



**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

## **SIMILARITY SEARCH IN UNSTRUCTURED DATA USING DATA-TRANSITIVE MODELS**

by

*David Bernhauer*

A dissertation thesis submitted to  
the Faculty of Information Technology, Czech Technical University in Prague,  
in partial fulfilment of the requirements for the degree of Doctor.

Doctoral degree study programme: Informatics  
Department of Software Engineering

Prague, August 2022

---

**Supervisor:**

prof. RNDr. Tomáš Skopal, Ph.D.  
Department of Software Engineering  
Faculty of Information Technology  
Czech Technical University in Prague  
Thákurova 9  
160 00 Prague 6  
Czech Republic

Copyright © 2022 David Bernhauer

---

# Abstract

Similarity search is becoming part of the applications we use daily, e.g., in recommendation systems or multimedia search applications. As the amount of data grows, so does the need to search this unstructured data efficiently and effectively. While various similarity indexing approaches provide efficiency, their constraints on the used similarity limit the effectiveness that represents the relevance of the results. Hence, there is a demand for indexing methods that impose as few constraints as possible and still manage to index big multimedia databases.

This thesis presents new indexability indicators (triangularity and ptolemaicity) that consider the data structure required by indexes. Moreover, they can also capture violations of these constraints and possibly the level of such violations. Both indicators use an analysis of relationships between objects in the database. We have analyzed high-dimensional data using these indicators, and experiments confirmed the expected properties of these indicators.

The second part deals with transforming non-metric distance measures to enable the indexing of non-metric similarity spaces using traditional metric approaches. Metric indexes are the de facto standard in similarity search, so it is possible to use many existing indexes. As a solution, we proposed TriGenGA as an extension of the TriGen algorithm to generate general modifiers using genetic algorithms. The results showed that such modifiers outperform the existing TriGen algorithm's efficiency and effectiveness.

Finally, we defined a data-transitive similarity meta-model that illustrates inherently non-metric similarity. The main focus is on the relevance of similarity search. It is challenging to design a high-quality similarity model in the case of data with many duplicates or few similarity links. A data-transitive similarity meta-model solves this problem by constructing a chain of similar objects that can link even mutually dissimilar objects. At the same time, the chain itself is an explanation of why two objects are relevant. Moreover, although this is a completely new approach, it is possible to apply common similarity approaches. We have successfully tested this meta-model within the domain of open datasets.

The thesis is structured as a commentary on already published papers.

---

**Keywords:**

similarity search, indexability, non-metric similarity models, TriGen, data-transitive similarity meta-model.

---

# Abstrakt

Podobnostní vyhledávání se stává součástí aplikací, které používáme každý den, např. doporučovací systémy nebo aplikace pro vyhledávání multimedií. S rostoucím množstvím dat roste i potřeba v těchto nestrukturovaných datech rychle a efektivně vyhledávat. Zároveň různé podobnostní přístupy k indexaci zajišťují rychlost vyhledávání, jejich omezení limitují efektivitu reprezentovanou relevancí výsledků. Proto vzniká poptávka po indexačních metodách, které kladou co nejmenší omezení a přesto umožňují indexování velkých multimediálních databází.

Tato práce prezentuje nové indikátory indexovatelnosti (angl. triangularity a ptolemaicity) které zohledňují indexy požadovanou strukturu dat. Kromě toho dokáží zachytit i porušení těchto omezení a případně úroveň takového porušení. Oba indikátory využívají analýzy vztahů mezi objekty v databázi. Využitím těchto indikátorů jsme provedli analýzu vysoce dimenzionálních dat. Experimenty potvrdily očekávané vlastnosti těchto indikátorů.

Druhá část se zabývá transformací nemetrických vzdáleností, která umožňuje indexaci nemetrických podobnostních prostorů pomocí tradičních metrických přístupů. Metrické indexy jsou de facto standardem v oblasti podobnostního vyhledávání, takže je možné využít mnoho již existujících indexů. Jako řešení jsme navrhli TriGenGA jako rozšíření algoritmu TriGen o generování obecných modifikátorů pomocí genetických algoritmů. Výsledky ukázaly, že takové modifikátory překonávají existující TriGen algoritmus v rychlosti i efektivitě.

Na závěr jsme definovali datově-tranzitivní podobnostní meta-model, který je ukázkou inherentně nemetrické podobnosti. Hlavní důraz je kladen na relevanci podobnostního vyhledávání. V případě dat s mnoha duplicitami či málo podobnostními propojeními je obzvláště obtížný úkol vytvořit kvalitní podobnostní model. Datově-tranzitivní podobnostní meta-model řeší tento problém pomocí sestavení řetězu podobných objektů, který může propojovat i zcela nepodobné objekty. Zároveň je takový řetěz vysvětlením, proč jsou dva objekty vzájemně relevantní. Navíc, přestože se jedná o zcela nový přístup, je na něj možné aplikovat běžné podobnostní přístupy. Tento meta-model jsme úspěšně otestovali v rámci domény otevřených dat.

Práce je strukturována jako komentář k již publikovaným článkům.

---

**Klíčová slova:**

podobnostní vyhledávání, indexovatelnost, nemetrické podobnostní modely, TriGen, datově-tranzitivní podobnostní meta-model.

---

# Contributions

In particular, the main contributions of the dissertation thesis are as follows:

1. INDEXABILITY ANALYSIS

New indexability indicator *triangularity* for metric and *ptolemaicity* for Ptolemaic similarity models that consider the structure of objects defined by constraints given by these similarity models. At the same time, these indicators deal with possible violations of these constraints.

2. TRIGENGA

Proposal of an extension of the original TriGen algorithm that can deal with multi-parameter modifiers. Using genetic algorithms, it can find a general modifier that outperforms the original algorithm.

3. DATA-TRANSITIVE SIMILARITY META-MODEL

Definition of data-transitive similarity meta-model as a robust relevance-oriented similarity. This provides a novel self-explanatory way to enrich similarity by using mutually similar objects from the database.





---

# Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, prof. RNDr. Tomáš Skopal, Ph.D. He has been a constant source of encouragement and insight during my research and helped me with numerous problems and professional advancements. He motivated me, especially at the beginning and the end of my study when the support was most needed. I also want to thank him for many hours of consultations that pushed me further in my scientific career.

Special thanks go to the staff of the Department of Software Engineering, who maintained a pleasant and flexible environment for my research. I would also like to thank my colleagues from the Department of Software Engineering at Charles University, who co-authored scientific publications and provided many tips and ideas for my research. I would like to express special thanks to the department management for providing most of the funding.

My research has also been partially supported by the Technology Agency of the Czech Republic (TAČR) as project No. TH03010276, by the Czech Science Foundation (GAČR) as project No. 19-01641S, by the Czech Science Foundation (GAČR) as project No. 17-22224S, and by the Grant Agency of the Czech Technical University in Prague, grants No. SGS17/210/OHK3/3T/18 and No. SGS20/213/OHK3/3T/18.

I would like to express thanks to my colleagues and students, who were a source of inspiration and motivation during my whole study. Many thanks belong to my friends who supported me during the tough times. Special thanks belong to Adam and Jan for their valuable comments and proofreading.

Last but not least, my greatest thanks go to my family members for their infinite patience, care, and support in my studies.

---

## Dedication

To my parents

*Jiřina Bernhauerová*

and

*Antonín Bernhauer*

for their endless love, support and encouragement.

---

# Contents

Notation	xv
<b>I Commentary</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Similarity Search . . . . .	3
1.1.1 Motivation . . . . .	4
1.2 Problem Statement . . . . .	4
1.3 Goals of the Thesis . . . . .	5
1.4 Structure of the Thesis . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Basic Concepts . . . . .	7
2.1.1 Similarity vs. Distance . . . . .	8
2.1.2 Similarity Search . . . . .	12
2.1.3 Quality Measures . . . . .	14
2.2 Metric-space Indexing . . . . .	16
2.2.1 Filtering . . . . .	16
2.2.2 Indexability . . . . .	18
<b>3 Non-metric Similarity Models</b>	<b>21</b>
3.1 Introduction . . . . .	21
3.2 Related Work . . . . .	22
3.3 Indexability . . . . .	24
3.4 TriGenGA . . . . .	25
3.5 Discussion . . . . .	27
<b>4 Data-Transitive Similarity Metamodel</b>	<b>29</b>

4.1	Introduction . . . . .	29
4.2	Related Work . . . . .	30
4.3	Our Approach . . . . .	31
4.4	Discussion . . . . .	33
<b>5</b>	<b>Conclusion</b>	<b>35</b>
5.1	Contributions of the Thesis . . . . .	35
5.2	Future Work . . . . .	36
	<b>Bibliography</b>	<b>39</b>
	<b>Reviewed Publications of the Author Relevant to the Thesis</b>	<b>45</b>
	<b>Remaining Publications of the Author Relevant to the Thesis</b>	<b>49</b>
	<b>Remaining Publications of the Author</b>	<b>51</b>
<b>II</b>	<b>Collection of Works</b>	<b>53</b>
1	Open Dataset Discovery using Context-enhanced Similarity Search	55
2	Modular Framework for Similarity-based Dataset Discovery using External Knowledge	87
3	Similarity vs. Relevance: From Simple Searches to Complex Discovery	119
4	Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs	135
5	Analysing Indexability of Intrinsically High-dimensional Data using TriGen	147
6	Inferred Social Networks: A Case Study	157
7	Advanced Behavioral Analyses Using Inferred Social Networks: A Vision	163
8	SIMILANT: An Analytic Tool for Similarity Modeling	175
9	Non-metric Similarity Search Using Genetic TriGen	181
10	Approximate Search in Dissimilarity Spaces using GA	191

---

## List of Figures

2.1	Comparison of several $L_p$ distances. . . . .	10
2.2	Range and $k$ NN query . . . . .	13
2.3	Illustration of filtering in metric spaces using pivots. . . . .	17
3.1	Example of TriGen modifier . . . . .	23
3.2	TriGenGA example . . . . .	26
3.3	TriGenGA error efficiency . . . . .	28
4.1	Example of inferred similarity of users in graph using clustering ATMs. . . . .	30
4.2	Example of data-transitive similarity of movies . . . . .	32



---

## Notation

$\mathbb{U}$	universe (domain) of valid objects
$\mathbb{S} \subset \mathbb{U}$	database of objects
$\mathbb{S}^* \subset \mathbb{S}$	random subset of database
$x, y, z, o_i \in \mathbb{S}$	objects from database
$q, q_i \in \mathbb{U}$	query objects from domain
$s : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$	similarity measure between pairs of objects in $\mathbb{U}$
$\delta : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}_0^+$	dissimilarity measure (distance) between pairs of objects in $\mathbb{U}$
$\mathcal{M} : \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$	distance modifier
$\mathcal{Q}_{k\text{NN}}(q, k)$	$k$ NN query with parameter $k$
$\mathcal{Q}_{\text{range}}(q, r)$	range query with parameter $r$





Part I  
Commentary



---

# Introduction

*In this chapter, we introduce the reader to the field of similarity search and the primary points of research. Section 1.1 introduces the reader to what similarity search deals with and where they may encounter it in everyday life. In Section 1.2, we define the current problems in this domain that are addressed in this thesis. The main goals are presented in Section 1.3. Section 1.4 describes the structure of the thesis.*

## 1.1 Similarity Search

Searching in structured databases is now standard, and almost no IT sector can do it without using relational databases. In this case, a data structure is an essential feature, otherwise query languages like SQL could not work properly. In many cases, the strict data structure of relational databases has proved too restrictive. Hence, document, graph, and other databases based on alternative data structures have come into popularity.

In recent years, however, the amount of unstructured and unannotated data has rapidly grown. Multimedia databases are becoming a part of our lives. This creates demand to navigate in these databases without the need for manual annotation or structure definition. Keyword search proves insufficient and often problematic, especially when expressing more complex concepts. Thus, similarity search comes with querying by example, which is applied in many domains.

The similarity is based on expert knowledge, unlike other domains (artificial intelligence, probabilistic models). Evaluating whether two database elements (e.g., images) are similar is a subjective matter. At the same time, it turns out that different definitions of similarity are suitable for different applications. For one use case, it may be useful to measure similarity based on color distribution. Meanwhile, for another use case, it may be better to compare sets of visual features. This property, complicates algorithms working with similarities. Therefore, some algorithms restrict themselves to subsets of similarities, making it challenging to use non-trivial similarities.

### 1.1.1 Motivation

Many streaming applications apply similarity search within their recommendation algorithms at the content-based recommendation level, searching for similar songs, clips, or products. [1, 2] They use a similarity-based approach to identify users with similar behavioral patterns and then match preferences at the collaborative filtering level. [3] Identifying a user from a photo, footage, or other data is another domain where we may encounter similarity search. [4]

Recently, similarity search has also gained importance due to the need for information verification (fact-checking). [5] First of all, thanks to similarity search, we can often easily trace the original source of information (e.g., an image), regardless of frequent artificial modifications. In the second place, similarity search can be used to verify the authenticity of information, such as the location captured in an image or video. Finding a similar photo from the exact location, where we have confidence in the annotation attached, can be challenging in unstructured data. [6]

With the growing amount of data comes the need to find efficient algorithms, that can handle even such unstructured data indexing. The definition of similarity is a rather subjective matter and efficient algorithms often have various constraints on how such similarity should look like. Exploration is also a relatively common target of similarity search. [7] Exploration allows us to find what we are looking for using a sequence of steps instead of one query. The results of such search could be more relevant to a user. A similarity directly defined as exploration will, by definition, not satisfy some of the constraints of indexes. Thus, the algorithms need to be adapted.

## 1.2 Problem Statement

In our research, we address two problem domains. The first domain is indexing non-metric similarities using existing metric indexes. Metric indexes are efficient, but we are limited to using only metric distance measures. This requires the use of techniques that allow conversion from non-metric to metric space while preserving properties important for similarity measures. Thus, one of the goals is to propose an approach that allows a more general construction of transformations from non-metric space to metric space, using a genetic algorithm approach. Secondary criterion is to keep indexing and search as efficient as possible.

The second domain is data-transitive similarity. This thesis aims to define what data-transitive similarity is, how it relates to exploration, and to show its practical use in a real-world use case. Another goal is to present the limitations associated with transitive similarity.

## 1.3 Goals of the Thesis

In this section, we briefly summarize the main goals of this dissertation:

1. INDEXABILITY ANALYSIS (see Chapter 3)  
To create novel indexability indicators that are more sensitive to the structure of metric (or Ptolemaic) similarity models. The new indicators should also be able to deal with the violation of triangular (or Ptolemaic) inequality, so that they can be used for indexability analysis of non-metric similarity models.
2. TRIGENGA (see Chapter 3)  
Propose a variant of the TriGen algorithm using a genetic algorithm to generate more general modifiers that optimize both efficiency and effectiveness better.
3. DATA-TRANSITIVE SIMILARITY META-MODEL (see Chapter 4)  
Define a new similarity model linking dissimilar objects to each other using a chain of similar objects (simulating exploration).

## 1.4 Structure of the Thesis

This thesis is a brief commentary on our publications. The commentary is divided into five chapters as follows:

**Chapter 1** summarizes the motivation and the basic goals of our efforts. It also presents a summary of the main contributions.

**Chapter 2** introduces basic concepts in similarity search, metric similarity models, and indexing.

**Chapter 3** discusses non-metric spaces, transformations to metric spaces and presents the current state-of-the-art. We also mention indexability indicators in the context of non-metric similarity. Finally, it presents the basic idea of transformations generated by genetic algorithms and experimental evaluation.

**Chapter 4** defines a data-transitive similarity meta-model and its important properties. We demonstrate the importance and relevance of the domain of open data search.

**Chapter 5** summarizes the presented research results and suggests possible future research topics.

All contributions are properly published and cited in Bibliography. Relevant papers are mentioned at the beginning of each section and their full parts are included in the second part of the thesis (Part II: Work 1–Work 10).



---

## Background

*This chapter is divided into two parts. Section 2.1 describes the basics of similarity search and quality measurement. Section 2.2 presents the basic principles and requirements for efficient search within metric spaces.*

### 2.1 Basic Concepts

Similarity as a concept is very individual and subjective. It is challenging to decide whether two objects are similar in everyday life. Nevertheless, people are very good at making relative comparisons between pairs of objects from the same universe, e.g., for the domain of animals, whether a fish and a whale are more similar than a cat and a dog. In our context, similarity is primarily a qualitative measure, formally defined by Definition 2.1.1, where  $\mathbb{U}$  is the universe (domain) of all objects. Thus the concept of similarity itself is not restrictive. In real cases, these similarity measures have further restrictions.

**Definition 2.1.1** (Similarity Measure). Let  $s$  be a pairwise similarity function  $s : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}$  defined in the way that  $\forall x, y, z \in \mathbb{U}$  if  $s(x, y) > s(x, z)$  holds, then object  $x$  is more similar to object  $y$  than object  $z$ .

The object can be a document, an image, a video, virtually anything. This raises the following question: if we are comparing animals, are we interested in their visual similarity or genetic similarity? For different kinds of similarity measures, a different subset of data, attributes, and features are appropriate. Therefore, similarity measures usually do not work directly with the original object but only with some simplified representation of it (vector, set, ...). Formally, we can define this transformation as a function  $\sigma : \mathbb{O} \mapsto \mathbb{U}$  that maps the original objects to the domain of the similarity measure. In information retrieval, such a result of the transformation is called a descriptor. In image processing, the same concept is called a feature. We will ignore this implementation detail for our purposes, and when we discuss objects, we will always implicitly consider their representations.

Since the definitions themselves can be quite abstract, let us give an example of several similarity measures. A frequently used similarity measure in the vector domain is the *Cosine similarity* [8]. According to Equation 2.1, Cosine similarity returns values in the interval  $\langle -1, 1 \rangle$ , where a value of  $-1$  means that the vectors are completely opposite each other, while a value of  $1$  means that the vectors point in the same direction. The *Jaccard index* [9], which is a popular similarity measure in the domain of sets, has different bounds. According to Equation 2.2, the Jaccard index returns values in the interval  $\langle 0, 1 \rangle$ , where a value of  $0$  means that the sets have no element in common (they are completely different). Conversely, a value of  $1$  means that the sets are identical.

$$s_{\text{Cosine}}(\vec{x}, \vec{y}) = \frac{\vec{x}\vec{y}}{\|\vec{x}\| \|\vec{y}\|} = \frac{\sum_{i=1}^N x_i y_i}{\sqrt{\sum_{i=1}^N x_i^2} \sqrt{\sum_{i=1}^N y_i^2}} \quad (2.1)$$

$$s_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2.2)$$

### 2.1.1 Similarity vs. Distance

Although the definition of the similarity measure does not require a formal restriction on the domain of values, such a restriction makes sense from a logical point of view. After all, no matter how we define similarity, we usually want to specify the situation where two objects are identical. However, the general definition of similarity measure is too benevolent and makes no assumptions about the identity of two objects. If two objects have a similarity of  $0.2$ , we cannot determine whether the objects are identical or not without knowing the details of the similarity measure.

Therefore, in the field of similarity search, we encounter the concept of distance measure more often. The *distance* measure  $\delta$  (sometimes also  $d$ ) is a kind of an inverse concept with respect to the similarity measure, formally defined by Definition 2.1.2. While this may not formally denote the identity property (see Definition 2.2.1), in most cases it will, or we are able to achieve this by a suitable transformation of the original distance measure. Since the distance measure is defined differently in various literature [10, 11], we will discuss these concepts and their differences later in Section 2.2 and Chapter 3.

**Definition 2.1.2** (Distance Measure). Let  $\delta$  be a *distance (dissimilarity) function*  $\delta : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}_0^+$  defined as the opposite of the similarity function  $s$ . It should satisfy *non-negativity* ( $\forall x, y \in \mathbb{U} \delta(x, y) \geq 0$ ) and ( $\forall x, y, z \in \mathbb{U} s(x, y) > s(x, z) \Leftrightarrow \delta(x, y) < \delta(x, z)$ ). The pair  $(\mathbb{U}, \delta)$  is a *dissimilarity space*.

To give a more specific idea, we will list some well-known distance measures categorized by different types of input data. The most trivial distance measure we can define is the *Discrete metric* [12] defined by Equation 2.3. The range of values are the numbers  $0$  and  $1$ , so for each pair of objects, it gives us information on whether the objects are



identical or not. Thus we can apply this distance measure over any set of objects that we are able to compare for equality. The practical use of such distance measure is very questionable.

$$\delta_{Discrete}(x, y) = \begin{cases} 0, & \text{for } x = y, \\ 1, & \text{for } x \neq y. \end{cases} \quad (2.3)$$

**Vector-based distances** One of the most common types of descriptors is the representation of an object by a vector of numbers. Some objects are inherently representable by a vector, e.g., the representation of a patient by his blood tests [13]. Other objects, such as text or images, are typically transformed into a vector implementation using embeddings [14]. The individual components of the vector then represent properties that can have real meaning (e.g., the dominant color of an image) or also abstract meaning (e.g., how blocky an image is).

We introduced cosine similarity  $s_{Cosine}$  in the previous section. In the case where we need to work more with distance, similarities can be converted to distance measures in several ways. For cosine similarity, we encounter two approaches most often. The first approach is to directly transform the similarity to a distance by changing the sign and applying an appropriate shift. The *Cosine distance* is defined as Equation 2.4 [15, 16]. An alternative approach is to view the distance in terms of angles and obtain the angle itself from the cosine similarity by applying trigonometric functions. *Angular distance* is defined as Equation 2.5 [15, 16]. Both of these distance measures have different properties (to be discussed later). Hence, the use of one over the other may be more advantageous in different situations, as stated by Cer et al. in [17].

$$\delta_{Cosine}(\vec{x}, \vec{y}) = 1 - s_{Cosine}(\vec{x}, \vec{y}) \quad (2.4)$$

$$\delta_{Angular}(\vec{x}, \vec{y}) = \frac{\arccos(s_{Cosine}(\vec{x}, \vec{y}))}{\pi} \quad (2.5)$$

The previous similarities describe distances based on the angles between two vectors. Regardless of their magnitudes, a special meaning is then given to the null vector  $\vec{0}$ , which cannot be used in this case. Instead, in some situations, the vectors represent points in space. For comparing such vectors, we commonly encounter the *Euclidean distance* defined as Equation 2.6. Deza and Deza [16] describe the Euclidean distance as "*as-the-crow-flies*", which aptly captures that such a distance is based more on the proximity of points to each other.

$$\delta_{Euclidean}(\vec{x}, \vec{y}) = L_2 = \sqrt{\sum_i^{\dim} (\vec{x}_i - \vec{y}_i)^2} \quad (2.6)$$

## 2. BACKGROUND

The *Manhattan distance* defined by Equation 2.7 takes a similar approach, which can be thought of as a path between two vectors along only parallel axes. As an analogy we can mention the *Chebyshev distance* defined by Equation 2.8, where the distance is equal to the maximum of the distances in each dimension. Figure 2.1 presents a visual comparison of these distance measures. These distance measures belong to the family of *Minkowski distances* or also known as  $L_p$  distances which can be described by Equation 2.9, where  $p \geq 1$ . By extending the Minkowski distances parameter to  $0 < p \leq 1$  we obtain *Fractional  $L_p$  distances*.

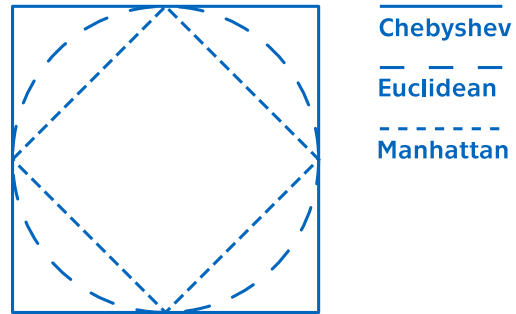


Figure 2.1: Comparison of several  $L_p$  distances. Line shows the shape of elements within the same distance from the center.

$$\delta_{\text{Manhattan}}(\vec{x}, \vec{y}) = L_1 = \sum_i^{\text{dim}} |\vec{x}_i - \vec{y}_i| \quad (2.7)$$

$$\delta_{\text{Chebyshev}}(\vec{x}, \vec{y}) = L_\infty = \max_i |\vec{x}_i - \vec{y}_i| \quad (2.8)$$

$$\delta_{\text{Minkowski}}^{p \geq 1}(\vec{x}, \vec{y}) = L_p = \left( \sum_i^{\text{dim}} |\vec{x}_i - \vec{y}_i|^p \right)^{\frac{1}{p}} \quad (2.9)$$

As a further generalization of the Euclidean distance, we can note the *Quadratic-form distance* defined as Equation 2.10, where  $\mathbb{A}$  is a non-singular matrix of  $\mathbb{R}^{\text{dim}, \text{dim}}$ . The matrix  $\mathbb{A}$  represents correlations between the dimensions. If matrix  $\mathbb{A}$  corresponds to a unit matrix (the dimensions are independent), we obtain an alternative notation for the Euclidean distance.

$$\delta_{\text{QFD}}^{\mathbb{A}}(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^\top \mathbb{A} (\vec{x} - \vec{y})} \quad (2.10)$$

**String-based distances** In the domain of texts, and especially shorter ones such as various headings, labels, and in general texts where it is very difficult to capture the meaning using embeddings (the context is quite small) or documents (text contains few unique words), we typically use a direct representation by text strings. However, we can also use this representation in many other domains, such as genetics where we can use strings to represent individual AGCT bases [18].

*Hamming distance* [16] assumes two strings of equal length on the input, defined as Equation 2.11. In simple terms, it is the number of positions at which the strings differ. This means that for the words peel and eels, the Hamming distance is equal to 3. In this

case, 4 is the maximum distance with respect to the length of the words. However, both words share the substring "eel".

$$\delta_{Hamming}(x[1..n], y[1..n]) = \sum_{i=1}^n \mathbf{1}_{x[i] \neq y[i]} = |\{i \in \mathbb{N} \mid x[i] \neq y[i] \wedge 1 \leq i \leq n\}| \quad (2.11)$$

Since equal string length is a fairly strict constraint, a different distance measure often used over strings is the *Levenshtein distance* [16], also known as *edit distance*, where distance represents the number of operations (insert, remove, substitute) needed to transform one string into the other. It is thus a generalization of Hamming distance. The distance is defined as Equation 2.12, where  $x[1..n]$  is a string  $x$  of length  $n$ . Since the function itself is recursive, such similarity is usually implemented using dynamic programming. Using the previous example, we get that "peel" and "eels" have edit distance equal to 2 (one per removing letter "p" and one per adding letter "s").

$$\delta_{Edit}(x[1..n], y[1..m]) = \begin{cases} |x| & \text{for } |y| = 0, \\ |y| & \text{for } |x| = 0, \\ \delta_{Edit}(x[2..n], y[2..m]) & \text{for } x[1] = y[1], \\ 1 + \max \begin{cases} \delta_{Edit}(x[2..n], y[1..n]) \\ \delta_{Edit}(x[1..n], y[2..n]) \\ \delta_{Edit}(x[2..n], y[2..n]) \end{cases} & \text{otherwise.} \end{cases} \quad (2.12)$$

**Time series-based distances** When processing data, we often take the time component into account. Therefore, we often encounter representations using time series, either univariate or multivariate. Thus, we are not directly comparing values at one point in time but how similar the events are. We can also use time series to represent, for example, simple gestures [19] or player behaviour in game [20].

In the context of time series, we can then encounter the *Dynamic time warping distance* (DTW, [21]) defined as Equation 2.13. From the definition, we can see a direct similarity to  $\delta_{Edit}$ , which is extended by the ground distance  $d(x, y)$ , which specifies the distance between two points at a particular point of time series  $\dot{x}^n = (\dot{x}_1, \dot{x}_2, \dots, \dot{x}_n)$ .

$$\delta_{DTW}^d(\dot{x}^n, \dot{y}^m) = \begin{cases} 0 & \text{for } n = 0 \wedge m = 0, \\ \infty & \text{for } n = 0 \vee m = 0, \\ d(\dot{x}_n, \dot{y}_m) + \max \begin{cases} \delta_{DTW}^d(\dot{x}^n, \dot{y}^{m-1}) \\ \delta_{DTW}^d(\dot{x}^{n-1}, \dot{y}^m) \\ \delta_{DTW}^d(\dot{x}^{n-1}, \dot{y}^{m-1}) \end{cases} & \text{otherwise.} \end{cases} \quad (2.13)$$

**Set-based distances** An exciting category of descriptors are sets of elements. Whether they are actual sets of some specific elements or features, these are interesting kinds of descriptors. Like a time series, a descriptor consists of multiple unrelated objects. A typical example is a set of keywords, where we can consider either exact matching in finding the intersection or at least partial matching and finding the best match (e.g., accepting typos).

When comparing sets, the simplest variant is the aforementioned Jaccard index. To obtain the distance measure, it is sufficient to make a similar adjustment as in the case of Cosine distance. *Jaccard distance* [16] is defined as Equation 2.14. Simply put, it treats elements as being either the same or different, similar to the Discrete metric. The result is the ratio of the intersection of sets to their union.

$$\delta_{Jaccard}(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \quad (2.14)$$

Some other distance measures, in turn, also consider similarities (distances) between elements of a set. For example, the *Hausdorff distance* [22] defined as Equation 2.15 considers the similarity of the elements  $d(x, y)$ , where  $x \in X, y \in Y$ . The goal is then to match elements of one set to the most similar elements of the other set.

$$\delta_{Hausdorff}^d(X, Y) = \max \left\{ \hat{\delta}_{asymHaus}^d(X, Y), \hat{\delta}_{asymHaus}^d(Y, X) \right\} \quad (2.15)$$

$$\hat{\delta}_{asymHaus}^d(X, Y) = \max_{x \in X} \min_{y \in Y} d(x, y)$$

### 2.1.2 Similarity Search

Similarity search differs from full-text search primarily in the form of querying. Standard keyword-based search or SQL-like languages are not sufficient since the data is not structured and often not annotated, e.g., we often do not have captions for photos. Instead, *query-by-example* style querying is used, where we already have an image (example) and search for images that are the similar or relevant.

The problem of similarity search can be formalized as a search for all relevant objects in the database  $\mathbb{S} \subset \mathbb{U}$ . Although the query to the database  $\mathbb{S}$  is from domain  $\mathbb{U}$ , the result of the search is always a subset of the database  $\mathbb{S}$ . The database  $\mathbb{S}$  is mostly assumed to be a static database because building the index is difficult. Nevertheless, there exist solutions for dynamic databases. [23, 24]

One basic type of query is the *range query* [10]. Such a query consists of a search pattern  $q$  (query) and a parameter  $r$  that conditions the distance of the objects in the query result. The formalization of a range query is captured in Definition 2.1.3. The  $r$  parameter is dependent on the distance  $\delta$  used; although it indirectly affects the size of the result set, it is not possible to estimate the exact size. We can only rely on the statement that as  $r$  grows, the size of the resulting set  $\mathcal{Q}_{range}(q, r)$  grows.

**Definition 2.1.3** (range query). Let  $q \in \mathbb{U}$  be a query object and  $r \in \mathbb{R}_0^+$  the range of the query. Then  $\mathcal{Q}_{range}(q, r)$  is the *range query*. The result of the range query is  $\mathcal{Q}_{range}(q, r) = \{x \in \mathbb{S} \mid \delta(q, x) \leq r\}$ .

Range query basically divides the database  $\mathbb{S}$  into two disjunctive subsets  $\mathcal{Q}_{range}(q, r)$  and  $\mathbb{S} \setminus \mathcal{Q}_{range}(q, r)$ . By applying several different  $r$  to the same query  $q$ , we can obtain a partition into several different similarity levels. The naive implementation of range query, illustrated by Algorithm 1, requires computing  $O(|\mathbb{S}|)$  distance comparisons. Thus, the goal of indexing algorithms is primarily to reduce the number of those comparisons. The range query is effectively used in the DBScan clustering algorithm [25] to analyze neighborhood density.

---

**Algorithm 1:** Naive implementation of *range query*

---

**Input:** database  $\mathbb{S} \subset \mathbb{U}$ , distance  $\delta$ , range query  $\mathcal{Q}_{range}(q, r)$

**Result:** set of relevant objects  $V$

$V \leftarrow \{\}$

**for**  $x \in \mathbb{S}$  **do**

**if**  $\delta(q, x) \leq r$  **then**  
    |  $V \leftarrow V \cup \{x\}$

**return**  $V$

---

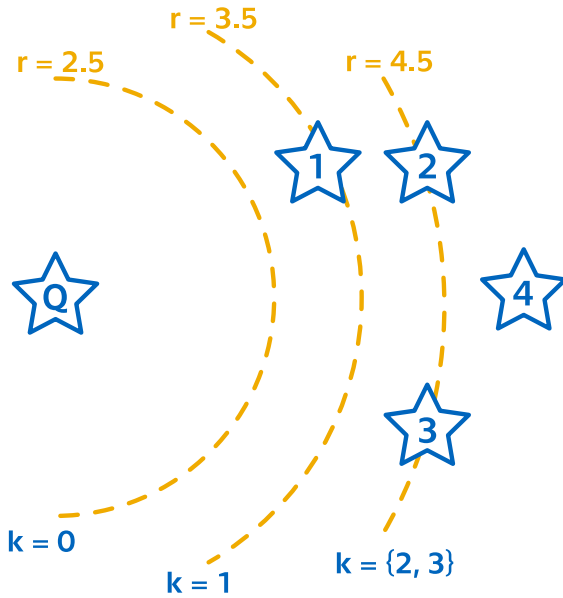


Figure 2.2: Example of a range (orange line) and a  $k$ NN query (blue numbers).

Range query has a rather complicated way of checking the size of the result set, and very often, we get into a situation where there is no object in the result set (Figure 2.2,  $r = 2.5$ ) or, on the contrary, all the objects from the database  $\mathbb{S}$ . This problem is addressed by the *kNN query* ( $k$ -nearest neighbors, [10]), which consists of a search pattern  $q$  and a parameter  $k$  that specifies how many nearest neighbors we want to retrieve. The formalization of the  $k$ NN query is captured in Definition 2.1.4. Although the parameter  $k$  explicitly specifies the number of nearest neighbors, it makes no claims about their distance to each other. The definition also shows that the resulting set  $V$  is not uniquely determined since the  $k$ th nearest neighbor can theoretically have the same distance as the  $(k + 1)$ th nearest neighbor. Figure 2.2 shows that stars #2 and #3 have the same distance  $\delta$ , so for  $k = 2$  there are 2 valid results.

**Definition 2.1.4** (*k*-nearest neighbors query). Let  $q \in \mathbb{U}$  be a query object and  $k \in \mathbb{Z}^+$  the number of nearest neighbors in the result  $V \subset \mathbb{S}$ . Then  $\mathcal{Q}_{k\text{NN}}(q, k)$  is the *k*-nearest neighbors query (alternatively *kNN query*). The result  $V$  of the *kNN* query is defined as  $|V| = k$ ,  $(\forall x \in V, \forall y \in \mathbb{S} \setminus V) \delta(q, x) \leq \delta(q, y)$ .

A naive implementation of *kNN* query is illustrated in Algorithm 2, which has time complexity  $O(|\mathbb{S}| \log(k))$ , but since the distances can be precomputed, it is not necessary to compute more than  $O(|\mathbb{S}|)$  distances. The constant  $k$  is usually small compared to the size of the database, and we can thus neglect  $\log(k)$ . Alternatively, instead of a heap, we can use the *Quickselect* algorithm, which finds the  $k$ th element in the set, splitting the set into elements that are smaller and larger than that element. Thus the total complexity will be just  $O(|\mathbb{S}|)$  in average.

---

**Algorithm 2:** Naive implementation of *kNN* query

---

```

Input: database  $\mathbb{S} \subset \mathbb{U}$ , distance  $\delta$ , kNN query  $\mathcal{Q}_{k\text{NN}}(q, k)$ 
Data: distance table  $d(q, \star)$  for  $q$ 
Result: set of relevant objects  $V$ 
/* precompute distance */
for  $X \in \mathbb{S}$  do
   $d(q, x) \leftarrow \delta(q, x)$ 
/* keep only  $k$  objects in heap */
HeapInit( $V$ )
for  $x \in \mathbb{S}$  do
  HeapAdd( $V, (d(q, x), x)$ )
  if  $|V| > k$  then
    HeapRemoveMax( $V$ )
return  $V$ 

```

---

### 2.1.3 Quality Measures

In order to be able to compare different similarity approaches, we need to define indicators to determine the quality of each approach. A typical indicator of the quality of an algorithm is its time complexity. As introduced in the previous section, the naive solution has asymptotically linear complexity with respect to the size of the database  $\mathbb{S}$ . It does not make much sense to improve the asymptotic complexity, because in the worst case it will always be linear. Speeding up will mainly be achieved by filtering out non-viable candidates, which will depend on many factors, including the query itself. In case of similarity search, we call this property *efficiency*.

Instead of asymptotic complexity, we can measure the actual runtime of the program. Such a metric has practical use but is dependent on the implementation itself and various optimizations. These may include the physical location of individual objects (or their

descriptors) in memory. Such a measurement is then influenced by other factors such as data retrieval from disk, cache, etc. The implementation itself or the chosen distance measure also play a significant role. We obtain quadratic complexity for *edit distance* or *DTW*, assuming we have enough memory to use dynamic programming. For *Hausdorff distance*, the situation is further complicated because the asymptotic complexity itself will depend on the ground distance.

Since the distance measure is one of the main factors influencing the resulting speed, measuring the average number of distance function calls over all queries makes sense. This also allows us to speed up the experiments by precomputing the distance matrix. Even for more complex similarity functions, we can easily and quickly evaluate different approaches.

The second direction we may be interested in for similarity search is the quality of the returned result, so-called *effectiveness*. Finding out the quality of the result makes sense if we experiment with different kinds of distances and have queries and their expected results. In this case, we check which model returns more or less relevant results from the user’s perspective. For a standard indexing method, this is not a meaningful comparison, as it depends only on the model itself. On the other hand, for approximation methods, we are mainly interested in quality compared to the naive approach or how much the result differs from the naive approach. Thus, in terms of effectiveness, we look at the following indicators.

*Precision* [26] is an indicator that directly defines the usefulness of the result. Precision is defined as the ratio of relevant (expected) objects in the result to the number of all retrieved objects. Practically, it determines the probability that an object in the result set is relevant. It is typically defined as precision-at- $k$  (or  $P@k$ , [27]), formally defined as Equation 2.16. Technically, for  $k = 0$ , we can define that precision is equal to 1.

$$P@k = \frac{|\text{relevant} \cap k \text{ retrieved}|}{|k \text{ retrieved}|} = \frac{|\text{relevant} \cap k \text{ retrieved}|}{k} \quad (2.16)$$

*Recall* [26] is an indicator that complements precision with information about whether we have retrieved all relevant objects. Recall can be defined as the ratio of relevant (expected) objects in the result to the number of all relevant objects in the database. Although we can determine precision from the query result alone, we need to know the entire database to determine the resulting recall. Typically, we determine the recall-at- $k$  (or  $R@k$ ), formally defined as Equation 2.17. Symmetrically to precision, for  $k = |S|$  we get that recall is equal to 1.

$$R@k = \frac{|\text{relevant} \cap k \text{ retrieved}|}{|\text{relevant}|} \quad (2.17)$$

Precision and recall are concepts that are opposed to each other in the general case. As  $k$  increases, recall increases, but precision decreases. Therefore, instead of  $P@k$  and  $R@k$ , we often encounter the so-called *precision-recall curve* (PR curve), which describes how precision changes as a function of recall. We can also encounter the 11-point variant of the precision-recall curve [28], where we consider recall only in the levels  $\{0, 0.1, 0.2, \dots, 1\}$ . To get a single value, we can consider, for example, the area under the curve for comparison.

*F-measure* (Equation 2.18, [29]) is another way to convert a precision and recall pair into a single quantity. In simple terms, it is the harmonic mean of precision and recall. But we may encounter a generalized version in the form of the  $F_\alpha$  measure, which is formally defined as Equation 2.19 ( $\alpha \in \langle 0, 1 \rangle$ ). Thus, the ideal value is 1, where we have 100% precision and recall.

$$F = \frac{2PR}{P + R} \quad (2.18)$$

$$F_\alpha = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} \quad (2.19)$$

## 2.2 Metric-space Indexing

Naive approaches to similarity search are insufficient with the rise of big data. The velocity and volume of data are increasing every year, and with them, the need for efficient search in unstructured data. For this purpose, we use various indexes. In our work, we focus on indexing metric spaces and related techniques.

*Metric space* is a prerequisite for many similarity indexes and applications. Metric space is defined as Definition 2.2.1. We use the non-negativity as an implicit property of the distance measure (Definition 2.1.2) in comparison to other definitions. At the same time, in some literature, we can see definitions where identity is replaced by *reflexivity* ( $\forall x \in \mathbb{U}, \delta(x, x) = 0$ ) and *non-negativity* (*positivity*,  $\forall x, y \in \mathbb{U}, x \neq y \implies \delta(x, y) > 0$ ).[10]

**Definition 2.2.1** (Metric space). *Metric space* is a pair  $(\mathbb{U}, \delta)$ . To qualify as metric space, the domain  $\mathbb{U}$  and the distance  $\delta$  must satisfy three requirements known as the *metric axioms*: identity, symmetry and triangle inequality.

$$\forall x, y \in \mathbb{U}, x = y \iff \delta(x, y) = 0 \quad (\text{identity})$$

$$\forall x, y \in \mathbb{U}, \delta(x, y) = \delta(y, x) \quad (\text{symmetry})$$

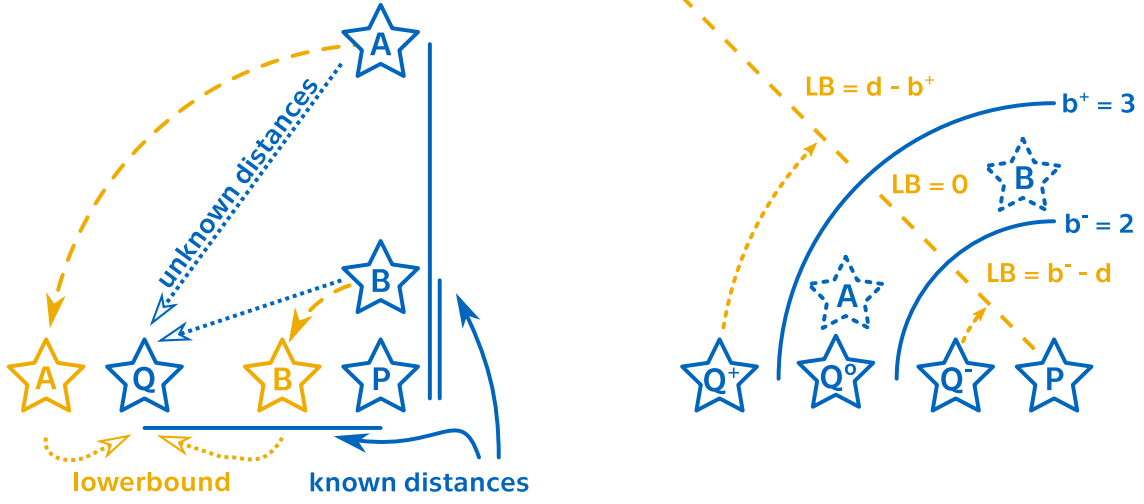
$$\forall x, y, z \in \mathbb{U}, \delta(x, z) \leq \delta(x, y) + \delta(y, z) \quad (\text{triangle inequality})$$

In the case of invalidity of the *identity* criterion, we are able to fill in this deficiency as a special case, or if  $\exists c \in \mathbb{R}, \forall x, y \in \mathbb{U}, \delta(x, x) = c \wedge \delta(x, y) \geq c$  holds, we can create a new distance measure  $\delta_0(x, y) = \delta(x, y) - c$  satisfying this rule. Similarly, in the case of a failure to satisfy *symmetry*, we can define a new distance measure  $\delta_{sym}$  as, for example,  $\delta_{sym}(x, y) = \min\{\delta(x, y), \delta(y, x)\}$ . [30]

### 2.2.1 Filtering

The most important of all rules is the *triangle inequality*. It is this property that allows efficient indexing of similarity. The inequality itself allows us to efficiently estimate the lower  $\delta_{LB}(x, y)$  and upper  $\delta_{UB}(x, y)$  bounds with partial knowledge of the distances.





(a) Filtering using 3 points

(b) Filtering using 2 points and interval

Figure 2.3: Illustration of filtering in metric spaces using pivots.

Suppose that for some objects  $x_1, x_2, \dots, x_n \in \mathbb{U}$  we know the distances between two consecutive objects  $d_{x_i x_{i+1}} = \delta(x_i, x_{i+1})$  and the distance  $\delta$  satisfies the metric axioms. Then from the triangle inequality it follows that  $\delta(x_1, x_3) \leq d_{x_1 x_2} + d_{x_2 x_3} = \delta_{UB}(x_1, x_3)$ , in general for  $a < b$ ,  $\delta(x_a, x_{b+1}) \leq \delta_{UB}(x_a, x_b) + d_{x_b x_{b+1}} = \delta_{UB}(x_a, x_{b+1})$ , so for  $\delta(x_1, x_n) \leq \sum_{i=1}^{n-1} d_{x_i x_{i+1}} = \delta_{UB}(x_1, x_n)$ . This knowledge can be trivially used, for example, to process a range query  $\mathcal{Q}_{range}(q, r)$ . If  $\delta_{UB}(q, x) \leq r$ , then we can say with confidence that the object  $x$  will be in the result of this query and there is no need to compute the exact  $\delta(q, x)$  distance.

Suppose that for some objects  $p, x, y \in \mathbb{U}$  we know the distances  $d_{px} = \delta(p, x)$ ,  $d_{py} = \delta(p, y)$  and the distance  $\delta$  satisfies the metric axioms. Then the triangle inequality implies  $d_{py} \leq d_{px} + \delta(x, y)$  and we can obtain the relation defined by Equation 2.21, simplified to just  $\delta_{LB}(x, y) = |d_{px} - d_{py}| \leq \delta(x, y)$ . Figure 2.3a illustrates both cases graphically. For most purposes, the object  $p$  is called *pivot*.

$$\delta_{LB}(x, y) = \begin{cases} d_{px} - d_{py} & \text{for } d_{px} > d_{py} \\ d_{py} - d_{px} & \text{otherwise} \end{cases} \leq \delta(x, y) \quad (2.21)$$

Suppose that for some objects  $p, x, y \in \mathbb{U}$  we know the distance  $d_{px} = \delta(p, x)$  and for  $\delta(p, y)$  we know the lower and upper bounds  $b_{py}^- \leq \delta(p, y) \leq b_{py}^+$  and the distance  $\delta$  satisfies the metric axioms. Unlike the previous instance, we get more cases, these are shown in Equation 2.22. After a brief analysis of each case, we can simplify it to  $\delta_{LB}(x, y) = \max \{d_{px} - b_{py}^+, 0, b_{py}^- - d_{px}\} \leq \delta(x, y)$ . Figure 2.3b visually illustrates all cases graphically.

$$\delta_{LB}(x, y) = \left\{ \begin{array}{ll} d_{px} - b_{py}^+ & \text{for } b_{py}^+ < d_{px} \\ 0 & \text{for } b_{py}^- \leq d_{px} \leq b_{py}^+ \\ b_{py}^- - d_{px} & \text{for } d_{px} < b_{py}^- \end{array} \right\} \leq \delta(x, y) \quad (2.22)$$

These principles are used by various indexes to filter the database and make queries more efficient (AESA [31], LAESA [32], M-tree [33], PM-tree [34], ...). Although in this paper we will only discuss indexing based on metric axioms, it is important to mention that not all indexes are based on this principle. For example, permutation indexes [35] do not rely on metric space at all and use information about the order of pivots according to the distance to the object for indexing. Ptolemaic indexes [36], on the other hand, are based on satisfying Ptolemy's inequality. Inverted index [37] is a popular structure in information retrieval domain.

## 2.2.2 Indexability

The indexes provide efficient structures, but the important question is whether a given space is indexable. The real efficiency of indexing algorithms and structures depends on the similarity space itself. Indexability can be affected by the chosen distance  $\delta$ . For example, the *discrete metric* is practically non-indexable. Since all objects are equidistant from each other, there is no internal structure to allow indexing.

Similarly, the poor indexability may be due to the combination of the chosen distance and the database  $\mathbb{S}$ . Thus, *Euclidean distance* may behave similarly when the dimension is high, and the number of elements is small. A high dimension will have the same effect as in the case of *discrete metric*, namely that all objects are similarly far apart. In an extreme case, we can have just one dimension reserved for each object, then the *Euclidean distance* becomes a *discrete metric*. This effect is well-known as the *curse of dimensionality* by Chavez et al. [38].

Chavez et al. proposed intrinsic dimensionality, which serves as an indexability indicator. Applying the Definition 2.2.2 of intrinsic dimensionality shows that intrinsic dimensionality grows with the higher mean of distances and lower variance of distances. Due to a large number of objects in the database  $\mathbb{S}$ , the intrinsic dimensionality is usually only estimated from a smaller random sample.

**Definition 2.2.2** (Intrinsic Dimensionality). The *intrinsic dimensionality* of a metric space is defined as  $IDim(\delta) = \frac{\mu^2}{2\sigma^2}$ , where  $\mu$  and  $\sigma^2$  are the mean and variance of its histogram of distances  $\delta$ .

Intrinsic dimensionality is practically only one number summarizing the whole complex model, this is quite practical for automatic evaluation. The disadvantage is that the same *IDim* can represent many different distributions. This is why in the case of manual analysis one works with the distance distribution itself. That provides information not only about the relationship between mean and variance, but also about the overall distribution

of distances. Unfortunately, both intrinsic dimensionality and distance distributions themselves only deal with the distance of two objects, but do not look further at the distance relationships between them.

Skopal [39] comes up with the idea of an indicator that relates to metric access methods based on ball partitioning. The *ball-overlap factor* (*BoF*, [39]) represents an indicator that tracks the ratio between the overlaps of  $k$ NN ball regions. Such an overlap implies inefficiency for indexes based on partitioning of objects into ball regions. If there are many overlaps, the indexing will not be very efficient. On the contrary, if there is no overlap, it is a perfect partition.



---

## Non-metric Similarity Models

*This chapter describes the problem of searching in a non-metric similarity model using metric indexes. The chapter is divided into five parts. Section 3.1 includes the introduction and motivation behind using metric approaches to solve the non-metric similarity problem and describes the basic terms and concepts. Section 3.2 then presents an overview of Related Work in the area. Section 3.3 describes new indexability indicators for analysis of high-dimensional data in the both metric and non-metric spaces. Section 3.4 introduces a generalization of TriGen using genetic algorithm. Finally, Section 3.5 discusses the findings and future work. The details are described in the author's papers Work 5, 9, and 10.*

### 3.1 Introduction

In the previous section, we described the principles of efficient search using the metric space model. The main advantage is a large number of existing indexes based on the metric model. The metric model introduces constraints in the form of metric axioms. However, many distance measures will not satisfy these metric axioms. This is the main difference between cosine and angular distance, where cosine distance is easy to compute but is not metric. Conversely, angular distance satisfies the metric axioms, but calculating *arccos* function is time-consuming. Other distances, such as Fractional  $L_p$  distance measures or DTW distance, are non-metric too.

This problem increases if we consider user-defined distance measures. Such a restriction is a huge limitation and prevents using more robust similarities. This chapter presents ways to modify the original similarity to preserve its properties while ensuring that the triangle inequality is satisfied. Such an approach will provide an efficient similarity search regardless of the distance chosen. Possible violations of some metric axioms (identity, symmetry) can be corrected easily; we have shown this in Section 2.2.1.

The most complicated part is to ensure the validity of the triangle inequality. For the purposes of this thesis, a *semimetric distance* is a distance measure that satisfies all

metric axioms except the triangle inequality. The triangular inequality is necessary for metric indexes to index the database efficiently. In this section, we introduce methods that attempt to deal with the problem using a *modifier* (a transformation function, Definition 3.1.1), as well as our proposed method that uses machine learning methods to ensure triangular inequality.

**Definition 3.1.1.** Let the function  $\mathcal{M}(d) : \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$  be increasing and  $\mathcal{M}(0) = 0$ . Then we call such a function a *modifier*. The modifier  $\mathcal{M}$  can be applied to the distance measure  $\delta$  as  $(\mathcal{M} \circ \delta)(x, y) = \mathcal{M}(\delta(x, y))$ .

Most of the methods assume that the chosen distance  $\delta$  (respectively modifier  $\mathcal{M}$ ) is  $\langle 0, 1 \rangle$ -bounded, i.e.  $\delta : \mathbb{U} \times \mathbb{U} \rightarrow \langle 0, 1 \rangle$  (respectively  $\mathcal{M} : \langle 0, 1 \rangle \rightarrow \langle 0, 1 \rangle$ ). In the general case, we could not apply such a method to many distances that are unbounded (e.g., *Minkowski distances*, *Hausdorff distance*). In our case, we always index a finite database  $\mathbb{S} \in \mathbb{U}$ , so for distances within the database, we can estimate  $d^+ = \max_{x, y \in \mathbb{S}} \delta(x, y)$ . Since queries  $q \in \mathbb{U}$  are usually not elements of the database  $\mathbb{S}$ , it makes sense to multiply  $d^+$  by a suitable constant (e.g.,  $(1 + \varepsilon)$ , where  $\varepsilon \geq 0$ ) and/or constrain the resulting distance  $\delta_{\langle 0, 1 \rangle\text{-bounded}}(x, y) = \min\left(\frac{\delta(x, y)}{d^+ \cdot (1 + \varepsilon)}, 1\right)$ . In the following sections, we will always assume that the distance and modifiers used are  $\langle 0, 1 \rangle$ -bounded.

## 3.2 Related Work

The simplest approach to obtain a metric distance from a semimetric distance is to use the following modifier

$$\mathcal{M}^M(d) = \begin{cases} 0 & \text{for } d = 0, \\ \frac{d+1}{2} & \text{otherwise,} \end{cases}$$

which transforms an arbitrary semimetric distance into a metric distance.[11] All other metric properties of the original distance are preserved. However, the problem with this approach is that the intrinsic dimensionality shown in Equation 3.1 is too high.

$$IDim(\mathcal{M}^M \circ \delta) = \frac{\mu_{\mathcal{M}^M \circ \delta}^2}{2\sigma_{\mathcal{M}^M \circ \delta}^2} = \frac{\left(\frac{\mu+1}{2}\right)^2}{2\left(\frac{\sigma}{2}\right)^2} = \frac{(\mu+1)^2}{2\sigma^2} = \frac{\overbrace{\mu^2}^{\leq 1} + \overbrace{2\mu+1}^{\geq 1}}{2\sigma^2} = IDim(\delta) + \frac{2\mu+1}{2\sigma^2} \quad (3.1)$$

**Constant Shift Embedding** A more elegant approach is the *Constant Shift Embedding* method [40], which uses a similar principle to the previous case  $\mathcal{M}^M$ . Authors came up with the idea that it is enough to shift the original distance  $\delta$  by a suitable constant  $0 \leq c \leq 1$ , i.e.  $\mathcal{M}_c^{CSE}(d) = \frac{d+c}{1+c}$ . This approach is a generalization of the previous method, where  $\mathcal{M}^M = \mathcal{M}_1^{CSE}$ . The constant  $c$  can be chosen based on the random sample  $\mathbb{S}^* \subset \mathbb{S}$  such that  $\forall x, y, z \in \mathbb{S}^*$  holds triangle inequality  $((\mathcal{M}_c^{CSE} \circ \delta)(x, z) \leq (\mathcal{M}_c^{CSE} \circ \delta)(x, y) + (\mathcal{M}_c^{CSE} \circ \delta)(y, z))$ .

**TriGen** An alternative approach was introduced by Skopal [30] with his TriGen algorithm. The algorithm can be considered a generalization of the Constant Shift Embedding approach. Over a random sample  $\mathbb{S}^*$ , it defines  $\epsilon(\mathbb{S}^*, \delta)$  as the percentage of triplets of objects from the sample  $\mathbb{S}^*$  that violate the triangle inequality using the distance  $\delta$ . The basic idea is that if we apply an arbitrary concave modifier  $\mathcal{M}^+$  to the distance  $\delta$ ,  $\epsilon(\mathbb{S}^*, \mathcal{M}^+ \circ \delta)$  will be lower (at least equal) than before the application. Such a modifier is called *triangle-generating*.

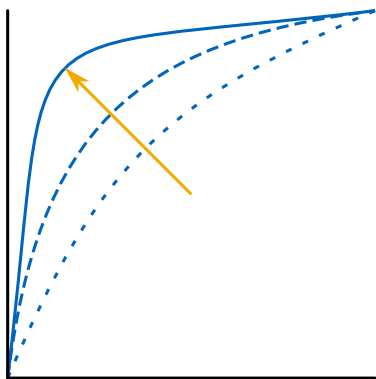


Figure 3.1: Example of TriGen modifier. Orange arrow denotes concavity.

An example of such modifier is *Fractional-power T-base* (Equation 3.2), which is concave for  $w > 0$  (Figure 3.1). The parameter  $w$  expresses the concavity and hence the power of the modifier to generate a triangular inequality. The algorithm defines a way to find the modifier with the best properties in general among the set of one-parameter modifiers. For each modifier, the algorithm uses a binary search to find  $w$  such that  $\epsilon(\mathbb{S}^*, \mathcal{M}_w)$  is less than or equal to the acceptable threshold  $\epsilon_{\mathbf{T}} \geq 0$ . Similarly, as in the case of Constant Shift Embedding, TriGen finds a parameter in such a way as to increase *intrinsic dimensionality* as little as possible. From the set of all matching modifiers  $\mathcal{M}_i$  (with the parameter  $w$  already found), the one that best satisfies the secondary criterion (e.g., *IDim*, *BoF*) is selected.

$$\mathcal{M}_w^{FP}(d) = \begin{cases} d^{\frac{1}{1+w}} & \text{for } w > 0 \text{ (triangle-generating),} \\ d & \text{for } w = 0, \\ d^{1-w} & \text{for } w < 0 \text{ (triangle-violating).} \end{cases} \quad (3.2)$$

The algorithm allows setting the threshold  $0 \leq \epsilon_{\mathbf{T}} \leq 1$ , which indirectly affects the quality of the search results. Indexes based on metric axioms can return bad results (filtering out something they should not). This will result in a just approximate search. On the other hand, the efficiency of such a search may be better.

Skopal [39] also generalizes the TriGen algorithm to *triangle-violating* modifiers. These are convex modifiers  $\mathcal{M}^-$ , which in contrast, can break the triangle inequality in the database. This again to the specified threshold  $\epsilon_{\mathbf{T}}$  to reduce *intrinsic dimensionality* and potentially increase indexability. TriGen thus tries to find the ideal compromise between effectiveness and efficiency. The main limitation remains in the form of individual modifiers.

**Other similar approaches** Several other properties have the potential to be used as an indexing tool in multimedia databases. Fagin et al. [41] utilize relaxed triangle inequality, which is defined as  $\delta(x, z) \leq \rho(\delta(x, y) + \delta(y, z))$ . In that case, authors use similar filtering methods as defined in Section 2.2.1.

Ptolemy’s indexing is another substitution for the triangle inequality. Lokoč et al. [36] presented the Ptolemy’s indexing with the Signature Quadratic Form Distance. Ptolemy’s inequality ( $\forall a, b, c, d \in \mathbb{U}$  holds  $\delta(a, d)\delta(b, c) \leq \delta(a, b)\delta(c, d) + \delta(a, c)\delta(b, d)$ ) presents information about two pivots, the object, and the query. It was shown that for efficient filtering, we need to find pivots close to the object and the query[36].

Connor et al. [42] recently presented good results in a similarity search with the *four-point property*. Similarly as Ptolemy’s inequality utilizes four points for lower bounding, the four-point property is based on embedding the original space into  $\mathbb{R}^3$  space as a tetrahedron (generally simplex).

All of the mentioned properties were constructed and observed in the data. The different approach presented by Bartoš [43, 44] infers the artificial inequalities using genetic programming. For inferring the artificial inequalities, the parallel approach was constructed and described in a doctoral thesis [45].

## 3.3 Indexability

The resulting effectiveness of similarity search is affected by several factors: data distribution, queries, distances, or the indexing method used. Indexability indicators such as *IDim*, distance distribution, or *BoF* try to capture properties that directly or indirectly affect the efficiency. Some, such as *IDim*, capture only the basic properties (distances). On the other hand, *BoF* targets specific indexing methods. Both expect metric similarity model. In our case, the situation is further complicated because the similarity space need not be metric, so we proposed new indicators *triangularity* and *ptolemaicity*. [A.5]

**Triangularity** First of all, we decided to analyze the triplets (of objects) themselves and define a *triangularity* over one triplet (Definition 3.3.1). The *triangularity* indicates, on a scale from 0 to 1, how triangular the triplet is. Values between 0 and  $\frac{1}{2}$  mean that the triangular inequality is violated, and a value of  $\frac{1}{2}$  means that the triplet forms a line segment. Values between  $\frac{1}{2}$  and 1 then represent valid triangles, and a value of 1 means that the triplet is an equilateral triangle.

**Definition 3.3.1.** Let  $(o_1, o_2, o_3) \in \mathbb{S}^3$  be a *triplet* of objects which forms distances  $(\delta(o_1, o_2), \delta(o_1, o_3), \delta(o_2, o_3))$ . Then for sorted distances  $(a, b, c)$ , where  $a \leq b \leq c$ , the  $Tri(a, b, c) = \frac{a+b}{2c}$  is the *triangularity* of triplet.

For the purposes of indexability analysis, we construct a distribution of triplets. We can construct all possible triplets for a random selection  $\mathbb{S}^* \subset \mathbb{S}$ . Or we can construct it directly as a random selection of triplets  $(o_1, o_2, o_3) \in \mathbb{S}^3$ . We then compute the triangularities for each triplet and construct a distribution.

In the distribution, the intervals and values mentioned above are indicative for us. The distribution of triplets in the intervals  $(0, \frac{1}{2})$  and  $(\frac{1}{2}, 1)$  determines indirectly the accuracy of the returned results using the metric space index. The traditional TriGen also



works with the ratio itself. The triplets distribution tells us not only whether triangle inequality is satisfied but also to what extent the criterion is satisfied.

Returning to the simplest version of three-point filtering (Section 2.2.1), in the case of an equilateral triangle  $Tri(a, b, c) = 1$ , the lower bound will always be  $\delta_{LB}(x, y) = 0$ . This means that such a triangle is not suitable for indexing. Conversely, the lower bound does not always have to be maximal for the line  $Tri(a, b, c) = 1/2$ . If we consider the assumption that  $d_{px} \approx 0$  (e.g., which we are able to satisfy with enough pivots), then values close to  $1/2$  will maximize  $\delta_{LB}$ . Thus, a distribution with a high number of triplets near  $1/2$  is likely to be well indexable.

**Ptolemaicity** Similarly, we proposed the *ptolemaicity* of quadruplets (of objects) as Definition 3.3.2. The processing procedure is similar, but the actual interpretability is difficult. We cannot simply say that  $1/2$  represents a segment, but it is still true that the value  $1/2$  is expected for efficient indexing. Similarly, ptolemaicity of 1 indicates poor indexability.

**Definition 3.3.2.** Let  $(o_1, o_2, o_3, o_4) \in \mathbb{S}^4$  be a *quadruplet* of objects which forms products of distance pairs  $(\delta(o_1, o_2) \cdot \delta(o_3, o_4), \delta(o_1, o_3) \cdot \delta(o_2, o_4), \delta(o_1, o_4) \cdot \delta(o_2, o_3))$ . Then for sorted values  $(a, b, c)$ , where  $a \leq b \leq c$ ,  $Pto(a, b, c) = \frac{a+b}{2c}$  is the *ptolemaicity* of quadruplet.

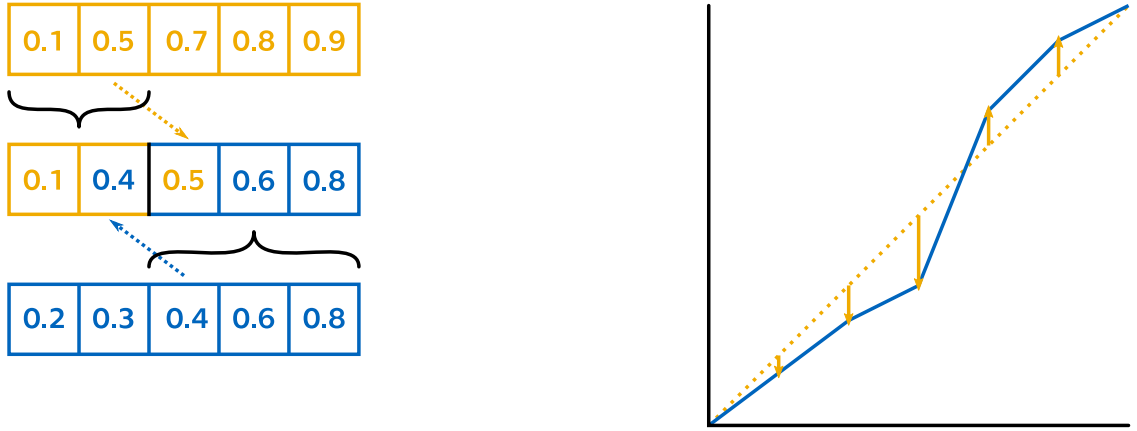
## 3.4 TriGenGA

The original TriGen algorithm only works with one-parameter modifiers. This limits it mainly in the number of possible combinations, and the expected effectiveness and efficiency are a function of the parameter  $w$ . As a solution to this problem, in our experiments we considered the modifier  $\mathcal{M}_{a,b}^{one-point}(d)$  defined as Equation 3.3 with two parameters  $a, b \in (0, 1)$ . Applying the binary search sequentially to each parameter may not find a valid or optimal solution. A brute force search for optimal parameters will be time-consuming since the number of possible combinations increases with the number of parameters and their values (which need not be discrete). Therefore, we decided to use genetic algorithms to estimate the parameters.

$$\mathcal{M}_{a,b}^{one-point}(d) = \begin{cases} b \cdot \frac{d}{a} & \text{for } d \leq a, \\ b + (1 - b) \cdot \frac{d-a}{1-a} & \text{otherwise.} \end{cases} \quad (3.3)$$

In [A.10], we introduced the concept of finding these parameters using a genetic algorithms approach. In our case, the genetic algorithm has a relatively simple structure, see Algorithm 3. The first step is to randomly generate a population of individuals of a given population size ( $pop\_size = 100$ ). The generation of the new population starts by selecting pairs with a preference for the best individuals based on a fitness function ( $fit(\vec{v})$ , Equation 3.6). These pairs will then generate new children (crossover, Figure 3.2a), which will undergo a possible mutation (with a probability of 5 %) and become candidates. A new population is created by merging the generated candidates with the old population

### 3. NON-METRIC SIMILARITY MODELS



(a) One-point crossover example, new child in middle.

(b) Example of TriGenGA modifier

Figure 3.2: Illustration of TriGenGA modifier and crossover of two parents.

and discarding half of the worst individuals. In the case of rapid population convergence, a catastrophic scenario replaces 50 % of the population with random individuals.

An individual is a modifier represented by a vector of numbers  $\vec{v} = (\vec{v}_0, \vec{v}_1, \dots, \vec{v}_n)$  of dimension  $n + 1$ , for which  $\vec{v}_0 = 0$ ,  $\vec{v}_n = 1$  and  $\forall i, j \in \mathbb{Z}_{n+1}$  holds  $i < j \implies \vec{v}_i < \vec{v}_j$ . Since some operations (crossover, mutation) may violate the increasing vector rule, the vector is reordered after each such operation to preserve the rule (Figure 3.2a). Such an operation does not affect the behavior of the genetic algorithm, only the meaning of the individual operations. The general modifier  $\mathcal{M}_{\vec{v}}^{\text{piecewise}}$  itself (Figure 3.2b) is then defined as a piecewise linear function by Equation 3.4.

$$\mathcal{M}_{\vec{v}}^{\text{piecewise}}\left(d \mid \frac{i}{n} \leq d \leq \frac{i+1}{n}\right) = \vec{v}_i + (\vec{v}_{i+1} - \vec{v}_i)(nd - i) \quad (3.4)$$

In [A.9], we have improved our genetic algorithm by improving the fitness function, which is the heart of the whole optimization algorithm. From preliminary results, we found that functions that change concavity (convexity) frequently show good results on a small sample but do not behave as expected when generalized. Therefore, we proposed the so-called ConFactor( $\vec{v}$ ) (Equation 3.5), which compares the number of concave segments  $c_{\vec{v}}^+$  and the number of convex segments  $c_{\vec{v}}^-$ . The resulting fitness function is then defined by Equation 3.6, with the first part taking care of primarily achieving the triangular inequality in the desired quantity and the second part trying to do a secondary optimization. Secondary optimization favors mainly functions that have the minimum number of inflection points.

$$\text{ConFactor}(\vec{v}) = \frac{|c_{\vec{v}}^+ - c_{\vec{v}}^-|}{c_{\vec{v}}^+ + c_{\vec{v}}^-} \quad (3.5)$$

**Algorithm 3:** TriGenGA: Genetic Algorithm for TriGen

---

```

Result: modifier  $\mathcal{M}_{\vec{v}}^{\text{piecewise}}$  with the best fitness score
Population  $\leftarrow$  GenerateRandomModifiers(pop_size)
while UnsuccessfulCatastrophe()  $\leq$  max. # of catastrophes do
    Parents  $\leftarrow$  TournamentSelectork(Population, fit( $\vec{v}$ ))
    for  $\forall$  pair of Parents do
        /* combine modifiers */
        Child  $\leftarrow$  OnePointCrossover(pair)
        if mutation probability succeeded then
            /* randomly modify modifier */
            Child  $\leftarrow$  Mutate(Child)
        Population  $\leftarrow$  Population  $\cup$  {Child}
    keep only pop_size best individuals in Population
    modify k to keep diversity in Population
    if best fitness score does not change for last L iterations then
        /* rapid convergence */
        Population  $\leftarrow$  CatastropheScenario(Population)
return  $\arg \max_{\vec{x} \in \text{Population}} \text{fit}(\vec{x})$ 

```

---

$$\text{fit}(\vec{v}) = \begin{cases} 1 - \epsilon(\mathbb{S}^*, \mathcal{M}_{\vec{v}}^{\text{piecewise}} \circ \delta) & \text{for } \epsilon(\mathbb{S}^*, \mathcal{M}_{\vec{v}}^{\text{piecewise}} \circ \delta) > \epsilon_{\mathbf{T}}, \\ 1 + \epsilon(\mathbb{S}^*, \mathcal{M}_{\vec{v}}^{\text{piecewise}} \circ \delta) \cdot \text{ConFactor}(\vec{v}) & \text{otherwise.} \end{cases} \quad (3.6)$$

## 3.5 Discussion

In [A.5], we illustrated how the triangularity and ptolemaicity of high-dimensional data change after applying the TriGen algorithm. In experiments with metric and Ptolemaic spaces, we showed that the TriGen algorithm could effectively modify the distance to speed up similarity search with minimal error. Even for pseudo-random (nearly unindexable) data, the TriGen algorithm found a transformation that improves indexing without loss of accuracy.

In [A.9], we showed that TriGenGA outperforms (Figure 3.3) the original TriGen algorithm. The choice of appropriate modifiers is what mainly limits the TriGen algorithm. On the other hand, TriGenGA is based on randomness so that each algorithm run may return a different result. Nevertheless, in experiments, TriGenGA outperformed the original algorithm in precision-efficiency ratio and much better estimated the actual search error compared to the threshold.

The widespread use of neural networks makes us wonder whether neural networks could be similarly used to find a modifier suitable for indexing. Since the indexing process is

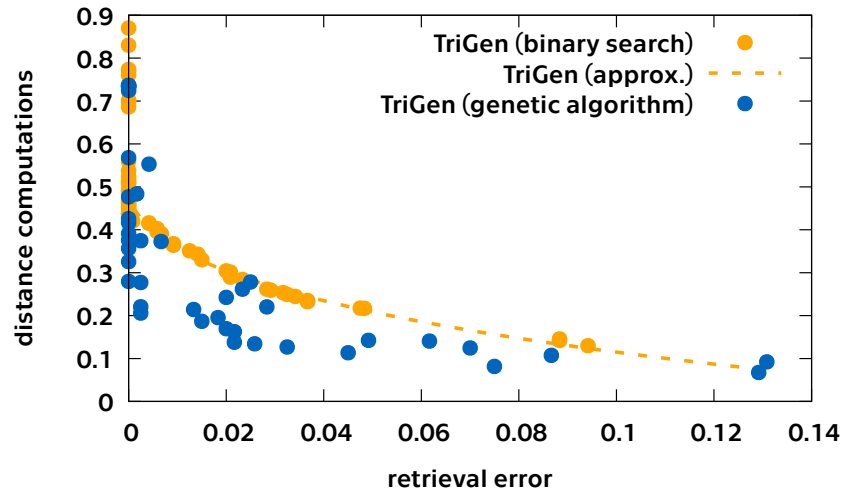


Figure 3.3: Comparison of precision and efficiency of original TriGen (orange, binary search) and TriGenGA (blue, genetic algorithm).

performed once, the time required to train such a network is negligible. In case of changes to the database, only minor fine-tuning of the network should be sufficient. A second open problem is how TriGenGA will behave for similarity searches based on similarity spaces other than metric (e.g., Ptolemaic, four-point property).

---

# Data-Transitive Similarity Metamodel

*This section describes the principles of the data-transitive metamodel in similarity search. Section 4.1 introduces the basic problems and real-world motivation for using transitive similarities. Section 4.2 describes related work and the current state-of-the-art. Section 4.3 then defines the data-transitive metamodel. Section 4.4 summarizes the findings of the experiments, open problems, and future work. The details are described in the author's papers Work 1, 2, 3, 4, 6, and 7).*

## 4.1 Introduction

Big data brings new challenges. The problem itself is not only the large volume and lack of annotation but often also sparse or no linking of such data at all. A large amount of unrelated information creates only a very weak link between database objects.

In [A.7, A.6], we had to deal with the domain of financial data and use anonymized information about clients, their payments, repayments, or ATM withdrawals to find links between individual clients and to create a social network (not only) for information propagation analysis. To create a link between clients based only on simple rules between two clients was unthinkable given the small sample of data (small institution). Therefore, these factual links were complemented by links based on the similarity of the entities (clients, shops, ATMs, ...) and aggregated by inferred links (see Figure 4.1).

These linkages allowed to connect users at several different levels. The first level was factual links of the type "clients withdraw money from the same ATM". These are then represented mainly by the common location of the two clients. The second level was "clients withdrawing from similar ATMs" (inferred links, see Figure 4.1). In this case, we lose information about the location of the ATM but gain information about the behavior of the clients. At the same time, their interconnections from the previous levels can be used to aggregate other information or infer links. In the end, we created a similarity-based social network, but the actual processing had to be done by analyzing the resulting graph.

At the same time, by exploiting external knowledge and aggregating it, we can gain

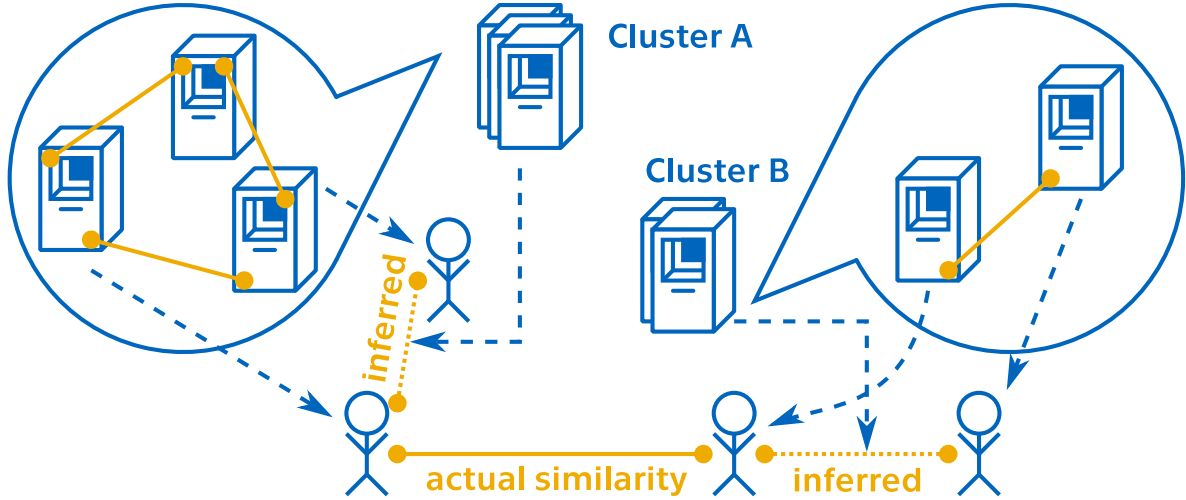


Figure 4.1: Example of inferred similarity of users in graph using clustering ATMs.

further linkages and get a more comprehensive view of the problem domain. External information does not always have to be available in sufficient quality, so alternative approaches must be applied. Moreover, it is important to emphasize the distinction between similarity and relevance, which in theoretical terms merge.

In [A.4], we addressed the possibility of using similarity search in the domain of open datasets. Open datasets are collections of data published by organizations for further use. The problem is a large number of datasets with minimal annotation, which makes it difficult to find similar or related datasets. The usefulness of open datasets lies in the ability to link and extract knowledge from these compilations. However, the organizations are unaware of the existence of similar datasets, so this linking is left to data analysts, journalists, and enthusiasts.

Finding similar datasets has proven to be challenging, and common open dataset catalogs only allow full-text searches. Our initial solution involved augmenting the basic similarity search with contextual information in the form of knowledge graphs or word embeddings. This contextual information proved to be essential, but linking datasets based on similarity was still only localized to nearly identical datasets.

## 4.2 Related Work

The metric space model [10] is considered the gold standard in the similarity search domain. However, metric space model only provides a tool for indexing the database (e.g., quick search). Although it is a relatively reasonable trade-off in many problems, the metric space model remains too limited for similarity modeling. The limitations increase with models aimed at higher search efficiency, such as Ptolemaic [46] or supermetric [47] mod-

els. In the last decade, approaches ranging from using a combination of metrics [48] to completely non-metric [11] approaches have been introduced to increase the effectiveness of similarity search. The main goal is to increase the relevance of the retrieved objects from the database while maintaining reasonable efficiency.

The query-by-example mechanism itself, in the form of single range and kNN queries, provides only basic search functionality. Therefore, linking similarities to relational databases tends to be complicated and requires using languages such as MySQL [49] or SimilarQL [50]. The concept of similarity joins [51] brings the similarity aspect of joining entire database tables. In the case of graph databases, inferred social network [A.6] in the form of a weighted graph and subsequent analysis of social networks or various graph metrics [52] is offered. In order to make queries more expressive, similarity queries can be extended to multi-modal retrieval using techniques such as late fusion [53] or content-based recommender systems [54].

### 4.3 Our Approach

In [A.3], we introduced the basic concept of data-transitive similarity, which addresses the problems outlined. The first problem is the distinction between similarity and relevance. User needs are very subjective, and the similarity approach most often leads to retrieving identical or very similar results to the query. In many cases, the goal is to find objects that are only very close or related in some way to the query.

For example, for a query on the movie "Spiderman: Homecoming", one can expect all the episodes of this movie series or movies of a similar genre. This respects the traditional idea of similarity. On the other hand, we can expect the movie "Dolittle" because they share some actors together. One level deeper, we can expect a "Doctor Strange" movie set in the same fictional world. This last example shows the similarity as a relevance that may not be obvious at first glance because apart from the shared music and comic book origins, the movies have almost nothing in common. However, both movies share a common intermediary, "Spider-Man: No Way Home", which connects the two aforementioned movies (see Figure 4.2).

Such an example is very similar to the well-known example of non-metric similarity, where "man" and "horse" are not similar at all, but both have a quite strong similarity to the mythical creature "centaur". Indeed, such similarity is analogous to exploration process and will almost certainly lead to a non-metric model. Our proposed meta-model can be described in general terms as a chain of sequentially similar objects, for which, however, it may not be true that the first and last objects of this chain are similar in the original model. The chain of similar objects makes them relevant in a particular context.

Formally, we can define such a data-transitive distance by Equation 4.1, where  $n$  stands for the length of the chain and the symbols  $\odot$  and  $\uplus$  stand for the outer and inner aggregation. Table 4.1 then represents a sample of the aggregations we used in our experiments. The main drawback of the approach used in [A.6] was the need to analyze the resulting network using graph-based methods. In the case of a data-transitive meta-model, we can

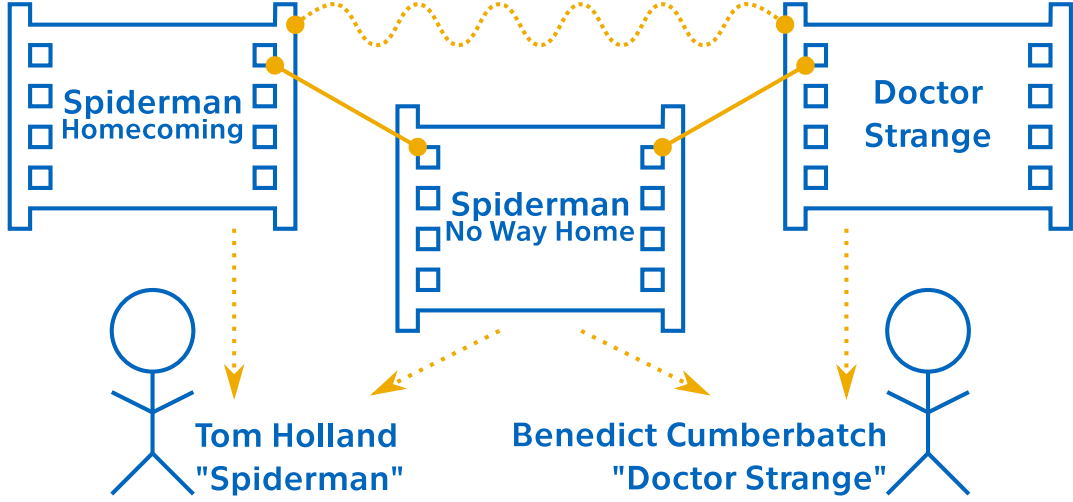


Figure 4.2: Example of data-transitive similarity of movies

---


$$\begin{aligned}
 \text{sum}(d_0, d_1, \dots, d_n) &= \sum_{j=0}^n d_j \\
 \min(d_0, d_1, \dots, d_n) &= \min \{d_0, d_1, \dots, d_n\} \\
 \max(d_0, d_1, \dots, d_n) &= \max \{d_0, d_1, \dots, d_n\} \\
 \text{prod}(d_0, d_1, \dots, d_n) &= \prod_{j=0}^n d_j \\
 \text{iproduct}(d_0, d_1, \dots, d_n) &= 1 - \prod_{j=0}^n (1 - d_j)
 \end{aligned}$$


---

 Table 4.1: Examples of inner  $\uplus$  and outer  $\odot$  aggregations.

use standard similarity methods and indexes, e.g., by applying TriGenGA, we can obtain a metric model and further index using standard metric approaches.

$$\hat{\delta}_{\uplus}^{\odot, n}(\mathbf{x}, \mathbf{y}) = \odot_{(i_1, \dots, i_n) \in \mathbb{S}^n} \uplus(\delta(\mathbf{x}, i_1), \delta(i_1, i_2), \dots, \delta(i_n, \mathbf{y})) \quad (4.1)$$

In our experiments, we also used optimizations that prevented identity-based transitivity. That is, the case where  $\delta(x, y) = 0$ . Such transitivity degenerates to standard similarity, so all chains containing the pair  $x, y$  for which  $\delta(x, y) \leq d_{5\%}$  did not participate (are excluded) in the construction of  $\hat{\delta}$ .<sup>1</sup>

---

<sup>1</sup> $d_{5\%}$  is first vigintile, respectively  $\frac{|\{(x, y) \in \mathbb{S}^2 \mid \delta(x, y) \leq d_{5\%}\}|}{|\mathbb{S}|} = 5\%$



## 4.4 Discussion

In [A.1], we applied a data-transitive meta-model specifically to the domain of open dataset exploration and tested the effectiveness of such a meta-model. User testing has shown that such models can be invaluable, specifically in cases where the individual objects are only weakly similar (for datasets, the textual labels are sparse). In particular, when we are interested in the broader context of relevant (not always similar) objects, this method outperformed standard similarity approaches.

The user testing had two objectives. The first goal was to develop a set of relevant answers for the queries. For each relevant answer, we added the percentage of users considering the object relevant. In a conservative setting, we only considered responses that were identified as relevant by all users. Traditional methods such as TF-IDF [55] or word embeddings performed better (see Table 4.2). Transitive methods started to dominate (see Table 4.2) over traditional methods when we lowered this criterion to a consensus of at least 80% of users. The second goal was to find out the subjective feeling of users about such an application, and in the SUS testing [56] performed, our application scored 70 points, which is a good result.

User consensus $\longrightarrow$		60 %	80 %	100 %
$\downarrow$ TF-IDF	Cosine	PR AUC		
<i>without transitive</i>		0.49	0.61	<b>0.89</b>
$\odot = \min$	$\uplus = \max$	<b>0.61</b>	<b>0.63</b>	0.40
$\odot = \min$	$\uplus = \text{iproduct}$	<b>0.61</b>	<b>0.63</b>	0.45
$\downarrow$ Word2Vec	Cosine	PR AUC		
<i>without transitive</i>		0.61	0.71	<b>0.89</b>
$\odot = \min$	$\uplus = \max$	0.56	<b>0.72</b>	0.73
$\odot = \min$	$\uplus = \text{iproduct}$	<b>0.76</b>	0.67	0.66

Table 4.2: Comparison of standard (without data-transitive) similarity models and data-transitive models using precision-recall area under curve per different levels of user consensus.

It also has the advantage of self-explainability since the intermediate chain itself explains the reason why a particular pair is relevant. However, for long chains or exotic aggregations it may be more challenging to explain the relevance. This risk is minimal with the knowledge of the distance measure and aggregations used. Self-explainability is a major advantage over neural networks, which suffer from this very shortcoming as a black-box approach.

The data-transitive meta-model should be seen as a completely new field that creates many new challenges. Although the indexing of such data-transitive distance can be solved by the aforementioned methods, including our proposed TriGenGA, the computation of data-transitive distance itself is a challenging problem. Finding the optimal value of the outer aggregation has an asymptotic complexity  $O(|S|^n)$ . The inner aggregation will have an asymptotic complexity of at least  $\Omega(n)$ . We can apply simple pruning for some

combinations, such as  $\odot = \min, \uplus = \max$ , but the general principle will be much more complicated or require approximate methods.

The second area is generally experiments with longer chains  $n > 1$  or combinations of several different lengths  $n$ . The first problem is the previously mentioned worse interpretation of the results. The second problem is how to combine such distances. Would it be more user-friendly to display the distances separately or aggregated (e.g.  $\min\{\delta(x, y), \hat{\delta}_{\uplus}^{\odot, n}(x, y)\}$ )?

---

## Conclusion

Research in similarity search has primarily focused on efficient retrieval using various indexing techniques. On the other hand, there has been an increasing demand for semantic-based querying. The tremendous growth of data has emphasized not only speed but also relevance. Traditional approaches no longer meet the needs of a user looking for alternatives, not duplicates. This thesis briefly presents new ways to deal with these problems, published in papers in which David Bernhauer was the co-author.

In the first part of this paper, we introduced the basic problems and the necessary theoretical background. We introduced new indexability indicators, focusing on indexes using metric or Ptolemaic axioms. Compared to standard indicators, these also account for the violation of the triangular (or Ptolemaic) inequality and their properties were tested together with an application of the TriGen algorithm. As part of this, we extended the TriGen algorithm to include multi-parameter functions and the estimation of their parameters by genetic algorithms. These can better adapt to a given distance and outperform the original TriGen algorithm.

We dealt with similarity modeling within the domain of financial data and open datasets. We have created The SimilAnT tool [A.8] (Work 8) for human-based analysis of similarity models and used it to visualize and find appropriate similarity models in these domains. We proposed a data-transitive similarity model for the open dataset domain that solved the problems of weak direct similarity due to sparse data. Compared to the financial domain, we were able to stay in the similarity area and thus use existing similarity approaches. User testing has shown that using data-transitive similarity effectively increases the relevance of results without requiring lengthy exploration.

### 5.1 Contributions of the Thesis

#### 1. INDEXABILITY ANALYSIS

In Chapter 3, we introduced the novel indexability indicators *triangularity* and *ptolemaicity*, which focus on capturing the structure of the data using properties of the

particular similarity model. They can also successfully capture the degree of violation of these properties, which existing approaches only capture with a single number (e.g., the percentage of triplets violating the triangle inequality). These indicators can be used for indexability analysis or, for example, as indicators of suitable pivots for certain types of indexation.

## 2. TRIGENGA

In Chapter 3, we developed an extension *TriGenGA* to the existing TriGen algorithm to generate modifiers using genetic algorithms. These allow finding the optimal modifier, eliminating the need to optimize multiple one-parameter modifiers that may not cover all possible transformations. We have shown that such a modifier can outperform the original algorithm in both efficiency and effectiveness. Moreover, TriGenGA does not have to be limited to metric spaces and can be used for other spaces (e.g., Ptolemaic) with appropriate modifications. However, this requires further experimentation.

## 3. DATA-TRANSITIVE SIMILARITY META-MODEL

In Chapter 4, we defined the *data-transitive similarity meta-model*. By applying it to a real open dataset domain, we showed that the data-transitive similarity meta-model can be used in situations where objects in the database are only sparsely similar to each other, and by using transitive similarity we are able to extend the original similarity. At the same time, we should also point out the main advantage of this approach, that the result of such transitive similarity is self-explainable, so that the context in which two objects are similar is always known. Moreover, this approach is compatible with similarity methods.

## 5.2 Future Work

Neural networks have proven to be an indispensable tool in the case of data preprocessing, whether for text descriptors (e.g., Word2Vec [57, 58]) or image descriptors (e.g., CNN [59]). Neural networks have already demonstrated the ability to index metric spaces [60]. Thus, the question is whether we can also use neural networks for efficient indexing of arbitrary similarity, which may greatly expand the range of usable distances and, with it, the freedom to design similarity. Compared to classical indexing methods, neural network-based methods may have the advantage of easy fine-tuning when enlarging the database. A similar principle can be used instead of the genetic algorithm in TriGen to derive a suitable distance modifier.

Data-transitive similarity is at the very beginning these days. It has proven to be an excellent concept in the case of database exploration, mainly because it provides an important explanation in the form of a sequence of similar objects. However, no experiments with more than one auxiliary object have been performed yet. In this area, it will be necessary to analyze the behavior for longer chains and also to design a way to present the results

across different lengths. A second useful feature is the densification of the similarity space when similarity arises between originally dissimilar objects. Such behavior depends on the chosen aggregation functions, and here we encounter another area that has only been explored in basic experiments.



---

## Bibliography

- [1] Jacobson, K.; Murali, V.; et al. Music Personalization at Spotify. In *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16*, New York, NY, USA: Association for Computing Machinery, 2016, ISBN 9781450340359, p. 373, doi: 10.1145/2959100.2959120. URL <https://doi.org/10.1145/2959100.2959120>
- [2] Deldjoo, Y.; Schedl, M.; et al. Recommender Systems Leveraging Multimedia Content. *ACM Comput. Surv.*, volume 53, no. 5, sep 2020, ISSN 0360-0300, doi: 10.1145/3407190. URL <https://doi.org/10.1145/3407190>
- [3] Lü, L.; Medo, M.; et al. Recommender systems. *Physics Reports*, volume 519, no. 1, 2012: pp. 1–49, ISSN 0370-1573, doi: <https://doi.org/10.1016/j.physrep.2012.02.006>, recommender Systems. URL <https://www.sciencedirect.com/science/article/pii/S0370157312000828>
- [4] Chen, D.; Yuan, Z.; et al. Similarity Learning With Spatial Constraints for Person Re-Identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [5] Martín, A.; Huertas-Tato, J.; et al. FacTeR-Check: Semi-automated fact-checking through semantic similarity and natural language inference. *Knowledge-Based Systems*, volume 251, 2022: p. 109265, ISSN 0950-7051, doi: <https://doi.org/10.1016/j.knosys.2022.109265>. URL <https://www.sciencedirect.com/science/article/pii/S0950705122006323>
- [6] Wang, X.-J.; Zhang, L.; et al. AnnoSearch: Image Auto-Annotation by Search. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 2006, pp. 1483–1490, doi: 10.1109/CVPR.2006.58.
- [7] Grošup, T. Methods of Multi-Modal Data Exploration. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR '19*, New York, NY,

- USA: Association for Computing Machinery, 2019, ISBN 9781450367653, p. 34–37, doi: 10.1145/3323873.3325858. URL <https://doi.org/10.1145/3323873.3325858>
- [8] Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, volume 24, no. 5, 1988: pp. 513–523.
- [9] Niwattanakul, S.; Singthongchai, J.; et al. Using of Jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, 2013, pp. 380–384.
- [10] Zezula, P.; Amato, G.; et al. *Similarity Search: The Metric Space Approach*. New York, NY 10013, USA: Springer, 2005th edition, 2005, ISBN 0-387-29146-6.
- [11] Skopal, T.; Bustos, B. On Nonmetric Similarity Search Problems in Complex Domains. *ACM Comput. Surv.*, volume 43, no. 4, Oct. 2011: pp. 34:1–34:50, ISSN 0360-0300, doi: 10.1145/1978802.1978813. URL <http://doi.acm.org/10.1145/1978802.1978813>
- [12] Kanas, S.; Lecko, A.; et al. Metric Spaces. *Formalized Mathematics*, volume 1, no. 3, 1990: pp. 607–610.
- [13] Sampo, J.; Luukka, P. Similarity classifier with generalized mean; ideal vector approach. In *International Conference on Fuzzy Systems and Knowledge Discovery*, Springer, 2006, pp. 1140–1147.
- [14] Ma, L.; Zhang, Y. Using Word2Vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 2895–2897.
- [15] Heckert, N. A.; Filliben, J. J. *NIST Handbook 148: DATAPLOT Reference Manual, Volume 2: LET Subcommands and Library Functions*. National Institute of Standards and Technology Handbook Series, 2003, accessed: 2022-05-20.
- [16] Deza, M.; Deza, E. *Encyclopedia of distances*. Dordrecht : New York: Springer Verlag, 2009, ISBN 9783642002335 9783642002342, oCLC: ocn310400730.
- [17] Cer, D.; Yang, Y.; et al. Universal Sentence Encoder. *arXiv:1803.11175 [cs]*, Apr. 2018, arXiv: 1803.11175. URL <http://arxiv.org/abs/1803.11175>
- [18] Huang, X. Q.; Hardison, R. C.; et al. A space-efficient algorithm for local similarities. *Comput. Appl. Biosci.*, volume 6, no. 4, Oct. 1990: pp. 373–381.
- [19] Kuzmanic, A.; Zanchi, V. Hand shape classification using DTW and LCSS as similarity measures for vision-based gesture recognition system. In *EUROCON 2007-The International Conference on "Computer as a Tool"*, IEEE, 2007, pp. 264–269.
- [20] Pinto, J. P.; Pimenta, A.; et al. Deep learning and multivariate time series for cheat detection in video games. *Machine Learning*, volume 110, no. 11, 2021: pp. 3037–3057.



- 
- [21] *Dynamic Time Warping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ISBN 978-3-540-74048-3, pp. 69–84, doi: 10.1007/978-3-540-74048-3\_4. URL [https://doi.org/10.1007/978-3-540-74048-3\\_4](https://doi.org/10.1007/978-3-540-74048-3_4)
- [22] Huttenlocher, D.; Klanderman, G.; et al. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 15, no. 9, 1993: pp. 850–863, doi: 10.1109/34.232073.
- [23] Stanoi, I.; Agrawal, D.; et al. Reverse Nearest Neighbor Queries for Dynamic Databases. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, pp. 44–53.
- [24] Achtert, E.; Kriegel, H.-P.; et al. Reverse K-nearest Neighbor Search in Dynamic and General Metric Databases. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-422-5, pp. 886–897, doi: 10.1145/1516360.1516462. URL <http://doi.acm.org/10.1145/1516360.1516462>
- [25] Ester, M.; Kriegel, H.-P.; et al. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, AAAI Press, 1996, pp. 226–231. URL <http://dl.acm.org/citation.cfm?id=3001460.3001507>
- [26] Carterette, B. *Precision and Recall*. Boston, MA: Springer US, 2009, ISBN 978-0-387-39940-9, pp. 2126–2127, doi: 10.1007/978-0-387-39940-9\_5050. URL [https://doi.org/10.1007/978-0-387-39940-9\\_5050](https://doi.org/10.1007/978-0-387-39940-9_5050)
- [27] Craswell, N. *Precision at n*. Boston, MA: Springer US, 2009, ISBN 978-0-387-39940-9, pp. 2127–2128, doi: 10.1007/978-0-387-39940-9\_484. URL [https://doi.org/10.1007/978-0-387-39940-9\\_484](https://doi.org/10.1007/978-0-387-39940-9_484)
- [28] Zhang, E.; Zhang, Y. *Eleven Point Precision-recall Curve*. Boston, MA: Springer US, 2009, ISBN 978-0-387-39940-9, pp. 981–982, doi: 10.1007/978-0-387-39940-9\_481. URL [https://doi.org/10.1007/978-0-387-39940-9\\_481](https://doi.org/10.1007/978-0-387-39940-9_481)
- [29] Zhang, E.; Zhang, Y. *F-Measure*. Boston, MA: Springer US, 2009, ISBN 978-0-387-39940-9, pp. 1147–1147, doi: 10.1007/978-0-387-39940-9\_483. URL [https://doi.org/10.1007/978-0-387-39940-9\\_483](https://doi.org/10.1007/978-0-387-39940-9_483)
- [30] Skopal, T. On Fast Non-metric Similarity Search by Metric Access Methods. In *Advances in Database Technology - EDBT 2006*, edited by Y. Ioannidis; et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ISBN 978-3-540-32961-9, pp. 718–736.
- [31] Vidal, E. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Pattern Recognition Letters*, volume 15, no. 1, 1994: pp. 1–7, ISSN 0167-8655, doi: <https://doi.org/10.1016/>

- 0167-8655(94)90094-9. URL <https://www.sciencedirect.com/science/article/pii/0167865594900949>
- [32] Micó, M. L.; Oncina, J.; et al. A new version of the nearest-neighbour approximating and eliminating search algorithm (AESAs) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, volume 15, no. 1, 1994: pp. 9–17, ISSN 0167-8655, doi: [https://doi.org/10.1016/0167-8655\(94\)90095-7](https://doi.org/10.1016/0167-8655(94)90095-7). URL <https://www.sciencedirect.com/science/article/pii/0167865594900957>
- [33] Ciaccia, P.; Patella, M.; et al. M-tree: An efficient access method for similarity search in metric spaces. In *Vldb*, volume 97, 1997, pp. 426–435.
- [34] Skopal, T. Pivoting M-tree: A Metric Access Method for Efficient Similarity Search. In *DATESO*, volume 4, Citeseer, 2004, pp. 27–37.
- [35] Esuli, A. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*, volume 48, no. 5, 2012: pp. 889–902, ISSN 0306-4573, doi: <https://doi.org/10.1016/j.ipm.2010.11.011>, large-Scale and Distributed Systems for Information Retrieval. URL <https://www.sciencedirect.com/science/article/pii/S0306457310001019>
- [36] Lokoč, J.; Hetland, M. L.; et al. Ptolemaic Indexing of the Signature Quadratic Form Distance. In *Proceedings of the Fourth International Conference on Similarity Search and Applications, SISAP '11*, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0795-6, pp. 9–16, doi: 10.1145/1995412.1995417. URL <http://doi.acm.org/10.1145/1995412.1995417>
- [37] Grossman, D. A.; Frieder, O. *Information retrieval: Algorithms and heuristics*, volume 15. Springer Science & Business Media, 2004.
- [38] Chávez, E.; Navarro, G.; et al. Searching in Metric Spaces. *ACM Comput. Surv.*, volume 33, no. 3, sep 2001: p. 273–321, ISSN 0360-0300, doi: 10.1145/502807.502808. URL <https://doi.org/10.1145/502807.502808>
- [39] Skopal, T. Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces. *ACM Trans. Database Syst.*, volume 32, no. 4, Nov. 2007, ISSN 0362-5915, doi: 10.1145/1292609.1292619. URL <http://doi.acm.org/10.1145/1292609.1292619>
- [40] Roth, V.; Laub, J.; et al. Optimal Cluster Preserving Embedding of Nonmetric Proximity Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 25, 01 2004: pp. 1540–1551, doi: 10.1109/TPAMI.2003.1251147.
- [41] Fagin, R.; Stockmeyer, L. Relaxing the Triangle Inequality in Pattern Matching. *Int. J. Comput. Vision*, volume 30, no. 3, Dec. 1998: pp. 219–231, ISSN 0920-5691, doi: 10.1023/A:1008023416823. URL <https://doi.org/10.1023/A:1008023416823>

- 
- [42] Connor, R.; Vadicamo, L.; et al. Supermetric Search with the Four-Point Property. In *Similarity Search and Applications*, edited by L. Amsaleg; M. E. Houle; E. Schubert, Cham: Springer International Publishing, 2016, ISBN 978-3-319-46759-7, pp. 51–64.
- [43] Skopal, T.; Bartoš, T. Algorithmic Exploration of Axiom Spaces for Efficient Similarity Search at Large Scale. In *Similarity Search and Applications*, edited by G. Navarro; V. Pestov, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-32153-5, pp. 40–53.
- [44] Bartoš, T.; Skopal, T.; et al. Efficient Indexing of Similarity Models with Inequality Symbolic Regression. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13*, New York, NY, USA: ACM, 2013, ISBN 978-1-4503-1963-8, pp. 901–908, doi: 10.1145/2463372.2463487. URL <http://doi.acm.org/10.1145/2463372.2463487>
- [45] Bartoš, T. *Indexing Arbitrary Similarity Models*. Dissertation thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2014.
- [46] Hetland, M. L.; Skopal, T.; et al. Ptolemaic access methods: Challenging the reign of the metric space model. *Inf. Syst.*, volume 38, no. 7, 2013: pp. 989–1006, doi: 10.1016/j.is.2012.05.011. URL <https://doi.org/10.1016/j.is.2012.05.011>
- [47] Connor, R.; Vadicamo, L.; et al. Supermetric search. *Information Systems*, volume 80, 2019: pp. 108–123, ISSN 0306-4379, doi: <https://doi.org/10.1016/j.is.2018.01.002>. URL <https://www.sciencedirect.com/science/article/pii/S0306437917301588>
- [48] Bustos, B.; Kreft, S.; et al. Adapting metric indexes for searching in multi-metric spaces. *Multimedia Tools and Applications*, volume 58, 06 2012: pp. 1–30, doi: 10.1007/s11042-011-0731-3.
- [49] Lu, W.; Hou, J.; et al. MSQl: efficient similarity search in metric spaces using SQL. *The VLDB Journal*, volume 26, no. 6, Dec 2017: pp. 829–854, ISSN 0949-877X, doi: 10.1007/s00778-017-0481-6. URL <https://doi.org/10.1007/s00778-017-0481-6>
- [50] Traina, C.; Moriyama, A.; et al. The SimilarQL framework: similarity queries in plain SQL. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, ACM, 2019, pp. 468–471, doi: 10.1145/3297280.3299736.
- [51] Silva, Y. N.; Pearson, S. S.; et al. Similarity Joins: Their implementation and interactions with other database operators. *Information Systems*, volume 52, 2015: pp. 149–162, ISSN 0306-4379, doi: 10.1016/j.is.2015.01.008, special Issue on Selected Papers from SISAP 2013.
- [52] Symeonidis, P.; Tiakas, E.; et al. Transitive Node Similarity for Link Prediction in Social Networks with Positive and Negative Links. In *Proceedings of the Fourth ACM*

- Conference on Recommender Systems, RecSys '10*, New York, NY, USA: Association for Computing Machinery, 2010, ISBN 9781605589060, p. 183–190, doi: 10.1145/1864708.1864744. URL <https://doi.org/10.1145/1864708.1864744>
- [53] Novak, D.; Zezula, P.; et al. Inherent Fusion: Towards Scalable Multi-Modal Similarity Search. *J. Database Manage.*, volume 27, no. 4, Oct. 2016: p. 1–23, ISSN 1063-8016, doi: 10.4018/JDM.2016100101. URL <https://doi.org/10.4018/JDM.2016100101>
- [54] Aggarwal, C. C. *Recommender Systems - The Textbook*. Springer, 2016, ISBN 978-3-319-29659-3, 1-498 pp.
- [55] Sammut, C.; Webb, G. I. (editors). *TF-IDF*. Boston, MA: Springer US, 2010, ISBN 978-0-387-30164-8, pp. 986–987, doi: 10.1007/978-0-387-30164-8\_832. URL [https://doi.org/10.1007/978-0-387-30164-8\\_832](https://doi.org/10.1007/978-0-387-30164-8_832)
- [56] Lewis, J. R. The System Usability Scale: Past, Present, and Future. *International Journal of Human-Computer Interaction*, volume 34, no. 7, 2018: pp. 577–590, doi: 10.1080/10447318.2018.1455307, <https://doi.org/10.1080/10447318.2018.1455307>. URL <https://doi.org/10.1080/10447318.2018.1455307>
- [57] Mikolov, T.; Chen, K.; et al. Efficient Estimation of Word Representations in Vector Space. 2013, doi: 10.48550/ARXIV.1301.3781. URL <https://arxiv.org/abs/1301.3781>
- [58] Mikolov, T.; Sutskever, I.; et al. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26, edited by C. Burges; L. Bottou; M. Welling; Z. Ghahramani; K. Weinberger, Curran Associates, Inc., 2013. URL <https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf>
- [59] Dara, S.; Tamma, P. Feature Extraction By Using Deep Learning: A Survey. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 1795–1801, doi: 10.1109/ICECA.2018.8474912.
- [60] Antol, M.; Ol'ha, J.; et al. Learned Metric Index — Proposition of learned indexing for unstructured data. *Information Systems*, volume 100, 2021: p. 101774, ISSN 0306-4379, doi: <https://doi.org/10.1016/j.is.2021.101774>. URL <https://www.sciencedirect.com/science/article/pii/S0306437921000326>

---

## Reviewed Publications of the Author Relevant to the Thesis

- [A.1] Bernhauer, D.; Nečaský, M.; Škoda, P.; Klímek, J.; Skopal, T. Open Dataset Discovery using Context-enhanced Similarity Search. *Knowledge and Information Systems*, 2022, *accepted with minor revision*. (Work 1)
- [A.2] Nečaský, M.; Škoda, P.; Bernhauer, D.; Klímek, J.; Skopal, T. Modular framework for similarity-based dataset discovery using external knowledge. *Data Technologies and Applications*, volume *ahead-of-print*, no. *ahead-of-print*, 2022, doi: 10.1108/DTA-09-2021-0261. (Work 2)
- [A.3] Skopal, T.; Bernhauer, D.; Škoda, P.; Klímek, J.; Nečaský, M. Similarity vs. Relevance: From Simple Searches to Complex Discovery. In *Similarity Search and Applications: 14th International Conference, SISAP 2021*, Dortmund, Germany: Springer-Verlag, 2021, doi: 10.1007/978-3-030-89657-7\_9. (Work 3)
- [A.4] Škoda, P.; Bernhauer, D.; Nečaský, M.; Klímek, J.; Skopal, T. Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services, iiWAS '20*, Chiang Mai, Thailand: Association for Computing Machinery, 2020, doi: 10.1145/3428757.3429973. (Work 4)

The paper has been cited in:

- Smits, J. J. *Improving integration Platforms as a Service through the addition of enterprise data catalog features*. 2022. Master's Thesis. University of Twente.
- [A.5] Bernhauer, D.; Skopal, T. Analysing Indexability of Intrinsically High-dimensional Data using TriGen. In *Similarity Search and Applications: 13th International Conference, SISAP 2020*, Copenhagen, Denmark: Springer, 2020, doi: 10.1007/978-3-030-60936-8\_20. (Work 5)

The paper has been cited in:

- Mic, V.; Raček, T.; Křenek, A.; Zezula, P. Similarity Search for an Extreme Application: Experience and Implementation. In *International Conference on Similarity Search and Applications*. Springer, Cham, 2021. p 265–279, doi: 10.1007/978-3-030-89657-7\_20.
- [A.6] Holubová, I.; Svoboda, M.; Bernhauer, D.; Skopal, T.; Paščenko, P. Inferred Social Networks: A Case Study. In *Proceedings of the 2019 International Conference on Data Mining Workshops, ICDMW'19*, Beijing, China: IEEE Computer Society, 2019, doi: 10.1109/ICDMW.2019.00019. (Work 6)

The paper has been cited in:

- Měkota, O. *Link Prediction in Inferred Social Networks*. 2021. Master's Thesis. Charles University
- [A.7] Holubová, I.; Svoboda, M.; Skopal, T.; Bernhauer, D.; Peška, L. Advanced Behavioral Analyses Using Inferred Social Networks: A Vision. In *Database and Expert Systems Applications*, Springer, 2019, doi: 10.1007/978-3-030-27684-3\_26. (Work 7)
- [A.8] Bernhauer, D.; Skopal, T.; Holubová, I.; Peška, L.; Svoboda, M. SIMILANT: An Analytic Tool for Similarity Modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19*, Beijing, China: ACM, 2019 doi: 10.1145/3357384.3357852. (Work 8)
- [A.9] Bernhauer, D.; Skopal, T. Non-metric Similarity Search Using Genetic TriGen. In *Similarity Search and Applications: 13th International Conference, SISAP 2019*, Newark, NJ, USA: Springer, 2019, doi: 10.1007/978-3-030-32047-8\_8. (Work 9)

The paper has been cited in:

- Connor, R.; Dearle, A.; Mic, V.; Zezula, P. On the application of convex transforms to metric search. *Pattern Recognition Letters*, volume 138, 2020, doi: 10.1016/j.patrec.2020.08.008.
  - Beecks, C.; Berns, F.; Schmidt, K. W. Ptolemaic indexing for managing and querying internet of things (IoT) data. In *2019 IEEE International Conference on Big Data*, (Big Data). IEEE, 2019, doi: 10.1109/Big-Data47090.2019.9005725.
- [A.10] Bernhauer, D.; Skopal, T. Approximate search in dissimilarity spaces using GA. In *GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary*

*Computation Conference Companion*, Prague, Czech Republic: Association for Computing Machinery, 2019, doi: 10.1145/3319619.3321907. (Work 10)

The paper has been cited in:

- Rezazadeh Hamedani, A.; Moattar, M. H.; Forghani, Y. Dissimilarity space reinforced with manifold learning and latent space modeling for improved pattern classification. *Journal of Big Data*, volume 8, no. 135, 2021, doi: 10.1186/s40537-021-00527-6.





---

## Remaining Publications of the Author Relevant to the Thesis

- [A.11] Bernhauer, D. *Approximate Search in Dissimilarity Spaces*. Ph.D. Minimum Thesis, Faculty of Information Technology, Prague, Czech Republic, 2019.
- [A.12] Bernhauer, D. *Approximate Search in Dissimilarity Spaces using GA*. Technical Report TR-FIT-19-03 Faculty of Information Technology, Prague, Czech Republic, 2019.



---

## Remaining Publications of the Author

- [A.13] Bernhauer, D.; Skopal, T. Recommender System as the Support for Binaural Audio. In *Augmented Reality and Virtual Reality. Progress in IS*. Springer, 2019, doi: 10.1007/978-3-030-06246-0\_17.



Part II  
Collection of Works



---

# Open Dataset Discovery using Context-enhanced Similarity Search

Bernhauer, D. (20 %); Nečaský, M (20 %).; Škoda, P. (20 %); Klímek, J. (20 %); Skopal, T. (20 %) Open Dataset Discovery using Context-enhanced Similarity Search. *Knowledge and Information Systems*, 2022, (*accepted with minor revision*). [A.1]

Preprint submitted to KAIS

## Open Dataset Discovery using Context-enhanced Similarity Search

David Bernhauer<sup>1,2</sup>, Martin Nečaský<sup>1</sup>, Petr Škoda<sup>1</sup>, Jakub Klímek<sup>1\*</sup> and Tomáš Skopal<sup>1</sup>

<sup>1</sup>Charles University, Faculty of Mathematics and Physics,  
Department of Software Engineering, Malostranské náměstí 25,  
Prague, 118 00, Czechia.

<sup>2</sup>Czech Technical University in Prague, Faculty of Information  
Technology, Department of Software Engineering, Thákurova 9,  
Prague, 160 00, Czechia.

\*Corresponding author(s). E-mail(s):

[jakub.klimek@matfyz.cuni.cz](mailto:jakub.klimek@matfyz.cuni.cz);

Contributing authors: [bernhdav@fit.cvut.cz](mailto:bernhdav@fit.cvut.cz);

[martin.necasky@matfyz.cuni.cz](mailto:martin.necasky@matfyz.cuni.cz); [petr.skoda@matfyz.cuni.cz](mailto:petr.skoda@matfyz.cuni.cz);

[tomas.skopal@matfyz.cuni.cz](mailto:tomas.skopal@matfyz.cuni.cz);

### Abstract

Today, open data catalogs enable users to search for datasets with full-text queries in metadata records combined with simple faceted filtering. Using this combination a user is able to discover a significant number of the datasets relevant to a user's search intent. However, there still remain relevant datasets that are hard to find because of the enormous sparsity of their metadata (e.g., several keywords). As an alternative, in this paper, we propose an approach to dataset discovery based on similarity search over metadata descriptions enhanced by various semantic contexts. In general, the semantic contexts enrich the dataset metadata in a way that enables the identification of additional relevant datasets to a query that could not be retrieved using just the keyword or full-text search. In experimental evaluation we show that context-enhanced similarity retrieval methods increase the findability of relevant datasets, improving thus the retrieval recall that is critical in dataset discovery scenarios. As a part of the evaluation, we created a catalog-like user interface for dataset discovery and recorded



2 *Open Dataset Discovery using Context-enhanced Similarity Search*

streams of user actions that served us to create the ground truth. For the sake of reproducibility, we published the entire evaluation testbed.

**Keywords:** Dataset, Discovery, Search, Similarity, Evaluation, Context

## 1 Introduction

Dataset discovery is an important task in the area of data integration [1]. Leading tech companies, including Google, have developed their own dataset search techniques [2]. Novel solutions for dataset search in various domains started to appear recently, such as the Datamed [3] – an open source index for discovery of biomedical datasets. The existing approaches emphasize the importance of quality metadata for dataset findability [4], pointing out that available metadata attached to a dataset does not always describe the dataset’s semantic content as well as the usability of that metadata in a particular retrieval task (or use case). Other works [5–7] confirm that dataset discovery is highly contextual depending on the particular user’s task. This makes the dataset discovery harder as it may not be sufficient to search for datasets only by classical keyword or full-text search. The mentioned works show the contextual dependency should be inherently reflected by the dataset search engines. More sophisticated approaches able to search for datasets being *similar* to an example dataset could be helpful in these scenarios. However, as shown by [1, 4], many existing dataset discovery solutions are based on basic keyword search. Examples of such solutions are the European Data Portal<sup>1</sup>, the Czech National Open Data Catalog<sup>2</sup> [8], or the Open Data Portal of the US Government<sup>3</sup>. [4] provides a detailed survey of existing discovery solutions and shows that existing solutions are mainly based on the basic keyword search in structured metadata records comprising the titles, descriptions and free keywords characterizing datasets.

In this paper, we evaluate several approaches to dataset discovery that are based on the similarity search. The similarity-based methods allow us to systematically incorporate the datasets’ context information into the search process, which is especially beneficial if the dataset’s metadata alone is very sparse. Then, it could provide better results than the traditional full-text search applied on sparse metadata. The hypothesis we want to research in this paper is formulated as follows:

**Research hypothesis:**

*Context-enhanced similarity-based dataset discovery extends the search results with relevant datasets that are not findable using the full-text search methods.*

---

<sup>1</sup><https://data.europa.eu>

<sup>2</sup><https://data.gov.cz>

<sup>3</sup><https://www.data.gov/>

## 1.1 Paper Contributions

The contributions presented in this paper are summarized as follows.

- We argue for similarity search in the role of an open dataset discovery technique and describe selected similarity-based methods for dataset search.
- We created a prototype of a user environment in the style of standard dataset catalog the users are familiar with. Beyond the standard catalog functionality, we also implemented an option for dataset similarity search.
- We hired and tasked professional independent users (not related to the authors of the paper) to use the catalog environment and solve several pre-defined dataset discovery scenarios (use cases). We recorded the streams of user actions in the environment for all the users and their tasks. Using the streams, we created a new ground truth for evaluation of the effectiveness of dataset discovery methods.
- We showed how to use the new ground truth in order to compare hundreds of different dataset search methods.
- By the analysis of user action streams we have confirmed the research hypothesis.
- We have accomplished and evaluated also the usability and user friendliness of the similarity-based dataset discovery using the standard SUS method.

## 2 Related work

The problem of dataset search and discovery could be addressed by various approaches. Domain-specific and small-scale dataset collections could be maintained by database administrators and thus described by a relational schema (list of simple-type attributes). Then the dataset discovery task would be as simple as querying records in an SQL database. However, most of the datasets are published on the web as open data where no admins exist, the scale is enormous and the semantic diversity of the datasets is unlimited. The dataset discovery techniques are rarely based on querying the actual content of datasets, as the datasets' content structure and semantics greatly varies (full-text, spreadsheets, database dumps, multimedia) while the individual dataset elements are too fragmented (numbers, strings, dates, pixels). The diversity of open datasets is so large that the common consistent description for open datasets proved to be a simple metadata record comprising of a few text-based attributes (title, description, keywords) describing the dataset.

### 2.1 Similarity Search

As the datasets' metadata is text-based, a natural method for dataset discovery is the full-text search, widely used in various web search engines. However, from a wider perspective the full-text search methods are just a subset of larger concept for retrieval of unstructured data – the similarity search.

In the area of similarity search, data objects are retrieved using the query-by-example paradigm. The query is defined as an example data object (a

4 *Open Dataset Discovery using Context-enhanced Similarity Search*

particular dataset in our case) and similarity scores are computed between the example and all the objects in the database<sup>4</sup>. The similarity scores then determine the ranking of objects with respect to the similarity of the query example. Finally, the most similar objects in the ranking are returned as the query result – mostly in the form of  $k$  nearest neighbors (kNN).

The research in the similarity search area had intensified some three decades ago by setting the metric space model as the golden standard [9]. The metric distances in place of (dis)similarity functions were introduced purely for database indexing reasons (i.e., for fast search). Though a good trade-off for many problems, the metric space model remains quite restrictive for modeling similarity. The restrictions are even more strict in follow-up models aiming at improving search efficiency, such as the ptolemaic [10] or supermetric [11] models.

In the era of Big data, however, the increasing diversity and complexity of data calls for more general schemes of similarity modeling. The metric space properties could be too restrictive in many fields [12], therefore approaches to non-metric similarity search get constantly more attention in the last decades. For the sake of robustness, the similarity models used in practical applications often favor partial matching over the global similarity (high similarity in objects' details implies higher mutual relevance of objects than a smaller "overall" similarity does). However, such a bias towards partial matching necessarily brings violations of the metric properties [12] (e.g., triangle inequality is broken). Consider the textbook example with a man, a horse and a centaur. The centaur is partially very similar to the man (upper body part) as well as to the horse (lower body part), but the man and the horse are completely dissimilar (a triangle for them cannot be created). In such situations, models dropping some of the metric space properties – the non-metric similarity models – could address the problem. The need for non-metric similarity models could be beneficial also in the domain of dataset discovery as they allow to more easily integrate the dataset contexts in various ways.

## 2.2 Similarity-based Dataset Discovery

In this paper, we are interested in similarity-based dataset discovery. The rationale for using similarity search is the usage of an example dataset as query instead of a free-form text. Where full-text query (typed into a search box by user) consists of a few arbitrary keywords, the example dataset carries richer semantics.

Also, example datasets really exist in some dataset catalog, so using them as queries cannot be a "blind shooting" (whereas arbitrary full-text search can). A limitation to this approach is the assumption that such an example dataset is provided by the user or by the system. This is not always possible especially at the beginning of the dataset discovery process, while in such cases

---

<sup>4</sup>Note that the data-engineering terminology could be a bit misleading. A *database* is generally a collection of database objects (or documents). In our case, the database object (document) is a particular *dataset* (e.g., a CSV file). Hence, in dataset discovery we search a database of datasets.

the full-text query is the only viable starting option. Nevertheless, once some datasets suitable for examples are determined during the discovery process, by them we obtain a consistent way of how to ask for other relevant datasets – the similar ones.

The similarity can be used during query processing where the query result obtained by full-text search can be enriched with similar datasets. Also, it could be used for result handling and presentation where similarity of retrieved datasets is applied to group datasets in the presentation or to enable exploration of retrieved datasets in case of large results.

A specific subset of techniques related to similarity-based dataset discovery is discussed in the recent survey [4]. It discusses techniques to extend a table by discovering tables through table similarity based on tabular schema similarity (e.g., [13, 14]) and semantic similarity using embedding approaches (e.g., [15]). These can be considered also as dataset discovery techniques because a table is just a special case of a dataset.

Several novel techniques for similarity-based dataset discovery using knowledge graphs have been proposed in the last few years. The Aurum system [16] was proposed to build, maintain and query an enterprise knowledge graph (EKG) which represents datasets and their structural elements, e.g., table columns, as nodes and relationships between them as edges. A relationship between two structural elements may represent content similarity, schema similarity, e.g., similarity of names of the columns, or key/foreign key pairs defined in the dataset schemas. The paper introduces an efficient model which utilizes EKG. Moreover, the introduced technique requires only a linear passage through datasets to build EKG. Dataset discovery is then performed on top of EKG. When a user selects a dataset, the tool offers other relevant datasets through the relationships in EKG.

A content and schema similarity technique was proposed in [17]. For schema similarity, the approach considers similarity of attribute (column) names. For content similarity, the approach considers various similarity models, e.g., based on value embedding. Content-based union and complement metrics between RDF datasets were proposed in [18], designed to compute similarity scores directly on the RDF triples forming the datasets.

Several papers propose dataset similarity techniques based on metadata similarity. A full-text search method proposed in [19] enables to measure similarity between datasets on the basis of papers citing the datasets and a citation network between datasets. In another study [20] the authors evaluate four different metadata-based models for searching spatially related datasets, i.e., datasets which are related because of the same or similar spatial area covered (where the user query is parsed and tagged by a geocode). A number of similarity models and indexes were proposed for semi-structured data [21–24], where mix of both textual and structured information is available. These multi-modal approaches could be used in cases the schema of the expected datasets is designed to grasp meaningful semi-structured content.

6 *Open Dataset Discovery using Context-enhanced Similarity Search*

There are also works which focus on methods useful for datasets published as Linked Data [25]. For example, content-based metrics for measuring connectivity between datasets using links and shared entities between the datasets were introduced in [26]. The authors of [27] also use shared entities to define similarity between datasets and extend this approach also to cluster similar entities. The metrics can be then used also for measuring similarity between datasets. A dataset recommendation approach is presented in [28] which identifies linking candidates based on the presence of schema overlap between datasets. A dataset recommendation method based on cosine similarity of sets of concepts present in datasets was introduced in [29]. Similarly, the authors of [30] recommend datasets based on the similarity of resource labels present in the datasets. Then they rank the recommended datasets based on cosine similarity and tf-idf weights and the coverage of labels present in the original dataset. Another work [31] presents a probabilistic Bayesian classifier for dataset recommendation. Such recommendation techniques can also be used for dataset discovery services as they recommend similar or related datasets.

### 2.2.1 Baseline: The European Data Portal

The European Data Portal<sup>5</sup> (EDP) offers a discovery feature based on dataset similarity besides the basic keyword search. For a dataset found by the keyword query search, similar datasets are also offered to the user. According to source code published at GitLab<sup>6</sup>, the portal uses a similarity-based model called TLSH [32, 33]. In TLSH, the keyword-based title and description metadata of the dataset are concatenated into a string. Then a locally sensitive hash is constructed from the concatenated string, aiming to produce similar hashes for similar datasets, and these hashes (bit strings) are compared using a proprietary similarity function. It was implemented as a technique for searching for duplicate or almost equal datasets, ignoring typing errors. Being the official similarity-based method in EDP search interface, we use TLSH as the baseline method in our evaluation.

## 3 Similarity Models for Dataset Discovery

The fundamental element for similarity search is the representation of objects for measuring similarity. In our case, we work with the metadata attached to datasets, which are text-based properties title, description, and keywords. The metadata representation is very sparse as the information included is very small (a few keywords) or even missing (e.g., title). An example of a real dataset metadata is:

```
title = "Floods in the 19th century",
description = "Flooded areas in a 19th century flood in the Pilsen region",
keywords = "floods; environment; GIS"
```

<sup>5</sup><https://data.europa.eu/en>

<sup>6</sup><https://gitlab.com/european-data-portal/metrics/edp-metrics-dataset-similarities/-/blob/master/src/main/java/io/piveau/metrics/similarities/SimilarityVerticle.java>

### 3.1 Direct similarity models

The first group of similarity models we used in our evaluation are models that measure direct similarity between a pair of datasets' descriptors. Such similarities are not dependent on other objects (datasets) in the database.

#### 3.1.1 Context-free models

The simplest similarity models are context-free, that is, there is no external knowledge used to evaluate the similarity score of two objects. Only the information stored in descriptors is used for the similarity computation. An example of such a simple similarity measure over text strings that we originally tried for dataset discovery was the Levenstein (or edit) distance [34]. However, it proved to be very impractical early on, especially for longer text strings, e.g., description, due to its high time complexity. Therefore, we did not use it as regular similarity for dataset discovery. Nevertheless, we used edit distance for presentation purposes; specifically for grouping similar datasets within the final user interface (section 4.1.1).

The first measure in our evaluation was the TLSh – Locality Sensitivity Hashing – presented in [32] and used the European Data Portal (EDP). Thus, it served us as a suitable baseline similarity model. Except for the extra language translation step, our implementation<sup>7</sup> of the algorithm is identical to the implementation in EDP. The concatenation of title and description is hashed by a hashing function (TLshH) and the subsequent similarity  $TlshD(\#x, \#y)$  is computed over this hash. This hashing function provides similar hashes for similar strings.

Another possibility is to represent dataset's textual metadata (denoted as a text document *doc*) as a vector in the bag of words model (BoW). The document *doc* is parsed for words, that are lemmatized, non-significant words (such as prepositions, conjunctions, stop-words (frequent words), etc.) are filtered out. We use the UDPipe [35] for lemmatization of *doc* with the English-ParTUT model for English metadata [36] and Czech-PDT model for Czech metadata [36]. These lemmatized words are called terms. A dictionary is created that contains all terms from the entire database (all datasets' metadata). Thus, the simplest option is to represent each *term* as a separate dimension, and the value of the vector for a given *doc* tells us the number of occurrences ( $\text{freq}_{term, doc}$ ) of that term in *doc*. The vector of a dataset associated with *doc* is defined as  $\mathbf{doc} = (\text{tf}(t_1, doc), \text{tf}(t_2, doc), \dots, \text{tf}(t_m, doc))$ , where *m* is the number of terms in the dictionary. Then it is sufficient to compare the vectors by, for example, the Cosine distance  $CosD(\mathbf{a}, \mathbf{b})$  defined by Equation 4. A more popular model is then TF-IDF [37], which extends the whole concept by word frequency analysis and thus emphasizes unique words as shown by Equation 2. The vector is built using the tf-idf score defined as Equation 3 instead of tf score (Equation 1).

---

<sup>7</sup><https://github.com/mff-uk/simpipes-components/blob/main/processors/compute-similarity/basic/linda/tlsh.py>

$$\text{tf}(term, doc) = \frac{\text{freq}_{term, doc}}{\sum_{t \in doc} \text{freq}_{t, doc}} \quad (1)$$

$$\text{idf}(term) = -\log \frac{|\{doc \in \mathcal{D} : term \in doc\}|}{|\mathcal{D}|} \quad (2)$$

$$\text{tf-idf}(term, doc) = \text{tf}(term, doc) \cdot \text{idf}(term) \quad (3)$$

$$\text{CosD}(\mathbf{a}, \mathbf{b}) = 1 - \frac{\sum_{i=1}^n \mathbf{a}_i \mathbf{b}_i}{\sqrt{\sum_{i=1}^n \mathbf{a}_i^2} \sqrt{\sum_{i=1}^n \mathbf{b}_i^2}} \quad (4)$$

We also evaluated set-based similarities. In particular, keywords, or titles (split into words), can be thought of as a set of words. Such a set can then be compared using Jaccard distance based on Jaccard similarity.  $JacD$  is defined in Equation 5. Alternatively, we can utilize the set of vectors from the next section. Since we would have a hard time finding duplicates in such vectors, as the vectors will usually be slightly different, we can use Hausdorff distance defined in Equation 6.  $HausD$  tries to find the minimum matching between two sets using a defined ground distance  $d$ . This ground distance  $d$ , in our case, is the Cosine distance  $CosD$  as mentioned above.

$$JacD(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \quad (5)$$

$$HausD_d(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(a, b) \right\} \quad (6)$$

### 3.1.2 Context-enhanced models

The problem of context-free models in our particular case of dataset discovery is the sparsity of metadata. Simply said, there is often nothing in common between the datasets if a similarity model relies just on keyword matching. This problem could be alleviated with a context to enhance the keywords by some background semantic concepts. Such an enhancement enables the find similarities at some higher-level of abstraction.

As an example, knowledge graphs/ontologies like WordNet [38], ConceptNet [39] or Wikidata [40] could be used to map metadata as well as user queries to knowledge graph nodes. The terms in metadata/query are enriched by keywords represented by the knowledge graph nodes and the query result is obtained using the standard full-text search or a kind of shortest path evaluation [40]. To boost the enriching effect even more, NLP-based preprocessing methods could be used to semantically disambiguate the terms later mapped to knowledge graph nodes [41–43].

Alternatively, NLP-processed text could be used alone as a light-weight variant to the ontology mapping. Word embeddings stand for such a light-weight alternative that "imports" the contextual knowledge directly into the data representations (vectors). Instead of knowledge graphs, embeddings represent machine-learning approach to create a background shared semantics to

be used for dataset representations. In particular, word embeddings transform each word or phrase into a vector representing a given concept. It is based on the principle that similar words occur in similar contexts. Vectors of similar words are then close to each other. We average all the vectors (one vector per term from query/metadata) into a single vector for a query/dataset. In our experiments, we tried the original Word2Vec embedding [44] with three different parameterizations. For English metadata, we use the `Word2Vec(enwiki)` model trained on English Wikipedia articles. For Czech metadata, we use the `Word2Vec(cswiki)` model trained on Czech Wikipedia articles and the `Word2Vec(1aw)` model trained on legislative texts.

Node2Vec [45] extends the idea of Word2Vec from the text domain to the graph domain.  $k$  random walks of length  $len$  are generated for each node. These random walks then represent standard sentences, and the embedding concept is identical to Word2Vec. For Node2Vec embedding, we trained several models with different parameters  $k$ ,  $len$  over the Wikidata graph. Concepts from Wikidata then serve not only as a node in the graph, but also as a term.

Similarly, we experimented with the *BERT* model [46]. BERT model provides not only embedding but also natural language processing, i.e., lemmatization, word splitting, etc. At the same time, the learning principle is slightly different from Word2Vec-based methods. Word2Vec methods are based on the prediction of surrounding words based on a specific word or vice versa, the prediction of a specific word based on surrounding words. The BERT model is trained by masking certain words and then predicting them. In addition to masking, it also uses Next Sentence Prediction, i.e., whether two sentences are consecutive in the text, for training.

Although we ultimately did not include these similarity models in the user evaluation that formed the ground truth, the results of automatic evaluation are available in the extended evaluation in the [section 4.2.3](#).

### 3.2 Data-transitive Similarity Meta-model

In this section, we utilize the meta-model of data-transitive similarity [47]. Unlike the direct similarity models, using data-transitive similarity all objects in the database influence the similarity score between a particular pair of objects. In a sense, the data-transitive similarity introduces a new level of context-enhanced similarity modeling; the context is the database itself.

The basic assumption of data-transitive similarity is a chain of objects from the database that are similar to each other, but the beginning and end of the chain could be quite dissimilar (yet relevant). The strength of the chain itself can be formalized as an aggregation of several consecutive ground distances. The [Equation 7](#) defines general form of data-transitive distance function  $\hat{d}$ , where  $\mathcal{D}$  is a set of objects (the database in practical applications),  $d$  is a ground distance (the direct similarity),  $n$  is the length of the chain. Operator  $\odot$  is an outer aggregation over all permutations of length  $n$  over elements of database  $\mathcal{D}$  (e.g., min, max, avg). Operator  $\oplus$  is an inner aggregation over the individual direct distances within a particular chain. [Table 1](#) shows examples



10 *Open Dataset Discovery using Context-enhanced Similarity Search*

of various inner aggregation functions. They are also the aggregation functions we worked with in our preliminary experiments. A more complex alternative may be a combination of several kinds of aggregations or distances.

$$\hat{d}_{\Psi}^{\odot, n}(\mathbf{x}, \mathbf{y}) = \bigodot_{(i_1, \dots, i_n) \in \mathcal{D}^n} \biguplus (d(\mathbf{x}, i_1), d(i_1, i_2), \dots, d(i_n, \mathbf{y})) \quad (7)$$

$\text{sum}(\delta_0, \delta_1, \dots, \delta_n) =$	$\sum_{j=0}^n \delta_j$
$\text{min}(\delta_0, \delta_1, \dots, \delta_n) =$	$\min \{\delta_0, \delta_1, \dots, \delta_n\}$
$\text{max}(\delta_0, \delta_1, \dots, \delta_n) =$	$\max \{\delta_0, \delta_1, \dots, \delta_n\}$
$\text{prod}(\delta_0, \delta_1, \dots, \delta_n) =$	$\prod_{j=0}^n \delta_j$
$\text{iproduct}(\delta_0, \delta_1, \dots, \delta_n) =$	$1 - \prod_{j=0}^n (1 - \delta_j)$

**Table 1:** Examples of inner aggregation  $\biguplus$ .

To summarize, the data-transitive similarity  $\hat{d}$  is a non-metric meta-model operating on top of a ground similarity model  $d$  and a particular database  $\mathcal{D}$ . The computation of a single data-transitive distance involves a series of similarity queries over the database.

In our experiments, we evaluated the data-transitive similarities defined in Equation 8, 9, 10, 11, and 12. For our user evaluation, we describe the selected similarity models in section 4.1.1.

$$\hat{d}_{\max}^{\min, 1}(\mathbf{x}, \mathbf{y}) = \min_{i_1 \in \mathcal{D}} \max(d(\mathbf{x}, i_1), d(i_1, \mathbf{y})) \quad (8)$$

$$\hat{d}_{\text{sum}}^{\min, 1}(\mathbf{x}, \mathbf{y}) = \min_{i_1 \in \mathcal{D}} (d(\mathbf{x}, i_1) + d(i_1, \mathbf{y})) \quad (9)$$

$$\hat{d}_{\text{prod}}^{\min, 1}(\mathbf{x}, \mathbf{y}) = \min_{i_1 \in \mathcal{D}} (d(\mathbf{x}, i_1) \cdot d(i_1, \mathbf{y})) \quad (10)$$

$$\hat{d}_{\text{iproduct}}^{\min, 1}(\mathbf{x}, \mathbf{y}) = \min_{i_1 \in \mathcal{D}} (1 - (1 - d(\mathbf{x}, i_1)) \cdot (1 - d(i_1, \mathbf{y}))) \quad (11)$$

$$\hat{d}_{\text{diff}}^{\max, 1}(\mathbf{x}, \mathbf{y}) = \max_{i_1 \in \mathcal{D}} |d(\mathbf{x}, i_1) - d(i_1, \mathbf{y})| \quad (12)$$

### 3.2.1 Notation

Since we worked with more than 200 different configurations in our experiments, we provide a simplified notation to understand how models work. The input descriptors are T title, D description, or K keywords, and their language (-en or -cs), or their concatenation (e.g. TD-en means concatenated title and description translated into English). The | sign means data processing, L lemmatization, SW removal of stop-words,. Or their transformation, split splitting string into a set of words, BoW splitting into a bag of words or avg

average of values. Data-transitive similarity in our model is a function with three parameters: ground distance, outer aggregation, inner aggregation. For all examples, we assume chains of length equal to 1. Thus, for example,  $\text{DT}(\mathbf{d}, \text{outer}, \text{inner})$  corresponds to  $\hat{d}_{\text{outer}}^{\text{inner},1}$ .

An example is  $\text{HausD}(\text{TDK-en} \mid \text{L} \mid \text{Word2Vec}(\text{cswiki}))$ , where we take the concatenation of all the metadata in the English translation, lemmatize it, apply Word2Vec embedding, and take the subsequent vectors as input for Hausdorff distance.

Another example is  $\text{DT}(\text{CosD}(\text{T-cs} \mid \text{L} \mid \text{SW} \mid \text{BoW}(\text{tf})), \text{min}, \text{max})$  which represents data-transitive meta-model  $\hat{d}_{\text{max}}^{\text{min},1}$  based on Cosine distance. Descriptors are titles in the Czech translation. These descriptors are lemmatized, and bag of words (based just on term frequencies) is created.

## 4 Evaluation

In this section we provide a thorough quantitative evaluation of the similarity-based methods for dataset discovery.

### 4.1 Methodology

Our evaluation of the individual methods for dataset search was based on several realistic use cases. In a particular use case we assigned a user with a specific goal of finding datasets needed to build a certain service, e.g., a mobile application showing flood areas, or a map of criminality, etc. The use case was specified with a short description of the service to be built and, in the cases of similarity-based methods, with a starting dataset which is relevant for the service. In the case of the fulltext-search method, we did not provide a starting dataset.

#### 4.1.1 Experimental Setup

We defined 6 use cases and hired 10 independent users to evaluate 6 different dataset similarity models in the context of the defined use cases. The 10 users were professionals who worked with open data at the national or local (municipal) level and who regularly used the Czech National Open Data Catalog<sup>8</sup>. The use cases were defined in Czech, but we offer an English translation here:

- UC1 Intellectual property protection** – As a programmer, I want to create an online service to find data about intellectual property protection.  
*Starting dataset:* Intellectual property rights checks, published by the Ministry of the Interior
- UC2 Medical facilities** – As a journalist, I want to create a map of medical facilities in the Czech Republic according to their addresses.  
*Starting dataset:* Doctors' offices in Brno, published by the Brno city
- UC3 Construction in Brno city** – I want to build a family house in Brno and as an analyst I want to know something about the area. That is, information

---

<sup>8</sup><https://data.gov.cz>

12 *Open Dataset Discovery using Context-enhanced Similarity Search*

about the processes of building a house in Brno, data about living in Brno, etc.

*Starting dataset:* People and housing in Brno, published by the Brno city

**UC4 Floods** – As a programmer, I want to make an application for displaying flood maps.

*Starting dataset:* Floods in the 19th century in Plzeň (Pilsen), published by the Plzeň city

**UC5 Climate status** – As a journalist, I want to create an application showing the state of the climate in the regions of the Czech Republic.

*Starting dataset:* Climate bonity, published by the Prague city

**UC6 Fined people** – As a programmer, I want to create a service for showing traders and companies that have been fined for their business practices. I have already found various data sets on fines, but they do not contain data on the fined traders and companies themselves. So I still need to supplement the data on fines with data on fined traders and companies.

*Starting dataset:* Imposed fines, published by the Czech Trade Inspection Authority

For all similarity models, the input is metadata records (title, description, keywords) for datasets. Since the metadata can be in different languages, they are automatically translated into English using LINDAT [48]. On the other hand, a poor translation can cause noise in the data and subsequent processing problems. Therefore, the information is presented to users in their native language, which allows us to objectively build ground truth and evaluate the quality of the models.

For clarity, we describe the abbreviations used in the figures, as defined in section 3.2.1. The baseline method we evaluated is the TLSH method ( $\text{tlsh} = \text{TlshD}(\text{TD-en} \mid \text{TlshH})$ ). As the basis of our similarity method, we implemented the TF-IDF method ( $\text{tf\_idf} = \text{CosD}(\text{TDK-en} \mid \text{L} \mid \text{SW} \mid \text{BoW}(\text{tf\_idf}))$ ) over the union of title, description, and keywords if the dataset has that metadata category present. We also evaluated data-transitive similarity models. We evaluated a total of 3 kinds of data-transitive similarities based on TF-IDF method ( $\text{tf\_idf}$ ):

- $\text{min\_max} = \text{DT}(\text{tf\_idf}, \text{min}, \text{max})$
- $\text{min\_sum} = \text{DT}(\text{tf\_idf}, \text{min}, \text{sum})$
- $\text{min\_iprod} = \text{DT}(\text{tf\_idf}, \text{min}, \text{iprod})$

The users did the evaluation using an extended version of the current user interface [49] of the Czech National Open Data Catalog, called LinkedPipes DCAT-AP Viewer. There are two extensions:

1. The detailed view of a dataset is extended with the list of similar datasets returned by a given similarity method (see Figure 1). In addition, when the list of similar datasets contains datasets, which are very similar among themselves, they can be displayed in groups. The granularity of grouping can be set by the user using the slider *Grouping similar datasets*.

**Agenda** [🔗](#) [👍](#)

**Ministry of Interior**

Agendas registered in the Register of Rights and Obligations in the sense of § 51 of Act No. 111/2009 Coll. on basic registers. The content is also available via API (SPARQL endpoint) <https://rpp-opendata.egon.gov.cz/odrpp/sparql>.

---

**agenda** **register of rights and obligations** **activity**

---

<b>Topic</b> Government and public sector <a href="#">🔗</a>	<b>Related geographical area</b> Czech Republic <a href="#">🔗</a>	<b>Documentation</b> <a href="#">View documentation</a>	<b>Update periodicity</b> daily <a href="#">🔗</a>
<b>Classification according to EuroVoc</b> decision-making power <a href="#">🔗</a> executive power <a href="#">🔗</a> competence of the institution <a href="#">🔗</a>		<b>Contact point</b> Basic Registry Administration Help Desk (SZR)	

---

**Similar datasets** Grouping similar datasets : Large

**Scope in agendas**  
Powers of public authorities or private data users in the agendas registered in the Register of Rights and Obligations in the sense of § 51 of Act No. 111/2009 Coll. on basic registers. The content is also available via API (SPARQL endpoint) <https://rpp-opendata.egon.gov.cz/odrpp/sparql>. [👍](#)

---

**Code list of activity types**  
Code list of types of activities used in the Register of Rights and Obligations in the sense of § 51 of Act No. 111/2009 Coll. on basic registers. The content is also available via API (SPARQL endpoint) <https://rpp-opendata.egon.gov.cz/odrpp/sparql>. [👍](#)

**Nomenclature of types of subjects**  
Code list of types of entities used in the Register of Rights and Obligations in the sense of § 51 of Act No. 111/2009 Coll. on basic registers. The content is also available via API (SPARQL endpoint) <https://rpp-opendata.egon.gov.cz/odrpp/sparql>. [👍](#)

**Code list of states**  
Code list of states used in the Register of Rights and Obligations in the sense of § 51 of Act No. 111/2009 Coll. on basic registers. The content is also available via API (SPARQL endpoint) <https://rpp-opendata.egon.gov.cz/odrpp/sparql>. [👍](#)

**Figure 1:** Dataset detail view with similar datasets (translated from Czech)

- It is possible to like a dataset or a group of similar datasets using the like button in the dataset title or in the list of similar datasets, marking it as relevant to the currently evaluated use case.

#### 4.1.2 Evaluation

During an evaluation session, a user searched for datasets either by using the full-text search offered by the original interface, or by browsing through similar datasets using their similarity computed with one of the similarity methods. When a user found a given dataset relevant for a given use case, they gave a like to it.

The evaluation consisted of three rounds – a warm-up round, the first round and the second round. During the warm-up round, the evaluators could try out the extended user interface as they pleased, getting familiar with it. In this round, the user interface contained both the full-text search functionality, and the similar datasets functionality.

In the first round, the users were first given a set of tasks, each task being an evaluation of a given similarity method on a given use case. All combinations of 6 use cases, 6 similarity methods and the full-text search method, amounting to the total of 42 tasks, were evaluated by each of the 10 users. The users had 7 minutes to evaluate each similarity-based task and 10 minutes to complete the full-text based task. The users did not know, which of the methods they are evaluating in each session, only that they are evaluating the full-text

14 *Open Dataset Discovery using Context-enhanced Similarity Search*

search method, or some of the similarity-based methods. One of the similarity methods was evaluated twice for verification purposes. Therefore, we actually tested 5 similarity methods, one of which was evaluated twice, and a full-text method. After the first round, the evaluators were asked to take at least 7 days break from the evaluations, before continuing with the second round.

In the second round, the users were given a new use case and a user interface which allowed them to use a similarity method we selected for them and the original full-text search functionality. We selected the method from the first round which was resulting in the biggest number of liked datasets overall, when we cleaned the results of one outlying user. This does not necessarily mean that the selected method is the best similarity method. However, it is good enough to test the usability of a dataset search user interface enhanced with similarity-based dataset searching functionality. This was the goal of the evaluation in the second round. In the second round, the users were given 15 minutes to finish the tasks. After finishing it we asked them to answer a couple of questions about the usability of the user interface with the combination of the full-text search functionality and the similar datasets functionality (SUS questionnaire). The questions and results are discussed in [section 4.4](#).

### 4.1.3 Collected Data

For each task we recorded each of the following user actions:

- **like** – a user liked a certain dataset or a group of datasets
- **dislike** – a user cancelled a like of a certain dataset or a group of datasets
- **navigation** – a user navigated in the user interface from one page to another, including the full-text search query
- **openWindow** – a user opened a new browser tab or window
- **closeWindow** – a user closed a new browser tab or window
- **autoStart** – the evaluation session started automatically, based on the unique URL provided to each of the users
- **finish** – user manually finished the evaluation by clicking **Finish evaluation** button
- **start** – user manually started the evaluation, or resumed previously finished evaluation

We recorded the sequence of the user actions as a stream of time-stamped events. The dataset is published and persisted in Zenodo [50]. It links to the dataset containing dataset metadata from the data catalog used for the evaluation [51]. We provide an example here, showing an event stream representing a user session, its members representing user actions, and we provide an example of a "Like" action of a dataset.

```
<https://.../tf_idf_en_min_iprod-...>
  a ldes:EventStream;
  odsim:method <https://.../tf_idf_en_min_iprod>;
  odsim:useCase <https://.../use-case-005>;
  odsim:user <https://.../U03>;
```

Preprint submitted to KAIS

*Open Dataset Discovery using Context-enhanced Similarity Search* 15

```
tree:member <https://.../2021-10-13T08:20:24.139Z>,  
  <https://.../2021-10-13T08:20:24.160Z>,  
  <https://.../2021-10-13T08:20:37.863Z>,  
  <https://.../2021-10-13T08:20:39.582Z>;  
ldes:timestampPath sosa:resultTime .
```

```
<https://.../2021-10-13T08:20:37.863Z>  
  a sosa:Observation;  
  sosa:resultTime "2021-10-13T08:20:37.863Z"^^xsd:dateTime;  
  odsim:dataset <https://.../Ostrava/37085167>;  
  odsim:action <https://.../actions/like> .
```

We then used the event streams to evaluate the similarity methods and the provided user interface for dataset search based on their similarity as follows.

First, we use the event streams to construct a new ground truth for evaluating dataset similarity methods in [section 4.2](#). In [section 4.2](#) we also show such evaluation of several dataset similarity methods (a superset of those used in the user evaluation). Second, we use the event streams to confirm the hypothesis from the introduction to this paper in [section 4.3](#). Finally, we evaluate the usability of the provided user interface using the System usability scale method in [section 4.4](#).

## 4.2 Ground Truth Evaluation

In this section, we describe the importance and process of creating ground truth testbed for automatic evaluation.

### 4.2.1 Ground Truth Testbed Construction

A user evaluation of different similarity models can be time-consuming and costly. For experimental evaluation, it is necessary to rely at least partially on automatic evaluation, which can serve as a good indicator of a retrieval method quality. However, an automatic evaluation cannot work without a clearly specified ground truth (search results provided or confirmed by a human user). Unfortunately, to the best of our knowledge there does not exist a public testbed for open dataset discovery that would also include a well-defined ground truth. Therefore, it is necessary first to build such a testbed.

The construction of ground truth involves many problems. Open data catalogs contain a large number of datasets from various providers. This means that it is not a controlled database, and it is impossible to specify which datasets can be found in it. Particular tasks to find datasets must have at least partial coverage in the database. The actual building of ground truth by a user (domain expert) is time-consuming and can be considered subjective. Therefore, it is quite logical to include multiple independent users in the process of ground truth construction.

For each use case, users selected several datasets that they considered relevant through our evaluation tool. The ground truth for each use case can then

16 *Open Dataset Discovery using Context-enhanced Similarity Search*

be constructed as a set of datasets that have been identified as relevant by at least  $q$  users. The choice of the parameter  $q$  can significantly affect the results, for  $q = 100\%$  the ground truth will be constructed very conservatively. Theoretically, there may be a situation where the result is an empty set, i.e., a user disagrees with the others. On the other hand, lower  $q$  values support creative users and their ideas and increase the total set of relevant datasets.

	UC1	UC2	UC3	UC4	UC5	UC6
$q = 10\%$	<b>5.71%</b>	0.86%	2.57%	1.60%	1.89%	<b>57.36%</b>
$q = 20\%$	<b>5.08%</b>	0.24%	1.65%	1.10%	1.24%	<b>51.67%</b>
$q = 30\%$	2.49%	0.17%	1.03%	0.90%	0.49%	<b>36.66%</b>
$q = 40\%$	0.43%	0.10%	0.68%	0.76%	0.32%	<b>6.03%</b>
$q = 50\%$	0.06%	0.10%	0.49%	0.70%	0.21%	0.05%
$q = 60\%$	<b>0.00%</b>	0.08%	0.44%	0.68%	0.17%	0.02%
$q = 70\%$	<b>0.00%</b>	0.08%	0.33%	0.62%	0.11%	0.02%
$q = 80\%$	<b>0.00%</b>	0.08%	0.27%	0.52%	0.10%	<b>0.00%</b>
$q = 90\%$	<b>0.00%</b>	0.05%	0.22%	0.27%	0.10%	<b>0.00%</b>
$q = 100\%$	<b>0.00%</b>	0.02%	0.17%	0.17%	0.10%	<b>0.00%</b>

**Table 2:** Number of relevant items (datasets) normalized by the database size for different  $q$  (starting dataset is excluded).

Table 2 shows the size of the resulting set of relevant datasets over each parameter  $q$ . We can see that for use cases UC1 and UC6 and large parameters  $q$  the users do not agree on any relevant dataset except the starting one (not included in the table nor in the automatic evaluation). Similarly, for low  $q$  values the resulting set of relevant datasets is too large ( $\geq 5\%$  of all datasets). Following the HCI recommendations, the user interface shows only up to ten most similar datasets in our setup (i.e., following the Google search ergonomiy). It corresponds to 0.1% of the database size (6302 datasets). Therefore, it makes sense to consider these use cases in the automatic evaluation only for reasonable values of  $q$ .

The ground truth for each relevant combination of a use case and a  $q$  value, i.e., those for which there is non-empty set of datasets, is published in Zenodo for reuse as a separate dataset [52] linked to the evaluated datasets from [51]. Each ground truth is modeled as a SKOS [53] collection of datasets in the data.

Sample of the ground truth data for UC1 and  $q = 50\%$ :

```
col:UC1_50 a skos:Collection;
skos:prefLabel "Ground truth of UC1 with q=50%"@en;
odsim:useCase "UC1";
odsim:q 50 ;
skos:member ds:https---...190910full,
<https://data.gov.cz/...ntrolniU/98197>,
ds:http---data.ctu.cz-ap...0-ae26-4be34d1d477f,
<https://data.gov.cz/...dy/C0bchodniI/98803>,
ds:http---data.mfcr...rav-dusevniho-vlastnictvi
```

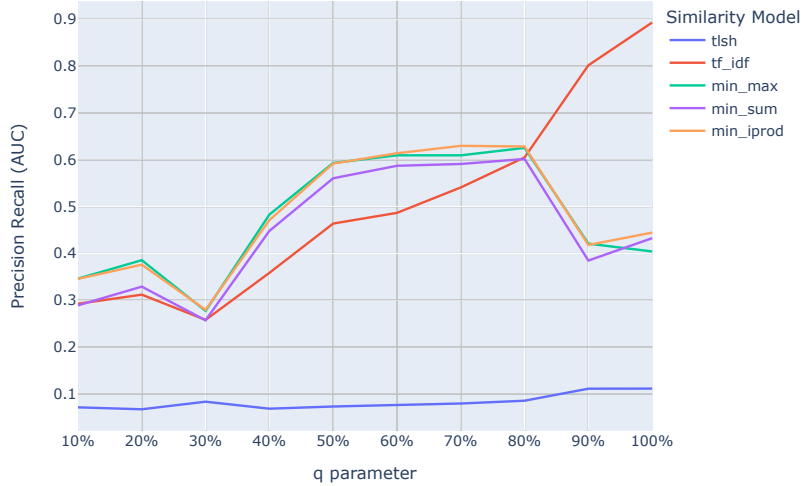


Figure 2: Precision-recall area under the curve score

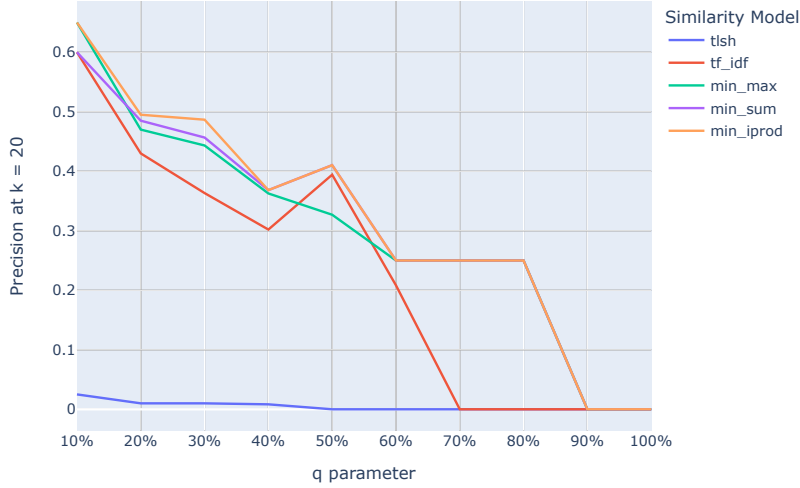
#### 4.2.2 Automatic Evaluation

In terms of automatic evaluation, we can look at two criteria, precision and recall [54]. If the search engine returns  $k$  results (datasets), the precision is a fraction of the results the users consider as relevant. Recall tries to capture the relationship between all relevant results and those obtained by a search engine. We use an 11-point precision-recall curve [55] to compare the search-engine behavior for different  $k$ . In the case of an ideal search engine, for any value of recall, precision should be 1, while a bad search engine will have almost zero precision values. To simplify the comparison, we compute the area under the precision-recall curve (PR AUC) and compare the different variants accordingly. This value is averaged over all valid use cases.

Figure 2 shows the relationship of the PR AUC of each similarity model as a function of  $q = \{10\%, 20\%, \dots, 80\%, 90\%, 100\%\}$ . In the case of the conservative setting of parameter  $q$ , we obtain the highest success rate by using the standard TF-IDF similarity. Decreasing the parameter  $q$  leads to higher success rates of similarity models based on data-transitivity. The main reason for this is the ability of data-transitive models to capture direct similarities and, more importantly, to capture associations between completely dissimilar datasets. The TLSH approach has proven to be insufficient.

The effectiveness gain when using data-transitive similarity models can be seen even more in Figure 3. Here we show the precision at  $k = 20$  of the returned results. To show the added value, we do not include the relevant





**Figure 3:** Precision at  $k = 20$ , without relevant datasets found by full-text search

results found by users using standard keyword search (full-text) in the results. Hence, the figure shows how data-transitive similarity models add extra value over the standard keyword search.

Overall, the variants of data-transitive similarity models behave very similarly, with minimal differences. Considering precision-recall curve and precision at  $k = 20$  results, the `min_iprod` variant performs best. However, we must stress that this is the model’s success rate over our use cases. Other models may be more appropriate for different use cases. This is shown by the difference between  $q = 50\%$  and  $q = 100\%$ ; due to the minimal number of datasets for  $q = 100\%$  (all users must agree).

### 4.2.3 Extended Evaluation

Apart from the few similarity models employed in the user evaluation, we performed automatic evaluations also for more than 200 other similarity models, however, they were evaluated with respect to the new Ground Truth. These models cover a variety of similarity functions, parameters, contexts, and metadata. A complete list would be exhaustive and uninformative, so below we list only the results of the most interesting models.

Table 3 shows the PR AUC values for the selected models and the selected parameters  $q$ . In total, we see three different approaches, all working with the union of all metadata (title, description, keyword) and cosine distance. The

Preprint submitted to KAIS

*Open Dataset Discovery using Context-enhanced Similarity Search* 19

Model \ $q$ parameter	50%	60%	80%	100%
CosD(TDK-en   L   BoW(tf-idf))	0.47	0.49	0.61	<b>0.90</b>
DT(CosD(TDK-en   L   BoW(tf-idf)), min, max)	0.60	0.61	0.63	0.41
CosD(TDK-en   L   SW   BoW(tf-idf))	0.46	0.49	0.61	0.89
DT(CosD(TDK-en   L   SW   BoW(tf-idf)), min, max)	0.59	0.61	0.63	0.40
DT(CosD(TDK-en   L   SW   BoW(tf-idf)), min, iprod)	0.59	0.61	0.63	0.45
CosD(TDK-en   L   Word2Vec(enwiki)   avg)	0.58	0.61	0.71	0.89
DT(CosD(TDK-en   L   Word2Vec(enwiki)   avg), min, max)	0.51	0.56	<b>0.72</b>	0.73
DT(CosD(TDK-en   L   Word2Vec(enwiki)   avg), min, iprod)	<b>0.71</b>	<b>0.76</b>	0.67	0.66

**Table 3:** PR AUC values for selected similarity models.

difference is in the processing; for the first two TF-IDF models we do not filter frequent words. Last three models utilize `Word2Vec[enwiki]` embedding, which converts individual words to vectors generated by the Word2Vec model learned over the English Wikipedia articles. It can be seen that stop-word filtering does not matter that much; on the other hand, the use of embeddings can be beneficial. Again, we have confirmed that the less conservative the  $q$  parameter setting we use, the more successful data-transitive models are compared to the other approaches.

Figure 4 shows the precision at  $k = 20$  returned results excluding results found by full-text search for  $q = 50\%$ . It may be interesting to observe that title and description are the biggest information carriers. Another interesting observation may be that models with language translation can be more efficient than the original ones. This may be due to several factors, the first factor is the use of english models that are trained on larger amounts of data. The second reason may be the translation itself, which prefers the same translations of words instead of using synonyms. Models using Node2Vec based on Wikidata hierarchy, BERT, or Hausdorff distance did not prove to be very effective for our use cases. Last but not least, it shows importance of data-transitive similarity models, which work very well as a complement to traditional full-text search.

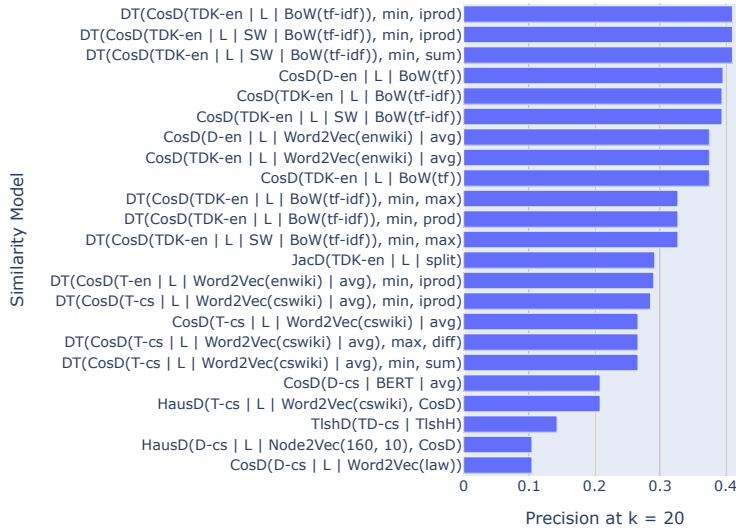
### 4.3 Evaluation of similarity search vs. fulltext search

In this section, we use the event streams recorded in our user evaluation to confirm the research hypothesis we already posed in section 1:

*Context-enhanced similarity-based dataset discovery extends the search results with relevant datasets that are not findable using the full-text search methods.*

First, we analyze the number of liked datasets per use case and method (either a method based on a similarity model or the full-text search method) by individual evaluators. The results are visualized in Figure 5 as box-plot diagrams for each use case.<sup>9</sup> The first observation is that users found some relevant datasets for each use case and search method. None of the methods outperforms on a significant number of use cases. We can see that the full-text

<sup>9</sup>The numbers can be computed using `query1.sparql` published with the dataset [50]

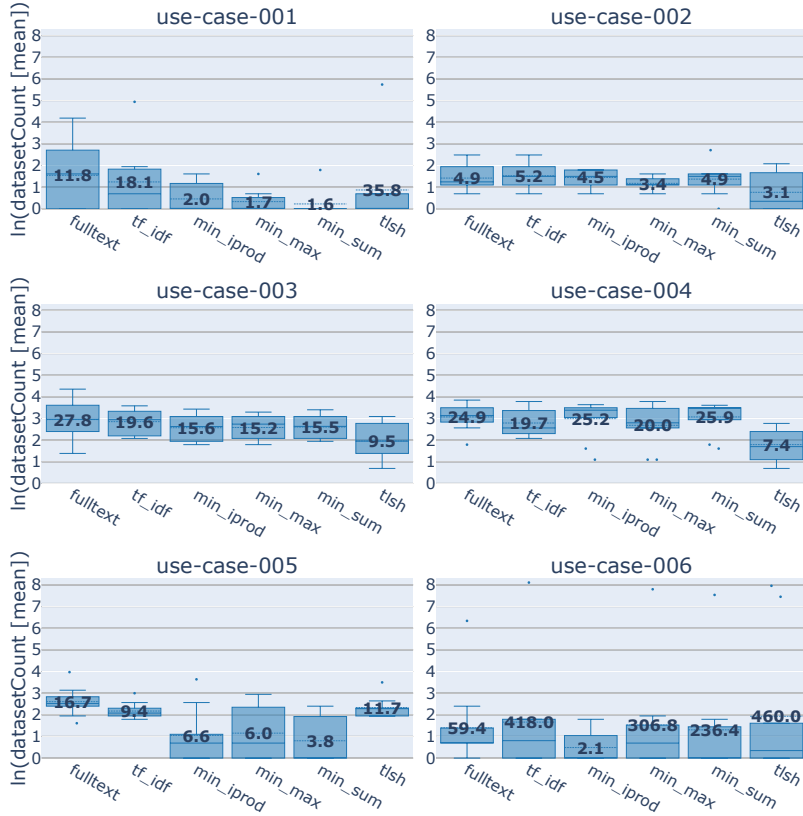


**Figure 4:** Precision at  $k = 20$ , without relevant datasets found by full-text search during extended evaluation for  $q = 50\%$

search performs quite well as the number of found datasets is high in comparison to the similarity methods. In other words, the evaluators were able to find significant amount of datasets using the classical full-text search method. They were also able to find some datasets using the similarity methods.

The previous evaluation does not confirm nor disprove our hypothesis. We need to know which datasets the evaluators found using a similarity method but not with the full-text method and, vice versa, which datasets they found using the full-text search method but not using none of the similarity methods. Figure 6 shows the number of datasets per use case and similarity method which were liked by evaluators without the datasets which were also liked when the full-text method was used.<sup>10</sup> In other words, it shows us how similarity methods complement the full-text method with showing datasets which were unnoticed by the evaluators when they used the full-text search. We can see that the majority of the evaluators liked datasets which they did not like when using the full-text search. This confirms the part of our hypothesis that the dataset similarity methods provide users with datasets which are not findable

<sup>10</sup>The numbers can be computed using `query2.sparql` published with the dataset [50]



**Figure 5:** Number of datasets liked by evaluators per use case and method ( $\ln$  scale).

by the full-text search method or which they did not notice when using the full-text method. We can therefore say that dataset similarity methods supplement the full-text search accordingly.

Figure 7 shows the numbers of datasets the evaluators liked when using the full-text search but which they did not like or did not notice when using the similarity search methods.<sup>11</sup> It shows that using only the similarity methods is not sufficient for the dataset search because for each use case there are some datasets the evaluators noticed (i.e. gave like to them) when using the full-text method but did not notice when using the similarity search methods.

These two evaluations confirm our hypothesis. None of the methods is sufficient on its own. Our evaluation shows that for various dataset search use cases there are datasets which were noticed by the evaluators using the classical

<sup>11</sup>The numbers can be computed using `query3.sparql` published with the dataset [50]

22 *Open Dataset Discovery using Context-enhanced Similarity Search*

full-text search. However, there are also relevant datasets which the evaluators did not notice when using the full-text search and needed a similarity method. Therefore, it is beneficial to combine the classical full-text search with similarity methods.

Let us demonstrate the evaluation on a concrete example. Table 4 shows a dataset *Forests*, which was found useful to the use case *Climate status* with the starting dataset *Climate bonity* by 4 users during the evaluation. The interesting fact is that the users found this dataset in the data catalog using some of the similarity-based methods and at the same time, no user found this dataset using the full-text search.

	<b>Title Keywords</b>	<b>Description</b>
<b>S</b>	<i>Climate bonity</i> <b>Prague</b>	Climate quality - a comprehensive description according to all evaluated climatological aspects. The data were created using ArcGIS 9.2, Spatial Analyst. The layer was transferred from the raster layer bonita, with a horizontal resolution of 25m. The following data were used for the implementation of this map: Digital reference map Prague-block map of the building Line layer of street sections Vector data of the thematic layer Úpn-transport-line layer of the road network Outputs from the Model Air Quality Assessment in Prague-update 2006 by ATEM s.r.o. Digital terrain model DMR25 Orthophotos of the Czech Republic. The resulting creditworthiness map was created on the basis of the weighted average of partial maps using map algebra.
<b>F</b>	<i>Forests</i> <b>GIS, overview</b>	Forest areas in the territory of Pilsen and its surroundings (derived from the cadastral map).

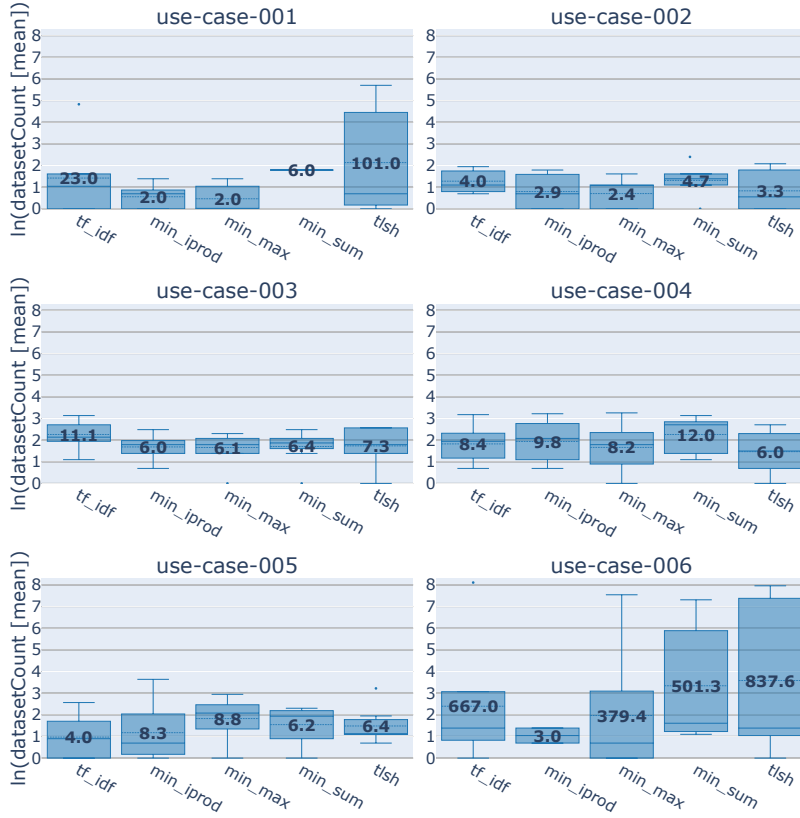
**Table 4:** Example of a starting dataset (**S**) and the dataset found useful by multiple users in the evaluation using similarity-based methods (**F**).

#### 4.4 System usability evaluation

To evaluate the usability of the user interface of the LinkedPipes DCAT-AP Viewer extended with the listing of datasets similar to a particular dataset based on our similarity methods, we used the System usability scale (SUS).<sup>12</sup> The SUS methodology uses a simple form consisting of the following 10 questions.

- Q1** I think that I would like to use this system frequently.
- Q2** I found the system unnecessarily complex.
- Q3** I thought the system was easy to use.
- Q4** I think that I would need the support of a technical person to be able to use this system.
- Q5** I found the various functions in this system were well integrated.

<sup>12</sup><https://measuringu.com/sus/>

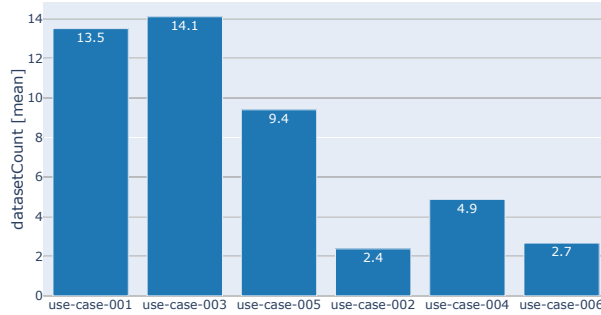


**Figure 6:** Average number of liked datasets per use case and model not found by the full-text method ( $\ln$  scale).

- Q6** I thought there was too much inconsistency in this system.  
**Q7** I would imagine that most people would learn to use this system very quickly.  
**Q8** I found the system very cumbersome to use.  
**Q9** I felt very confident using the system.  
**Q10** I needed to learn a lot of things before I could get going with this system.

Each question can be answered on a scale from 1 to 5 where 1 means *Strongly disagree* and 5 means *Strongly agree*. The resulting SUS score is computed as  $SUS = 2.5 * (20 + r(Q1) - r(Q2) + r(Q3) - r(Q4) + r(Q5) - r(Q6) + r(Q7) - r(Q8) + r(Q9) - r(Q10))$  where  $r(QN)$  represents the result given in the  $N$ th question.

We received 10 answers from our 10 users, as can be seen in [Table 5](#) yielding an average SUS score of **70**. According to the original description of the



**Figure 7:** Average number of liked datasets per use case found only by the full-text method.

U. / Q.	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS
U01	4	2	2	3	1	5	1	3	1	4	30
U02	3	3	4	4	4	3	3	4	2	4	45
U03	1	2	5	4	2	4	5	1	2	1	57.5
U04	4	1	4	1	4	2	4	2	4	1	82.5
U05	5	2	5	2	5	1	4	2	4	1	87.5
U06	3	1	4	1	4	4	5	2	2	1	72.5
U07	5	2	5	1	4	2	5	1	4	2	87.5
U08	5	2	4	2	3	3	5	2	4	2	75
U09	5	2	4	1	4	2	5	2	3	1	82.5
U10	4	2	4	1	3	2	5	1	3	1	80

**Table 5:** System Usability Scale evaluation – answers and SUS scores

SUS methodology published by U.S. General Services Administration (GSA) Technology Transformation Service<sup>13</sup>, "based on research, a SUS score above a 68 would be considered above average and anything below 68 is below average". More recent study [56] describes 7 grades of SUS scores from 1: *Worst imaginable* to 7: *Best imaginable* with grades 4: *OK* with the SUS mean score 50.9, 5: *Good* with 71.4 and 6: *Excellent* with 85.5. Therefore, the resulting score shows that our interface for searching data sets in a catalog extended with the similarity methods is considered good for the users.

It is worth noting that besides the standard SUS questions, we asked each user for a free-text generic comment regarding the evaluation. From the comment of U01 it was clear that their answers were heavily biased by the quality of the dataset metadata in the platform, which is something we cannot influence.

<sup>13</sup><https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>

## 5 Conclusions

In this paper we proposed an approach to dataset discovery based on similarity search over metadata descriptions enhanced by various semantic contexts. In experimental evaluation we have shown that context-enhanced similarity retrieval methods increase the findability of relevant datasets. As a part of the evaluation we created a catalog-like user interface for dataset discovery and recorded streams of user actions that served us to create the ground truth. We published the entire evaluation testbed.

Based on the evaluation, we recommend the developers of open dataset catalogs to implement the option of similarity-based retrieval into the catalog search engines. The similarity-based search alone does not outperform full-text or keyword search in all dataset discovery scenarios. Though, it could be a useful complementary search option that together with the full-text search could improve the effectiveness of the dataset discovery. The similarity-based search could be especially useful for improving the retrieval recall (completeness of search results). In particular, the retrieval recall can be improved for more opened discovery usecases similar to those presented in the evaluation. These usecases start with an open-ended goal to build a certain service and with a starting dataset found using the classical keyword search. For each usecase, there were datasets that were similar to the given dataset but could not be found using keywords. However, we also showed that it is not possible to choose the best similarity technique. It is necessary to implement more different similarity techniques and combine them.

**Acknowledgments.** This work was supported by the Czech Science Foundation (GAČR), grant number 19-01641S.

## References

- [1] Miller, R.J., Nargesian, F., Zhu, E., Christodoulakis, C., Pu, K.Q., Andritsos, P.: Making open data transparent: Data discovery on open data. *IEEE Data Eng. Bull.* **41**(2), 59–70 (2018). <http://sites.computer.org/debull/A18june/p59.pdf>
- [2] Brickley, D., Burgess, M., Noy, N.F.: Google dataset search: Building a search engine for datasets in an open web ecosystem. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 1365–1375. ACM, ??? (2019). <https://doi.org/10.1145/3308558.3313685>. <https://doi.org/10.1145/3308558.3313685>
- [3] Chen, X., Gururaj, A.E., Ozyurt, B., Liu, R., Soysal, E., Cohen, T., Tiryaki, F., Li, Y., Zong, N., Jiang, M., Rogith, D., Salimi, M., Kim, H.-e., Rocca-Serra, P., Gonzalez-Beltran, A., Farcas, C., Johnson, T., Margolis, R., Alter, G., Sansone, S.-A., Fore, I.M., Ohno-Machado, L., Grethe, J.S., Xu, H.: DataMed – an open source discovery index for finding biomedical



26 *Open Dataset Discovery using Context-enhanced Similarity Search*

- datasets. *Journal of the American Medical Informatics Association* **25**(3), 300–308 (2018). <https://doi.org/10.1093/jamia/ocx121>
- [4] Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.D., Kacprzak, E., Groth, P.: Dataset search: a survey. *VLDB J.* **29**(1), 251–272 (2020). <https://doi.org/10.1007/s00778-019-00564-x>
- [5] Gregory, K., Groth, P., Scharnhorst, A., Wyatt, S.: Lost or found? discovering data needed for research. *Harvard Data Science Review* **2**(2) (2020). <https://doi.org/10.1162/99608f92.e38165eb>
- [6] Gregory, K.M., Cousijn, H., Groth, P., Scharnhorst, A., Wyatt, S.: Understanding data search as a socio-technical practice. *Journal of Information Science* **46**(4), 459–475 (2020). <https://doi.org/10.1177/0165551519837182>
- [7] Koesten, L.: A User Centred Perspective on Structured Data Discovery. In: Companion Proceedings of the The Web Conference 2018. WWW '18, pp. 849–853. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018). <https://doi.org/10.1145/3184558.3186574>
- [8] Klímek, J.: DCAT-AP representation of Czech National Open Data Catalog and its impact. *J. Web Semant.* **55**, 69–85 (2019). <https://doi.org/10.1016/j.websem.2018.11.001>
- [9] Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search - The Metric Space Approach. *Advances in Database Systems*, vol. 32. Springer, Boston, MA, USA (2006). <https://doi.org/10.1007/0-387-29151-2>
- [10] Hetland, M.L., Skopal, T., Lokoc, J., Beecks, C.: Ptolemaic access methods: Challenging the reign of the metric space model. *Inf. Syst.* **38**(7), 989–1006 (2013). <https://doi.org/10.1016/j.is.2012.05.011>
- [11] Connor, R., Vadicamo, L., Cardillo, F.A., Rabitti, F.: Supermetric search. *Information Systems* **80**, 108–123 (2019). <https://doi.org/10.1016/j.is.2018.01.002>
- [12] Skopal, T., Bustos, B.: On nonmetric similarity search problems in complex domains. *ACM Comput. Surv.* **43**(4), 34–13450 (2011). <https://doi.org/10.1145/1978802.1978813>
- [13] Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., Yu, C.: Finding Related Tables. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD '12, pp. 817–828. Association for Computing Machinery, New York, NY, USA (2012). <https://doi.org/10.1145/2213836.2213962>

- [14] Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD '12, pp. 97–108. Association for Computing Machinery, New York, NY, USA (2012). <https://doi.org/10.1145/2213836.2213848>
- [15] Zhang, S., Balog, K.: Ad Hoc Table Retrieval Using Semantic Similarity. In: Proceedings of the 2018 World Wide Web Conference. WWW '18, pp. 1553–1562. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018). <https://doi.org/10.1145/3178876.3186067>
- [16] Fernandez, R.C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., Stonebraker, M.: Aurum: A Data Discovery System. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018, pp. 1001–1012. IEEE Computer Society, USA (2018). <https://doi.org/10.1109/ICDE.2018.00094>
- [17] Bogatu, A., Fernandes, A.A.A., Paton, N.W., Konstantinou, N.: Dataset Discovery in Data Lakes. In: 36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020, pp. 709–720. IEEE, USA (2020). <https://doi.org/10.1109/ICDE48307.2020.00067>
- [18] Mountantonakis, M., Tzitzikas, Y.: Content-Based Union and Complement Metrics for Dataset Search over RDF Knowledge Graphs. *J. Data and Information Quality* **12**(2) (2020). <https://doi.org/10.1145/3372750>
- [19] Altaf, B., Akujuobi, U., Yu, L., Zhang, X.: Dataset Recommendation via Variational Graph Autoencoder. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 11–20 (2019). <https://doi.org/10.1109/ICDM.2019.00011>
- [20] Degbelo, A., Teka, B.B.: Spatial search strategies for open government data: A systematic comparison. *CoRR* **abs/1911.01097** (2019) <https://arxiv.org/abs/1911.01097>
- [21] Tekli, J., Chbeir, R.: Minimizing user effort in XML grammar matching. *Information Sciences* **210**, 1–40 (2012). <https://doi.org/10.1016/j.ins.2012.04.026>
- [22] Tekli, J., Chbeir, R., Traina, A.J.M., Traina, C., Fileto, R.: Approximate XML structure validation based on document–grammar tree similarity. *Information Sciences* **295**, 258–302 (2015). <https://doi.org/10.1016/j.ins.2014.09.044>
- [23] Hovy, E., Navigli, R., Ponzetto, S.P.: Collaboratively built semi-structured

28 *Open Dataset Discovery using Context-enhanced Similarity Search*

- content and Artificial Intelligence: The story so far. *Artificial Intelligence* **194**, 2–27 (2013). <https://doi.org/10.1016/j.artint.2012.10.002>. Artificial Intelligence, Wikipedia and Semi-Structured Resources
- [24] Tekli, J., Chbeir, R., Traina, A.J.M., Traina, C.: SemIndex+: A semantic indexing scheme for structured, unstructured, and partly structured data. *Knowledge-Based Systems* **164**, 378–403 (2019). <https://doi.org/10.1016/j.knosys.2018.11.010>
- [25] Berners-Lee, T.: Linked Data (2006). <https://www.w3.org/DesignIssues/LinkedData.html>
- [26] Mountantonakis, M., Tzitzikas, Y.: Scalable Methods for Measuring the Connectivity and Quality of Large Numbers of Linked Datasets. *J. Data and Information Quality* **9**(3) (2018). <https://doi.org/10.1145/3165713>
- [27] Wagner, A., Haase, P., Rettinger, A., Lamm, H.: Entity-Based Data Source Contextualization for Searching the Web of Data. In: *The Semantic Web: ESWC 2014 Satellite Events*, pp. 25–41. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11955-7\\_3](https://doi.org/10.1007/978-3-319-11955-7_3)
- [28] Ben Ellefi, M., Bellahsene, Z., Dietze, S., Todorov, K.: Dataset recommendation for data linking: An intensional approach. In: *The Semantic Web. Latest Advances and New Domains*, pp. 36–51. Springer, Cham (2016)
- [29] Ellefi, M.B., Bellahsene, Z., Dietze, S., Todorov, K.: Dataset Recommendation for Data Linking: An Intensional Approach. In: Sack, H., Blomqvist, E., d’Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings. Lecture Notes in Computer Science*, vol. 9678, pp. 36–51. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-34129-3\\_3](https://doi.org/10.1007/978-3-319-34129-3_3)
- [30] Martins, Y.C., da Mota, F.F., Cavalcanti, M.C.: DSCrank: A Method for Selection and Ranking of Datasets, pp. 333–344. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-49157-8\\_29](https://doi.org/10.1007/978-3-319-49157-8_29)
- [31] Leme, L.A.P.P., Lopes, G.R., Nunes, B.P., Casanova, M.A., Dietze, S.: Identifying Candidate Datasets for Data Interlinking. In: Daniel, F., Dolog, P., Li, Q. (eds.) *Web Engineering*, pp. 354–366. Springer, Berlin, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-39200-9\\_29](https://doi.org/10.1007/978-3-642-39200-9_29)
- [32] Oliver, J., Cheng, C., Chen, Y.: TLSH – A Locality Sensitive Hash. In: *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 7–13 (2013). <https://doi.org/10.1109/CTC.2013.9>

- [33] Dutkowski, S., Schramm, A.: Duplicate Evaluation - Position paper by Fraunhofer FOKUS. Technical report, Fraunhofer FOKUS (2015). [https://www.w3.org/2016/11/sdsvoc/SDSVoc16\\_paper\\_24](https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_24)
- [34] Miller, F.P., Vandome, A.F., McBrewster, J.: Levenshtein Distance. Alphascript Publishing (2009). <https://www.abebooks.com/products/isbn/9786130216900>
- [35] Straka, M., Hajič, J., Straková, J.: UDPipe: Trainable pipeline for processing CoNLL-u files performing tokenization, morphological analysis, POS tagging and parsing. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pp. 4290–4297. European Language Resources Association (ELRA), Portorož, Slovenia (2016). <https://www.aclweb.org/anthology/L16-1680>
- [36] Straka, M., Straková, J.: Universal Dependencies 2.5 Models for UDPipe (2019-12-06). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University (2019). <http://hdl.handle.net/11234/1-3131>
- [37] Sammut, C., Webb, G.I. (eds.): TF-IDF, pp. 986–987. Springer, Boston, MA (2010). [https://doi.org/10.1007/978-0-387-30164-8\\_832](https://doi.org/10.1007/978-0-387-30164-8_832)
- [38] About WordNet. Princeton University, USA (2010). Princeton University. <https://wordnet.princeton.edu/>
- [39] Speer, R., Chin, J., Havasi, C.: ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA, pp. 4444–4451. AAAI Press, California, USA (2017). <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972>
- [40] Škoda, P., Matějčík, J., Skopal, T.: Visualizer of Dataset Similarity Using Knowledge Graph. In: Satoh, S., Vadicamo, L., Zimek, A., Carrara, F., Bartolini, I., Aumüller, M., Jónsson, B.P., Pagh, R. (eds.) Similarity Search and Applications - 13th International Conference, SISAP 2020, Copenhagen, Denmark, September 30 - October 2, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12440, pp. 371–378. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-60936-8\\_29](https://doi.org/10.1007/978-3-030-60936-8_29)
- [41] Tekli, J., Charbel, N., Chbeir, R.: Building Semantic Trees from XML Documents. *Web Semant.* **37**(C), 1–24 (2016). <https://doi.org/10.1016/j.websem.2016.03.002>

- 30 *Open Dataset Discovery using Context-enhanced Similarity Search*
- [42] Pilehvar, M.T., Navigli, R.: A Large-Scale Pseudoword-Based Evaluation Framework for State-of-the-Art Word Sense Disambiguation. *Comput. Linguist.* **40**(4), 837–881 (2014). [https://doi.org/10.1162/COLI\\_a\\_00202](https://doi.org/10.1162/COLI_a_00202)
- [43] Tekli, J.: An Overview on XML Semantic Disambiguation from Unstructured Text to Semi-Structured Data: Background, Applications, and Ongoing Challenges. *IEEE Transactions on Knowledge and Data Engineering* **28**(6), 1383–1407 (2016). <https://doi.org/10.1109/TKDE.2016.2525768>
- [44] Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient Estimation of Word Representations in Vector Space (2013). <http://arxiv.org/abs/1301.3781>
- [45] Grover, A., Leskovec, J.: node2vec: Scalable Feature Learning for Networks. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, August 13-17, 2016, pp. 855–864. ACM, USA (2016). <https://doi.org/10.1145/2939672.2939754>
- [46] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018). <https://arxiv.org/abs/1810.04805>
- [47] Skopal, T., Bernhauer, D., Škoda, P., Klímek, J., Nečaský, M.: Similarity vs. Relevance: From Simple Searches to Complex Discovery. In: Reyes, N., Connor, R., Kriege, N.M., Kazempour, D., Bartolini, I., Schubert, E., Chen, J. (eds.) *Similarity Search and Applications - 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 - October 1, 2021, Proceedings. Lecture Notes in Computer Science*, vol. 13058, pp. 104–117. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-89657-7\\_9](https://doi.org/10.1007/978-3-030-89657-7_9)
- [48] Kořarko, O., Variš, D., Popel, M.: LINDAT Translation service. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University (2019). <http://hdl.handle.net/11234/1-2922>
- [49] Klímek, J., Škoda, P.: LinkedPipes DCAT-AP Viewer: A Native DCAT-AP Data Catalog. In: van Erp, M., Atre, M., López, V., Srinivas, K., Fortuna, C. (eds.) *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks Co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018. CEUR Workshop Proceedings*, vol. 2180. CEUR-WS.org, Aachen, Germany (2018). <https://ceur-ws.org/Vol-2180/>

Preprint submitted to KAIS

*Open Dataset Discovery using Context-enhanced Similarity Search* 31

[paper-32.pdf](#)

- [50] Škoda, P., Klímek, J.: Data collected from user evaluation of dataset search using similarity methods. Zenodo (2021). <https://doi.org/10.5281/zenodo.5788427>
- [51] Klímek, J., Škoda, P.: Dump of metadata from the Czech National Open Data Catalog, 2020-04-20, State Administration of Land Surveying and Cadastre datasets removed. Zenodo (2021). <https://doi.org/10.5281/zenodo.4433464>
- [52] Klímek, J., Bernhauer, D.: Ground truths for dataset search using similarity methods generated from a user evaluation. Zenodo (2021). <https://doi.org/10.5281/zenodo.5788444>
- [53] Bechhofer, S., Miles, A.: SKOS Simple Knowledge Organization System Reference. W3C Recommendation, W3C (August 2009). <https://www.w3.org/TR/2009/REC-skos-reference-20090818/>
- [54] Baeza-Yates, R., Ribeiro-Neto, B.A.: Modern Information Retrieval - the Concepts and Technology Behind Search, Second Edition. Pearson Education Ltd., Harlow, England (2011). <http://www.mir2ed.org/>
- [55] Zhang, E., Zhang, Y.: In: LIU, L., ÖZSU, M.T. (eds.) Eleven Point Precision-recall Curve, pp. 981–982. Springer, Boston, MA (2009). [https://doi.org/10.1007/978-0-387-39940-9\\_481](https://doi.org/10.1007/978-0-387-39940-9_481)
- [56] Lewis, J.R.: The system usability scale: Past, present, and future. *International Journal of Human–Computer Interaction* **34**(7), 577–590 (2018). <https://doi.org/10.1080/10447318.2018.1455307>

---

# Modular Framework for Similarity-based Dataset Discovery using External Knowledge

Nečaský, M. (20 %); Škoda, P. (20 %); Bernhauer, D. (20 %); Klímek, J. (20 %); Skopal, T. (20 %) Modular framework for similarity-based dataset discovery using external knowledge. *Data Technologies and Applications*, volume *ahead-of-print*, no. *ahead-of-print*, 2022, doi: 10.1108/DTA-09-2021-0261. [A.2]

The current issue and full text archive of this journal is available on Emerald Insight at:  
<https://www.emerald.com/insight/2514-9288.htm>

# Modular framework for similarity-based dataset discovery using external knowledge

Similarity-based dataset discovery framework

Martin Nečaský and Petr Škoda

*Department of Software Engineering, Faculty of Mathematics and Physics,  
Charles University, Prague, Czechia*

David Bernhauer

*Department of Software Engineering, Faculty of Mathematics and Physics,  
Charles University, Prague, Czechia and*

*Department of Software Engineering, Faculty of Information Technology,  
Czech Technical University in Prague, Prague, Czechia, and*

Jakub Klímek and Tomáš Skopal

*Department of Software Engineering, Faculty of Mathematics and Physics,  
Charles University, Prague, Czechia*

---

Received 27 September 2021  
Revised 3 December 2021  
Accepted 17 December 2021

## Abstract

**Purpose** – Semantic retrieval and discovery of datasets published as open data remains a challenging task. The datasets inherently originate in the globally distributed web jungle, lacking the luxury of centralized database administration, database schemes, shared attributes, vocabulary, structure and semantics. The existing dataset catalogs provide basic search functionality relying on keyword search in brief, incomplete or misleading textual metadata attached to the datasets. The search results are thus often insufficient. However, there exist many ways of improving the dataset discovery by employing content-based retrieval, machine learning tools, third-party (external) knowledge bases, countless feature extraction methods and description models and so forth.

**Design/methodology/approach** – In this paper, the authors propose a modular framework for rapid experimentation with methods for similarity-based dataset discovery. The framework consists of an extensible catalog of components prepared to form custom pipelines for dataset representation and discovery.

**Findings** – The study proposes several proof-of-concept pipelines including experimental evaluation, which showcase the usage of the framework.

**Originality/value** – To the best of authors' knowledge, there is no similar formal framework for experimentation with various similarity methods in the context of dataset discovery. The framework has the ambition to establish a platform for reproducible and comparable research in the area of dataset discovery. The prototype implementation of the framework is available on GitHub.

**Keywords** Dataset, Discovery, Search, Framework, Similarity, Knowledge graph

**Paper type** Research paper

## 1. Introduction

The number of datasets available on the web increases tremendously. For example, the number of datasets published by public authorities in European countries increased from 880k datasets in August 2019 [1] to 1140k datasets in November 2021 [2]. Also Google *observed an explosive growth in the number of available datasets in recent years* according to [Benjelloun et al. \(2020\)](#).

---

© Martin Nečaský, Petr Škoda, David Bernhauer, Jakub Klímek and Tomáš Skopal. Published by Emerald Publishing Limited. This article is published under the Creative Commons Attribution (CC BY 4.0) licence. Anyone may reproduce, distribute, translate and create derivative works of this article (for both commercial and non-commercial purposes), subject to full attribution to the original publication and authors. The full terms of this licence may be seen at <http://creativecommons.org/licenses/by/4.0/legalcode>.

This work was supported by the Czech Science Foundation (GAČR), Grant Number 19-01641S.



Data Technologies and  
Applications  
Emerald Publishing Limited  
2514 9288  
DOI 10.1108/DTA-09-2021-0261



---

---

## DTA

Although there exist dataset catalogs providing search for datasets, their retrieval features are restricted to simple keyword search based on textual metadata recorded in the catalog. These simple search methods presume that their users, the data consumers, know exactly what they are searching for and which search query leads to the expected results. However, this assumption is usually not valid, and, in principle, it neglects the very purpose of the catalogs. When users know which datasets they are searching for, they usually also know who publishes a dataset and how the publisher titles the dataset. With this knowledge, it is quite straightforward to locate a dataset on the publisher's website using a generic search engine. The genuine purpose of data catalogs emerges when users do not exactly know which datasets they are searching for and how to find them. This is a usual situation where the users know only a few keywords and topics that roughly characterize the needed data. The problem of missing information about data is inherently related to the big data phenomenon and is generally discussed as the problem of data findability by [Zezula \(2015\)](#). In their studies, [Gregory et al. \(2020a\)](#), [Koesten \(2018\)](#) and [Degbelo \(2020\)](#) show that users typically need to search for more than a single, isolated dataset. Typically, the users wish to find multiple datasets similar to each other in some way, and this is where the pure metadata-based search methods come up short. The studies also show that dataset discovery depends on the context of the user's needs and discovery tasks. Various works such as [Fernandez et al. \(2018\)](#), [Zhang and Balog \(2018\)](#) and [Mountantonakis and Tzitzikas \(2018\)](#) also show that dataset content can be important for building dataset discovery services. Therefore, it is not easy to construct a dataset discovery service on top of a single similarity discovery method. It is necessary to be able to experiment with various combinations of different methods and compare them. This leads us to the following research questions we try to solve in this paper:

- RQ1.* How can we support such experiments with different similarity dataset discovery methods?
- RQ2.* How can we support combining the methods to more complex pipelines for computing dataset similarities?
- RQ3.* How can we evaluate and compare different pipelines?

In this paper, we introduce a modular framework for rapid experimentation with methods for similarity-based dataset discovery, using the perspective of software engineering. We are aware that the development of an ultimate and universal method for dataset discovery would be an infeasible effort. This is based on our previous work – [Škoda et al. \(2019\)](#), [Skopal et al. \(2021\)](#) – where we already experimented with various similarity discovery methods. We have measured them on various real search scenarios, and we showed that none of the evaluated methods performs best on all the scenarios. In this paper, we do not propose yet another method. Instead, we focus on answering the research question above by proposing a framework for experiments with various dataset similarity methods.

Therefore, the framework is not proposed as a complete solution to particular dataset discovery problems, but it should rather act as an extensible modular toolbox for experimentation with various dataset discovery pipelines, including future ones. It supports experimentation by providing a predefined and extensible set of compatible components which can be combined to more complex pipelines which can then be measured, evaluated and compared. Although the framework is designed as generic and extensible, its retrieval model is based on the similarity search paradigm that proved to be an effective general mechanism for retrieval of complex data. Another feature of the framework is its presumption of external knowledge in the process of dataset discovery, which is essential to retrieval using different dataset contexts. In [Škoda et al. \(2020\)](#), we proposed a framework for evaluation of dataset discovery methods. This paper focuses not only on the evaluation but also on the experimentation with the methods and their combinations.

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

### 1.1 Paper contributions

We identify four contributions in this paper:

- (1) The modular and extensive architecture of the framework provides a formal meta-model solution under which particular steps or methods can be easily connected together into dataset discovery pipelines.
- (2) The framework includes a proof-of-concept component catalog that could be used as a predefined source for assembling discovery pipelines.
- (3) We demonstrate the framework benefits using a proof-of-concept implementation and evaluation of several pipelines constructed within the framework.
- (4) The extensible architecture of the framework could provide other scientists both formal and system platforms for reproducible and repeatable research in the area of dataset discovery.

Similarity-based dataset discovery framework

---

The rest of the paper is organized as follows. In [Section 2](#), we give an overview of the dataset discovery area and various discovery methods with the main focus on similarity-based methods. We also show that there is a lack of a framework introduced in this paper. In [Section 3](#), we introduce the framework. We present its layered architecture and describe its layers in detail. In [Section 4](#), we prove the viability of our approach using a proof-of-concept where we construct concrete pipelines for measuring dataset similarity and its presentation to users willing to discover datasets. We conclude in [Section 5](#).

## 2. Related work

### 2.1 Importance of dataset discovery

Finding related datasets, or shortly dataset discovery, is one of the important tasks in data integration as shown by [Miller et al. \(2018\)](#). [Chapman et al. \(2020\)](#) recognize dataset discovery as a research field with its unique technical challenges and open questions. Large companies such as Google develop their own dataset search techniques and solutions ([Brickley et al., 2019](#)). New solutions for dataset search in specific domains started to appear recently. For example, [Chen et al. \(2018\)](#) introduced *Datamed*, which is an open source discovery index for finding biomedical datasets. The field of dataset discovery is not studied only from the technical point of view but also from a more social point of view. [Gregory et al. \(2020a\)](#) investigate how researchers discover data they need for their research projects on the base of the large survey among researchers (1,677 respondents from 105 countries). [Gregory et al. \(2020b\)](#) investigate the same problems by analyzing existing research literature and interviewing scientists who need to discover datasets for their work. [Koesten \(2018\)](#) interviewed 20 data professionals asking them questions on how they search for datasets. These recent studies and surveys show that dataset discovery is an important problem which needs further research. [Gregory et al. \(2020a, b\)](#) conclude that dataset search engines could help searchers looking for data outside their domain to better identify new possible sources of data. [Gregory et al. \(2020a\)](#) moreover show that data needed for a research task can be diverse data from different sources of various types. [Degbelo \(2020\)](#) formulate 27 open data user needs as a synthesis of current findings from recent literature focusing on smart city data. They structure the statements to 10 categories. One of the categories is called *serendipitous resource discovery (SRD)* which involves user questions such as “Are there datasets that I never thought of, that could also be relevant to my tasks?” [Degbelo \(2020\)](#) emphasize the need of cross-linking to other datasets related to the dataset being looked at by the user.

### 2.2 Dataset discovery techniques

All the studies emphasize the role of quality metadata for dataset findability, while [Chapman et al. \(2020\)](#) point out that available metadata do not always describe what is actually in a

---

---

DTA

dataset and whether a described dataset fits for a given task. [Gregory et al. \(2020a, b\)](#) and [Koesten \(2018\)](#) confirm that dataset discovery is highly contextual depending on the current user's task. The studies show that this contextual dependency must be reflected by the dataset search engines. This makes the task of dataset discovery harder as it may not be sufficient to search for datasets only by classical keyword-based search. More sophisticated approaches which are able to search for similar or related datasets could be helpful in these scenarios. As shown by [Chapman et al. \(2020\)](#) and [Miller et al. \(2018\)](#), many existing dataset discovery solutions are based on simple keyword query search. This is what is typically implemented in open data portals such as NODC or EDP. There are also open data portal mash-ups. For example, EDP collects metadata records about open datasets from national portals of individual member states and provides search features across the whole European Union. Recent works further extend these basic approaches. [Brickley et al. \(2019\)](#) describe Google Dataset Search. The authors explain in the paper how dataset metadata is crawled from the web and cleansed. The metadata is then mapped to the Google's knowledge graph, which is then used for dataset duplicates detection and for dataset discovery. [Chen et al. \(2020\)](#) enrich metadata records with labels based on the dataset content. [Chapman et al. \(2020\)](#) describe the whole dataset discovery process comprising querying for datasets, query processing resulting in a list of datasets, result handling and its presentation. It also surveys recent techniques for these individual steps.

### *2.3 Similarity-based dataset discovery techniques*

In this paper, we are interested in similarity-based dataset discovery techniques. Dataset similarity can be used either during query processing where the query result can be enriched with similar datasets or for result handling and presentation where similarity of retrieved datasets can be used to group datasets in the presentation or to enable exploration of retrieved datasets in case of large results. For example, EDP offers a discovery feature based on dataset similarity besides the basic keyword query search. For a dataset found by the keyword query search, similar datasets are also offered to the user. According to source code published at GitLab [3], the portal uses TFSH presented by [Oliver et al. \(2013\)](#). Firstly, they concatenate the title and description of the dataset. The locally sensitive hash is constructed from the concatenated string, which should produce a similar hash for a similar dataset, and these hashes are compared. Originally, the method was introduced by [Dutkowski and Schramm \(2015\)](#). It was implemented as a technique for searching for duplicate or almost equal datasets, ignoring typing errors.

Similarity-based dataset discovery is discussed in the recent survey by [Chapman et al. \(2020\)](#). It discusses techniques to extend a table by discovering tables through table similarity based on tabular schema similarity (e.g. [Das Sarma et al., 2012](#), [Yakout et al., 2012](#)) and semantic similarity using embedding approaches (e.g. [Zhang and Balog, 2018](#)). These can be considered also as dataset discovery techniques because a table is just a special case of a dataset. Several novel techniques for similarity dataset discovery have been proposed in literature in the last few years. [Fernandez et al. \(2018\)](#) propose Aurum. It is a system to build, maintain and query an enterprise knowledge graph (EKG) which represents datasets and their structural elements, for example, table columns, as nodes and relationships between them as edges. A relationship between two structural elements may represent content similarity, schema similarity, for example, similarity of names of the columns, or key/foreign key pairs defined in the dataset schemas. The paper introduces an efficient model which exploits EKG. Moreover, the introduced technique requires only a linear passage through datasets to build EKG. Dataset discovery is then performed on top of EKG. When a user selects a dataset, the tool offers other relevant datasets through the relationships in EKG. [Bogatu et al. \(2020\)](#) propose a technique based on content and schema similarity. For schema

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

Similarity-based dataset discovery framework

---

similarity, the approach considers similarity of column names. For content similarity, the approach considers various similarity models, for example, based on value embedding. [Mountantonakis and Tzitzikas \(2020\)](#) propose union and complement metrics between RDF datasets. The metrics are content-based and computed directly on the RDF triples forming the datasets. Several papers propose dataset similarity techniques based on metadata similarity. [Altaf et al. \(2019\)](#) describe a method which enables to measure similarity between datasets on the base of papers citing the datasets and a citation network between datasets. [Degbelo and Teka \(2019\)](#) evaluate four different metadata-based models for searching spatially related datasets, that is, datasets which are related because of the same or similar spatial area covered. The first model is a full-text search model. The second one parses and geocodes user's query. The other two models map user's query to knowledge graphs, WordNet ([Fellbaum, 2005](#)) and ConceptNet ([Speer et al., 2017](#)), enrich the query with the neighborhoods from these knowledge graphs and use the result for the full-text search.

There are also works which focus on methods useful for datasets published as Linked Data ([Berners-Lee, 2006](#)). For example, [Mountantonakis and Tzitzikas \(2018\)](#) introduce content-based metrics for measuring connectivity between datasets using links and shared entities between the datasets. [Wagner et al. \(2014\)](#) also use shared entities to define similarity between datasets and extend this approach also to clusters of similar entities. The metrics can be then used also for measuring similarity between datasets. [Ellefi et al. \(2016\)](#) present a dataset recommendation approach which identifies linking candidates based on the presence of schema overlap between datasets. [Ellefi et al. \(2016\)](#) introduce a dataset recommendation method based on cosine similarity of sets of concepts present in datasets. Similarly, [Martins et al. \(2016\)](#) recommend datasets based on the similarity of resource labels present in the datasets. Then they rank the recommended datasets based on their TF-IDF score and their coverage of labels present in the original dataset. [Leme et al. \(2013\)](#) present a probabilistic Bayesian classifier for dataset recommendation. Such recommendation techniques can also be used for dataset discovery services as they recommend similar or related datasets.

### 2.4 Data catalogs and metadata

Any dataset discovery method depends on the ability to find and access available datasets. To make a dataset available and accessible, the current practice is to describe it with metadata which is published in a data catalog. A typical data catalog consists of a database of dataset metadata records and a pair of query interfaces, one for machines, for example, a SPARQL endpoint, and one for humans, for example, a user interface, the latter typically using the former. Examples of such data catalogs are the official portal for European data (EDP) [4], the Czech National Open Data Catalog (NODC) [5] described by [Klímek \(2019\)](#), or the US government's open data portal [6]. In addition, there are many more data catalogs within enterprises. Those typically contain non-open data, but otherwise, they work in an identical way as the open data catalogs.

For dataset metadata, there is the DCAT W3C Recommendation ([Browning et al., 2020](#)), a vocabulary specifying the metadata fields and their representation based on RDF ([Cyganiak et al., 2014](#)). For data portals within the European Union, there is an application profile of DCAT called DCAT-AP [7], further specifying the metadata fields, improving metadata interoperability. Furthermore, there are additional application profiles of DCAT-AP for individual countries and use cases.

According to DCAT, a dataset is "A collection of data, published or curated by a single agent, and available for access or download in one or more representations [8]." The dataset metadata record according to DCAT contains various kinds of fields. Some fields are textual, such as dataset title, dataset description and keywords describing a dataset. Other fields, such as, in the case of DCAT-AP, update frequency, file format or language used, contain values from code lists from the EU Vocabularies [9].

---

---

DTA

### *2.5 Modularity and reusability of existing solutions*

Based on the dataset metadata, data catalogs offer dataset search and retrieval functionalities for their users. As the studies mentioned above show, the users typically need to search for more than a single, isolated dataset. Typically, the users wish to find multiple datasets related to each other in some way, and this is where the pure metadata-based search methods present in today's data catalogs come up short. The studies also show that dataset discovery depends on the context of the user's needs and discovery tasks. Therefore, it is not easy to construct a dataset discovery service on top of a single similarity discovery technique. It is necessary to be able to experiment with various combinations of various techniques and compare them. A framework supporting such experimentation, possibly providing predefined components which can be combined and compared, would be helpful for anyone who needs to find a viable solution for their domain and contexts. The framework shall support extracting metadata from data catalogs, processing the extracted metadata and computing similarities between described datasets, and presenting the resulting similarities to human users. The framework shall be modular. It shall provide a set of various components for metadata extraction, metadata processing, similarity measuring and also various human interfaces for presenting the similarities to human users. One could say that any ETL framework shall be sufficient. An ETL framework, as explained by [El-Sappagh et al. \(2011\)](#), is a framework for defining data manipulation processes, each starting with extracting data from its sources, its various transformations and loading the result to a database. The database is then used for various purposes, including presentation of the data to human users. However, what is important is not a particular ETL framework but a set of ETL components prepared for a certain task. In our case, this task is dataset discovery and measuring similarity of datasets in particular. From this point of view, a given technique for metadata extraction or similarity computation shall be packed as a component. Moreover, the components need to have standardized and compatible outputs and inputs so that it is possible to combine them and introduce new components for experimentation.

Unfortunately, the only standardization in the existing solutions are the DCAT and DCAT-AP metadata standards described above. The existing techniques for metadata extraction and similarity computation surveyed in the subsections above are isolated and incompatible solutions without open and modular architecture. This makes them hard to combine and compare. To the best of our knowledge, the existing works in the field of dataset discovery do not address this problem, and our paper is the first which addresses the problem of defining such open and modular architecture for dataset discovery. For example, [Brickley et al. \(2019\)](#) describe the architecture of the Google dataset search solution. The architecture comprises components for metadata extraction; metadata manipulation including its normalization, cleansing and deduplication; and a user interface for searching datasets using the cleansed metadata. The EDP architecture comprises components for harvesting metadata from national data catalogs in EU countries, components for metadata manipulation including a component for computing dataset similarities using TLSH. [Fernandez et al. \(2018\)](#) describe the architecture of Aurum comprising a component for extracting dataset profiles of tabular datasets, a component for building relationships between datasets and a component for querying the resulting search index by human users. We can see that all these works somehow describe a dataset discovery architecture, but none of them defines it as open and modular architecture which enables to reuse and combine various solutions.

### **3. Architecture of dataset discovery framework**

In this section, we introduce our dataset discovery framework. We present its basic concepts, its architecture and conceptual model of its components. As can be seen from the overview of dataset similarity techniques in [Section 2](#), the framework needs to accommodate various

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

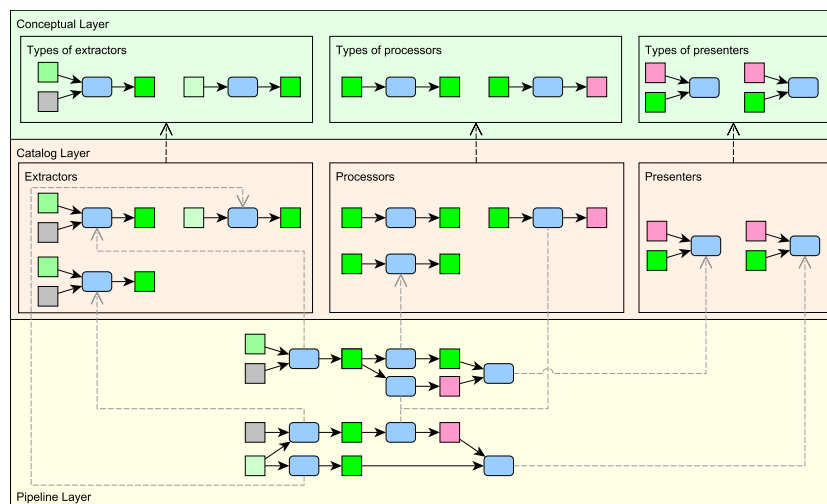
solutions, from simple metadata similarity techniques to techniques employing knowledge graphs. The framework must be modular so that it is possible to combine the techniques. To achieve modularity, it is necessary to encapsulate various techniques as framework components with well-defined inputs and outputs.

The basic concepts of our framework are *similarity production pipelines* and *similarity presentations*. A similarity production pipeline is a data processing pipeline which takes a data catalog as an input together with other possible inputs and produces similarity of datasets in the catalog. It comprises various components which either extract data from external sources, process data or present data to end users. A similarity presentation then presents similarity to human users through a user interface. It comprises a presentation component which takes an output of one or more similarity production pipelines as an input. The framework supports experts in building their own similarity production pipelines and similarity presentations. The experts can experiment with various combinations of components suitable for their domain and use cases and compare them.

The architecture is depicted in Figure 1. It comprises three layers of abstraction: pipeline (yellow), catalog (orange) and conceptual (green). The pipeline layer comprises concrete assembled similarity production pipelines, each complemented with a similarity presentation. A similarity production pipeline connects various production components together, defines in which order the components are executed and how the components pass their outputs. A similarity presentation then takes the outputs of one or more pipelines and presents them to human users using a presentation component. Two sample similarity production pipelines are outlined in Figure 1. Their components are depicted as blue boxes with rounded corners. Component inputs and outputs are depicted as squares in different colors. Each pipeline ends with a similarity presentation.

The catalog layer comprises a *catalog* of supported components which can be placed in pipelines and used for presentations. A component specifies expected input data, how the input data is processed and what output is provided. The sample catalog depicted in Figure 1 comprises eight different components, each depicted as a blue box with rounded corners and with their expected inputs and their outputs as colored squares. The gray dashed arrows

Similarity-based dataset discovery framework



**Figure 1.** The architecture of the dataset similarity discovery framework

---

---

DTA

oriented from the pipeline layer to the catalog layer depict how the individual components chosen from the catalog layer are placed in the similarity pipelines or used for similarity presentation.

The conceptual layer is an abstract layer which defines the *conceptual model* of similarity production pipelines and similarity presentations. The conceptual model defines supported *component types*. Each component in the catalog is an instance of a component type from the conceptual model. A component type specifies an action but it is abstracted from a specific method and specific form of input and output data. The input and output data specification is abstracted to conceptual *entity types* which describe possible inputs and outputs at the conceptual level without technical details. [Figure 1](#) outlines the conceptual model. It shows that the conceptual model defines component types with their input and output entity types. From the left to the right, each component type depicted as a blue box has one or more instances at the catalog layer. The first component type depicted on the left of the conceptual layer has two components as its instances in the catalog layer, the second has one component as an instance and so forth. Each component type instance, that is, a specific component, adheres to its component type. It means that it provides a specific implementation of the action defined by the component type. To perform the action, it takes inputs and provides outputs which adhere to the input and output types, respectively, predefined by the component type.

In this section, we describe each layer in detail. We describe the conceptual model as a fixed set of component types, with fixed set of entity types as their inputs and outputs. We then define how components in a catalog shall be defined. However, we do not define a fixed catalog. A definition of a specific catalog of components is up to a specific implementation of the framework and a given catalog can be arbitrarily extended with new components. However, any component must be an instance of one of the types from the conceptual level which is fixed. This section ends with a formal definition of similarity pipelines and presentations which shows how components from a catalog can be combined together. A specific component catalog, specific similarity pipelines and presentations and how they can be executed are presented in [Section 4](#).

### 3.1 Conceptual layer

The conceptual layer defines the conceptual model for similarity production pipelines and similarity applications. It ensures compatibility of components in pipelines as it defines supported entity and component types. Each component type represents some action with a given input and output. This defines not only supported actions but also possible ordering of components in pipelines because it is necessary that their inputs and outputs are compatible.

**3.1.1 Entity types.** An entity type represents types of data entities handled during similarity production pipelines execution. The conceptual model of possible entity types is depicted in [Figure 2](#). We distinguish the following entity types in this paper:

- (1) **Dataset** represents a dataset  $D$  in a collection of datasets  $\{D_1, \dots, D_n\}$  or shortly  $\{D_i\}_1^n$  or  $\{D_i\}$  when  $n$  is not important. The goal of a similarity production pipeline is to produce similarities of datasets in  $\{D_i\}$  among each other.
- (2) **Knowledge** represents external knowledge which is used in a similarity production pipeline. We distinguish three subtypes in this paper:
  - **KnowledgeGraph** represents external knowledge structured as a graph  $G = (N, E)$  where  $N$  is a set of nodes,  $E \subseteq N \times P \times (N \cup L)$  is a set of edges,  $P$  is a set of all possible node properties and  $L$  is a set of all possible literal values. A literal value is simply a string. For an edge  $(n, p, o)$ ,  $n$  is called subject,  $p$  is called predicate and  $o$  is called object. An edge  $(n, p, o)$  s.t.  $o \in N$  is called object edge.

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

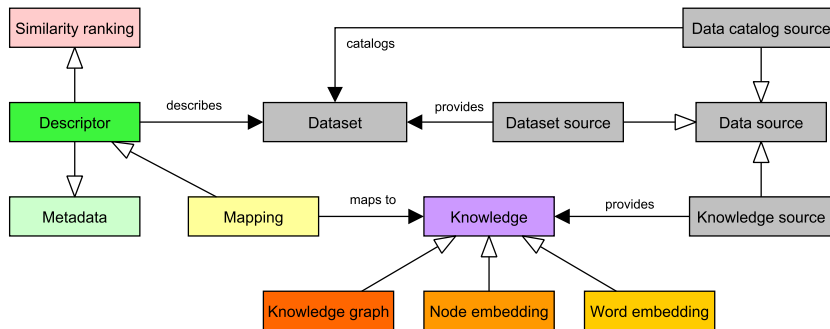
Similarity-based dataset discovery framework

- An edge  $(n, p, o)$  s.t.  $o \in L$  is called literal edge. A reader may notice that the knowledge graph model corresponds to the RDF data model (Cyganiak et al., 2014), but this is not important at the conceptual layer.
- `NodeEmbedding` represents external knowledge structured as a node embedding  $\mathcal{N}_G : N \rightarrow \mathbb{R}^n$  of a knowledge graph  $G$  where  $N$  is the set of nodes of  $G$ .
  - `WordEmbedding` represents external knowledge structured as a word embedding  $\mathcal{W} : L \rightarrow \mathbb{R}^n$  where  $L$  is the set of all possible literal values.
- (3) `Descriptor` represents a descriptor of a dataset  $D$ . A descriptor of  $D$  is any collection of data which describes  $D$ , for example, vector, a histogram, time-series, a set of descriptors or a combination of other dataset descriptors. We distinguish three subtypes:
- `Metadata` represents a descriptor which describes  $D$  using metadata. Currently, we consider only one possible kind of metadata descriptors which correspond to the DCAT standard (see Section 2.4).
  - `Mapping` represents a descriptor which describes  $D$  by mapping its representation  $\{d^1, \dots, d^m\}$  to an external knowledge. A representation  $\{d^1, \dots, d^m\}$  of  $D$  is any set of elements which characterize  $D$ . It is a generic concept which may comprise elements of the data schema of  $D$ , all or chosen data elements from the content of  $D$ , and it can also be  $D$  itself. A mapping of  $D$  to an external knowledge comprises particular mappings of the elements of the representation.

If the external knowledge is a knowledge graph  $G$ , then the mapping is a set  $\{(D, d^i, \{n_1^i, \dots, n_{k_i}^i\})\}_{i=1}^m$  which maps each element  $d^i$  of a representation  $\{d^1, \dots, d^m\}$  of  $D$  to a set of nodes  $\{n_1^i, \dots, n_{k_i}^i\}$  from  $G$ .

If the external knowledge is an embedding  $E$ , then the mapping is a set  $\{(D, d^i, \{v_1^i, \dots, v_{k_i}^i\})\}_{i=1}^m$  which maps each element  $d^i$  of a representation  $\{d^1, \dots, d^m\}$  of  $D$  to vectors  $\{v_1^i, \dots, v_{k_i}^i\}$  where each vector is a result of applying the embedding  $E$  and some mapping logic represented by a concrete mapping component.

- `SimilarityRanking` represents a descriptor which describes  $D$  in a collection  $\{D_i\}_1^n$  using similarity of  $D$  with other datasets in  $\{D_i\} \setminus \{D\}$  as a pair  $(D, \{(D_1, s_1), \dots, (D_n, s_n)\})$  where  $s_i = \text{similarity}(D, D_i)$  for some function measuring similarity of datasets.



**Figure 2.** The conceptual model of possible entity types for similarity production pipelines



DTA

- (4) `DataSource` represents an external source of a certain kind of data. An instance of a data source type is a concrete data source which can be accessed through an API to extract data. We distinguish the following data source types.
- `DataCatalogSource` represents a data catalog source  $DS_{cat}$  which provides an API for extracting metadata descriptors. Currently, we consider only data catalog sources providing DCAT metadata descriptors through a SPARQL endpoint or for a bulk download.
  - `KnowledgeSource` represents an external knowledge source  $DS_k$  which provides an API for extracting external knowledge. It can be a SPARQL endpoint for extracting an RDF knowledge graph, a bulk download of a word embedding or a set of documents on which a word embedding can be trained.
  - `DatasetSource` represents a datasets source  $DS_d$  which enables downloading a dataset described by a metadata descriptor. It can simply be the web where the metadata descriptor provides a URL for downloading the content of the dataset.

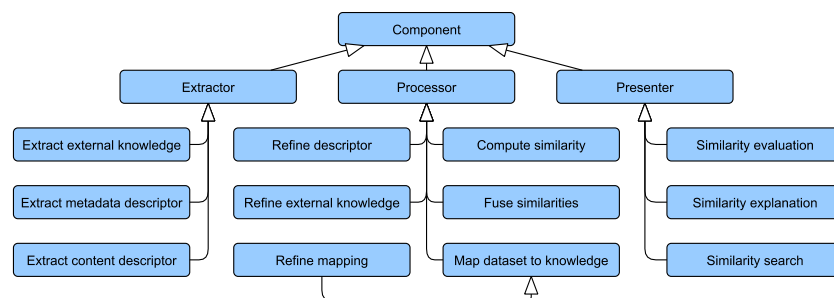
**3.1.2 Component types.** A component type represents an operation. It is abstracted from a concrete algorithm for that operation and its implementation. The concrete algorithm and implementation is provided by a concrete component which is an instance of the component type.

Formally, an operation  $operation(in_1, \dots, in_m) : out$  takes input parameters  $\{in_1, \dots, in_m\}$  and produces an output  $out$ . An input  $in_j$  is a regular expression  $T, T^+, T^*$  or  $T^?$  meaning that one instance, one or more instances, zero or more instances, or zero or one instance of an entity type  $T$  is provided on the input. The same is for the operation output.

The top-level component type is `Component`. It represents all possible components. There are three more specific subtypes: `Extractor`, `Processor` and `Presenter`. The full list of supported component types with their inheritance hierarchy is shown in [Figure 3](#).

**3.1.2.1 Extractors.** `Extractor` represents an extractor. An extractor is a component which performs an operation of extracting data from a data source. The conceptual model of extractors is shown in [Figure 4](#). We distinguish the following subtypes of `Extractor`:

- (1) `ExtractExternalKnowledge` represents components performing  $extract_k(KnowledgeSource) : Knowledge$  which extracts an external knowledge from a given external knowledge source.
- (2) `ExtractMetadataDescriptor` represents components performing



**Figure 3.** The conceptual model of possible components types for similarity production pipelines

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

Similarity-based dataset discovery framework

$extract_{meta}(DataCatalogSource) : Metadata^+$   
which extracts metadata about datasets from a given data catalog source.

- (3) ExtractContentDescriptor represents components performing

$extract_{desc}(DatasetSource, Metadata^+) : Descriptor^+$   
which extracts from a dataset source a descriptor for each dataset with a provided metadata descriptor.

3.1.2.2 Processors. Processor represents a processor. A processor is a component which performs an operation for processing input entities to output entities. The conceptual model of processors is shown in Figure 5.

- (1) RefineDescriptor represents components performing

$process_{refd}(Descriptor^+) : Descriptor^+$   
which transforms input descriptors to output descriptors.

- (2) RefineExternalKnowledge represents components performing

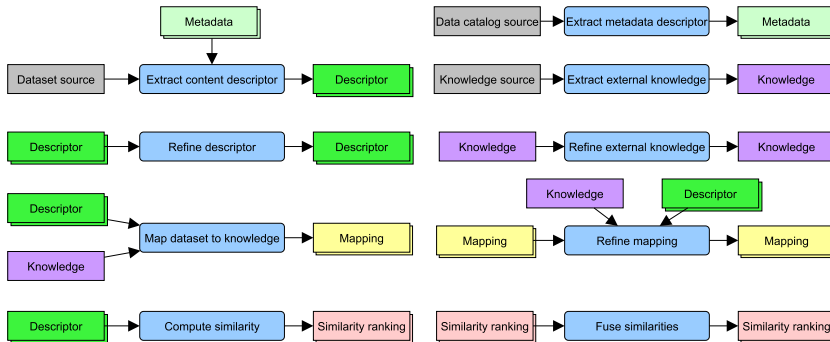
$process_{refk}(Knowledge) : Knowledge$   
which transforms an external knowledge to another external knowledge, for example, transforming literal objects of literal edges to other literal objects in a knowledge graph, removing some object edges from a knowledge graph or applying some vector operation on the vectors in an embedding.

- (3) MapDatasetToKnowledge represents components performing

$process_{map}(Descriptor^+, Knowledge) : Mapping^+$   
which maps a set of datasets described by the input descriptors to an input external knowledge. The resulting output mapping is created on the base of the input descriptors, and it maps each dataset to the external knowledge.

- (4) RefineMapping represents components performing

$process_{refm}(Mapping^+, Descriptor^*, Knowledge^?) : Mapping^+$   
which transforms an input mapping of datasets to an external knowledge to another mapping of the same datasets to the same external knowledge or another external knowledge specified as an optional input. The new mapping is created using the optional input descriptors of the datasets.



**Figure 4.** The conceptual model of possible extractor component types for similarity production pipelines

**Figure 5.** The conceptual model of possible processor component types for similarity production pipelines

DTA

- (5) ComputeSimilarity represents components performing

$$process_{sim}(Descriptor^+) : SimilarityRanking^+$$

which computes a similarity of each dataset described by an input descriptor with other datasets described by the other input descriptors. The component uses the input descriptors to compute the similarity.

- (6) FuseSimilarities represents components performing

$$process_{fuse}(SimilarityRanking^+) : SimilarityRanking^+$$

which performs multimodal fusion of two or more input similarities for a dataset to a single similarity. For each dataset with specified similarities on the input, an output similarity is produced.

3.1.2.3 Presenters. Presenter represents a presenter. The conceptual model of presenters is shown in Figure 6. A presenter is a component which uses the products of dataset similarity pipelines to present them to human users and provide them with some functionality.

- (1) SimilarityEvaluation represents components performing

$$present_{eval}(SimilarityRanking^+, Descriptor^+)$$

which presents dataset similarities to a human user who evaluates the similarities. It uses input descriptors of the datasets for presenting the datasets.

- (2) SimilaritySearch represents components performing

$$present_{sear}(SimilarityRanking^+, Descriptor^+)$$

which enables a human user to choose a dataset and then shows datasets similar to the chosen one on the base of the input similarities. It uses input descriptors of the datasets for presenting the datasets.

- (3) SimilarityExploration represents components performing

$$present_{expl}(SimilarityRanking^+, Descriptor^+, Knowledge^*)$$

which explains similarities between datasets to a human user. For the explanation, it uses the input similarity, dataset descriptors and optionally also external knowledge of different kinds. Among the input descriptors, there are also mappings of the datasets to the external knowledge if provided on the input. Descriptors may be used for presentation as well as explanation purposes.

### 3.2 Catalog layer

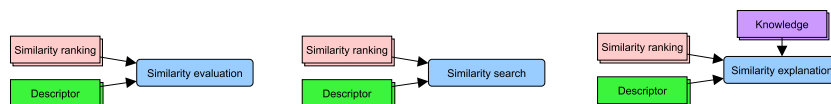
The catalog layer contains the catalog of components experts may use to build their similarity production pipelines and similarity presentations. Each component  $c$  in the catalog is an instance of a component type  $T$  from the conceptual level which we write as  $c \in T$ . A component  $c$  is described in the catalog with a record using the following structure:

*type*: specifies the type of  $c$  from the conceptual layer

*input*: specifies the inputs of  $c$  from the conceptual layer using the entity types from the conceptual layer

*output*: specifies the outputs of  $c$  from the conceptual layer using the entity types from the conceptual layer

**Figure 6.**  
The conceptual model of possible presenter component types for similarity production pipelines



## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

*description*: describes verbosely the behavior of  $c$

*implementation*: specifies a link or links to the source code and documentation of one or more implementations of  $c$

*configurable*: specifies whether and how the behavior of  $c$  can be configured

*configurations*: if  $c$  is configurable there is a list of predefined configurations; for each predefined configuration, a name and configuration specification is provided

Similarity-  
based dataset  
discovery  
framework

---

As can be seen from the structure, a component has one or more executable implementations. The implementations consume the same conceptual entities on the input, produce the same conceptual entities on the output but they technically differ in data formats used to express the inputs and outputs. Furthermore, a component's behavior can be further specified by configuring it. In case of a configurable component, some concrete configurations may be predefined in the catalog. These configurations are then used by their name in pipeline specifications in the next section which makes pipeline specifications clearer.

### 3.3 Pipeline layer

The components in the catalog layer represent concrete operations with their concrete implementations. The pipeline layer contains concrete similarity production pipelines and similarity presentations which put components together. In this section, we introduce a formal algebraic model of similarity production pipelines and similarity presentations. A pipeline expressed using the algebraic model cannot be directly executed. The algebraic expression must be interpreted and translated to an executable script as described in Section 4.4. The algebraic model enables one to define a pipeline without implementation details so that it is more suitable for comparing different pipelines. Using the algebraic model, it is easier to see what are the conceptual differences between given pipelines.

A *pipeline fragment*  $F$  is an expression.

- (1)  $c(F_1, \dots, F_m)$  where  $c \in \mathbb{T}$  is a component of type  $T$  performing an operation  $operation(in_1, \dots, in_m) : out$  and  $F_1, \dots, F_m$  are pipeline fragments s.t.  $\forall j \in \{1, \dots, m\}$   $F_j$  is compatible with the input parameter  $in_j$  of *operation*.  $F_j$  is *compatible* with an input parameter  $in_j$  of a component type  $T$  iff the type of the result of execution of  $F_j$  is the same as the type of  $in_j$ .
- (2)  $F_1 \cup F_2$  where  $F_1, F_2$  are pipeline fragments with their outputs being of the same entity type, or
- (3)  $s$  where  $s \in DataSource$ .

An *execution* of  $F = c(F_1, \dots, F_m)$  means executing the operation defined by  $c$  with the results of executing  $F_1, \dots, F_m$  passed as parameters. An execution of  $F = F_1 \cup F_2$  means creating the set union of the results of executing  $F_1$  and  $F_2$ . An execution of  $F = s$  where  $s \in DataSource$  is undefined.

A pipeline fragment  $F$  is a *similarity production pipeline* iff  $F = c(F_1, \dots, F_m)$  where  $c \in ComputeSimilarity \cup FuseSimilarity$ . A pipeline fragment  $F$  is a *similarity presentation* iff  $F = c(F_1, \dots, F_m)$  where  $c \in Presenter$ .

The introduced algebraic model does not allow for specification of component configurations. If a component described in the catalog is configurable, its configurations must be defined and named in the catalog, and only these named configurations can be used in pipeline specifications.

### 4. Dataset discovery framework proof-of-concept

In this section, we demonstrate the dataset discovery framework on a concrete component catalog and similarity production pipelines constructed using the components from

---

DTA

the catalog. To demonstrate the framework, we apply the pipelines on a specific open data catalog and external knowledge. We also show examples of similarity presenters which we used for user evaluation of the resulting similarities, for example in [Škoda et al. \(2020\)](#).

#### 4.1 Data and external knowledge used for proof-of-concept

For the framework demonstration purposes, we work with metadata from the Czech National Open Data Catalog (NODC) [10] described in [Klímek \(2019\)](#), which is regularly harvested by the European Data Portal (EDP). This metadata describes open datasets published by public institutions in Czechia, such as the Czech Statistical Office, the city of Prague or the Czech Social Security Administration. For illustration of how the framework can be used, we use the most basic, textual metadata fields: *title*, *description* and *keywords* – which contain textual descriptions of the datasets in Czech. The collection we work with contains approximately 6,600 datasets from 39 publishers.

Besides dataset metadata, we utilize multiple types of external knowledge. We choose to employ Wikidata ([Vrandečić and Kröttsch, 2014](#)), a collaboratively edited knowledge-base with free access, as a knowledge graph source. In general, the Wikidata model is built around the entities and their relations. The entities may represent concepts or real objects or people. The relations are of various types, but for the framework demonstration we utilize only instance of [11] and subclass of [12] relations. We have also used this model for computing Node2Vec ([Grover and Leskovec, 2016](#)) text (labels) and concept (nodes) embeddings with different hyperparameters. For comparison, we utilize two standard word embeddings: Word2Vec ([Mikolov et al., 2013](#)) model trained on Czech Wikipedia articles and BERT ([Devlin et al., 2018](#)) using the *BERT-base, multilingual cased* model.

Formally, we define the following entities (see [Section 3.1.1](#)), which will be used in the proof of concept by components in pipelines:

(1) NODC

type: `CatalogDataSource`

description: Snapshot of the DCAT-AP metadata dump from the Czech National Open Data Catalog in the RDF TriG ([Carothers and Seaborne, 2014](#)) file format from 2020-04-20. For practical reasons, datasets of the State Administration of Land Surveying and Cadastre were omitted. It was approximately 120,000 datasets with very similar metadata, which did not add any value for the purpose of the framework demonstration, but unnecessarily increased the time and hardware requirements of the demonstration.

download: `2020.04.20-data.gov.cz-no-cuzk.trig` [13] ([Klímek and Škoda, 2021a](#)).

(2) W2V [CSWiki]

type: `KnowledgeSource`

description: Word2Vec model for word embedding trained on Czech Wikipedia articles.

download: `cswiki-latest-pages-articles.word2vec` [14] ([Bernhauer and Skopal, 2020](#)).

(3) N2V [Wikidata Labels/80/40]

type: `KnowledgeSource`

description: Node2Vec model for word embedding trained on Czech labels of items from the Wikidata knowledge graph, using the `instance of` and `subclass of` directed edges. For this model, length of random walk is 80 nodes and number of random walks per node is 40.

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

- Similarity-based dataset discovery framework
- 
- download: `labels.80.40.d` [15] (Bernhauer and Skopal, 2021d).
- (4) N2V [Wikidata Labels/160/40]  
type: KnowledgeSource  
description: Node2Vec model for word embedding trained on Czech labels of items from the Wikidata knowledge graph, using the `instance of` and `subclass of` directed edges. For this model, the length of the random walk is 160 nodes and the number of random walks per node is 40.
- download: `labels.160.40.d` [16] (Bernhauer and Skopal, 2021c).
- (5) N2V [Wikidata KG/80/40]  
type: KnowledgeSource  
description: Node2Vec model for node (concept) embedding trained on the Wikidata knowledge graph using the `instance of` and `subclass of` directed edges. For this model, the length of the random walk is 80 nodes and the number of random walks per node is 40.
- download: `concepts.80.40.d` [17] (Bernhauer and Skopal, 2021b).
- (6) N2V [Wikidata KG/40/10]  
type: KnowledgeSource  
description: Node2Vec model for node (concept) embedding trained on the Wikidata knowledge graph using the `instance of` and `subclass of` directed edges. For this model, the length of the random walk is 40 nodes and the number of random walks per node is 10.
- download: `concepts.40.10.d` [18] (Bernhauer and Skopal, 2021a).
- (7) BERT  
type: KnowledgeSource  
description: BERT is pretrained model for NLP tasks presented in Devlin *et al.* (2018). All pretrained models are officially available at <https://github.com/google-research/bert>. In our experiments, we have used BERT-base, multilingual cased model.
- download: `multi_cased_L-12_H-768_A-12.zip` [19] (Devlin *et al.*, 2018).
- (8) Wikidata knowledge graph  
type: KnowledgeSource  
description: Dump of Wikidata as available at <https://dumps.wikimedia.org/other/wikidata/>. The dump contains information about Wikidata entities and their relations.
- download: `20181217.json.gz` [20] (Klímek and Škoda, 2021b).

### 4.2 Proof-of-concept component catalog

The full list of available components and links to their implementations can be found in Appendix A. In this section, we list a subset of the components, which is used later by the selected similarity production pipelines in Section 4.3. The components are color-coded

---

DTA

according to the type of their outputs: knowledge (violet), SimilarityRanking (pink), descriptor (green) and mapping (orange).

*4.2.1 Extractors.*

- `extractMetadata`

**type:** ExtractMetadataDescriptor

**input:** catalog  $\in$  DataCatalogSource

**output:**  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$

**description:** Extracts metadata descriptors for all datasets from a given catalog. An extracted descriptor contains a title, description and keywords of a dataset if provided by the catalog.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/extractors/extract-metadata-descriptor/dcat-ap-extractor>

- `getVectorEmbeddingModel`

**type:** ExtractExternalKnowledge

**input:** knowledgesource  $\in$  KnowledgeSource

**output:** knowledge  $\subset$  Knowledge

**description:** Extracts external knowledge as an embedding model (i.e. Word2Vec, Node2Vec, BERT, ...).

**implementation:** There is no implementation as these models can be just downloaded.

**configurable:** Kind of embedding used.

**configurations:**

`getWord2VecModel` - Word2Vec embedding in Gensim format

`getNode2VecModel` - Node2Vec embedding in Gensim format

- `wikidataHierarchyExtractor`

**type:** ExtractExternalKnowledge

**input:** knowledgesource  $\in$  KnowledgeSource

**output:** knowledge  $\subset$  Knowledge

**description:** Extracts hierarchy, made of instance of and subclass of edges, from the Wikidata knowledge graph.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/extractors/extract-external-knowledge/wikidata-hierarchy-extractor>

- `wikidataLabelsExtractor`

**type:** ExtractExternalKnowledge

**input:** knowledgesource  $\in$  KnowledgeSource

**output:** knowledge  $\subset$  Knowledge

**description:** Extracts item labels and aliases from the Wikidata knowledge graph.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/extractors/extract-external-knowledge/wikidata-labels-extractor>

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

### 4.2.2 Processors.

- **projectDescriptor**  
type: RefineDescriptor  
input:  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$   
output:  $\{\text{descriptor}'_i\}_{i=1}^n \subset \text{Metadata}^+$   
description: Performs property projection on input descriptors.  
implementation: <https://github.com/mff-uk/simpipes-components/tree/main/processors/refine-descriptor/json-to-csv>  
configurable: Projected metadata property.  
configurations:
  - projectToTitle - title property
  - projectToDescription - description property
  - projectToKeywords - keywords property
- **concatenate**  
type: RefineDescriptor  
input:  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$   
output:  $\{\text{descriptor}'_i\}_{i=1}^n \subset \text{Metadata}^+$   
description: Refines two textual descriptors into one concatenated descriptor.  
implementation: <https://github.com/mff-uk/simpipes-components/tree/main/processors/refine-descriptor/join>
- **doLemmatization**  
type: RefineDescriptor  
input:  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$   
output:  $\{\text{descriptor}'_i\}_{i=1}^n \subset \text{Metadata}^+$   
description: Refines a textual descriptor by lemmatisation, i.e. transforms each word into its lemma and eliminates stop-words. In this case, the lemmatisation and optional stop-word elimination could be separated, but it is more convenient work with NLP processor just once.  
implementation: <https://github.com/mff-uk/simpipes-components/tree/main/processors/refine-descriptor/udpipe>
- **mapToAverageVector**  
type: MapDatasetToKnowledge  
input:  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$ ,  $\text{knowledge} \subset \text{Knowledge}$   
output:  $\{\text{mapping}_i\}_{i=1}^n \subset \text{Mapping}^+$

Similarity-  
based dataset  
discovery  
framework

---



---

## DTA

---

- description:** Maps textual metadata to provided embedding using the average vector over all words.
- implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/processors/map-dataset-to-knowledge/vectorize>
- configurable:** Kind of descriptor used.
- configurations:**
- `mapTextToAverageVector` - textual descriptor
  - `mapConceptsToAverageVector` - Wikidata concepts
- **instanceToClass**

**type:** RefineMapping

**input:**  $\{\text{mapping}_i\}_{i=1}^n \subset \text{Mapping}^+$ ,  $\text{knowledge} \subset \text{Knowledge}$

**output:**  $\{\text{mapping}_i\}_{i=1}^n \subset \text{Mapping}^+$

**description:** Map entities to their instance of ancestors.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/processors/refine-mapping/instance-to-class>
  - **bagOfWordsMapper**

**type:** MapDatasetToKnowledge

**input:**  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Metadata}^+$ ,  $\text{knowledge} \subset \text{Knowledge}$

**output:**  $\{\text{mapping}_i\}_{i=1}^n \subset \text{Mapping}^+$

**description:** Maps textual metadata to provided textual entities.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/processors/map-dataset-to-knowledge/bag-of-words-mapper>
  - **computeSimilarity**

**type:** ComputeSimilarity

**input:**  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Descriptor}^+$

**output:**  $\{\text{similarityRanking}_j\}_{j=1}^n \subset \text{SimilarityRanking}^+$

**description:** Computes similarity for scalar-based descriptors such as texts, sets of words or vectors.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/processors/compute-similarity/basic>

**configurable:** Kind of similarity used.

**configurations:**

    - `computeJaccardSimilarity` - Jaccard distance, usable for texts and sets of words
    - `computeCosineSimilarity` - Cosine distance, usable for texts and vectors
    - `computeTLSHSimilarity` - TLSH distance, usable for texts

### 4.2.3 Presenters.

- (1) `evaluateExactSize`

**type:** SimilarityEvaluation

**input:**  $\{\text{similarityRanking}_j\}_{j=1}^n \subset \text{SimilarityRanking}^+$ ,  
 $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Descriptor}^+$

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

Similarity-based dataset discovery framework

---

**description:** Evaluates similarity using the provided baseline. For every baseline's dataset, the  $k$ NN query (Papadias, 2009) similarity search is performed.  $k$  is the expected number of similarity datasets. Average ratio of observed and expected is presented to the user.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/presenters/similarity-evaluation/exact-size>

(2) evaluateTopK

**type:** SimilarityEvaluation

**input:**  $\{\text{similarityRanking}_j\}_{j=1}^n \subset \text{SimilarityRanking}^+$ ,  
 $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Descriptor}^+$

**description:** Evaluates similarity using the provided baseline. For every baseline's dataset, the  $k$ NN query similarity search is performed.  $k$  is specified by user through parameter, and it is a constant for each test case. Average ratio of observed and expected is presented to the user.

**Implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/presenters/similarity-evaluation/top-k>

(3) evaluatePRCurve

**type:** SimilarityEvaluation

**input:**  $\{\text{similarityRanking}_j\}_{j=1}^n \subset \text{SimilarityRanking}^+$ ,  
 $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Descriptor}^+$

**Description:** Evaluates similarity using the provided baseline. For every baseline's dataset, the 11-point PR curve (Zhang and Zhang, 2009) is computed and presented to the user.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/presenters/similarity-evaluation/pr-curve>

(4) OpenDataInspectorEvaluation

**type:** SimilarityEvaluation

**input:**  $\{\text{descriptor}_i\}_{i=1}^n \subset \text{Descriptor}^+$ ,  
 $\{\text{similarityRanking}_j\}_{j=1}^n \subset \text{SimilarityRanking}^+$

**description:** OpenDataInspector is a standalone tool. The evaluation module (see Škoda *et al.*, 2020) allows domain experts to evaluate similarity production pipeline results.

**implementation:** <https://github.com/mff-uk/simpipes-components/tree/main/presenters/similarity-evaluation/odin-similarity>

### 4.3 Proof-of-concept pipelines

In this section, we present four similarity production pipelines as examples. The full list of currently available pipelines can be found in Appendix B. The pipelines demonstrate the practical usability of our framework. For example, the first pipeline presented in Section 4.3.1 is an implementation of the solution employed by the EDP (see Section 2). In Section 2, we also

---

DTA

discussed discovery solutions which employed an external knowledge in a form of a knowledge graph, for example, [Brickley et al. \(2019\)](#), The fourth pipeline presented in [Section 4.3.4](#) is an implementation of a dataset discovery solution employing an external knowledge in a form of a knowledge graph. In our case, the external knowledge is the Wikidata knowledge graph. A similar solution is employed by Google in their dataset search architecture with their own knowledge graph as described by [Brickley et al. \(2019\)](#). Various pipelines implementing existing as well as novel solutions can be constructed in our framework. The results of similarity production pipelines can be used, for instance, for their evaluation, both automatically and with assistance of domain experts. Later, in [Section 4.5](#), we show the usage of appropriate presenters.

*4.3.1 TLSH similarity production pipeline.* TLSH similarity production pipeline corresponds to the similarity feature implemented by the EDP. Note that `computeTLSHSimilarity` can be split into hash computation and similarity function computation, but in this case, these two parts are interconnected.

```
computeTLSHSimilarity(  
  concatenate(  
    projectToTitle( extractMetadata(NODC) ),  
    projectToDescription( extractMetadata(NODC) )  
  )  
)
```

*4.3.2 Metadata-based similarity production pipeline.* The metadata-based similarity production pipeline is one of the most straightforward pipelines. It relies on suitable usage of dataset metadata such as title, keywords and description. The metadata is pre-processed by lemmatization and by removing stop-words. Then, it is compared by the Jaccard similarity function.

```
computeJaccardSimilarity(  
  doLemmatization(  
    projectToTitle( extractMetadata(NODC) )  
  )  
)
```

*4.3.3 Text-based similarity production pipeline using Word2Vec embedding.* This similarity production pipeline is an example of text embedding used as external knowledge. The lemmatization phase can be followed by various embedding mappings. In this case, we use a mapping to a Word2Vec model trained on Czech Wikipedia articles. It helps to deal with synonyms and words with similar meaning.

```
computeCosineSimilarity(  
  mapTextToAverageVector(  
    doLemmatization(  
      projectToTitle( extractMetadata(NODC) )  
    ),  
    getWord2VecModel(w2v [CSWiki])  
  )  
)
```

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

*4.3.4 Concept-based similarity production pipeline with additional external knowledge.* In some cases, it can be useful to enhance a similarity model with additional external knowledge. For example, one part of external knowledge can provide information about mapping words to concepts. Another part can embed the concepts into a vector space. In our case, both come from the same database (i.e. Wikidata), but they are processed in different ways. Firstly, we have mapped words and phrases to Wikidata concepts. Secondly, we have trained a Node2Vec model using the Wikidata knowledge graph and applied a concept-to-vector embedding. In contrast with the Word2Vec embedding, the external knowledge in this pipeline is built using the Wikidata concept hierarchy instead of the natural language processing of the Wikipedia articles.

Similarity-  
based dataset  
discovery  
framework

---

```
computeCosineSimilarity(  
  mapConceptsToAverageVector(  
    instanceToClass(  
      bagOfWordsMapper(  
        projectToTitle( extractMetadata(NODC) ),  
        wikidataLabelsExtractor(Wikidata knowledge graph)  
      ),  
      wikidataHierarchyExtractor(Wikidata knowledge graph)  
    ) ∪  
    instanceToClass(  
      bagOfWordsMapper(  
        projectToDescription( extractMetadata(NODC) ),  
        wikidataLabelsExtractor(Wikidata knowledge graph)  
      ),  
      wikidataHierarchyExtractor(Wikidata knowledge graph)  
    ) ∪  
    instanceToClass(  
      bagOfWordsMapper(  
        projectToKeywords( extractMetadata(NODC) ),  
        wikidataLabelsExtractor(Wikidata knowledge graph)  
      ),  
      wikidataHierarchyExtractor(Wikidata knowledge graph)  
    ),  
    getNode2VecModel(N2V [Wikidata KG/40/10])  
  )  
)
```

### 4.4 Pipeline implementation

In the previous section, we showcased a few selected pipeline definitions. The definitions capture all important information that is relevant in order to compare different pipelines. However, as the pipelines are described using the catalog layer model (see [Section 3.2](#)), additional steps must be carried out to get an executable pipeline implementation. In the rest of this section, we describe what a user needs to do in order to obtain an executable version of a pipeline in a step-by-step fashion. In addition, we provide an example of each step motivated by the pipeline described in [Section 4.3.3](#).

---

---

## DTA

The first step is to employ the component catalog (see [Section 4.2](#)), resolve component configurations and obtain links to component implementations in our GitHub repository: <https://github.com/mff-uk/SimPipes-Components>. In the component repository, each component has an implementation, input and output data sample and a README.md file with details on how to use the component. The component description consists of: textual description, requirements, input description, output description, configuration and example execution script.

For example, the `projectDescriptor` component performs a projection of a given property as described in the initial part of the README .md. However, the component implementation also changes the data format from JSON to CSV, as can be seen from the following input and output descriptions:

```
Format: Directory of JSON files.
Contents: Dataset descriptor.
Sample: Input sample

Format: CSV file.
Contents: CSV with iri and required property.
Sample: Output sample
```

Each input or output description consists of format specification, human readable description of the content and link to a data sample.

Another important part of the description is the component implementation configuration. The configuration is used not only to set the inputs and outputs of components, but also to provide additional parameters to the implemented algorithms.

```
input - Path to datasets descriptor files.
output - Path to output file.
property - Name of property to project.
linePerValue - For each value create a new line.
```

The last part of the component description is an example of the execution script. Note that the name of the component entry script, `json-to-csv.py`, does not have to correspond to the component name, JSON To CSV, nor to the component type name, `refine-descriptor`.

```
python3 json-to-csv.py \
  --datasets ./input-sample/datasets \
  --output ./output/output.csv \
  --property title
```

The next step is to create a component instance configuration. Most of the time, this needs to be done manually, as the user needs to understand the configuration description in the component catalog [Section 4.2](#) and create a corresponding configuration for the component instance based on the README.md file in the component repository.

For example, `projectToTitle` is a named configuration of the above-mentioned `projectDescriptor` component. The configuration description of `projectToTitle` states that the projected property is set to `title`. From the description configuration, the user should be able to figure out that this can be archived by setting the `property` configuration to `title`. As `title` is already in the example, the `property` argument remains the same here. The `input` and `output` arguments of the script should be changed to match the rest of the pipeline.

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

Similarity-based dataset discovery framework

---

Once the configurations are ready, the user can use them to obtain commands that can be used to execute the given components, and put those commands into a script that forms a backbone of the pipeline implementation. However, as the pipelines algebra captures the pipeline at the conceptual level, it leaves out some implementation details like utility components.

Utility components are stored in the `processors/utilities` directory in the repository. They do not change the contents or meaning of the data passed among the individual components in the pipeline, but they might be necessary for operations like data format changes, which make the inputs and outputs of the individual components in the pipeline compatible. Therefore, the user now needs to take a look at the implementation pipeline backbone and possibly insert necessary utility components.

A good example of the necessity to use a utility component is the pipeline union described in [Section 3.3](#) and used in [Section 4.3.4](#). The pipeline produces several different descriptors by using the `instanceToClass` component on descriptors computed from title, description and keywords. In the next step, all the data should be put together by union. While the union has no counterpart in the component catalog, there is a component in the component repository that can carry on this operation called `json-union`.

With the complete script to run all the components, the last step is to make sure that requirements of each component are fulfilled. An example of such a script can be found in [Appendix C](#).

Some components may require external data or installation of additional libraries, which are described in the component's `README.md` file. As the components are written in Python, their dependencies are provided as `requirements.txt` files. The installation of the dependencies is then as simple as running `pip3 install -r requirements.txt` for the used components. Once all the requirements are met and the script is ready, the pipeline can be executed simply by running the script. While the process is relatively straightforward, the user needs to have good knowledge of the available components and their configurations as well as basic knowledge of data processing techniques.

### 4.5 Proof-of-concept similarity presentation

The possibilities of usage of the results of similarity production pipelines is out of scope of this paper. Nevertheless, we present at least two usage examples: manual and automatic evaluation of results of various pipelines.

**4.5.1 Manual evaluation.** The manual evaluation process is fully described in [Škoda et al. \(2020\)](#), showcasing, among others, the first three similarity production pipelines from [Section 4.3](#). Note that the results of this evaluation led us to later replace the first pipeline ([Section 4.3.1](#)) with the last pipeline ([Section 4.3.4](#)) in further evaluation, which is used for illustration in this paper.

The core idea of the manual evaluation protocol is to employ a set of predefined use cases. Each use case defines a textual description of the user's intent, which is used to set up a scenario in which a user performs the search. In addition, the use case contains a collection of the so-called *starting datasets* – datasets already verified to be relevant to the user and their scenario. The user scenario definition is important as different scenarios may lead to different results even with the same set of starting datasets. For the purpose of the evaluation, all team members followed the same protocol as described in [Škoda et al. \(2020\)](#).

We carried out the evaluation on selected pipelines from [Section 4.3](#) using the `OpenDataInspectorEvaluation` presenter:

---

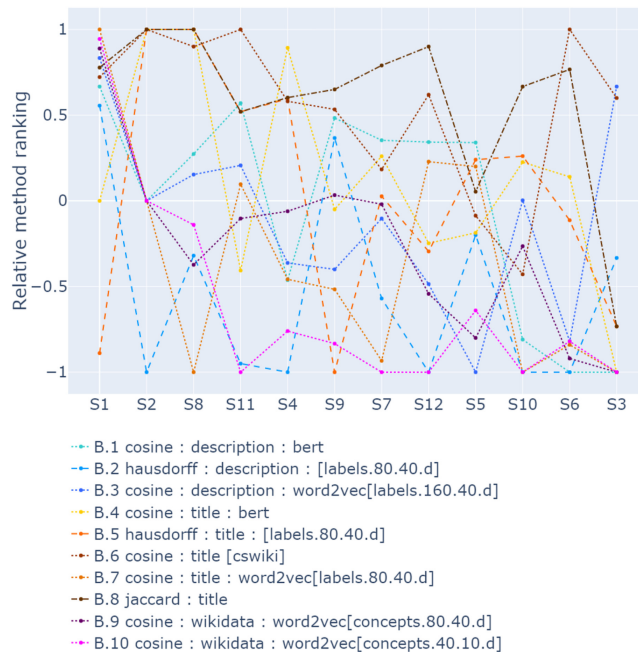
---

DTA

```
OpenDataInspectorEvaluation(  
  extractMetadata(NODC),  
  computeJaccardSimilarity(  
    doLemmatisation(  
      projectToTitle( extractMetadata(NODC) )  
    )  
  )  
)
```

The presenter provides not only an environment to run the evaluation but also the scripts that can be used to gain basic insight into the evaluation and plot the results. In Figure 7, we can see an example of one of those outputs. The values S1, S2 . . . S12 on the x-axis represent the different use-cases as defined in Škoda *et al.* (2020). There are results for 10 methods, whose algebraic definitions can be found in Appendix B. To briefly summarize the results, it is clear that there is no one method that outperforms all others in all use-cases, which shows the need for further research into the various types of use cases, various similarity production pipelines and into the problem of how to select the best similarity for a given use case. These findings correspond with those in Škoda *et al.* (2020).

4.5.2 Automatic evaluation. During our previous experiments (Škoda *et al.*, 2020), we quickly found out that the time required for manual evaluation increases rapidly with the increasing number of similarity production pipelines to be evaluated and their configurations such as different embeddings and projections. This led us to investigate the possibilities of



**Figure 7.**  
Normalized average  
performance per model

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

automating at least a part of the evaluation process. We devised an automated similarity production pipeline screening process – an automatic evaluation of the success potential of different similarity production pipelines in the form of presenters. These presenters can provide a quick estimate of the pipeline efficiency so that we can focus on a much lower number of candidates for the manual evaluation.

However, the automatic evaluation relies on a user-defined baseline. To obtain the baseline, we manually added the list of all relevant datasets from the given catalog for each use case in [Skoda et al. \(2020\)](#).

The `evaluateExactSize` and `evaluateTopK` presenters perform a kNN query search, where  $k$  is the number of expected answers, or the specified constant, respectively. For each use-case, the accuracy is computed as the ratio of expected datasets in the result. The presenter `evaluateTopK` reflects more precisely the user-based evaluation as it is not essential to have an exact match. For the users, it is a success if the expected datasets are found in the first few results. The `evaluatePRCurve` presenter provides a more comprehensive view of the issue. Instead of one number, the PR curve says how much effort the users have to make to achieve the required precision.

In our experiments, these methods provided a good estimate for comparing similar pipelines, especially with different kinds of external knowledge. They are also useful for filtering out models with a great number of settings of hyperparameters. Example of results for similarity production pipelines in [Appendix B](#) is shown in [Table 1](#). The accuracy found in the automatic evaluation corresponded very well with manual evaluation. As expected, methods based on word embeddings with a large dictionary (e.g. Word2Vec or BERT based on Wikipedia) have been very successful. Worse, due to the specific domain, were approaches based on smaller databases (e.g. Node2Vec).

Since we already know that different similarity production pipelines are successful for different use cases and the results of the automatic evaluation represent an aggregation of accuracy over all use cases, the results are not conclusive, and therefore we use them only for the initial screening process.

**4.5.3 Run-time evaluation.** During the testing of various similarity models (pipelines), we can measure pipeline run-time in addition to precision, recall, accuracy and other statistical metrics. We can measure the run-time of entire pipelines, but we can also measure the run-time of individual components due to the modularity of the proposed framework. This can be very useful when comparing two implementations of the same component, where we replace one implementation with another in an identical pipeline.

[Table 1](#) shows the approximate run-time of each pipeline. Individual runs do not use precomputed parts from previous runs but can use parallel processing. The computations were run on a virtual machine running 8 cores with 32 GB of RAM running Ubuntu 21.10. Testing on a virtual machine of course means that the measurements were influenced by

Similarity-based dataset discovery framework

---

Pipeline	Accuracy	Run-time
B.1 cosine: description: bert	10.7%	~ 20m
B.2 hausdorff: description: [labels.80.40.d]	4.0%	~ 8m
B.3 cosine: description: word2vec[labels.160.40.d]	19.4%	~ 6m
B.4 cosine: title: bert	21.2%	~ 20m
B.5 hausdorff: title: [labels.80.40.d]	7.1%	~ 5m
B.6 cosine: title [cswiki]	34.9%	~ 4m
B.7 cosine: title: word2vec[labels.80.40.d]	12.7%	~ 3m
B.8 jaccard: title	32.4%	~ 2m
B.9 cosine: wikidata: word2vec[concepts.80.40.d]	14.3%	~ 3h
B.10 cosine: wikidata: word2vec[concepts.40.10.d]	3.0%	~ 7h

**Table 1.** Results of `evaluateTopK` with  $K = 20$  for pipelines presented in [Appendix B](#)



---

---

DTA

other virtual machines running on the same hardware, and are therefore only very approximate. Since our server does not support GPU acceleration, parts of pipelines B.1 and B.4 were run externally using GPU acceleration within Google Colab service [21].

The longer run-time of pipelines B.9 and B.10 is caused by long-running knowledge extraction components. Those components execution time alone accounts for roughly 3 and 3.5 *h*, respectively. The reasons for such long execution times are the size of the Wikidata knowledge graph dump and the proof-of-concept single threaded implementation. As a result, there is plenty of room for improvement that could significantly speed up the execution.

One of the indisputable advantages of our framework is the ability to use existing outputs from common parts of pipelines. This allows us to test different pipelines that share similar properties efficiently. At the same time, if pipelines are designed appropriately, it is also possible to use parallel processing and, therefore, experimental evaluation of several pipelines simultaneously. As a result, the framework allows efficient experimental evaluation despite the inefficient implementation of components, which is very common in the experimentation phase. Although our proof-of-concept implementation is not scalable in terms of production deployment, scalability of experiments is achieved by reusing already computed parts, which is one of the goals of our framework.

## 5. Conclusions

In this paper, we have introduced a modular framework for experimentation with dataset discovery methods. We have presented the framework from a software-engineering perspective, providing:

- (1) formal conceptual definitions of framework components,
- (2) a catalog of ready-to-use component implementations,
- (3) production pipelines focusing on similarity-based methods and utilization of external knowledge, such as knowledge graphs and embedding models and
- (4) publication of the framework on GitHub.

An interested reader can dive from the conceptual level to the more detailed implementation level, where the algebraic definitions of pipelines are translated into scripts that could be directly executed as a piece of experimental software. The framework, including the implementation, was published on GitHub and is ready to be freely used and extended.

## Notes

1. <https://data.europa.eu/catalogue-statistics/Evolution>
2. Measured by a SPARQL query for selecting the number of distinct datasets on <https://data.europa.eu/data/sparql>
3. <https://gitlab.com/european-data-portal/metrics/edp-metrics-dataset-similarities/-/blob/master/src/main/java/io/piveau/metrics/similarities/SimilarityVerticle.java>
4. <https://data.europa.eu>
5. <https://data.gov.cz>
6. <https://www.data.gov/>
7. <https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe>
8. <https://www.w3.org/TR/vocab-dcat-2/#Class:Dataset>
9. <https://op.europa.eu/en/web/eu-vocabularies/authority-tables>

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

10. <https://data.gov.cz>
11. <https://www.wikidata.org/wiki/Q21503252>
12. <https://www.wikidata.org/wiki/Q21514624>
13. <https://zenodo.org/record/4433464/files/2020.04.20-data.gov.cz-no-cuzk.trig?download=1>
14. <https://zenodo.org/record/3975038>
15. <https://zenodo.org/record/4433699>
16. <https://zenodo.org/record/4433737>
17. <https://zenodo.org/record/4433778>
18. <https://zenodo.org/record/4433795>
19. [https://storage.googleapis.com/bert\\_models/2018\\_11\\_23/multi\\_cased\\_L-12\\_H-768\\_A-12.zip](https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip)
20. <https://zenodo.org/record/4436356/files/20181217.json.gz?download=1>
21. <https://colab.research.google.com/>

Similarity-  
based dataset  
discovery  
framework

---

### References

- Altaf, B., Akujuobi, U., Yu, L. and Zhang, X. (2019), "Dataset recommendation via variational graph autoencoder", *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 11-20, doi: [10.1109/ICDM.2019.00011](https://doi.org/10.1109/ICDM.2019.00011).
- Benjelloun, O., Chen, S. and Noy, N.F. (2020), "Google dataset search by the numbers", *Proceedings, Part II. Vol. 12507 of Lecture Notes in Computer Science, The Semantic Web - ISWC 2020-19th International Semantic Web Conference*, Springer, Athens, Greece, November 2-6, 2020, pp. 667-682, doi: [10.1007/978-3-030-62466-8\\_41](https://doi.org/10.1007/978-3-030-62466-8_41).
- Berners-Lee, T. (2006), "Linked data", available at: <https://www.w3.org/DesignIssues/LinkedData.html>.
- Bernhauer, D. and Skopal, T. (2020), *Word2Vec Model - Czech Wikipedia*, Zenodo, Geneva, doi: [10.5281/zenodo.3975038](https://doi.org/10.5281/zenodo.3975038).
- Bernhauer, D. and Skopal, T. (2021a), *Node2Vec Model - Czech Wikidata (Knowledge Graph/Concepts/L40/Rw10)*, Zenodo, Geneva, doi: [10.5281/zenodo.4433795](https://doi.org/10.5281/zenodo.4433795).
- Bernhauer, D. and Skopal, T. (2021b), *Node2Vec Model - Czech Wikidata (Knowledge Graph/Concepts/L80/Rw40)*, Zenodo, Geneva, doi: [10.5281/zenodo.4433778](https://doi.org/10.5281/zenodo.4433778).
- Bernhauer, D. and Skopal, T. (2021c), *Node2Vec Model - Czech Wikidata (Knowledge Graph/Labels/L160/Rw40)*, Zenodo, Geneva, doi: [10.5281/zenodo.4433737](https://doi.org/10.5281/zenodo.4433737).
- Bernhauer, D. and Skopal, T. (2021d), *Node2Vec Model - Czech Wikidata (Knowledge Graph/Labels/L80/Rw40)*, doi: [10.5281/zenodo.4433699](https://doi.org/10.5281/zenodo.4433699).
- Bogaty, A., Fernandes, A.A.A., Paton, N.W. and Konstantinou, N. (2020), "Dataset discovery in data lakes", *36th IEEE International Conference on Data Engineering, ICDE 2020*, IEEE, Dallas, TX, USA, April 20-24, 2020, pp. 709-720, doi: [10.1109/ICDE48307.2020.00067](https://doi.org/10.1109/ICDE48307.2020.00067).
- Brickley, D., Burgess, M. and Noy, N.F. (2019), in Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R. and Zia, L. (Eds), "Google dataset search: building a search engine for datasets in an open web ecosystem", *The World Wide Web Conference, WWW 2019*, ACM, San Francisco, CA, USA, May 13-17, 2019, pp. 1365-1375, doi: [10.1145/3308558.3313685](https://doi.org/10.1145/3308558.3313685).
- Browning, D., Beltran, A.G., Perego, A., Winstanley, P., Albertoni, R. and Cox, S. (2020), *Data Catalog Vocabulary (DCAT) - Version 2. W3C Recommendation*, W3C, available at: <https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/>.

---

---

---

  
DTA

- Carothers, G. and Seaborne, A. (2014), *RDF 1.1 TriG. W3C Recommendation*, W3C, available at: <https://www.w3.org/TR/2014/REC-trig-20140225/>.
- Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.D., Kacprzak, E. and Groth, P. (2020), "Dataset search: a survey", *VLDB Journal*, Vol. 29 No. 1, pp. 251-272, doi: [10.1007/s00778-019-00564-x](https://doi.org/10.1007/s00778-019-00564-x).
- Chen, X., Gururaj, A.E., Ozyurt, B., Liu, R., Soysal, E., Cohen, T., Tiryaki, F., Li, Y., Zong, N., Jiang, M., Rogith, D., Salimi, M., Kim, H.-e., Rocca-Serra, P., Gonzalez-Beltran, A., Farcas, C., Johnson, T., Margolis, R., Alter, G., Sansone, S.-A., Fore, I.M., Ohno-Machado, L., Grethe, J.S. and Xu, H. (2018), "DataMed – an open source discovery index for finding biomedical datasets", *Journal of the American Medical Informatics Association*, Vol. 25 No. 3, pp. 300-308, doi: [10.1093/jamia/ocx121](https://doi.org/10.1093/jamia/ocx121).
- Chen, Z., Jia, H., Heflin, J. and Davison, B.D. (2020), "Leveraging schema labels to enhance dataset search", in Jose, J.M., Yilmaz, E., Magalhães, J., Castells, P., Ferro, N., Silva, M.J. and Martins, F. (Eds), *Advances in Information Retrieval*, Springer International Publishing, Cham, pp. 267-280.
- Cygniak, R., Lanthaler, M. and Wood, D. (2014), *RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation*, W3C, available at: <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R. and Yu, C. (2012), "Finding related tables", *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD'12*, Association for Computing Machinery, New York, NY, pp. 817-828, doi: [10.1145/2213836.2213962](https://doi.org/10.1145/2213836.2213962).
- Degbelo, A. (2020), "Open data user needs: a preliminary synthesis", *Companion Proceedings of the Web Conference 2020, WWW'20*, Association for Computing Machinery, New York, NY, pp. 834-839, doi: [10.1145/3366424.3386586](https://doi.org/10.1145/3366424.3386586).
- Degbelo, A. and Teka, B.B. (2019), "Spatial search strategies for open government data: a systematic comparison", CoRR abs/1911.01097, available at: <https://arxiv.org/abs/1911.01097>.
- Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018), "Bert: pre-training of deep bidirectional transformers for language understanding", arXiv preprint arXiv:1810.04805, available at: <https://arxiv.org/abs/1810.04805>.
- Dutkowski, S. and Schramm, A. (2015), "Duplicate evaluation - position paper by fraunhofer FOKUS", *Tech. rep., Fraunhofer FOKUS*, available at: [https://www.w3.org/2016/11/sdsvoc/SDSVoc16\\_paper\\_24](https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_24).
- El-Sappagh, S., Hendawi, A. and El-Bastawissy, A. (2011), "A proposed model for data warehouse etl processes", *Journal of King Saud University - Computer and Information Sciences*, Vol. 23, pp. 91-104.
- Ellefi, M.B., Bellahsene, Z., Dietze, S. and Todorov, K. (2016), "Dataset recommendation for data linking: an intensional approach", in Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P. and Lange, C. (Eds.), *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Proceedings*, Springer, Heraklion, Crete, Greece, May 29 - June 2, 2016, pp. 36-51, doi: [10.1007/978-3-319-34129-3\\_3](https://doi.org/10.1007/978-3-319-34129-3_3).
- Fellbaum, C. (2005), "WordNet and wordnets", in Brown, K. (Ed.), *Encyclopedia of Language and Linguistics*, 2nd ed., Elsevier, Oxford, pp. 665-670.
- Fernandez, R.C., Abedjan, Z., Koko, F., Yuan, G., Madden, S. and Stonebraker, M. (2018), "Aurum: a data discovery system", *34th IEEE International Conference on Data Engineering, ICDE 2018*, IEEE Computer Society, Paris, France, April 16-19, 2018, pp. 1001-1012, doi: [10.1109/ICDE.2018.00094](https://doi.org/10.1109/ICDE.2018.00094).
- Gregory, K., Groth, P., Scharnhorst, A. and Wyatt, S. (2020a), "Lost or found? Discovering data needed for research", *Harvard Data Science Review*, Vol. 2 No. 2, available at: <https://hdsr.mitpress.mit.edu/pub/gw3r97ht>.

## 2. MODULAR FRAMEWORK FOR SIMILARITY-BASED DATASET DISCOVERY USING EXTERNAL KNOWLEDGE

---

- Gregory, K.M., Cousijn, H., Groth, P., Scharnhorst, A. and Wyatt, S. (2020b), "Understanding data search as a socio-technical practice", *Journal of Information Science*, Vol. 46 No. 4, pp. 459-475, doi: [10.1177/0165551519837182](https://doi.org/10.1177/0165551519837182).
- Grover, A. and Leskovec, J. (2016), "node2vec: scalable feature learning for networks", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Klímek, J. (2019), "DCAT-AP representation of Czech national open data catalog and its impact", *Journal of Web Semantics*, Vol. 55, pp. 69-85, doi: [10.1016/j.websem.2018.11.001](https://doi.org/10.1016/j.websem.2018.11.001).
- Klímek, J. and Škoda, P. (2021a), *Dump of Metadata from the Czech National Open Data Catalog, 2020-04-20, State Administration of Land Surveying and Cadastre Datasets Removed*, Zenodo, Geneva, doi: [10.5281/zenodo.4433464](https://doi.org/10.5281/zenodo.4433464).
- Klímek, J. and Škoda, P. (2021b), *Wikidata Dump from 2018-12-17 in JSON*, Zenodo, Geneva, doi: [10.5281/zenodo.4436356](https://doi.org/10.5281/zenodo.4436356).
- Koesten, L. (2018), "A user centred perspective on structured data discovery", *Companion Proceedings of the The Web Conference 2018. WWW'18. International World Wide Web Conferences Steering Committee*, CHE, Republic and Canton of Geneva, pp. 849-853, doi: [10.1145/3184558.3186574](https://doi.org/10.1145/3184558.3186574).
- Leme, L.A.P.P., Lopes, G.R., Nunes, B.P., Casanova, M.A. and Dietze, S. (2013), "Identifying candidate datasets for data interlinking", in Daniel, F., Dolog, P. and Li, Q. (Eds), *Web Engineering*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 354-366.
- Martins, Y.C., da Mota, F.F. and Cavalcanti, M.C. (2016), *DSCrank: A Method for Selection and Ranking of Datasets*, Springer International Publishing, Cham, pp. 333-344, doi: [10.1007/978-3-319-49157-8\\_29](https://doi.org/10.1007/978-3-319-49157-8_29).
- Mikolov, T., Chen, K., Corrado, G.S. and Dean, J. (2013), "Efficient estimation of word representations in vector space", available at: <http://arxiv.org/abs/1301.3781>.
- Miller, R.J., Nargesian, F., Zhu, E., Christodoulakis, C., Pu, K.Q. and Andritsos, P. (2018), "Making open data transparent: data discovery on open data", *IEEE Database Engineering Bulletin*, Vol. 41 No. 2, pp. 59-70, available at: <http://sites.computer.org/debull/A18june/p59.pdf>.
- Mountantonakis, M. and Tzitzikas, Y. (2018), "Scalable methods for measuring the connectivity and quality of large numbers of linked datasets", *Journal of Data and Information Quality*, Vol. 9 No. 3, doi: [10.1145/3165713](https://doi.org/10.1145/3165713).
- Mountantonakis, M. and Tzitzikas, Y. (2020), "Content-based union and complement metrics for dataset search over RDF knowledge graphs", *Journal of Data and Information Quality*, Vol. 12 No.2, doi: [10.1145/3372750](https://doi.org/10.1145/3372750).
- Oliver, J., Cheng, C. and Chen, Y. (2013), "TLSH – a locality sensitive hash", *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 7-13.
- Papadias, D. (2009), *Nearest Neighbor Query*, Springer, Boston, MA, pp. 1890-1890, doi: [10.1007/978-0-387-39940-9\\_245](https://doi.org/10.1007/978-0-387-39940-9_245).
- Škoda, P., Klímek, J., Nečaský, M. and Skopal, T. (2019), "Explainable similarity of datasets using knowledge graph", *Similarity Search and Applications - 12th International Conference, SISAP 2019, Proceedings*, Springer, Newark, NJ, USA, October 2-4, 2019, pp. 103-110, doi: [10.1007/978-3-030-32047-8\\_10](https://doi.org/10.1007/978-3-030-32047-8_10).
- Škoda, P., Bernhauer, D., Nečaský, M., Klímek, J. and Skopal, T. (2020), "Evaluation framework for search methods focused on dataset findability in open data catalogs", *Proceedings of The 22nd International Conference on Information Integration and Web-based Applications and Services (iiWAS2020)*, Chiang Mai, Thailand, November 2020, pp. 200-209, available at: [http://www.iivas.org/conferences/iivas2020/proceedings/proceedings\\_iivas\\_2020.pdf](http://www.iivas.org/conferences/iivas2020/proceedings/proceedings_iivas_2020.pdf).
- Skopal, T., Bernhauer, D., Skoda, P., Klímek, J. and Nečaský, M. (2021), "Similarity vs relevance: from simple searches to complex discovery", *Similarity Search and Applications - 14th International Conference, SISAP 2021, Proceedings*, Springer, Dortmund, Germany, September 29 - October 1, 2021, pp. 104-117, doi: [10.1007/978-3-030-89657-7\\_9](https://doi.org/10.1007/978-3-030-89657-7_9).

Similarity-based dataset discovery framework

---

---

DTA

- Speer, R., Chin, J. and Havasi, C. (2017), "Conceptnet 5.5: an open multilingual graph of general knowledge", in Singh, S.P. and Markovitch, S. (Eds), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI Press, San Francisco, California, USA, February 4-9, 2017, pp. 4444-4451, available at: <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972>.
- Straka, M. and Straková, J. (2019), *Universal Dependencies 2.5 Models for UDPipe (2019-12-06)*, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, Prague, available at: <http://hdl.handle.net/11234/1-3131>.
- Vrandečić, D. and Krötzsch, M. (2014), "Wikidata: a free collaborative knowledgebase", *Communications of the ACM*, Vol. 57 No. 10, pp. 78-85, doi: [10.1145/2629489](https://doi.org/10.1145/2629489).
- Wagner, A., Haase, P., Rettinger, A. and Lamm, H. (2014), "Entity-based data source contextualization for searching the web of data", *The Semantic Web: ESWC 2014 Satellite Events*, Springer International Publishing, Cham, pp. 25-41.
- Yakout, M., Ganjam, K., Chakrabarti, K. and Chaudhuri, S. (2012), "Infogather: entity augmentation and attribute discovery by holistic matching with web tables", *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD'12*, Association for Computing Machinery, New York, NY, USA, pp. 97-108, doi: [10.1145/2213836.2213848](https://doi.org/10.1145/2213836.2213848).
- Zezula, P. (2015), "Similarity searching for the big data - challenges and research objectives", *Mobile Networks and Applications*, Vol. 20 No. 4, pp. 487-496, doi: [10.1007/s11036-014-0547-2](https://doi.org/10.1007/s11036-014-0547-2).
- Zhang, S. and Balog, K. (2018), "Ad hoc table retrieval using semantic similarity", *Proceedings of the 2018 World Wide Web Conference. WWW'18. International World Wide Web Conferences Steering Committee*, CHE, Republic and Canton of Geneva, pp. 1553-1562, doi: [10.1145/3178876.3186067](https://doi.org/10.1145/3178876.3186067).
- Zhang, E. and Zhang, Y. (2009), *Eleven Point Precision-Recall Curve*, Springer, Boston, MA, pp. 981-982, doi: [10.1007/978-0-387-39940-9\\_481](https://doi.org/10.1007/978-0-387-39940-9_481).

#### Appendix

The Appendix is available online for this article.

#### Corresponding author

Jakub Klímek can be contacted at: [jakub.klimek@matfyz.cuni.cz](mailto:jakub.klimek@matfyz.cuni.cz)

---

For instructions on how to order reprints of this article, please visit our website:  
[www.emeraldgroupublishing.com/licensing/reprints.htm](http://www.emeraldgroupublishing.com/licensing/reprints.htm)  
Or contact us for further details: [permissions@emeraldinsight.com](mailto:permissions@emeraldinsight.com)



---

## Similarity vs. Relevance: From Simple Searches to Complex Discovery

Skopal, T. (20 %); Bernhauer, D. (20 %); Škoda, P. (20 %); Klímek, J. (20 %); Nečaský, M. (20 %) Similarity vs. Relevance: From Simple Searches to Complex Discovery. In *Similarity Search and Applications: 14th International Conference, SISAP 2021*, Dortmund, Germany: Springer-Verlag, 2021, doi: 10.1007/978-3-030-89657-7\_9. [A.3]



## Similarity vs. Relevance: From Simple Searches to Complex Discovery

Tomáš Skopal<sup>1</sup>, David Bernhauer<sup>1,2</sup>, Petr Škoda<sup>1</sup>, Jakub Klímek<sup>1</sup>,  
and Martin Nečaský<sup>1</sup>

<sup>1</sup> SIRET Research Group, Faculty of Mathematics and Physics, Charles University,  
Prague, Czech Republic

{tomas.skopal,david.bernhauer,petr.skoda,jakub.klimek,  
martin.necasky}@matfyz.cuni.cz

<sup>2</sup> Faculty of Information Technology, Czech Technical University in Prague,  
Prague, Czech Republic

**Abstract.** Similarity queries play the crucial role in content-based retrieval. The similarity function itself is regarded as the function of relevance between a query object and objects from database; the most similar objects are understood as the most relevant. However, such an automatic adoption of similarity as relevance leads to limited applicability of similarity search in domains like entity discovery, where relevant objects are not supposed to be similar in the traditional meaning. In this paper, we propose the meta-model of data-transitive similarity operating on top of a particular similarity model and a database. This meta-model enables to treat directly non-similar objects  $\mathbf{x}$ ,  $\mathbf{y}$  as similar if there exists a chain of objects  $\mathbf{x}$ ,  $i_1, \dots, i_n$ ,  $\mathbf{y}$  having the neighboring members similar enough. Hence, this approach places the similarity in the role of relevance, where objects do not need to be directly similar but still remain relevant to each other (transitively similar). The data-transitive similarity concept allows to use standard similarity-search methods (queries, joins, rankings, analytics) in more complex tasks, like the entity discovery, where relevant results are often complementary or orthogonal to the query, rather than directly similar. Moreover, we show the data-transitive similarity is inherently self-explainable and non-metric. We discuss the approach in the domain of open dataset discovery.

## 1 Introduction

When searching data, we can choose from a multitude of available models and paradigms. Some models assume exact data structure and semantics, such as the relational database model (and SQL) or graph database model (RDF+SPARQL, XML+XQuery). In such models, the relevance of a data entity to a particular query is binary (relevant/not relevant); specified by a binary predicate. The precision and recall in retrieval of structured data is always 100% as there is no uncertainty expected. Also, structured query languages offer high expressive power that allows the user to specify the relevance of data in many ways.



On the other side of the data universe, when searching in unstructured or loosely structured data (like multimedia, text, time series), we do not have enough a-priori information on how to model the data features for exact search. In such situation the similarity search models could be used, representing a universal way of content-based retrieval in unstructured data. Instead of formulating a structured query aiming at binary relevance, in similarity search we use a ranking of the database objects determined by their similarity score to a query example (the query-by-example paradigm). Hence, the relevance is relaxed from binary to multiple-value. When compared to retrieval of structured data, the similarity search is more like an “emergency solution” for unstructured data. The expressive power of similarity queries is limited to a ranking induced by numeric aggregation of differences between the query example and the database objects; keeping it a black-box search for the user. The low expressive power of the query-by-example paradigm leads to a paradox – we search for what we already have. Specifically, we query for as good results as possible, having the best result already at hand – the query example. Of course, in practical applications the query-by-example paradigm makes sense, because the query example itself does not contain the whole information we search for. For instance, searching by the photo of Eiffel tower we not only get another Eiffel tower image, but also some context (the Wikipedia web page the result image was embedded in). Nevertheless, the context (external information attached to data) does not remove the essence of the paradox – based purely on the similarity of results, the query example itself is always the best result<sup>1</sup>.

Historically, the low expressive power of similarity search has been accepted in the major application area – the multimedia retrieval. Here the semantics to be captured in multimedia objects (the descriptors) is rather vague, general and bound to human common knowledge. The similarity search is thus a perfect method for multimedia retrieval as the similarity concept itself is vague and general (and so is the human cognition – the inspiration for similarity search). When combined with descriptor models employing high-level “canonized” semantics, such as the bag of words using the vocabulary of deep features [11], then even the cosine similarity can perform well. Unfortunately, the domain experts are not always so lucky to work with nicely shaped semantic descriptors, while then the low expressive power of similarity search is fully revealed. A solution to this could be a proposal of similarity-aware relevance of data objects to an example object (query) that enables much more complex aggregation than just evaluating the direct similarity (the “exempleness” of the results). If we find a way of how to extend the concept of similarity into a relevance, we would be able to use the existing similarity search methods in more expressive retrieval scenarios. For example, consider a fashion e-shop where a user searches for a product by an example image, e.g., shoes. The result could not only consist of similar shoes, but it could also return related accessories (handbag, belt) sharing some design features with the shoes [14].

<sup>1</sup> Let’s omit another problem; where to acquire such a “holy grail” example in real-world problems.

### 3. SIMILARITY VS. RELEVANCE: FROM SIMPLE SEARCHES TO COMPLEX DISCOVERY

---

106 T. Skopal et al.

In the following, we continue the discussion in the specific domain of open datasets discovery. Unlike in multimedia retrieval, where direct audiovisual similarity to a query usually leads to good results, in open datasets with sparse descriptors we often do not find anything directly (non-trivially) similar. Here the similarity extended towards more general relevance could improve the retrieval effectiveness in a fundamental way.

#### 1.1 Discovery of Open Datasets by Similarity

The similarity search models can utilize not only content features but also metadata (if available). The focus on metadata can be efficient and effective in domains where the content of the objects is too heterogeneous so that it is hard to extract features for measuring similarity (or relevance). On the other hand, such objects could be catalogued by a community to enable search of the objects by metadata.

This is the case of the domain of open datasets search and discovery [12]. There are various datasets published on the internet which are catalogued in open data catalogs [18]. They are extremely heterogeneous in structure and semantics so that modeling them by content is nearly impossible (consider tables and spreadsheets without schema, full-text reports, database dumps, geographical and map data, logs, etc.). Open data catalogs provide descriptive metadata about the datasets in a single place where potential consumers can search for datasets. However, the problem of metadata is that they are often sparse and poor. In the open data domain, dataset publishers usually limit their descriptive metadata to briefly describe the core semantics of their datasets (by title, keywords, text description). No broader context of a dataset including some description of its relationships to other datasets is specified in the metadata. Using such sparse metadata for similarity retrieval is therefore limited. We confirmed this in our previous work [26] where we showed that various similarity methods do not perform very well when applied to the descriptive metadata of open datasets.

In our experiments, we noticed situations where two datasets are relevant to each other but none of the similarity models is able to identify this relevance. Let us demonstrate this on a concrete example of open data published by public authorities in Czechia. The datasets are catalogued in the National Open Data Catalog (NODC)<sup>2</sup>. There are two datasets entitled *IDOL Integrated Transport System Tariff Zones* and *Traffic intensity on sections of motorways*. The similarity of both datasets based on their metadata descriptions is low according to various similarity models presented in [26]. However, when we reviewed the datasets manually we found out that they are very relevant to each other. The first one is related to public transport. The second one is related to transport on motorways. So when users find one of the datasets, they would like to get also the other dataset as well. What makes them relevant to each other is the background semantics which is not directly expressed in the descriptive metadata. Since it is not expressed in the metadata, no similarity model can work with

<sup>2</sup> <https://data.gov.cz/english/>.

this. However, there is a third dataset in NODC titled *BKOM transport year-book*. The similarity models identify its similarity with the original two datasets on the base of available metadata. So using the third dataset we could say that the two original datasets are relevant to each other because they are both similar to the third one. In other words, they are *transitively similar* when using other datasets as a context. What is also interesting in the example is that metadata about the third dataset express explicitly the concept of transport. So, the third dataset is not just an intermediary dataset between the two. It explains why they are relevant, contributing thus to the discussion on *explainability* of similarity search.

## 2 Related Work

Before presenting the meta-model of data-transitive similarity, we discuss several related points.

### 2.1 Similarity Modeling

The research in the similarity search area had intensified some three decades ago by setting the metric space model as the golden standard [25]. The metric distances in place of (dis)similarity functions were introduced purely for database indexing reasons (i.e., for fast search). Though a good trade-off for many problems, the metric space model remains quite restrictive for modeling similarity. The restrictions are even more strict in follow-up models aiming at improving search efficiency, such as the ptolemaic [15] or supermetric [9] models. As mentioned in the previous section, this might not be a problem in case the descriptors are canonized and semantic (such as histograms referring to a vocabulary of deep features). However, for the lower-semantic cases there were alternative approaches to indexing similarity proposed in the past 15 years, ranging from dynamic combinations of multiple metrics [5] for multi-modal retrieval to completely unrestricted, non-metric approaches [23]. The rationale for their introduction was to increase the expressive power of similarity search (and effectiveness) and still provide an acceptable retrieval efficiency.

### 2.2 Retrieval Mechanisms

No matter if we choose metric or non-metric similarity, the expressive power of retrieval is also affected by the retrieval mechanism used. The query-by-example paradigm constitutes the basic functionality of similarity search in form of kNN or range queries. The similarity joins enable the use of similarity within the database JOIN operators [22]. The similarity queries could be also used with additional post-processing techniques for multi-modal retrieval and analytics, such as the late fusion [21] and content-based recommender systems [1]. Last but not least, there appear proposals and frameworks helping with the integration of similarity search constructs into query languages, such as SimilarQL [24], or MSQL [19]. The ultimate goal is to establish higher-level declarative query models for similarity search [3].

#### 2.3 Dataset Discovery

Finding related datasets, also known as dataset discovery, is one of the important tasks in data integration [20]. Large companies such as Google have developed their own dataset search techniques and solutions [4]. New solutions for dataset search in specific domains started to appear recently. For example, *Datamed* [8] is an open source discovery index for finding biomedical datasets. The existing works emphasize the role of quality metadata for dataset findability while [6] points out that available metadata does not always describe what is actually in a dataset and whether a described dataset fits for a given task. Other studies [12, 13, 16] confirm that dataset discovery is highly contextual depending on the current user's task. The studies show that this contextual dependency must be reflected by the dataset search engines. This makes the task of dataset discovery harder as it may not be sufficient to search for datasets only by classical keyword-based search. More sophisticated approaches being able to search for similar or related datasets could be helpful in these scenarios. As shown by [6, 20] many existing dataset discovery solutions are based on simple keyword search. Discovery of datasets by similarity is discussed in the recent survey [6]. Several papers propose dataset retrieval techniques based on metadata similarity. In [2] a method is described which enables to measure similarity between datasets on the base of papers citing the datasets and a citation network between datasets. In [10] four different metadata-based models are evaluated for searching spatially related datasets, i.e., datasets which are related because of the same or similar spatial area covered. To the best of our knowledge, none of the approaches does apply the following technique of data-transitive similarity in dataset discovery.

### 3 Data-Transitive Similarity

In this section, we introduce the meta-model of data-transitive similarity. The original inspiration was the omnipresent database operation JOIN, used in many data management use cases for interconnecting relevant pieces of information. In relational databases the join operations allow to connect data records by means of shared attribute(s). In an extensive interpretation, the mechanism in database joins has roots in an identification of relevant entities by partial matches (equality predicate) or by partial similarity (inequality predicate). Analogously, by introducing data-transitive similarity we aim at consecutively joining similar objects and evaluating the overall relevance as an aggregation over the partial similarity scores.

The basic assumption of data-transitive similarity is thus a chain of objects from the database that are similar to each other, but the beginning and end of the chain could be quite dissimilar (yet relevant). Remember the well-known example with the human and the horse, illustrating the violation of the triangle inequality [23]. These two creatures tend to be quite dissimilar, yet they can be relevant (transitively similar). The relevance here can be ensured by a connecting object in the middle of the chain – a horseman, or more poetically a centaur, creature that is half man and half horse. The data-transitive similarity itself,

however, can be more complex; the connecting agent may not be a single object, but a whole chain of objects. This chain also serves as an explanation of why the two objects are relevant and in what context (addressing the explainability issue).

The connection itself can be formalized as an aggregation of several consecutive ground distances. The Eq. 1 defines general form of data-transitive distance function  $\hat{d}$ , where  $\mathcal{D}$  is a set of objects (the database in practical applications),  $d$  is a ground distance (the direct similarity),  $n$  is the length of the chain. Operator  $\bigcirc$  is an outer aggregation over all permutations of length  $n$  over elements of database  $\mathcal{D}$  (e.g., min, max, avg). Operator  $\biguplus$  is an inner aggregation over the individual direct distances within a particular chain. Table 1 shows examples of various inner aggregation functions. They are also the aggregation functions we worked with in our preliminary experiments. A more complex alternative may be a combination of several kinds of aggregations or distances.

$$\hat{d}_{\biguplus}^{\bigcirc, n}(\mathbf{x}, \mathbf{y}) = \bigcirc_{(i_1, \dots, i_n) \in \mathcal{D}^n} \biguplus(d(\mathbf{x}, i_1), d(i_1, i_2), \dots, d(i_n, \mathbf{y})) \quad (1)$$

**Table 1.** Examples of inner aggregation  $\biguplus$ .

$\text{sum}(\delta_0, \delta_1, \dots, \delta_n) = \sum_{j=0}^n \delta_j$
$\text{min}(\delta_0, \delta_1, \dots, \delta_n) = \min\{\delta_0, \delta_1, \dots, \delta_n\}$
$\text{max}(\delta_0, \delta_1, \dots, \delta_n) = \max\{\delta_0, \delta_1, \dots, \delta_n\}$
$\text{prod}(\delta_0, \delta_1, \dots, \delta_n) = \prod_{j=0}^n \delta_j$
$\text{iproduct}(\delta_0, \delta_1, \dots, \delta_n) = 1 - \prod_{j=0}^n (1 - \delta_j)$

To summarize, we define the data-transitive similarity  $\hat{d}$  as a meta-model operating on top of a ground similarity model  $d$  and a particular database  $\mathcal{D}$ . The computation of a single data-transitive distance involves a series of similarity queries over the database. The computational complexity of the data-transitive similarity thus involves not just the complexity of  $d$  but also the size of the database  $|\mathcal{D}|$ . Depending on the implementation, the worst-case time complexity  $O(\hat{d})$  can vary from  $O(d)$  to  $O(d)O(|\mathcal{D}|^n)$ , assuming  $n$  as a constant or  $n \ll |\mathcal{D}|$ .

From the definitions above it immediately follows that data-transitive distances are not metric distances – not only due to the possibly non-linear combination of the particular ground distances, but mainly due to the database-dependent nature of the distance topology (non-uniform distribution of points in the data universe and its impact on the chain members).

One might say that such advanced relevance constructions should not be modeled at the level of similarity, as they are part of higher retrieval models closer to the application level (e.g., a part of content-based recommender system). However, we want to stress that we intentionally included the data-transitive similarity into the family of generic pair-wise non-metric similarities. As such, it

### 3. SIMILARITY VS. RELEVANCE: FROM SIMPLE SEARCHES TO COMPLEX DISCOVERY

---

110 T. Skopal et al.

can be plugged into any search engine that supports non-metric similarities. This would not be possible if designed as a proprietary late-fusion retrieval model.

#### 3.1 Implementation

The fundamental problem we have addressed in the data-transitive similarity design was determining the number of intermediaries (the chain length  $n$ ) to form a transitive similarity. Although our model assumes an arbitrary  $n$ , determining the specific value is not a straightforward problem itself. A significant issue may be that for some objects, there is no intermediary to form transitive similarity. In general, the number of intermediaries may not be constant, and for different objects this value needs to be chosen dynamically.

Thus, for our experiment, we have applied a simplification in this regard and assume that data-transitive similarity has at most one intermediary (i.e.,  $n = 1$ ). Therefore, we always have a triplet: a query, an intermediary, and a result. This decision reduces the number of hyperparameters with respect to longer chains (e.g., number of intermediaries, different aggregation functions). This approach also has the advantage of a higher level of explainability. For longer chains of intermediaries, we need to discuss whether each part of the sequence makes sense for given transitivity. Whereas in the case of a single intermediary, we can argue with a reasonable certainty whether the query and result are relevant from the perspective of the intermediary explanation.

The second problem is the transitivity involving duplicates or near-duplicates in the chain – intermediaries very  $d$ -close to the query or to the result. Such duplicate intermediaries usually do not add any value. Therefore, small distances  $d$  (the first 5% of distance distribution) are not considered (in fact, all such distances are set to infinity to become disqualified in  $\hat{d}$ ).

Third, all ground distances are required to be normalized to 0–1 because some aggregations ( $\uplus = \text{prod}$ ,  $\uplus = \text{iproduct}$ ) require a bounded distance. In our implementation, we do not implement any optimizations, while to compute the data-transitive similarity we need to iterate over all database objects in the role of an intermediary. At the moment, optimizations for reduction of the set of intermediaries are beyond the subject of our research.

#### 3.2 Open Dataset Testbed

For the open dataset testbed presented in Sect. 1.1, we considered title, description, and keywords metadata. Since the original data provided by the National Open Data Catalog are in the Czech language, we used the automatic English translation [17], followed by the words lemmatization and filtering non-meaningful words (we consider only nouns, adjectives, verbs, and adverbs). In addition, we ignored several experimentally detected stop-words (data, dial, export, etc.). The metadata descriptors were represented in the bag of words model (BoW) with tf-idf weights.

Over these descriptors, the ground cosine distance was computed as  $d_{\text{cos}}(\mathbf{x}, \mathbf{y}) = 1 - s_{\text{cos}}(\mathbf{x}, \mathbf{y})$  (where  $s_{\text{cos}}$  is cosine similarity) for all pairs of objects (all pairs of datasets in our case). Figure 1 shows the distribution of distances  $d_{\text{cos}}$  over this testbed. We can see that most of the datasets are not  $d_{\text{cos}}$ -similar, and the testbed exhibits high intrinsic dimensionality [7]. This is due to the relatively sparse metadata (average about 20 words). For some datasets, some parts, such as description or keywords are empty; there is only the title description.

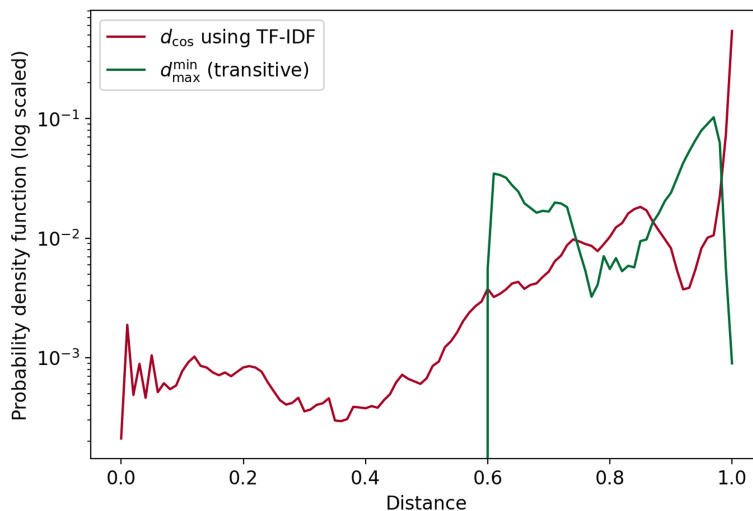


Fig. 1. Distance distribution of  $d_{\text{cos}}$  and  $\hat{d}_{\text{max}}^{\text{min}}$  transitive similarity

In our experiment, we took only one intermediary, while  $\hat{d}_{\text{max}}^{\text{min}}$  (Formula 2) was chosen as the data-transitive similarity function, since it exhibited the most robust aggregation in our preliminary experiments. Figure 1 shows how the distribution of  $\hat{d}_{\text{max}}^{\text{min}}$ -distances is different when compared to  $d_{\text{cos}}$ . Smaller distances (below approx. 0.6) are eliminated due to the removal of near-duplicate dataset pairs (set to 5% closest datasets), as mentioned in the previous subsection. The rest of the  $\hat{d}_{\text{max}}^{\text{min}}$ -distance domain is split into two categories representing relevance (more relevant around 0.7, less relevant around 0.9), with many  $d_{\text{cos}}$ -dissimilar datasets moving into the category of more  $\hat{d}_{\text{max}}^{\text{min}}$ -relevant datasets.

$$\hat{d}_{\text{max}}^{\text{min}}(\mathbf{x}, \mathbf{y}) = \min_{\forall i \in \mathcal{D}} \max \{d(\mathbf{x}, i), d(i, \mathbf{y})\} \quad (2)$$

## 4 Evaluation

As we have already discussed in [26], the findability evaluation in the open dataset discovery is complicated from several points of view. The database contains a relatively large number of datasets, but there is no sufficient ground

### 3. SIMILARITY VS. RELEVANCE: FROM SIMPLE SEARCHES TO COMPLEX DISCOVERY

---

112 T. Skopal et al.

truth for dataset similarity. To overcome the lack of ground truth, in this paper we evaluate the concept of relevance which is closer to dataset discovery, rather than direct context-independent similarity of datasets.

#### 4.1 Methodology

Our evaluation targets the additional value of data-transitive similarity search over the standard (direct  $d_{cos}$ ) similarity search. First, the search for similar datasets using standard  $d_{cos}$ -similarity search is performed. Let us represent this search as a  $k_d$  NN query, where  $k_d$  is the number of results. Then, there are  $k_t$  results displayed to the user using data-transitive similarity based  $k_t$  NN query, while filtering out results of the previous  $k_d$  NN query. For our experiment, we assume  $k_d = 100$  and  $k_t = 20$ .

The user (evaluator) is given a list of triplets (query, intermediary, result) and then evaluates each such triplet as relevant or non-relevant. A triplet is relevant if the user finds a possible use case for the query dataset and the result dataset and, at the same time, the intermediary dataset reasonably connects the two datasets. Let us repeat that the user is only confronted with results that were not findable by standard (direct) similarity search. A total of 5 users (evaluators) participated in the evaluation.

During the evaluation, we encountered the problem that some pairs of datasets are only relevant if we ignore specific fine-grained attributes of the datasets. The first observed attribute is the information about the publisher, e.g., contracts of the Ministry of Finance and invoices of the Ministry of Finance. The second attribute is the time or date of repeatedly published datasets, e.g., the list of companies for the year 2020. The third attribute is the localization specified in the datasets, e.g., hospitals in Prague vs. hospitals in Brno. For the evaluation, we decided to ignore these attributes as they only contribute to fragmentation of the datasets that are otherwise relevant to each other. However, this problem might disappear if we consider more than just one intermediary in the data-transitivity model (subject of future evaluations).

As part of the experiment, we evaluated the relevance of the results for a set of prepared queries. This set was created based on previous experiments presented in [26]. A total of 64 transitive results were found for 11 different queries.

#### 4.2 Results

During the evaluation, we looked at two main criteria: consistency and effectiveness. For every triplet, we have computed its score as sum of 0 (non-relevant) and 1 (relevant) ratings of all evaluators. In our case, the score ranges from 0 (all evaluators claim the triplet is non-relevant) to 5 (all evaluators claim the triplet is relevant). Figure 2 (left) shows the number of triplets with particular score, Fig. 2 (right) shows the number of triplets per data-transitive distance ranges and distribution of scores inside these ranges.

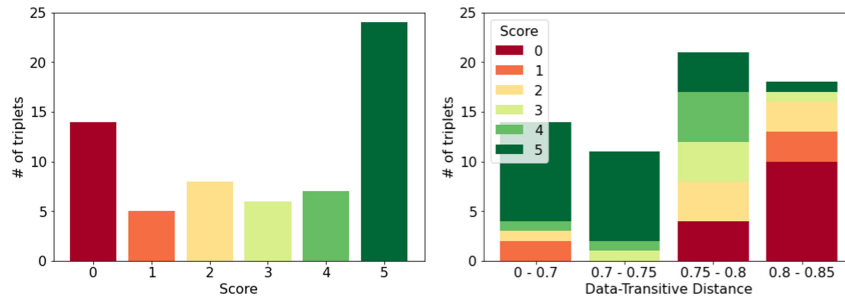
The consistency is validated based on the evaluators' agreement on the relevance of the evaluated triplets. Figure 2 (left) shows that in almost 78.13% of



the cases, majority of evaluators (scores 0–1 and 4–5) agreed on the triplets’ relevances. This observation confirms that the overall evaluation results are not just random noise.

Effectiveness is measured as the ratio of relevant datasets to all returned results. This gives us a measure of how much data-transitive similarity can improve the standard search. At Fig. 2 (left), we see that in 57.81% of the cases, the triplet was marked as relevant by a majority of evaluators (score 4–5).

Although the overall effectiveness may not seem significant, we must stress that all the relevant results found were not achievable by the direct similarity search (as already mentioned in Sect. 4.1). For 65.63% of the datasets,  $d_{cos}$  distances to query are maximal. We can also notice in Fig. 2 (right) that the data-transitive similarity model complies with the general thesis of similarity search (more distant datasets are less relevant and vice versa).



**Fig. 2.** The left figure shows the distribution of triplet ratings (how many triplets were rated by a particular relevancy score). For example, the score = 3 means that 3 evaluators thought the triplet was relevant (they rated it 1) and 2 evaluators thought the triplet was not relevant (they rated it 0). The right figure shows the distribution of ratings according to each data-transitive distance interval.

### 4.3 Qualitative Analysis

In Table 2 we see an example of triplet  $(Q, I, R)$  that was evaluated as relevant in our experiment (small data-transitive distance  $d_{\max}^{\min}(Q, R)$  through  $I$ ). If we analyze the distance structure, the query dataset ( $Q$ ) “Floods in the 19th century” does not have the “water” keyword in the metadata. However, thanks to the intermediary “5-year water” dataset ( $I$ ), we have both “water” and “flood” in metadata and so the query dataset is transitively similar to the result dataset ( $R$ ) “Water reservoirs”. In the original similarity (the direct ground distance  $d_{cos}$ ), the query  $Q$  and the result  $R$  datasets have maximum distance; they have nothing in common. In the data-transitive similarity search, however, the dataset  $R$  is within the first 20 results thanks to the connection with  $I$ . The relevance here can be explained by the fact that reservoirs can affect flooding and so the dataset  $R$  might be useful in flood prevention planning.

### 3. SIMILARITY VS. RELEVANCE: FROM SIMPLE SEARCHES TO COMPLEX DISCOVERY

---

114 T. Skopal et al.

**Table 2.** Example of Query, Intermediary, Result triplet: floods vs water. Title, keywords and description metadata are provided for each dataset.

	Title	
	Keywords	Description
	<i>Floods in the 19th century</i>	
Q	Floods, Environment, GIS	Flooded areas in a 19th century flood in the Pilsen region
	<i>5-year water</i>	
I	GIS, Floods, Environment	Flooding areas of n-year water in the Pilsen region
	<i>Water reservoirs under the management of the river basin and the forest of the Czech Republic under the territorial jurisdiction of the river Vltava</i>	
R	water tanks, water management	The shp file contains points representing water reservoirs whose permitted volume of buoyant or accumulated water exceeds 1 000 000 m3 or to which the Forests of the Czech Republic, p. The registers are updated continuously, the dataset only once a year. The current data can be viewed on the water information portal VODA – <a href="http://www.voda.gov.cz">www.voda.gov.cz</a>

The second example (Table 3) shows the imbalance of some descriptions, where the query dataset “Housing Young 2017” description has 3 paragraphs of text and the result dataset “BUG<sup>3</sup> - Economy and Labour Market” description has only one sentence. Although these datasets share some keywords, the

**Table 3.** Example of Query, Intermediary, Result triplet: housing vs labour. Title, keywords and description metadata are provided for each dataset.

	Title	
	Keywords	Description
	<i>Housing Young 2017</i>	
Q	sociology, housing research, housing young, housing, Brno	The main objective of the Youth Housing survey conducted in 2017 was to identify and describe the housing needs of young people living in Brno, as well as their preferences in this area. ... 3 paragraphs of text here ...
	<i>BUG - people and housing</i>	
I	Brno urban Grid, housing, people, BUG	Datasets from the Brno Urban Grid - theme people and housing
	<i>BUG - Economy and Labour Market</i>	
R	BUG, labour market, economy, Brno Urban Grid	Datasets from the Brno Urban Grid application - theme of economy and labour market

<sup>3</sup> BUG = Brno Urban Grid.

resulting position in ranking is too far when using the direct distance  $d_{cos}$ , so that the user cannot find the dataset. With the data-transitive similarity using the intermediary “BUG - people and housing” dataset the problem is mitigated. In this case, we are able to explain the relevance between the housing of young people and the state of the labour market.

## 5 Conclusion and Future Work

We proposed an extended concept of similarity search by introducing the meta-model of data-transitive similarity operating on top of a particular similarity model. In the evaluation focused on the open data domain, we have demonstrated that the user is able to find relevant datasets that were not findable using standard (direct) similarity search. Moreover, as the data-transitive similarity is a variant of pair-wise non-metric similarity, it can be plugged into any search engine that supports non-metric similarities. It also confirms the necessity of non-metric approaches in complex retrieval tasks, such as the entity discovery.

In the future we plan to investigate more general chains of intermediaries, as well as internal indexing techniques for the data-transitive similarity computation itself. We also plan to experiment with other domains that require more complex explainable similarity approaches.

**Acknowledgments.** This work was supported by the Czech Science Foundation (GAČR), grant number 19-01641S.

## References

1. Aggarwal, C.C.: Recommender Systems. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-29659-3>
2. Altaf, B., Akujuobi, U., Yu, L., Zhang, X.: Dataset recommendation via variational graph autoencoder. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 11–20 (2019). <https://doi.org/10.1109/ICDM.2019.00011>
3. Augsten, N.: A roadmap towards declarative similarity queries. In: Bohlen, M., Pichler, R., May, N., Rahm, E., Wu, S.H., Hose, K. (eds.) Advances in Database Technology - EDBT 2018, pp. 509–512. Advances in Database Technology - EDBT, OpenProceedings.org, January 2018. <https://doi.org/10.5441/002/edbt.2018.59>
4. Brickley, D., Burgess, M., Noy, N.F.: Google dataset search: building a search engine for datasets in an open web ecosystem. In: The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019, pp. 1365–1375. ACM (2019). <https://doi.org/10.1145/3308558.3313685>
5. Bustos, B., Kreft, S., Skopal, T.: Adapting metric indexes for searching in multi-metric spaces. *Multimedia Tools Appl.* **58**, 1–30 (2012). <https://doi.org/10.1007/s11042-011-0731-3>
6. Chapman, A., et al.: Dataset search: a survey. *VLDB J.* **29**(1), 251–272 (2020)
7. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. *ACM Comput. Surv.* **33**(3), 273–321 (2001)

### 3. SIMILARITY VS. RELEVANCE: FROM SIMPLE SEARCHES TO COMPLEX DISCOVERY

---

116 T. Skopal et al.

8. Chen, X., et al.: DataMed - an open source discovery index for finding biomedical datasets. *J. Am. Med. Inform. Assoc.* **25**(3), 300–308 (2018). <https://doi.org/10.1093/jamia/ocx121>
9. Connor, R., Vadicamo, L., Cardillo, F.A., Rabitti, F.: Supermetric search. *Inf. Syst.* **80**, 108–123 (2019)
10. Degbelo, A., Teka, B.B.: Spatial search strategies for open government data: a systematic comparison. *CoRR abs/1911.01097* (2019). <https://arxiv.org/abs/1911.01097>
11. Gkelios, S., Sophokleous, A., Plakias, S., Boutalis, Y., Chatzichristofis, S.A.: Deep convolutional features for image retrieval. *Exp. Syst. Appl.* **177**, 114940 (2021)
12. Gregory, K., Groth, P., Scharnhorst, A., Wyatt, S.: Lost or found? Discovering data needed for research. *Harvard Data Sci. Rev.* **2**(2) (2020). <https://doi.org/10.1162/99608f92.e38165eb>. <https://hdr.mitpress.mit.edu/pub/gw3r97ht>
13. Gregory, K.M., Cousijn, H., Groth, P., Scharnhorst, A., Wyatt, S.: Understanding data search as a socio-technical practice. *J. Inf. Sci.* **46**(4), 459–475 (2020)
14. Grosup, T., Peska, L., Skopal, T.: Towards augmented database schemes by discovery of latent visual attributes. In: Herschel, M., Galhardas, H., Reinwald, B., Fundulaki, I., Binnig, C., Kaoudi, Z. (eds.) *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, 26–29 March 2019*, pp. 670–673. *OpenProceedings.org* (2019). <https://doi.org/10.5441/002/edbt.2019.83>
15. Hetland, M.L., Skopal, T., Lokoc, J., Beecks, C.: Ptolemaic access methods: challenging the reign of the metric space model. *Inf. Syst.* **38**(7), 989–1006 (2013)
16. Koesten, L.: A user centred perspective on structured data discovery. In: *Companion Proceedings of the The Web Conference 2018, WWW 2018, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE*, pp. 849–853 (2018). <https://doi.org/10.1145/3184558.3186574>
17. Košarko, O., Variš, D., Popel, M.: LINDAT translation service (2019). <http://hdl.handle.net/11234/1-2922>, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
18. Kučera, J., Chlapek, D., Nečaský, M.: Open government data catalogs: current approaches and quality perspective. In: Kó, A., Leitner, C., Leitold, H., Prosser, A. (eds.) *EGOVIS/EDEM 2013. LNCS*, vol. 8061, pp. 152–166. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40160-2\\_13](https://doi.org/10.1007/978-3-642-40160-2_13)
19. Lu, W., Hou, J., Yan, Y., Zhang, M., Du, X., Moscibroda, T.: MSQL: efficient similarity search in metric spaces using SQL. *VLDB J.*, 3–26 (2017). <https://www.microsoft.com/en-us/research/publication/msql-efficient-similarity-search-metric-spaces-using-sql/>
20. Miller, R.J., Nargesian, F., Zhu, E., Christodoulakis, C., Pu, K.Q., Andritsos, P.: Making open data transparent: Data discovery on open data. *IEEE Data Eng. Bull.* **41**(2), 59–70 (2018). <http://sites.computer.org/debull/A18june/p59.pdf>
21. Novak, D., Zezula, P., Budikova, P., Batko, M.: Inherent fusion: towards scalable multi-modal similarity search. *J. Database Manage.* **27**(4), 1–23 (2016)
22. Silva, Y.N., Pearson, S.S., Chon, J., Roberts, R.: Similarity joins: their implementation and interactions with other database operators. *Inf. Syst.* **52**, 149–162 (2015). <https://doi.org/10.1016/j.is.2015.01.008>. Special Issue on Selected Papers from SISAP 2013
23. Skopal, T., Bustos, B.: On nonmetric similarity search problems in complex domains. *ACM Comput. Surv.* **43**(4) (2011). <https://doi.org/10.1145/1978802.1978813>

24. Traina, C., Moriyama, A., da Rocha, G.M., Cordeiro, R.L.F., de Aguiar Ciferri, C.D., Traina, A.J.M.: The SimilarQL framework: similarity queries in plain SQL. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, 8–12 April 2019, pp. 468–471. ACM (2019). <https://doi.org/10.1145/3297280.3299736>
25. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer, Boston (2006). <https://doi.org/10.1007/0-387-29151-2>
26. Škoda, P., Bernhauer, D., Nečaský, M., Klímek, J., Skopal, T.: Evaluation framework for search methods focused on dataset findability in open data catalogs. In: Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services, pp. 200–209 (2020)



---

# Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs

Škoda, P. (20 %); Bernhauer, D. (20 %); Nečaský, M. (20 %); Klímek, J. (20 %); Skopal, T. (20 %) Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services, iiWAS '20*, Chiang Mai, Thailand: Association for Computing Machinery, 2020, doi: 10.1145/3428757.3429973. [A.4]

The paper has been cited in:

- Smits, J. J. *Improving integration Platforms as a Service through the addition of enterprise data catalog features*. 2022. Master's Thesis. University of Twente.

## Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs

Petr Škoda  
skoda@ksi.mff.cuni.cz  
Charles University

Department of Software Engineering,  
Faculty of Mathematics and Physics  
Prague, Czech Republic

David Bernhauer  
bernhdav@fit.cvut.cz  
Charles University

Department of Software Engineering,  
Faculty of Mathematics and Physics  
Prague, Czech Republic  
Czech Technical University in Prague  
Department of Software Engineering,  
Faculty of Information Technology  
Prague, Czech Republic

Martin Nečaský  
necasky@ksi.mff.cuni.cz  
Charles University

Department of Software Engineering,  
Faculty of Mathematics and Physics  
Prague, Czech Republic

Jakub Klímeck  
klimek@ksi.mff.cuni.cz  
Charles University

Department of Software Engineering,  
Faculty of Mathematics and Physics  
Prague, Czech Republic

Tomáš Skopal  
skopal@ksi.mff.cuni.cz  
Charles University

Department of Software Engineering,  
Faculty of Mathematics and Physics  
Prague, Czech Republic

### ABSTRACT

Many institutions publish datasets as Open Data in catalogs, however, their retrieval remains problematic issue due to the absence of dataset search benchmarking. We propose a framework for evaluating findability of datasets, regardless of retrieval models used. As task-agnostic labeling of datasets by ground truth turns out to be infeasible in the general domain of open data datasets, the proposed framework is based on evaluation of entire retrieval scenarios that mimic complex retrieval tasks. In addition to the framework we present a proof of concept specification and evaluation on several similarity-based retrieval models and several dataset discovery scenarios within a catalog, using our experimental evaluation tool. Instead of traditional matching of query with metadata of all the datasets, in similarity-based retrieval the query is formulated using a set of datasets (query by example) and the most similar datasets to the query set are retrieved from the catalog as a result.

### CCS CONCEPTS

• **Information systems** → **Web searching and information discovery**; **Similarity measures**; **Resource Description Framework (RDF)**; **Ontologies**; **Recommender systems**.

### KEYWORDS

open data, findability, similarity, catalogs

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*iiWAS '20, November 30-December 2, 2020, Chiang Mai, Thailand*

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8922-8/20/11...\$15.00

<https://doi.org/10.1145/3428757.3429973>

### ACM Reference Format:

Petr Škoda, David Bernhauer, Martin Nečaský, Jakub Klímeck, and Tomáš Skopal. 2020. Evaluation Framework for Search Methods Focused on Dataset Findability in Open Data Catalogs. In *The 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20)*, November 30-December 2, 2020, Chiang Mai, Thailand. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3428757.3429973>

### 1 INTRODUCTION

In order to support public oversight and public sector information reuse, one of the legal duties of public-sector institutions, government and municipality offices is to publish data related to their services. Some of the data is wrapped into datasets and published on the Web as Open Data. The datasets are described by metadata provided by the publisher and are registered in catalogs such as the European Data Portal (EDP)<sup>1</sup>, or the Czech National Open Data Catalog (NODC)<sup>2</sup>. The catalogs provide a basic search functionality, however, low findability<sup>3</sup> of datasets still remains a major issue. The reason is the metadata is very brief, often reduced to a few keywords, a short description and title. Also, due to the distributed nature of open data the datasets are acquired in different contexts and unrestricted domains, with different background knowledge of individual data publishers, and in different languages and/or domain-specific slangs. In consequence, the low-level search methods provided by the catalogs based on keyword search, structured queries or entity search, are often insufficient [4]. To demonstrate the problem, in Figure 1 see a visualization of NODC dataset publishers represented by green nodes connected to keywords they use in dataset metadata represented by black and red nodes. Black nodes, a clear majority, represent keywords unique to individual

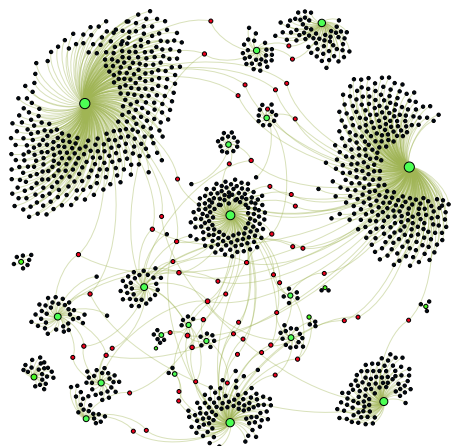
<sup>1</sup><https://www.europeandataportal.eu>

<sup>2</sup><https://data.gov.cz/english/>

<sup>3</sup>Note that findability is one of four key FAIR principles of data management [23] – findability, accessibility, interoperability, and reusability.



publishers, while red nodes represent keywords shared by at least two publishers. Hence, the datasets of different publishers cannot be simply matched based on keywords alone.



**Figure 1: Dataset publishers from NODC (green nodes) and their usage of unique (black nodes) and shared keywords (red nodes) in dataset metadata.**

A natural step to improve the findability of datasets is to develop alternative retrieval and integration models, including augmentation of the metadata by other contexts, such as embedding into knowledge bases (as motivated in [18]). However, to the best of our knowledge there does not exist an evaluation method (not even a benchmark) for datasets from open data catalogs that would enable researchers to compare alternative dataset retrieval models to the baseline search engines provided by the catalogs.

### 1.1 Problem Motivation

In this paper we propose a framework for dataset retrieval evaluation, given the specific properties of datasets mentioned above. We must emphasize that retrieval of datasets for a particular user scenario is much harder than, for example, retrieval of a document from a collection of texts. In the well-established field of text or multimedia retrieval, there exists a number of evaluation techniques and benchmarks, such as TREC [22], where the retrieval is supposed to work with full information that allows (almost) perfect findability, e.g., the content-based search. For example, when searching for images of dogs in a collection of photographs, the relevant images actually contain the dogs. However, in the case of datasets, the semantics is often not captured in the metadata at a level sufficient for perfect match. The users of data catalogs are forced to use low-level queries the catalog interfaces provide, but they would rather search datasets relevant to a complex scenario that cannot be simply decomposed into a set of low-level queries, e.g., search for datasets

important for zoning decision when a highway is to be built. Such a discrepancy is also mentioned in a recent dataset search survey [4] that calls these scenarios *constructive dataset search* and provides some examples. In consequence, a viable evaluation method cannot abstract from the user scenarios by assembling a testbed of datasets annotated by classic ground truth labels for a given set of prepared queries. Instead, the user scenarios and user interaction must enter the evaluation procedure.

### 1.2 Paper Contributions

In this paper, we present two contributions. First, we propose a general evaluation framework for dataset retrieval based on arbitrary retrieval models that provide ranking (section 3). The framework is centered around user scenarios representing the necessary domain-specific evaluation contexts. We also present an evaluation tool for human experts that provides interface for rapid and intuitive user evaluations within the framework. Second, we present a proof of concept specification and evaluation of the framework, employing various similarity-based retrieval models (applied to datasets' metadata) and the TLSH method used by EDP as a baseline model (section 4 and section 5).

## 2 RELATED WORK

Finding related datasets, or shortly data discovery, is one of the important tasks in data integration [14]. It has been recently recognized as a research field with its unique challenges and open questions [4]. In this section, we summarize recent research papers in the field and we also specifically address how dataset retrieval models introduced in these papers were evaluated.

### 2.1 Finding related datasets

Two important data discovery problems need to be solved [14]: finding joinable datasets and finding unionable datasets where the user starts with a dataset or datasets and searches for other related datasets which can be joined or unioned with the original ones.

As shown in [14] many existing dataset discovery solutions are based on simple keyword query search. This is what is typically implemented in open data portals such as NODC or EDP. There are also open data portal mash-ups. For example, EDP collects metadata records about open datasets from national portals of individual member states and provides search features across the whole European Union. Besides the basic keyword query search, EDP also offers a discovery feature based on dataset similarity. According to source code published at GitLab<sup>4</sup>, the portal uses TLSH presented in [16]. Firstly, they concatenate the title and description of the dataset. The locally sensitive hash is constructed from the concatenated string, which should produce a similar hash for a similar dataset, and these hashes are compared. Originally, this method [9] was implemented as a technique for searching duplicate (almost equal, ignoring typing errors) datasets.

Several novel techniques for dataset discovery have been proposed in literature in the last years. In [10], the authors propose Aurum, a system to build, maintain and query an enterprise knowledge graph (EKG) which represents datasets and their structural

<sup>4</sup><https://gitlab.com/european-data-portal/metrics/edp-metrics-dataset-similarities/-/blob/master/src/main/java/io/piveau/metrics/similarities/SimilarityVerticle.java>

## 4. EVALUATION FRAMEWORK FOR SEARCH METHODS FOCUSED ON DATASET FINDABILITY IN OPEN DATA CATALOGS

---

elements, e.g., table columns, as nodes and relationships between them as edges. A relationship between two structural elements may represent content similarity, schema similarity, e.g., similarity of names of the columns, or key/foreign key pairs defined in the dataset schemas. The paper introduces an efficient model which exploits EKG. Moreover, the introduced technique requires only a linear passage through datasets to build EKG. Dataset discovery is then performed on top of EKG. When a user selects a dataset, the tool offers other relevant datasets through the relationships in EKG. Similarly, [1] proposes a technique based on content and schema similarity. For schema similarity, the approach considers similarity of column names. For content similarity, the approach considers various similarity models, e.g., based on value embeddings.

Besides approaches which analyze dataset's structure and content there are also approaches that perform dataset discovery on metadata about datasets, e.g., their titles, descriptions and characterizing keywords. For example, Google Dataset Search [2] belongs to this group. The authors explain in the paper how dataset metadata is crawled from the Web and cleansed. The metadata is then mapped to the Google's knowledge graph which is then used for dataset duplicates detection and for dataset discovery. Another example of metadata-based approaches is [8] where authors evaluate four different metadata-based models for searching spatially related datasets, i.e., datasets which are related because of the same or similar spatial area covered. The first models is a full-text search model. The second one parses and geocodes user's query. The other two models map user's query to knowledge graphs, WordNet [21] and ConceptNet [19], enrich the query with the neighbourhoods from these knowledge graphs and use the result for the full-text search.

### 2.2 Evaluation of dataset retrieval

Many of the existing works on dataset discovery introduce predefined usage scenarios and use the scenarios either to explain their approaches or for some kind of evaluation. The paper [14] explains data discovery problems of finding joinable and unionable datasets on two scenarios of a data journalists searching for research data. There is no true evaluation in the paper. The user scenarios are presented only as a demonstration of proposed dataset discovery techniques. Also Google Dataset Search [2] does not present any systematic evaluation. The paper only shows that user's feedback was collected without any further details about the feedback collection process. A discussion about user's requirements then follows based on the collected feedback.

The evaluation of the Aurum system [10] defines three usage scenarios in three real companies. For evaluation, the authors conducted a survey with 4 users. The users used the system as defined by the scenarios. The authors asked them to rate the usefulness of Aurum's discovery, time savings compared to manual discovery, and how likely they would use the tool in their everyday work. The users answered these questions by picking a number between 0 and 5 where 0 is the worst evaluation.

An attempt to evaluate dataset discovery systematically can be found in [8]. Here, the authors define 4 simple full-text search queries and ask users to evaluate relevance of the discovered dataset w.r.t. the queries. Another systematically oriented evaluation can be found in [1]. The paper defines a ground truth consisting of

two parts. The first part is a synthetic ground truth which consists of approx. 5,000 tables synthetically derived from 32 base tables extracted from Canadian open government data. The tables were constructed by random projections and selections of the source tables. Two datasets are considered related when one was derived from the other. The second part is a real ground truth consisting of approx. 700 UK open government datasets. Relationships between the datasets were defined manually by a human expert. Two datasets are considered related when the human expert defined a relationship between them. The precision and recall of the dataset discovery algorithm  $D^3L$  introduced in the paper are measured.

The last approach to dataset discovery evaluation could be considered as an evaluation framework which could be reused to evaluate other approaches as well. However, as we point out in this paper, it is crucial to consider different usage scenarios and different users who may have different needs and background knowledge. The evaluation method presented in [1] however does not consider different scenarios and users.

## 3 EVALUATION FRAMEWORK

As detailed in Section 2.2, the recent works in the field of dataset discovery do not evaluate their dataset retrieval models systematically nor based on user scenarios. However, systematic evaluation based on user scenarios is crucial to demonstrate their usefulness. This finding led us to design and construct our own framework for evaluating dataset retrieval models. Before we introduce the framework itself, let us summarize important prerequisites of dataset discovery which are implied by the motivation in Section 1.1 and findings of the recent literature in the field presented in Section 2.1.

First, research papers in the field motivate their work with user scenarios. The user in such scenarios is always a professional who is able to process raw data, e.g., a data scientist, data journalist or researcher. The user is not simply a common user supposed to search web pages and now switching to search datasets. Second, different professionals solve different problems. Therefore, they have different expectations of what is discovered given the problem. Their concrete discovery problem is crucial because it influences how a discovery method should be evaluated. As it is written in Section 2.1, the user starts with initial datasets and expects that other related datasets needed to solve their problem are returned by the discovery. However, what datasets are relevant is influenced by both the user and the problem. In the literature, the best way to describe concrete dataset discovery problems is to describe them as user scenarios. This shows that evaluation based on a universal ground truth cannot provide the real picture of performance of a retrieval model.

A universal ground truth cannot reflect different users and different discovery problems. A human evaluation simulating concrete dataset discovery scenarios is necessary. This was also confirmed by our early attempt to create a universal ground truth (i.e., scenario-independent) for datasets by means of labeling a set of query datasets. In this attempt, datasets were labeled by users as relevant or not to a set of query datasets using three scores of relevance (relevant, somewhat relevant, not relevant). We have found out that users were consistent in labeling the datasets based on low-level similarity (such as keyword matching). However, when asked

for labeling high-level relevance, such as good complementarity with query dataset, the users started to develop their own personal ad-hoc scenarios, creating actually a universe needed to construct a complement (in mathematical meaning). This led to wild, non-systematic evaluations and inconsistent labeling across the users. In conclusion of the early attempt, we have realized the evaluation must take into account user scenarios (and user experts familiar with them) in order to instruct the evaluators using a well-defined evaluation protocol and to prevent trivial "everything-is-related-to-everything" user reasoning.

Hence, we resorted to development of evaluation framework that bounds the users (evaluators) with user scenarios. This defines the following requirements on a dataset discovery evaluation framework:

- The framework must enable evaluators to evaluate dataset retrieval models on different user scenarios expressed as a set of steps with a prescribed structure so that the scenarios are specified in the same structure and the same level of detail.
- The framework must support evaluation by different human evaluators who go through the scenarios and specify the performance of the evaluated models in their context.
- The framework must combine evaluation results of the evaluators from different scenarios.
- A graphical user interface is necessary to enable the evaluators to work effectively as there may be 10s of scenarios, 10s of retrieval models and 10s of datasets returned by the models which means 100s - 1000s of artifacts which need to be evaluated.

As we have shown in Section 2.2, none of the recent works in the dataset discovery field evaluate their dataset retrieval models in this systematic way considering predefined structured scenarios and different human evaluators. Our goal is to create an evaluation framework which would be as simple as possible while still fulfilling the requirements specified above.

### 3.1 Evaluation Methodology

Our evaluation framework is based on usage scenarios. In each scenario we consider a user with the goal to find datasets needed to, e.g., build a service such as a mobile or a map based application. The user starts either by entering keyword search queries or by browsing a data catalog. In both cases, the user finds initial datasets. The user then explores each initial dataset and finds out that they do not contain all necessary data. This is inevitable as the user is not able to find datasets completely covering the goal using just simple search queries or by simply browsing the catalog. As the initial datasets are not complete, the user needs to continue searching to find other complementary datasets which fill in what is missing in the initial ones. One possibility is a manual search where the user iterates the previous steps. The user analyzes manually what is missing in the initial datasets. This leads to other queries or to browsing the catalog again. The second possibility is that the catalog offers the other datasets automatically, using the initial datasets as queries (i.e., the query-by-example concept). This would save a lot of user's manual effort.

Our goal is to show how different retrieval models may fulfill the second (query-by-example) possibility. To evaluate the models we define scenarios considering the first possibility. Each scenario  $S$  has the following structure:

- $G$ : goal specification
- $p^{init}$ : process of finding the initial datasets
- $Q$ : the set of initial datasets (query datasets)
- $panalyze$ : process of analyzing the initial datasets and search for the complementary datasets manually
- $C$ : the list of complementary datasets found manually

We evaluate each retrieval model  $\mathcal{M}$  using the set  $D$  of all datasets in the catalog and each defined scenario according to the following methodology.

- (1) The set of query datasets  $Q$  is taken as the input.
- (2) For each dataset  $d \in D$  its rank is computed as  $rank_{\mathcal{M}}(d, Q)$ .
- (3) Top  $K$  datasets in  $D$  with the smallest  $rank_{\mathcal{M}}(d, Q)$  are returned as  $C_{\mathcal{M}}$ .
- (4)  $C_{\mathcal{M}}$  is compared with  $C$  automatically and by human experts.

In our evaluation we show how different retrieval models are appropriate for the second option. We consider automated as well as human-based comparison. This is important as the automated comparison gives us results quickly and consistently across the scenarios but it cannot reflect the fact that each human user searches for datasets with a different problem in mind and with different background knowledge. Therefore, we consider only the human-based comparisons in the rest of this paper.

### 3.2 Evaluation Tool for Human Experts

In order to support the evaluation process we implement a web-based application <sup>5</sup> (see its layout in Figure 2).

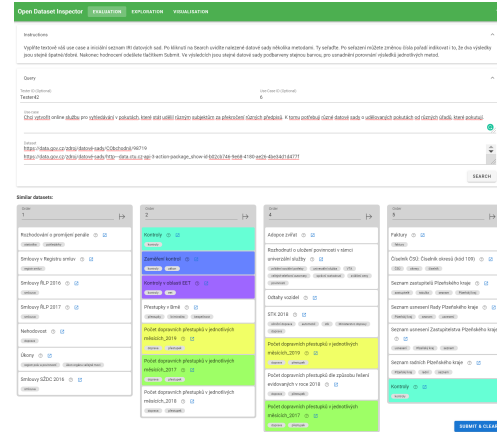


Figure 2: Evaluation user interface layout.

<sup>5</sup><https://github.com/mff-uk/open-dataset-inspector>

## 4. EVALUATION FRAMEWORK FOR SEARCH METHODS FOCUSED ON DATASET FINDABILITY IN OPEN DATA CATALOGS

As we expect the evaluation framework to consist of multiple scenarios, the application must support evaluation of different scenarios. Given a scenario  $S$  that consists of a set of query datasets  $Q$ , a goal specification  $G$ , and expected complementary datasets  $C$  found manually by the scenario author, the application presents the set of the most similar datasets  $C_M$  to the user for each evaluated model  $M$ . The user is asked to rank the models by reordering them from best to worst while taking  $C$  into account. As two models might be same in the eyes of the user, we allow the user to provide the same ranking for multiple models.

The basic evaluation workflow consists of the following steps:

- (1) Fill in the user's identification.
- (2) Provide the goal specification  $G$ .
- (3) Fill in the set of query datasets  $Q$ .
- (4) Click on the "search" button.
- (5) Revise for each evaluated model the most similar returned datasets. Take  $C$  into account for the revision.
- (6) Rank the models by reordering them using drag&drop or by text input specifying the ordering.
- (7) Click on the "submit & clear" button.

Although the workflow is simple and straightforward, the amount of datasets which must be revised by the user in Step 5 can be overwhelming. To tackle this issue, we utilize the fact that the models often return the same datasets. Being able to spot those shared datasets easily would make the user's work more effective. We solve this by using colors. When a single dataset is returned by more than one model it is colored with a unique color (see Figure 2). Therefore, the user may easily visually identify the presence and position of a shared dataset across different models. When the dataset is relevant, the user may search whether a dataset with the same color appears in  $C_{M'}$  of another model  $M'$ , which helps with deciding about the ranking.

Another issue may arise with higher number of models. A systematic bias may be introduced by the initial model ordering. We mitigate this by randomly changing the initial model ordering.

Once the user clicks on the "submit & clear" button, we save the user identification, the user scenario identification and the ranking.

### 4 PROOF OF CONCEPT SPECIFICATION

In the previous section we introduced the structure of a user scenario and showed a simple evaluation methodology. We have also presented a user interface for human evaluators. In this section, we specify a PoC (proof of concept) to demonstrate how our framework works in practice.

#### 4.1 Input Data

For our experiments, we use metadata from the Czech National Open Data Catalog (NODC)<sup>6</sup> [11], which is regularly harvested by the European Data Portal (EDP). This metadata describes open datasets published by public institutions in Czechia, such as the Czech Statistical Office<sup>7</sup>, the city of Prague<sup>8</sup> or the Czech Social

Security Administration<sup>9</sup>. The metadata uses DCAT-AP<sup>10</sup>, an RDF-based [5] vocabulary based on the W3C Recommendation DCAT [3], and a European recommendation for metadata in European data portals.

According to DCAT, a dataset is *A collection of data, published or curated by a single agent, and available for access or download in one or more representations*.<sup>11</sup> The dataset metadata record according to DCAT-AP can contain various fields, some textual, some with values from code lists from, e.g., the EU Vocabularies<sup>12</sup>. However, for our experiments so far, we have used the most basic, textual metadata fields: *title*, *description* and *keywords*, which contain textual descriptions of the datasets in Czech. The collection we work with contains approximately 6600 datasets from 39 publishers.

In our experiments, we view the dataset metadata from the NODC as just one of many *dataset contexts*. Another dataset context we use is an embedding of the dataset metadata texts in a Word2Vec model trained on all texts from the Czech Wikipedia [7]. Yet another dataset context we use is an embedding of the dataset metadata texts in a Word2Vec model trained on texts from the Czech legislation [6]. Both models are trained using the Python `gensim` [17] implementation of Word2Vec [13] with the following parameters in addition to default settings: vector dimension = 400, window size = 10, word minimum count = 10, sample =  $10^{-5}$ , the best of the evaluated configurations. Czech Wikipedia provides us with a dataset context which extends our knowledge about datasets with the general encyclopedic knowledge. Czech legislation defines the domain of Czech public sector institutions, their competences and duties. Under these competences and duties public institutions publish their open data and catalog them in NODC. Theoretically, Czech legislation provides us with a dataset context which extends our knowledge about datasets with specification about legal reasons why datasets exist, under which circumstances publishers create and publish datasets and about semantic relationships of datasets to other legal concepts. In our PoC we therefore include this as one of the external sources of knowledge. For training the Word2Vec model based on Czech legislation we used a publicly available structured representation of the Czech code of law<sup>13</sup> [15].

#### 4.2 Similarity-based Retrieval Models

In our experiments, we compare 12 similarity-based retrieval models (similarity models in short), including TLSH similarity used by EDP. The remaining models are based on one of the metadata fields: *title*, *description*, or *keywords*. Every model consists of a descriptor, i.e., preprocessed metadata record, and a distance function representing inverse similarity. Table 1 presents the used similarities. Table 2 gives the list of the used models. According to Section 3.1,  $rank_M(d, Q) = \min_{q \in Q} (dist_M(d, q))$  where  $dist_M(d, q)$  is the distance between  $d$  and  $q$  returned by similarity model  $M$ .

The keywords-based model works with a set of phrases and uses Jaccard distance. A phrase is a sequence of words separated by space compared as one string. Title and description are divided into words,

<sup>6</sup><https://data.gov.cz>

<sup>7</sup><https://data.gov.cz/datasets?publisher=Český%20statistický%20úřad>

<sup>8</sup><https://data.gov.cz/datasets?publisher=HLAVNÍ%20MĚSTO%20PRAHA>

<sup>9</sup><https://data.gov.cz/datasets?publisher=Česká%20správa%20sociálního%20zabezpečení>

<sup>10</sup><https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semantic/solution/dcat-application-profile-data-portals-europe>

<sup>11</sup><https://www.w3.org/TR/vocab-dcat-2/#Class:Dataset>

<sup>12</sup><https://op.europa.eu/en/web/eu-vocabularies/authority-tables>

<sup>13</sup><https://esbirka.opendata.cz/sparql>

Similarity	Domain	Formula (distance form)
cosine	vectors	$dist(\vec{x}, \vec{y}) = \begin{cases} 1 - \frac{\vec{x} \cdot \vec{y}}{ \vec{x}   \vec{y} }, & \text{for }  \vec{x}  \neq 0 \wedge  \vec{y}  \neq 0, \\ +\infty, & \text{otherwise.} \end{cases}$
jaccard	sets	$dist(X, Y) = \begin{cases} 1 - \frac{ X \cap Y }{ X \cup Y }, & \text{for }  X \cup Y  \neq 0, \\ 0, & \text{otherwise.} \end{cases}$
hausdorff	sets of vectors	$dist(X, Y) = \max \left\{ \begin{array}{l} \sup_{x \in X} \inf_{y \in Y} d(x, y) \\ \sup_{y \in Y} \inf_{x \in X} d(x, y) \end{array} \right\}$ where $d(x, y)$ is ground distance (cosine in our case)

Table 1: Similarity overview

and these words are simplified into lemma using UDPipe [20]. For the title, we have constructed the set of words and used Jaccard distance. In the case of the description, we have created a vector represented by the bag of words model. Each word represents one dimension in vector, and the value is the number of words in the description. These vectors are compared by cosine distance.

A more robust approach is embedding words into vector space using the Word2Vec. We have trained two Word2Vec models, based on Czech Wikipedia (suffix [cswiki]) and Czech legislation (suffix [law]). Extending metadata by Word2Vec embedding adds external knowledge, which allows matching similar words where equality is not sufficient. For every word, we have found a corresponding vector in the model (if it exists). For models with Hausdorff distance, we have used this set of vectors directly as the descriptor. In the case of cosine distance, we have constructed the average vector.

Model identifier = similarity:metadata	Descriptor type
tlsh : title description	concatenated string
jaccard : title	set of words
cosine : title [cswiki]	avg. W2V vector
cosine : title [law]	avg. W2V vector
hausdorff : title [cswiki]	set of W2V vectors
hausdorff : title [law]	set of W2V vectors
jaccard : keywords	set of keywords
cosine : description	bag of words
cosine : description [cswiki]	avg. W2V vector
cosine : description [law]	avg. W2V vector
hausdorff : description [cswiki]	set of W2V vectors
hausdorff : description [law]	set of W2V vectors

Table 2: Similarity models used in experiments

### 4.3 Users

5 users participated in the evaluation. We will refer to the users using  $U_1 \dots U_5$ . The users  $U_1$  and  $U_2$  are open data experts as they cooperate with different public institutions in Czechia and help them with publishing their datasets. They are also the authors of the user scenarios. The user  $U_3$  is also related to the open data field as he is the main programmer of the open source software LinkedPipes DCAT-AP viewer [12] which is used by the Czech Ministry of the Interior to run the NODC. The users  $U_4$  and  $U_5$  are

academic researchers in the field of similarity search in general. They are new to the field of open data.

### 4.4 User Scenarios

We defined 12 user scenarios  $S_1 \dots S_{12}$  for the evaluation. Each scenario has the structure defined in Section 3.1.

Table 3 shows some basic characteristics of the scenarios. It shows the author and the size of  $Q$  and  $C$  for each scenario. Moreover, it classifies the scenarios according to the discovery problem which needs to be solved in each scenario. This is motivated by [14] which distinguishes two discovery problems. First, finding joinable datasets means that the user searches for related datasets which can be joined with the original ones ( $\bowtie$ ). Second, finding unionable datasets means that the user searches for related datasets which can be unioned with the original ones. In addition to [14] we distinguish the problem of unioning datasets representing the same type of entities ( $\cup$ ) and unioning datasets representing different types of entities but sharing some common semantics ( $\cup$ ).

ID	Problem	Author	Q	C
$S_1$	$\bowtie$	$U_1$	1	5
$S_2$	$\cup$	$U_1$	1	2
$S_3$	$\cup$	$U_1$	1	3
$S_4$	$\cup$	$U_1$	1	1
$S_5$	$\bowtie$	$U_1$	1	5
$S_6$	$\cup$	$U_1$	2	3
$S_7$	$\cup$	$U_2$	3	8
$S_8$	$\cup$	$U_2$	2	7
$S_9$	$\cup$	$U_2$	3	7
$S_{10}$	$\cup$	$U_1$	6	25
$S_{11}$	$\cup$	$U_1$	1	27
$S_{12}$	$\cup$	$U_1$	1	3

Table 3: Characterization of evaluation scenarios

In the following section, we show  $S_1$  in its full detail. *pinit* and *panalyze* show how  $Q$  and  $C$ , respectively, were identified when we prepared the scenarios. However, they are not important for the evaluation itself. Therefore, for the rest of the scenarios, we present only  $G$ ,  $Q$  and  $C$ .

## 4. EVALUATION FRAMEWORK FOR SEARCH METHODS FOCUSED ON DATASET FINDABILITY IN OPEN DATA CATALOGS

4.4.1 **S<sub>1</sub> - Registered trademarks.**  $G_{S_1}$ : "The user intends to build an online service which shows companies and their registered trademarks."

$P_{S_1}^{init}$ :

- (1) The user searches for datasets using a keyword query "trademarks".
- (2) NODC returns a list of datasets with a dataset *Trademarks - full export*.
- (3) The user discovers the dataset *Trademarks - full export* in the list.

$Q_{S_1}$ : { *Trademarks - full export* }

$P_{S_1}^{analyze}$ :

- (1) In the documentation, the user finds out that there is only identification of the owners of the trademarks. Details about the owner companies are in other datasets.
- (2) The user searches for company datasets manually using different keywords. To discover all relevant datasets, the user needs to know possible types of companies as these are used in the metadata of these datasets.

$C_{S_1}$ : { *Joint-stock companies, Limited companies, Institutes, Foundations, Funded organizations* }

4.4.2 **S<sub>2</sub> - Intellectual property protection.**  $G_{S_2}$ : "The user needs to build an online service for searching intellectual property and how companies protect it. However, the user does not exactly know what kind of intellectual property can be protected."

$Q_{S_2}$ : { *Inspections in the field of intellectual property rights* }

$C_{S_2}$ : { *Trademarks - full export, Patents and utility models - full export* }

4.4.3 **S<sub>3</sub> - Medical facilities.**  $G_{S_3}$ : "The user needs to create a map of all medical facilities of all possible kinds in the country."

$Q_{S_3}$ : { *Ambulances* }

$C_{S_3}$ : { *Pharmacies and medical facilities in Prague, Medical facilities in Pilsen, National registry of healthcare providers* }

4.4.4 **S<sub>4</sub> - Map of town districts.**  $G_{S_4}$ : "The user wants to create an overview of town districts in the country."

$Q_{S_4}$ : { *Districts of Prague* }

$C_{S_4}$ : { *Parts of the city of Pilsen* }

4.4.5 **S<sub>5</sub> - Sanctioned companies.**  $G_{S_5}$ : "The user wants to build an online service which shows companies sanctioned because of their bad business practices."

$Q_{S_5}$ : { *Sanctions by Czech Trade Inspection Authority* }

$C_{S_5}$ : { *Joint-stock companies, Limited companies, Institutes, Foundations, Funded organizations* }

4.4.6 **S<sub>6</sub> - Sanctions given to companies.**  $G_{S_6}$ : "The user wants to build an online service for searching sanctions of different kinds given to companies. These sanctions can be, e.g. sanctions for bad business practices or for violations of the rights of employees."

$Q_{S_6}$ : { *Sanctions by Czech Trade Inspection Authority, Inspections and sanctions in electronic communications* }

$C_{S_6}$ : { *Technical inspections of motor vehicles, Building construction supervision, Inspections of social security duties of employers* }

4.4.7 **S<sub>7</sub> - Building a house in Brno city.**  $G_{S_7}$ : "The user wants to build a house in the city of Brno and is interested in all the staff relevant to building, housing and living in the location."

$Q_{S_7}$ : { *Housing research in Brno 2019, Housing of youngsters 2017, People and housing* }

$C_{S_7}$ : { *Building permits statistics - national statistics, Codelist of building authorities, Criminality in Brno, Elementary schools in Brno, Kindergartens in Brno, Safety in Brno, Memorable trees in Brno, City parks* }

4.4.8 **S<sub>8</sub> - Floods.**  $G_{S_8}$ : "The user intends to create an application for mapping floods in the country."

$Q_{S_8}$ : { *Flood areas in the city of Děčín, Floods in 19th century in Pilsen* }

$C_{S_8}$ : { *Flood 2002 in Pilsen, 5-year flood in Pilsen, 20-year flood in Pilsen, 100-year flood in Pilsen, Flood areas in Prague 2013, Flood areas of Vltava and Berounka rivers, Flood protection on Vltava and Berounka rivers* }

4.4.9 **S<sub>9</sub> - Thieves.**  $G_{S_9}$ : "The user needs to create a map with an overview of places with something valuable and can be potentially stolen by a thief."

$Q_{S_9}$ : { *Waste depots in Pilsen, Railways in Pilsen, Archeological locations in Pilsen* }

$C_{S_9}$ : { *Electrical waste depots in Pilsen, Railway tracks in the Ostrava city, Tram tracks in the Ostrava city, Tram lines in Pilsen, Fire hydrants in Pilsen, Cable-lines in Pilsen, Bombing the city of Brno in WW2* }

4.4.10 **S<sub>10</sub> - Statistics of mortality, morbidity and health in population.**  $G_{S_{10}}$ : "The user intends to build an application which will show mortality, morbidity and health statistics on a map."

$Q_{S_{10}}$ : { *Deaths in Czechia, Incapacity for work due to illness, Life expectancy in Czech districts by Czech statistical office (4 datasets for years 2015 - 2018)* }

$C_{S_{10}}$ : { *COVID-19 statistics (5 datasets), Sports injuries statistics, Injuries at work statistics, Traffic accident injuries statistics, Injuries at home statistics, Life expectancy by the Ministry of health, Standardized mortality (6 datasets per disease groups), Statistical measures of working disabilities (6 datasets with different statistical dimensions), Number of sickness benefits paid from social security insurance (3 datasets per different levels of administrative territories)* }

4.4.11 **S<sub>11</sub> - State of the climate.**  $G_{S_{11}}$ : "The user intends to create an application about the state of climate in different country regions."

$Q_{S_{11}}$ : { *Climate in Prague - basics* }

$C_{S_{11}}$ : { *Discharge of waste into water and water sampling (4 datasets for different rivers), Amount of landfilled and incinerated waste, Meteorological characteristics in Brno, Measuring stations in Pilsen, Air quality in Pilsen, Health and environment - Brno, Planted and remediated trees in Brno, Air pollution, Development of air pollution by NO<sub>2</sub> and PM<sub>10</sub>, REZZO2 - middle sources, REZZO1 - big sources, Meteostations, Air quality in Pilsen, History and actual pollution data from smart street lamps in Praha Karlín district, Costs of environment protection and their economic impact in Czech districts, Climate in Prague - details (6 datasets from different quality viewpoints), Costs of environment protection in Brno, Development* }

of air pollution by NO2 and PM10 in Brno, Potential of green roofs in Prague}

4.4.12  $S_{12}$  - **Food production.**  $G_{S_{12}}$  : "The user wants to create an overview of the state of production of food in the country."

$Q_{S_{12}}$  : { Harvest of crops in districts}

$C_{S_{12}}$  : { Production of meat, State of cattle and pigs, Sowing areas in Czechia}

## 5 PROOF OF CONCEPT EVALUATION

The output of the user evaluation is a ranking of given models for a given usage scenario. We do not restrict the user in the ranking they can set. As a result, we need to standardize the ranking first, before we draw any conclusions from them.

Let us assume that  $r_i \in R$  is a ranking provided by the user. We compute the normalized ranking as follows:

$$\bar{r}_i = -(r_i - \text{baseline}(R)) / \text{scale}(R) \quad (1)$$

where  $\text{baseline}(R)$  is the score of the baseline model for the given user in a given user scenarios and

$$\text{scale}(R) = \max |r_i - \text{baseline}(R)|_{r_i \in R} \quad (2)$$

We utilize  $tlsh$  : title description as the baseline model. The  $r_i$  values are in the  $< -1, 1 >$  interval where the higher the value, the better.

As we can see from Figure 3, almost all the models outperform the baseline model implemented by the EDP. The performance is relative to the  $tlsh$  : title description reference model, that is why the performance of this model is constantly zero. Most of the models were at least once ranked as one of the best model, with relative ranking equal to one. In similar fashion almost all of the models were at least once ranked as the worst, or as being in group with the worst ranking models. The  $\text{cosine:}title$  [cswiki] and  $\text{jaccard:}title$  are the two overall best ranking models. But rather than overall performance of models we are interested in performance per user and per scenario.

### 5.1 Results per User

Let us first focus on the performance per user. In order to do so, we compute a model performance for each user as an average performance of the model across all scenarios evaluated by the user. In addition, we scale the averages of each user so that the best model's score equals to 1.

As we can see from Figure 4, there are 4 groups of models. The first group created by  $\text{cosine:}title$  [cswiki] and  $\text{jaccard:}title$  is the best performing. The next group created by  $\text{cosine:}description$  [cswiki],  $\text{cosine:}description$  [law],  $\text{cosine:}title$  [law],  $\text{cosine:}description$  seems to be the second best performing.

It can be seen that the groups are consistent across users. That means that, in spite of the differences among the users, i.e., their background knowledge as discussed earlier in the paper, an agreement on performance of the models was achieved thanks to the predefined scenarios used for the evaluation.

The users had to order 12 models. Together with 7 datasets per model, this may not be a trivial task. It happened that some users invested their effort only into ranking a few best models, giving the rest of the models the same ranking, even though they were qualitatively different. This might have been caused by fatigue from

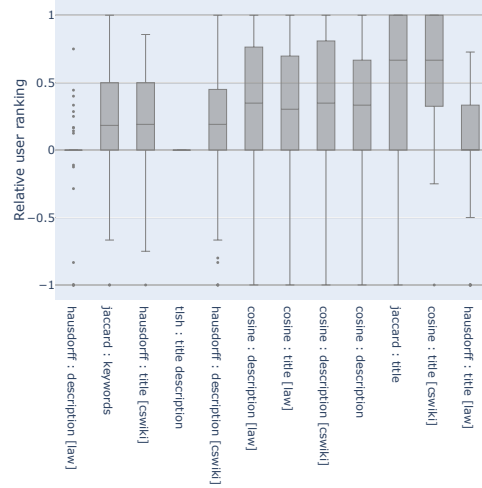


Figure 3: Box plot of normalized model scores

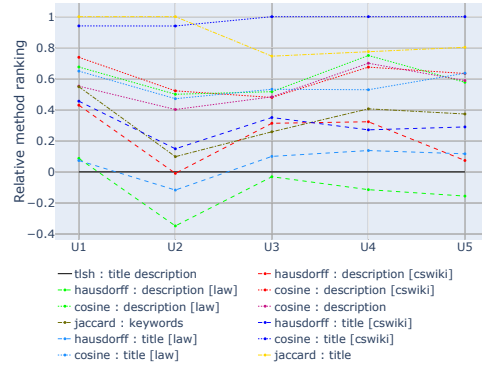


Figure 4: Model performance per user as normalized mean of relative user ranking

the evaluation process or by the user feeling that it is only the best performing models that matter.

In order to gain insight into the process of how users evaluate the models, we counted for how many scenarios a user used a given ranking and plot those values Figure 5. We can see that there are two groups. The  $U_5$  and  $U_4$  tend to rate with finer distinction among the models then the other users, i.e.,  $U_5$ ,  $U_4$  use 12 ranks while other users use 7-9. Another explanation might be that the second group of users starts to see models as bad much faster and does not distinguish how bad the models are after a certain threshold.

## 4. EVALUATION FRAMEWORK FOR SEARCH METHODS FOCUSED ON DATASET FINDABILITY IN OPEN DATA CATALOGS

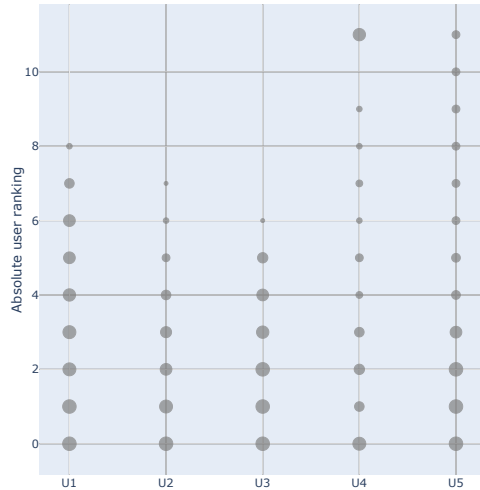


Figure 5: Counts of rankings categories used by users. Bigger dots mean more frequently used ranking. More dots mean finer ranking.

This might be a reason why better rated models are rated more consistently by the users than the worst performing ones.

### 5.2 Results per scenario

While it is easy to spot the best scoring model on per-user basis, this is not true if we consider performance over scenarios in Figure 6. As we can see there is no single best performing model. Even when we consider the two best performing models on per user basis the *cosine : title [cswiki]* and *jaccard : title*, they achieve the best performance only in total 6 scenarios (see Figure 7 and Figure 8).

It is thus clear that a model performance depends heavily on the scenarios at hand. At the same time, if we consider *cosine : title [cswiki]* and *jaccard : title* there is no clear connection between their performance and use-case characteristics (see Table 3). This suggests that a model performance does not strictly depend on the type of the problem, size of the query  $Q$  or number of complementary datasets  $C$ , but rather on the scenario itself. However, detailed interpretation of the results is out of scope of this manuscript.

If we consider the used metadata (see Table 2), then description based models score best for 3, keyword based for 1 and title based for 7 scenarios. This suggests that different metadata are suitable for different scenarios. At the same time it shows that users actually consider all available metadata during the evaluation.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a general framework for evaluating findability of datasets. The framework provides evaluation of arbitrary dataset retrieval models within defined user scenarios. We

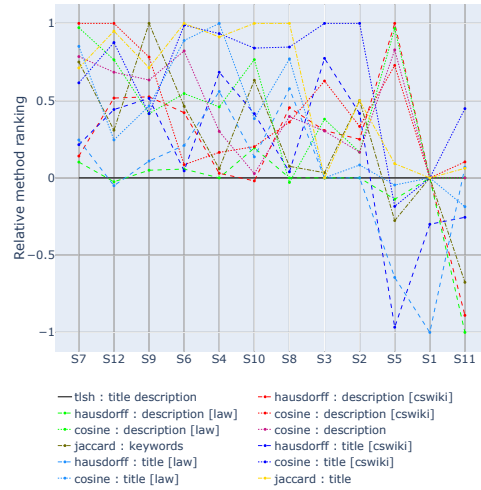


Figure 6: Normalized average performance per model

also presented an evaluation tool for human experts that provides interface for rapid and intuitive user evaluations within the framework. Finally, we presented a proof of concept specification and evaluation of the framework, employing various similarity-based retrieval models applied to datasets' metadata. In the proof of concept we verified that suitability of a given retrieval model is heavily dependent on particular user scenario and the expertise of the user/evaluator.

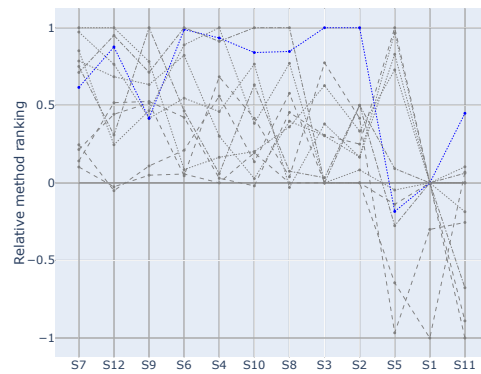
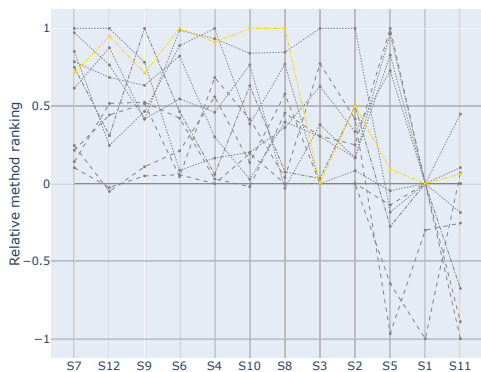


Figure 7: Normalized average performance per model with highlighted *cosine : title [cswiki]* model





**Figure 8: Normalized average performance per model with highlighted *jaccard : title* model**

The evaluation framework introduced in this paper can also be used as the base for a full dataset discovery benchmark in the future. The lack of such benchmark is one of the most widely recognized problems of the dataset discovery field [4].

#### ACKNOWLEDGMENTS

This work was supported by the Czech Science Foundation (GAČR), grant number 19-01641S.

#### REFERENCES

- [1] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 709–720. <https://doi.org/10.1109/ICDE48307.2020.00067>
- [2] Dan Brickley, Matthew Burgess, and Natasha F. Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryan W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 1365–1375. <https://doi.org/10.1145/3308558.3313685>
- [3] David Browning, Alejandra Gonzalez Beltran, Andrea Perego, Peter Winstanley, Riccardo Albertoni, and Simon Cox. 2020. *Data Catalog Vocabulary (DCAT) - Version 2*. W3C Recommendation. W3C. <https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/>.
- [4] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *Vldb* 7, 29, 1 (2020), 251–272. <https://doi.org/10.1007/s00778-019-00564-x>
- [5] Richard Cyganiak, Markus Lanthaler, and David Wood. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C. <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- [6] Bernhauer David and Skopal Tomáš. 2020. *Word2Vec model - Czech legislation*. <https://doi.org/10.5281/zenodo.3975084> This work was supported by the Czech Science Foundation (GAČR), grant number 19-01641S.
- [7] Bernhauer David and Skopal Tomáš. 2020. *Word2Vec model - Czech wikipedia*. <https://doi.org/10.5281/zenodo.3975038> This work was supported by the Czech Science Foundation (GAČR), grant number 19-01641S.
- [8] Auriol Degbelo and Brhane Bahrishum Tekla. 2019. Spatial Search Strategies for Open Government Data: A Systematic Comparison. *CoRR* abs/1911.01097 (2019), 10. [arXiv:1911.01097](http://arxiv.org/abs/1911.01097) <http://arxiv.org/abs/1911.01097>
- [9] Simon Dutkowski and Andreas Schramm. 2015. *Duplicate Evaluation - Position paper by Fraunhofer FOKUS*. Technical Report. Fraunhofer FOKUS. [https://www.w3.org/2016/11/sdsvoc/SDSVoc16\\_paper\\_24](https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_24)
- [10] Raul Castro Fernandez, Ziawasch Abedjan, Famiem Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 1001–1012. <https://doi.org/10.1109/ICDE.2018.00094>
- [11] Jakub Klímeček. 2019. DCAT-AP representation of Czech National Open Data Catalog and its impact. *J. Web Semant.* 55 (2019), 69–85. <https://doi.org/10.1016/j.websem.2018.11.001>
- [12] Jakub Klímeček and Petr Škoda. 2018. LinkedPipes DCAT-AP Viewer: A Native DCAT-AP Data Catalog. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - 10 - 12th, 2018 (CEUR Workshop Proceedings)*, Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna (Eds.), Vol. 2180. CEUR-WS.org, 4. <http://ceur-ws.org/Vol-2180/paper-32.pdf>
- [13] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. <http://arxiv.org/abs/1301.3781>
- [14] Renée J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, and Periklis Andritsos. 2018. Making Open Data Transparent: Data Discovery on Open Data. *IEEE Data Eng. Bull.* 41, 2 (2018), 59–70. <http://sites.computer.org/debull/A18june/p59.pdf>
- [15] Martin Nečaský, Tomáš Knap, Jakub Klímeček, Irena Holubová, and Barbora Vidová-Hladká. 2013. Linked Open Data for Legislative Domain - Ontology and Experimental Data. In *Business Information Systems Workshops*, Witold Abramowicz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 172–183.
- [16] J. Oliver, C. Cheng, and Y. Chen. 2013. TLSH - A Locality Sensitive Hash. In *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, 7–13.
- [17] Radim Řeháček and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. <http://is.muni.cz/publication/884893/en>
- [18] Tomáš Skopal, Jakub Klímeček, and Martin Nečaský. 2019. Improving Findability of Open Data Beyond Data Catalogs. In *Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, iiWAS 2019, Munich, Germany, December 2-4, 2019*. ACM, 413–417. <https://doi.org/10.1145/3366030.3366095>
- [19] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, Satinder P. Singh and Shaul Markovitch (Eds.). AAAI Press, 4444–4451. <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972>
- [20] Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, 88–99. <https://doi.org/10.18653/v1/K17-3009>
- [21] Princeton University. 2010. *About WordNet*. Princeton University. <https://wordnet.princeton.edu/>
- [22] Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.
- [23] Mark Wilkinson, Michel Dumontier, Jlsbrand Jan Aalbersberg, Gaby Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Olavo Bonino da Silva Santos, Philip Bourne, Jildau Bouwman, Anthony Brookes, Tim Clark, Merce Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris Evelo, Richard Finkers, and Barend Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3 (03 2016), <https://doi.org/10.1038/sdata.2016.18>



---

## Analysing Indexability of Intrinsically High-dimensional Data using TriGen

**Bernhauer, D. (50 %); Skopal, T. (50 %)** Analysing Indexability of Intrinsically High-dimensional Data using TriGen. In *Similarity Search and Applications: 13th International Conference, SISAP 2020, Copenhagen, Denmark*: Springer, 2020, doi: 10.1007/978-3-030-60936-8\_20. [A.5]

The paper has been cited in:

- Mic, V.; Raček, T.; Křenek, A.; Zezula, P. Similarity Search for an Extreme Application: Experience and Implementation. In *International Conference on Similarity Search and Applications*. Springer, Cham, 2021. p 265–279.



## Analysing Indexability of Intrinsically High-Dimensional Data Using TriGen

David Bernhauer<sup>1,2</sup>  and Tomáš Skopal<sup>1</sup> 

<sup>1</sup> SIRET RG, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

[bernhdav@fit.cvut.cz](mailto:bernhdav@fit.cvut.cz), [skopal@ksi.mff.cuni.cz](mailto:skopal@ksi.mff.cuni.cz)

<sup>2</sup> Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

**Abstract.** The TriGen algorithm is a general approach to transform distance spaces in order to provide both exact and approximate similarity search in metric and non-metric spaces. This paper focuses on the reduction of intrinsic dimensionality using TriGen. Besides the well-known intrinsic dimensionality based on distance distribution, we inspect properties of triangles used in metric indexing (the triangularity) as well as properties of quadrilaterals used in ptolemaic indexing (the ptolemaicity). We also show how LAESA with triangle and ptolemaic filtering behaves on several datasets with respect to the proposed indicators.

### 1 Introduction

The real-world datasets for similarity search often exhibit high intrinsic dimensionality manifested by distance distribution with low variance and high mean [5]. The reason could be the high complexity of the similarity model within a given domain (lot of independent features), but often this is just a consequence of automated feature extraction processes, e.g., the inference of deep features [6]. Intrinsically high-dimensional data cannot be used for efficient exact search but, luckily, there have been developed many approximate methods [9] to tackle this problem for the price of a lower retrieval precision. Some of these methods elegantly avoid the direct problem of high intrinsic dimensionality by not indexing actual distances, but just permutations of pivots [4, 7]. These methods enabled competitive application of similarity search in real-world domains where maximal retrieval precision is not as critical as the performance. However, we must keep in mind these methods are limited in tuning the precision at runtime (from query to query) as well as they are restricted to pivot-based indexing schemes.

The TriGen algorithm [11] was proposed as a universal method for fast exact and approximate search in metric and non-metric spaces. So far, it was not analyzed as a method for (intrinsic) dimensionality reduction. In this paper

---

This research has been supported by Czech Science Foundation (GAČR) project Nr. 19-01641S.

© Springer Nature Switzerland AG 2020  
S. Satoh et al. (Eds.): SISAP 2020, LNCS 12440, pp. 261–269, 2020.  
[https://doi.org/10.1007/978-3-030-60936-8\\_20](https://doi.org/10.1007/978-3-030-60936-8_20)

we empirically analyze this missing aspect. We also investigate the impact of TriGen modifications on the potential of ptolemaic indexing [8] that achieves better performance than metric indexing (though limited to ptolemaic metrics).

## 2 Background

When indexing data for fast similarity search, we face two fundamental concepts – the data indexability and the indexing model.

### 2.1 Indexability

The indexability generally refers to an ability to search efficiently a dataset  $\mathbb{S} \subset \mathbb{U}$  under a similarity model  $(\mathbb{U}, d)$ , regardless the indexing method used. The key is the distribution of data or, specifically, in case of similarity search it is the distribution of distances  $d(x, y)$  among data objects  $x, y \in \mathbb{S}$ . The classic indexability indicator for a metric space model  $(\mathbb{U}, d)$  is the intrinsic dimensionality [5], defined as the ratio of squared mean and doubled variance of the distance distribution;  $\text{iDim}(\mathbb{S}, d) = \frac{\mu^2}{2\sigma^2}$ . The lower iDim, the better indexability.

Alternatively, the ball overlap factor (BOF) [11] describes the ability to partition the dataset into non-overlapping ball-shaped regions. The BOF counts for how many object pairs will constitute overlapping balls (each ball radius is the distance to the ball center's  $k$ th nearest neighbor).

### 2.2 TriGen Transformation

The TriGen algorithm [11] transforms the input distance space  $(\mathbb{U}, d)$  by use of triangle-generating or -violating modifiers and a dataset sample  $\mathbb{S}^* \subseteq \mathbb{S} \subset \mathbb{U}$  into a target space  $(\mathbb{U}, f(d))$ . A modifier  $f : R \rightarrow R_0^+$  must be an increasing function with  $f(0) = 0$  to preserve the ordering of distances<sup>1</sup> and thus search results with respect to sequential scan. The triangle-generating (concave) modifiers “inflate” all the triangles in the space to become more equilateral; then the dataset is less indexable as the intrinsic dimensionality increases. The triangle-violating (convex) modifiers have the opposite effect – “squeezing” the triangles and lowering the intrinsic dimensionality. The idea behind the triangle-violating modifiers is that they lower the intrinsic dimensionality (more efficient search) for the price of a retrieval error (some triangles break which shows in incorrect filtering by querying). The indexability indicators, like the intrinsic dimensionality or BOF, together with the T-error measuring the ratio of broken triangles, guide TriGen to determine the right modifier.

Unlike other methods that map the source distance space into the Euclidean space, the TriGen model is based solely on transformation of distances, hence there is no need for an expensive and static embedding of metric objects into vectors. In consequence, once a modifier is computed for a particular problem, its

<sup>1</sup> Ranking of objects  $x_i \in \mathbb{U}$  based on  $d(q, x_i)$  is the same as based on  $f(d(q, x_i))$ .

change (e.g., a precision guarantee) can be easily recomputed and the already created index just updated (no change in descriptors). This allows to switch between several TriGen modifiers at query time, providing thus flexible exact-to-approximate search (e.g., the NM-tree [10]). Other TriGen follow-ups include extensions to non-symmetric distances [3], and the genetic TriGen variants [1, 2].

In this paper, we inspect the TriGen in the role of dimensionality reduction method. In high-dimensional datasets (as measured by intrinsic dimensionality), all of the non-trivial triangles tend to be almost-equilateral. Then application of TriGen with triangle-violating modifiers could act as a lossless dimensionality reduction method by squeezing the triangles without the violation of triangle inequality (breaking the triangles by squeezing them too much). Our hypothesis is, the higher the (intrinsic) dimensionality of data is, the more almost-equilateral the triangles are, and so the more aggressive modifier could be applied while still keeping the triangles unbroken. Simply said, we analyze the question if TriGen could “cancel” the curse of dimensionality (to some extent) in similarity search.

### 2.3 Metric and Ptolemaic Indexing

The metric access methods (metric indexes) [5] use some construction of lower bounds using the triangle inequality. In the simplest case of pivot tables (aka LAESA), the three objects in the triangle are the query object  $q$ , a dataset object  $x$ , and a pivot  $p$  (i.e.,  $LB_{\Delta}(q, x) = |d(q, p) - d(p, x)|$ ). If the triangle is equilateral,  $LB_{\Delta}(q, x) = 0$  and so the dataset object  $x$  cannot be filtered by the lower bound. On the other hand, if the triangle is (squeezed to) a line segment, the lower bound gets maximal (i.e.,  $LB_{\Delta}(q, x, p) = d(q, x)$ ) and so it is “super-effective” for filtering.

Similarly, ptolemaic access methods (ptolemaic indexes) [8] use some construction of lower bounds using the Ptolemy’s inequality that operates on quadrilaterals (quadruplets, respectively). In the simplest (LAESA) case there are four objects in the quadrilaterals: the query object  $q$ , a dataset object  $x$ , and two pivots  $p_1, p_2$ , while a lower bound can be derived as

$$LB_{pt}(q, x, p_1, p_2) = \frac{|d(q, p_1) \cdot d(x, p_2) - d(q, p_2) \cdot d(x, p_1)|}{d(p_1, p_2)} \quad (1)$$

As the quadrilaterals are more complex than triangles, there is not a single best or worst quadrilateral example for the lower bound construction. Also the inflating and squeezing effect of TriGen modifiers is not clear in case of quadrilaterals, and so for ptolemaic indexing.

### 3 Triangle and Quadrilateral Distribution

The intrinsic dimensionality, as an indexability indicator, considers only distances themselves but does not consider that some distance combinations cannot be present in triangles at the same time, which is important for the filtering by metric access methods. The BOF compensates this issue, but it cannot be easily

generalized for Ptolemaic inequality or non-metric cases. Therefore, we define the *triangularity* to quantify the shape of triangle on a real-value scale from equilateral triangle, through line segment to broken triangle. Similarly, we define *Ptolemaicity* to quantify the shape of quadrilateral on a scale from tetrahedron, through line segment to broken equilateral.

Hence, we need to aggregate three distances forming a triangle into one number, with extremes for equilateral triangles and line segments. We could adopt the TriGen criteria (presented in [5]) used for determining the number of triangles that do not satisfy the triangle inequality. The *triangularity* is defined for a triangle  $a = d(x, y), b = d(y, z), c = d(x, z)$  by Eq. 2 – this ratio determines how “equilateralish” (or “inflated”) a triangle is. The triangularity is 1 for equilateral triangle, 1/2 means the triangle forms line segment (“squeezed”), and for values below 1/2 the triangle is broken (does not satisfy the triangle inequality).

$$\text{Triangularity}(a, b, c) = \frac{a + b}{2c}, \text{ where } a \leq b \leq c \quad (2)$$

After TriGen preprocessing, we expect the distribution will be shifted to line segments (“squeezed”) instead of almost-equilateral triangles. Knowledge of this common property makes the *triangularity* a good indicator of datasets with statistically high probability to exhibit bad indexability.

Moreover, we try TriGen for Ptolemaic indexing, though the TriGen modifiers were originally proposed for indexing using the lower bounds based on triangle inequality and not the Ptolemy’s inequality (Eq. 3). We would like to find out how the Ptolemy’s inequality holds in comparison with the triangle inequality. We define *ptolemaicity* of a quadrilateral as Eq. 4, where  $d(w, x)d(y, z), d(w, y)d(x, z) \leq d(w, z)d(x, y)$ . The greatest ptolemaicity value is 1, which represents regular tetrahedron and results in bad indexability. ptolemaicity 1/2 represents a line segment and for values below 1/2 the equilateral is broken (does not satisfy Ptolemy’s inequality).

$$(\forall w, x, y, z \in \mathbb{U}) d(w, x)d(y, z) + d(w, y)d(x, z) \geq d(w, z)d(x, y) \quad (3)$$

$$\text{Ptolemaicity}(w, x, y, z) = \frac{d(w, x)d(y, z) + d(w, y)d(x, z)}{2d(w, z)d(x, y)} \quad (4)$$

## 4 Analysis of High-Dimensional Data

We have analyzed several datasets and looked at the intrinsic dimensionality and the retrieval efficiency (using the LAESA algorithm). Two low-dimensional datasets are from SISAP datasets: the 20-dimensional NASA dataset, and the 112-dimensional Colors dataset. As high-dimensional datasets we used a sample of the 2048-dimensional AlexNet image (V3C1) dataset, and several artificial datasets of dimensionality 2 to 2048 (randomly generated vectors). For all datasets we have used the Euclidean space, which is both metric and ptolemaic.

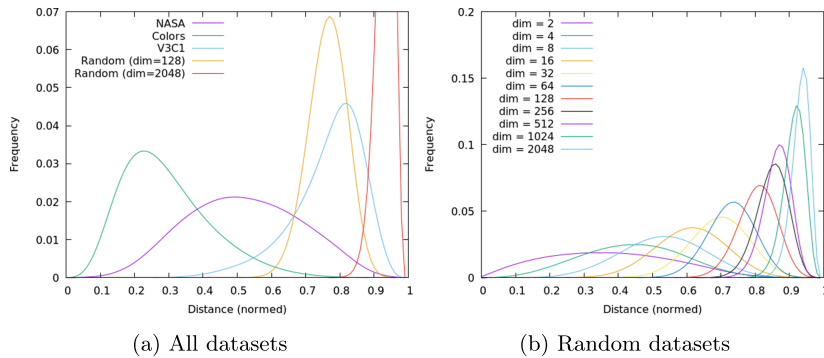
## 5. ANALYSING INDEXABILITY OF INTRINSICALLY HIGH-DIMENSIONAL DATA USING TRI-GEN

**Table 1.** Datasets statistics (iDim, distance computations with metric LAESA).

Dataset (dim)	without TriGen		with TriGen (zero error)	
	iDim	Dist. Comp.	iDim	Dist. Comp.
NASA (20)	$5.184 \pm 0.007$	2.12%	$4.593 \pm 0.007$	1.15%
Colors (112)	$2.742 \pm 0.003$	2.63%	$2.553 \pm 0.003$	2.08%
Random (128)	$181.328 \pm 0.304$	100%	$28.663 \pm 0.022$	95.78%
Random (2048)	$1967.66 \pm 184.295$	100%	$37.035 \pm 0.175$	99.3%
V3C1 (2048)	$30 \pm 0.050$	86.65%	$9.215 \pm 0.012$	45.39%

In Table 1 on the left, we present intrinsic dimensionality comparison and efficiency improvement of the metric LAESA (with randomly chosen 50 pivots) against sequential search. The iDim of Colors dataset is lower than iDim NASA dataset, however, LAESA performs better on NASA. Note the embedding dimensionality and iDim are dramatically different in case of V3C1 and Colors. Figure 1 shows distance distribution histograms for all datasets.

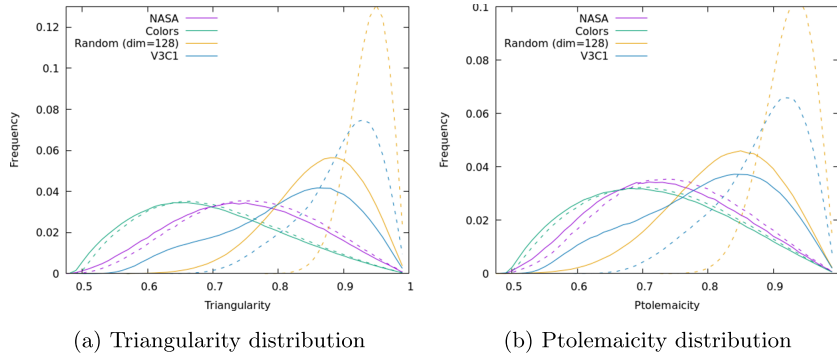
In Fig. 2a (dashed), we present triangularity distribution. As we expected, the distribution is shifted to the right side for high-dimensional datasets. This is the main assumption for transforming metric space using the TriGen into a more indexable one. Similarly, we have visualized the ptolemaicity distribution in Fig. 2b (dashed), which displays the same properties.



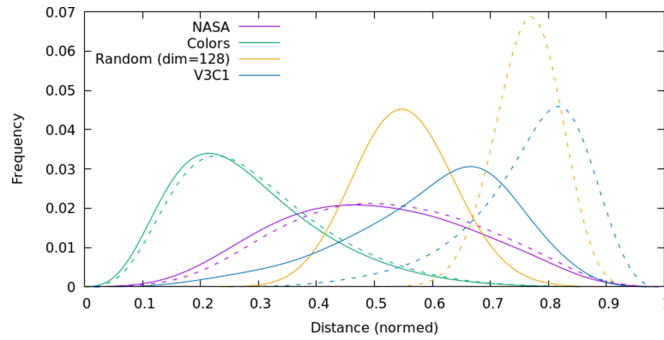
**Fig. 1.** Distance distribution comparison

Both triangularity and ptolemaicity distributions are similar, which means TriGen could be used for modification of Ptolemaic space, too. If the TriGen transforms both spaces consistently then, based on figures, Ptolemy’s inequality is violated earlier, because there is a higher number of line segments.





**Fig. 2.** Distribution of triangularity or ptolemaicity in datasets before (dashed) and after (solid) TriGen modifications.



**Fig. 3.** Dist. distribution before (dashed) and after (solid) TriGen modifications.

#### 4.1 TriGen Modifications

In the first part of our experiment, we have configured TriGen to zero error tolerance. The measured retrieval error (as defined in [11]) was also zero, hence, we achieved faster and still exact search. Figure 3 shows the change of distance distributions in datasets after TriGen modifications were made.

Table 1 on the right describes basic indicators after TriGen modifications, and we observe that triangle-violating modifications reduced the intrinsic dimensionality. The retrieval efficiency improved for all datasets (for some only slightly, but two times for NASA and V3C1). It indicates the presence of an inner structure beyond all conventional indicators, except for Random (2048) that is not indexable for exact search. However, TriGen can still transform a seemingly not-indexable dataset (V3C1, Random(128)) into partially indexable even for exact search.

Both triangularity (Fig. 2a) and ptolemaicity (Fig. 2b) distributions are flatter and shifted to the left as we expected. The ptolemaicity distribution is flatter than triangularity distribution, which means that Ptolemy’s inequality is more prone to a violation when used with TriGen.

4.2 Comparison of Real Performance

The TriGen algorithm controls the ratio of triangles satisfying the triangle inequality (so-called T-error tolerance) by a weight parameter that determines the convexity/concavity of the modifier. In the previous experiments we set T-error tolerance = 0 that (empirically) guarantees zero retrieval error. In Fig. 4a, we can see the dependence of distance computations and retrieval error on the weight (V3C1 dataset). We used just the triangle-violating (squeezing) modifications where -10 weight is heavy squeezing and -0.1 weight is almost no squeezing. We used LAESA with 50 randomly chosen pivots utilizing metric filtering, ptolemaic filtering, or both, and compared it with the sequential search.

The important observation is the ptolemaic filtering<sup>2</sup> has a similar pattern as the metric filtering. The general difference is in the shift of the ptolemaic curves to the right. The combination of triangle and ptolemaic filtering utilizes the benefits of both approaches. Triangle filtering deals with retrieval error caused by the Ptolemy’s inequality violation and the Ptolemy’s filtering deals with better efficiency, because of its ability to create better lower bounds.

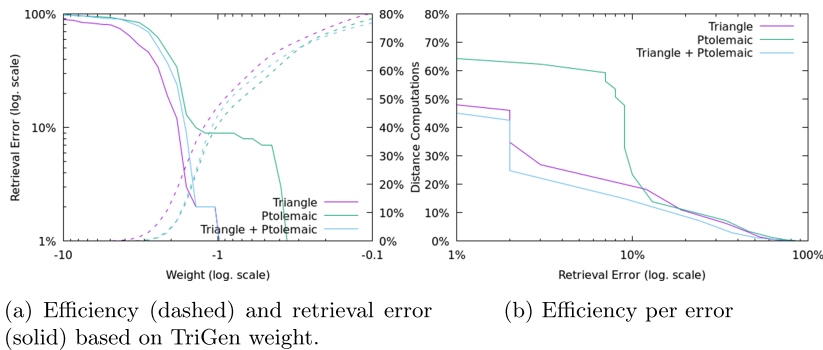


Fig. 4. Efficiency and retrieval error (LAESA with 50 pivots on V3C1 dataset).

Another point of view is presented in Fig. 4a, where pairs of efficiency and retrieval error values from Fig. 4b are aggregated into single efficiency per error value. So, we get rid of the TriGen weight parameter and only observe how the real efficiency is dependent on real retrieval error, obtaining more readable results than when depicted individually.

<sup>2</sup> We used simple random selection of pivot pairs in ptolemaic filtering instead the better but slower Balanced heuristic [8].

### 4.3 Discussion

The intrinsic dimensionality is not always sufficient to predict the real efficiency of an indexing algorithm. First, because of some inner structure that can hardly be described by a single number. Second, the high number of low distances, triangularities, or ptolemaicities does not imply better indexability. A good example can be randomly generated vectors with one outlier, which will shift the whole histogram to the left.

The TriGen can be used for both precise and approximate search. The combination of both filtering inequalities improves not only efficiency but also lowers the retrieval error. There is a possibility in the future to try other kinds of inequalities and their ability to scale with TriGen.

## 5 Conclusions

We have introduced structure-sensitive empirical measures for the analysis of metric and Ptolemaic spaces and defined the triangularity and the ptolemaicity as the quantifiers of triangle and quadrilateral shapes. Analysis of high-dimensional data shows that it is possible to use TriGen as dimensionality reduction method that improves the efficiency of similarity search.

Although the TriGen was designed for transforming non-metric spaces into metric ones, we have shown that the inverse application on high-dimensional data is possible as well and efficient for both exact and approximate search. Moreover, experiments indicate that TriGen could be used with different types of filtering inequalities (like Ptolemy's). The combination of several filtering inequalities synergically deals with the advantages (better efficiency) and disadvantages (worse precision) of the individual methods.

## References

1. Bernhauer, D., Skopal, T.: Approximate search in dissimilarity spaces using GA. In: GECCO, pp. 279–280. ACM (2019)
2. Bernhauer, D., Skopal, T.: Non-metric similarity search using genetic TriGen. In: Amato, G., Gennaro, C., Oria, V., Radovanović, M. (eds.) SISAP 2019. LNCS, vol. 11807, pp. 86–93. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-32047-8\\_8](https://doi.org/10.1007/978-3-030-32047-8_8)
3. Boytsov, L., Nyberg, E.: Pruning algorithms for low-dimensional non-metric k-NN search: a case study. In: Amato, G., Gennaro, C., Oria, V., Radovanović, M. (eds.) SISAP 2019. LNCS, vol. 11807, pp. 72–85. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-32047-8\\_7](https://doi.org/10.1007/978-3-030-32047-8_7)
4. Chavez, E., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. *IEEE TPAMI* **30**(9), 1647–1658 (2008)
5. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. *ACM Comput. Surv.* **33**(3), 273–321 (2001)
6. Donahue, J., et al.: DeCAF: a deep convolutional activation feature for generic visual recognition. In: ICML, pp. I-647–I-655. JMLR.org (2014)

## 5. ANALYSING INDEXABILITY OF INTRINSICALLY HIGH-DIMENSIONAL DATA USING TRIGEN

---

Analysing Indexability of Intrinsically High-Dimensional Data Using TriGen 269

7. Esuli, A.: Use of permutation prefixes for efficient and scalable approximate similarity search. *Inf. Process. Manag.* **48**(5), 889–902 (2012)
8. Hetland, M.L., Skopal, T., Lokoč, J., Beecks, C.: Ptolemaic access methods: challenging the reign of the metric space model. *Inf. Syst.* **38**(7), 989–1006 (2013)
9. Patella, M., Ciaccia, P.: Approximate similarity search: a multi-faceted problem. *J. Discret. Algorithms* **7**(1), 36–48 (2009)
10. Skopal, T., Lokoč, J.: NM-tree: flexible approximate similarity search in metric and non-metric spaces. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2008*. LNCS, vol. 5181, pp. 312–325. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85654-2\\_30](https://doi.org/10.1007/978-3-540-85654-2_30)
11. Skopal, T.: Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Trans. Database Syst.* **32**(4), 29-es (2007)

---

## Inferred Social Networks: A Case Study

Holubová, I. (20 %); Svoboda, M. (20 %); Bernhauer, D. (20 %); Skopal, T. (20 %); Paščenko, P. (20 %) Inferred Social Networks: A Case Study. In *Proceedings of the 2019 International Conference on Data Mining Workshops*, ICDMW'19, Beijing, China: IEEE Computer Society, 2019, doi: 10.1109/ICDMW.2019.00019. [A.6]

The paper has been cited in:

- Měkota, O. *Link Prediction in Inferred Social Networks*. 2021. Master's Thesis. Charles University

© 2019 IEEE. Reprinted, with permission, from Holubová, I.; Svoboda, M.; Bernhauer, D.; Skopal, T.; Paščenko, P., *Inferred Social Networks: A Case Study*, *Proceedings of the 2019 International Conference on Data Mining Workshops*, 2019.

# Inferred Social Networks: A Case Study

Irena Holubová\*, Martin Svoboda\*, David Bernhauer\*†, Tomáš Skopal\*, Petr Paščenko‡

\*Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

†Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

‡Profinit EU, s.r.o., Prague, Czech Republic

Email: {holubova,svoboda,skopal}@ksi.mff.cuni.cz, bernhdav@fit.cvut.cz, petr.pascenko@profinit.eu

**Abstract**—The behavior, environment, and characteristics of clients form a crucial source of information for various businesses. There exists a number of supervised as well as unsupervised data mining or other approaches that allow analyzing the respective data. In our ongoing project, focusing primarily on the financial sector, we suggest an innovative strategy that will overcome persisting shortcomings of the state-of-the-art methods using an analysis of a social network of clients. In addition, we do not assume the existence of such a network, but from a given set of client financial activities, we are able to infer a social network representing their relationships and behavior. Using real-world data and selected use cases from our domain, we show (a part of) the process of construction of an inferred social network, i.e., what kind of “hidden” information can, for example, be found and exploited.

**Keywords**—inferred social networks, similarity search, client behavior analysis.

## I. Introduction

The behavior, environment, and characteristics of clients form a crucial source of information for various businesses in order to increase their revenues, detect potential problems, prevent unwanted situations, etc. During the first stage of our ongoing project, we focus on clients of banks and other similar financial institutions providing basic banking products such as deposit accounts, payment cards, or cash loans. However, analogous principles can be observed in other areas as well, such as landline/mobile phone operators or electricity/gas suppliers.

In the financial sector, banks primarily evaluate various behavioral or credit scores of both potential and existing clients in order to classify them as fraudulent or legitimate, with various risk levels involved. There are basically three main issues that banks essentially need to solve: (1) fraud detection, (2) debt services default, and (3) customer churn. In addition, various life situations (e.g., finishing studies, birth of a child, promotion, loss of a job, retirement, etc.) can be interesting to offer appropriate financial products to relevant clients in the right moment.

There exists a number of data mining approaches that solve some of the indicated problems [9]. Most of the solutions use some supervised method [3], [7], where we can predict the future behavior of clients based on a learning data set where the corresponding targets were well known or identified manually. Unsupervised approaches [8] use characteristics of clients in order to group

them into clusters on the basis of their mutual similarity and maximum dissimilarity between the clusters.

In this paper, we describe a novel solution inspired by both of these approaches and based on an analysis of a social network of clients. Of course, the idea of utilizing social networks in this context was already proposed, e.g., for fraud detection [4], [6] or peer-to-peer lending [5], but we go much further. First, we do not assume the existence of such a network (as this is a strong assumption), but based on the available client characteristics and history of financial transactions we construct an inferred social network. For this purpose, we exploit both direct facts and features inferred from the available data indirectly using various similarity methods and statistical approaches, all that with additional measures of reliability and time-limited validity of the information. Such networks will permit the banks to view and analyze behavior and mutual relationships of clients from a new perspective, and provide otherwise impossible insights.

Structure of the text: In Section II we provide a description of the inferred social network and the process of its inference. Then we provide a set of examples of information we can use for construction of the output network in Section III. In Section IV we conclude.

## II. Inferred Social Network and Its Construction

In our first real-world scenario from the financial sector, we assume the following input (anonymized) data:

- 1) client personal and other socio-demographic characteristics (e.g., name, addresses, education, etc.),
- 2) history of financial transactions (e.g., credit/debit card payments, cash withdrawals, bank transfers, regular/overdue loan installments, etc.), and,
- 3) third-party information related to the non-person entities (e.g., shops, institutions, ATMs, etc.).

In all the cases, only data belonging to a particular monitored time interval  $I = [t_{start}, t_{end}]$  is covered.

As defined in detail in [2], we build the social network of bank clients at two different levels of granularity, representing a dual view of the same problem from different perspectives: (1) high-level view, where the social network only consists of vertices for bank clients, and (2) low-level view, where there are vertices also for different

kinds of entities (e.g., institutions, companies, etc.).<sup>1</sup>

We define a high-level network to be a multigraph  $G_H = (V_H, E_H)$ , where the set of vertices  $V_H$  represents individual bank clients and the set of edges  $E_H$  relationships among the clients. Each client  $v \in V_H$  has a set of properties  $\{p_1, p_2, \dots, p_n\}$ , each of which has its name  $p_{name}$  (e.g., age) and value  $p_{value}$  (e.g., 58). Each edge  $e \in E_H$  has its relationship type  $e_{type}$  (e.g., colleague). Edges (and in some cases properties of vertices) also have reliability  $e_{reliability} \in [0, 1]$  (e.g., 0.76 for information certain for 76%), and validity  $e_{validity} = [t_{start}, t_{end}]$  representing the time interval of validity of the information.

The low-level network  $G_L = (V_L, E_L)$  is defined analogously, just the vertices can be of different types and thus have also a vertex type  $v_{label}$  – see Figure 1.

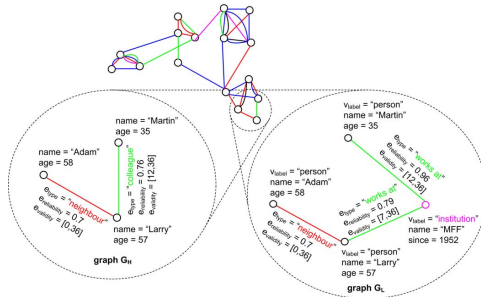


Fig. 1. High- and low-level views of a sample inferred social network

The process of inference of graph  $G_H$  (or  $G_L$ ) consists of two orthogonal approaches which we denote as (1) rule-based extraction and (2) similarity-based extraction [2].

#### A. Rule-Based Extraction

The more straightforward inference path is based on the definition of a set of domain-specific rules, each of which defines a way of constructing a particular type (or types) of relationships in the target graph  $G_H$  (or  $G_L$ ) depicted in Figure 1 using different colors. We distinguish:

- factual relationships directly present in the input data (e.g., a married couple),
- aggregated relationships derived using various information aggregations (e.g., co-workers defined as people working in the same company), and
- transitive relationships derived on the basis of a statistically significant amount of occurrences of specific events (e.g., a household defined as people with the same home address, frequent mutual money transfers, frequent payments in similar shops, etc.).

<sup>1</sup>Note that in general there are  $n + 2$  views since there exists a number of possible “intermediate” levels between  $G_H$  and  $G_L$ .

#### B. Similarity-Based Extraction

The process of data extraction in this inference path is more complex, and, to the best of our knowledge, also unique in the given context. We use a set of similarity models, each of which expresses a particular similarity of entities defined by certain selected features. These features (represented as descriptors characterizing the particular entities) can be based on single fixed values, time-varying series, accumulated values (e.g., per week or month), etc., or their combinations.

For each model  $l$ , an output is formed by a dense square matrix  $M_l$ , where for each two vertices  $v_i, v_j \in V_H$  (or  $V_L$ ), the measure of their mutual similarity  $sim_l(v_i, v_j) \in [0, 1]$  with regards to the corresponding descriptors is stored at  $M_l[i, j]$ . Using similarity thresholds, combination of multiple similarity results (e.g., a weighted sum), clustering of clients into categories, etc., we construct a sparse matrix  $M'_l$  which restricts the result just to the most interesting information with a reasonable size. Each  $M'_l$  forms a new type of edges (i.e., a new edge color in Figure 1).

### III. Real-World Use Cases

The real-world data we acquired from one of the smaller and newly emerged banks cover almost two calendar years. They contain strongly anonymized data about approximately 320 thousand clients, their accounts and cash loans, as well as their transaction history involving transfer (approx. 25 million), installment (approx. 2 million), and payment card (approx. 38 million) transactions.

We will demonstrate the potential of the inferred social networks using a set of real-world use cases. They were chosen with regards to the particular input data we currently have; more detailed or less anonymized information would enable extraction of different information.

#### A. Use Case 1: Merchant Category Codes

In the first use case, we analyze the Merchant Category Codes (MCC) associated with all payment card transactions done by the bank clients. Such a code is assigned to a particular merchant by the card company when it first starts accepting payment cards (and so it may not always fully reflect the actual scope of activities). Although there are thousands of such codes introduced in general, we only have about 550 codes in our data. To simplify the classification, we organized these codes into 17 proprietary classes (e.g., airlines, hotels, restaurants, wholesale, etc.).

As depicted in Figure 2, we get 9 clusters of clients with regards to percentages of their financial transactions in particular MCC classes. At the  $Y$ -axis, we can see our proprietary MCC classes; at the  $X$ -axis, we can see the average percentages of such transactions in particular clusters. From an analysis of the clusters, we can conclude that clients from clusters 4 and 5 like to travel (because of the high percentage of payments in MCC classes `travel_expense`, `vehicle_expense`, `other_travel` etc.). Clients from cluster 1 primarily use

## 6. INFERRED SOCIAL NETWORKS: A CASE STUDY

their account for ATM withdrawals, cashback, money exchange, etc. (indicated by MCC class money). Clusters 8 and 9 involve clients who primarily pay using their cards (as indicated by MCC classes risk and wholesale). Finally, clients from clusters 2 and 3 use both ATM withdrawals and credit card payments.

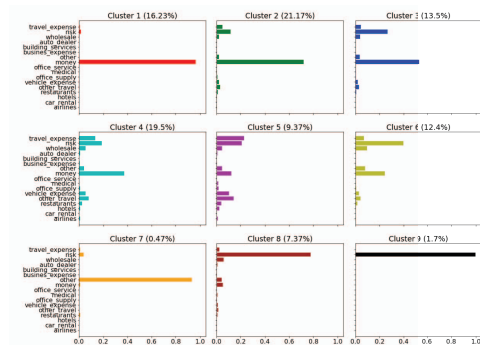


Fig. 2. Clusters of clients similar with regards to MCC codes of all of their financial transactions

### B. Use Case 2: Payments in Foreign Countries

Another information we have in our data is the country where the money withdrawal or credit card payment was executed. On the basis of this information, we can identify relationships between clients who have similar patterns in visiting foreign countries. We can study the seasons of payments (e.g., summer/winter, Christmas/Easter etc.), the amounts and frequency of payments etc. As third-party information, we can use the general knowledge about the countries and group them into clusters (related, e.g., to geography, size, GDP, etc.).

To clarify the outlined ideas, consider the graph of weekly payments during one year (starting from the first week in January) in 8 selected and commonly appearing countries provided in Figure 3. As we can see, for example, Germany (DEU) and Austria (AUT) have a very similar curve shape especially at the beginning and end of the year, corresponding to their popularity during winter holidays. Italy (ITA) also exhibits many similarities with them. Conversely, Croatia (HRV) is primarily a very popular summer holiday destination. These four countries also have a peak at the beginning of the summer holidays. Greece (GRC) is also a popular summer holiday destination, although not that much and without the peak.

If we cluster all the countries used in our data with a reasonable threshold, we obtain 3 clusters.

### C. Use Case 3: ATM Withdrawals

The last but not least use case we describe is an analysis of ATMs used by the bank clients. The idea behind is

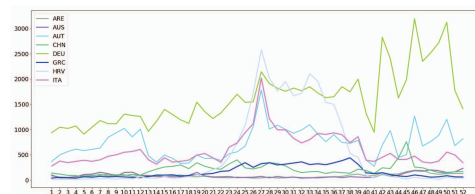


Fig. 3. Weekly payments in selected foreign countries

that if we were able to classify the ATMs (similarly to the case of classification of countries), we would also be able to classify clients themselves, e.g., according to their frequency of usage of ATMs of similar types. For this purpose, we study hourly withdrawals per week.

Using a reasonable threshold, we get 10 clusters of ATMs depicted in Figure 4, which contains the curves of average numbers of withdrawals from the ATMs per hour in a week (starting from Monday 00:00 – 00:59 and ending Sunday 23:00 – 23:59). As we can see, most of the clusters exhibit the same trend that higher numbers of withdrawals are during the working days. Usually, there is a peak in the morning and the afternoon.

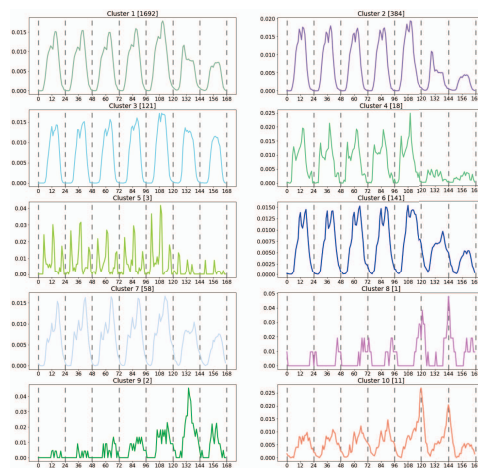


Fig. 4. Clusters of ATMs with similar hourly withdrawals per week

However, there are also interesting special cases. For example, we can say that cluster 4 contains ATMs located close to business centers, since the peak withdrawals are before and after usual office hours, and the amount of withdrawals significantly drops during weekends. In cluster 5, there are apparently three peaks each day. After an analysis of the location of the ATMs, we have concluded



that they are localized close to factories running three shifts (e.g., car factories, coal mines etc.). On the other hand, clusters 8, 9, and 10 have an apparent peak in the Friday/Saturday evening/night. These ATMs are located in the center of big cities, close to bars, clubs, pubs etc.

Note that in general, we do not strictly need the interpretation of the particular clusters. However, the backward check of the data and usage of third-party information enables us to better understand the differences and the information we acquire. Hence, we can cluster the clients not only according to the amounts and times of withdrawals in similar ATMs, but also based on a high probability that they have similar types of employers or similar free-time activities.

#### D. An Example of an Inferred Social Network

Let us elaborate more on Use Case 3 from the previous section by inferring an example social network (i.e., its graphs  $G_L$ ,  $G_H$ , respectively). We consider fact-based and similarity-based relations in the example – see graph  $G_L$  in Figure 5 on the left. First, a similarity-based clustering on ATM descriptors is performed, resulting in a number of clusters. Each cluster contains ATMs sharing a certain withdrawal pattern, as discussed in the previous section. ATMs within clusters are then turned into vertices of multigraph  $G_L$  with edges connecting vertices (ATMs) with the centroid ATM (its vertex) they belong to. To keep the figure readable, we depict only the subgraph representing cluster 10 (with centroid ATM at address “666 Bar Avenue”), which could be labeled “weekend fun”, as suggested in Use Case 3. The type of edges “sim32” stands for a particular model used for computing similarities – here the hourly withdrawals per week are used as descriptors and dynamic time warping distance as the similarity measure [1]. Reliabilities of edges within this subgraph correspond to the similarities.

Second, linking of bank clients to ATMs is performed using edges of type “withdrawals”. A client vertex is connected with an ATM vertex if, in the transaction history, there are at least  $w$  withdrawals of that client from that ATM ( $w$  is a parameter). As the client-ATM linking is based on facts, the edge reliabilities are set to 1.

An additional aggregation logic could be applied when turning (collapsing, or interpreting) the graph  $G_L$  into graph  $G_H$ , see Figure 5 on the right. As in  $G_H$  only the person vertices are expected, the edges between persons and other types of entities must be aggregated, resulting in inter-person edges only. In this example, we defined an edge of type “WF ATM usage” that is to be instantiated between person vertices that are linked to ATM vertices within the “weekend fun” cluster of vertices (there exists a path on edges of types “withdrawals”, [“sim32”], [“sim32”], “withdrawals”). Their reliabilities are computed as an aggregation (e.g., max, multiplication, etc.) on reliabilities of edges on the path. Other inter-person edges are created

for clients using the same ATMs (edge type “sharing ATM”, reliability 1).

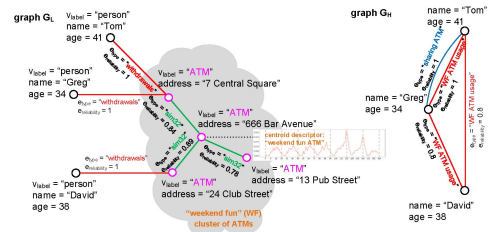


Fig. 5. An inferred social network – bank clients related by weekend usage of ATMs

#### IV. Conclusion

Not only in the financial sector, the analysis of characteristics and behavior of clients forms a key source of knowledge for various real-world businesses. The state-of-the-art approaches are primarily based on rather obvious, direct and only factual kinds of information we have about the clients. In this paper, we described an innovative solution which utilizes also inferred information and thus provides a richer and novel insight.

#### Acknowledgements

Supported by the TAČR project no. TH03010276.

#### References

- [1] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proc. VLDB Endow.*, 1(2):1542–1552, August 2008.
- [2] Irena Holubová, Martin Svoboda, Tomáš Skopal, David Bernhauer, and Ladislav Peška. Advanced Behavioral Analyses Using Inferred Social Networks: A Vision. In *Proc. of DEXA 2019 Workshops*, pages 210–219, 2019.
- [3] Sheikh Rabiul Islam, William Eberle, and Sheikh Khaled Ghafoor. Mining Bad Credit Card Accounts from OLAP and OLTP. *CoRR*, abs/1807.00819, 2018.
- [4] Christen Kirchner and Joseph Gade. Implementing Social Network Analysis for Fraud Prevention. CGI Group Inc., Düsseldorf, Germany, 2011.
- [5] Mingfeng Lin, Nagpuranand R. Prabhala, and Siva Viswanathan. Judging Borrowers by the Company They Keep: Friendship Networks and Information Asymmetry in Online Peer-to-Peer Lending. *Management Science*, 59(1):17–35, 2013.
- [6] Sanni Lookman and Selmin Nurcan. A Framework for Occupational Fraud Detection by Social Network Analysis. In *Proc. of CAiSE '15*, pages 221–228, Stockholm, Sweden, 2015.
- [7] E. W. T. Ngai, Yong Hu, Y. H. Wong, Yijun Chen, and Xin Sun. The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and an Academic Review of Literature. *Decision Support Systems*, 50(3):559–569, 2011.
- [8] Jon T. S. Quah and M. Sriganesh. Real-Time Credit Card Fraud Detection Using Computational Intelligence. *Expert Syst. Appl.*, 35(4):1721–1732, 2008.
- [9] Jilei Zhou. Data Mining for Individual Consumer Credit Default Prediction under E-commerce Context: A Comparative Study. In *Proc. of ICIS '17*, Seoul, South Korea, 2017.








---

## Advanced Behavioral Analyses Using Inferred Social Networks: A Vision

Holubová, I. (20 %); Svoboda, M. (20 %); Skopal, T. (20 %); Bernhauer, D. (20 %); Peška, L. (20 %) Advanced Behavioral Analyses Using Inferred Social Networks: A Vision. In *Database and Expert Systems Applications*, Springer, 2019, doi: 10.1007/978-3-030-27684-3\_26 [A.7]



# Advanced Behavioral Analyses Using Inferred Social Networks: A Vision

Irena Holubová<sup>1</sup> , Martin Svoboda<sup>1</sup> , Tomáš Skopal<sup>1</sup> ,  
David Bernhauer<sup>1,2</sup> , and Ladislav Peška<sup>1</sup> 

<sup>1</sup> Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic  
{holubova,svoboda,skopal,peska}@ksi.mff.cuni.cz

<sup>2</sup> Faculty of Information Technology, Czech Technical University in Prague,  
Prague, Czech Republic  
berhdav@fit.cvut.cz

**Abstract.** The success of many businesses is based on a thorough knowledge of their clients. There exists a number of supervised as well as unsupervised data mining or other approaches that allow to analyze data about clients, their behavior or environment. In our ongoing project focusing primarily on bank clients, we propose an innovative strategy that will overcome shortcomings of the existing methods. From a given set of user activities, we infer their social network in order to analyze user relationships and behavior. For this purpose, not just the traditional direct facts are incorporated, but also relationships inferred using similarity measures and statistical approaches, with both possibly limited measures of reliability and validity in time. Such networks would enable analyses of client characteristics from a new perspective and could provide otherwise impossible insights. However, there are several research and technical challenges making the outlined pursuit novel, complex and challenging as we outline in this vision paper.

**Keywords:** Inferred social networks · Similarity · Behavioral analysis

## 1 Introduction

The behavior, environment, and characteristics of clients form a significant source of information for various businesses in order to increase their revenues, detect potential problems, prevent unwanted situations, or at least suppress their negative impact. During the first stage of our ongoing project, we are focusing on clients of banks and other similar financial institutions providing basic banking products such as deposit accounts, payment cards or cash loans. However, analogous principles and challenges can be observed in other areas too, such as, e.g., landline or mobile phone operators, electricity or gas suppliers, or other.

This work was supported in part by the Technology Agency of the Czech Republic (TAČR) project number TH03010276 and by Czech Science Foundation (GAČR) project number 19-01641S.

© Springer Nature Switzerland AG 2019  
G. Anderst-Kotsis et al. (Eds.): DEXA 2019 Workshops, CCIS 1062, pp. 210–219, 2019.  
[https://doi.org/10.1007/978-3-030-27684-3\\_26](https://doi.org/10.1007/978-3-030-27684-3_26)

In the financial sector, banks primarily evaluate various behavioral or credit scores of both potential and existing clients in order to classify them as fraudulent or legitimate, with various risk levels involved. There are basically three main issues that banks essentially need to solve: (1) fraud detection, (2) debt services default, and (3) customer churn. In addition, various life situations (e.g., finishing studies, getting married, the birth of a child, promotion, loss of a job, retirement, etc.) can be interesting for banks, so that they can offer appropriate financial products to relevant clients, and offer them effectively and in the right moment.

For the purpose of the analysis of bank clients, there are several suitable sources of information. The main options involve: (1) socio-demographic or other information provided by the clients themselves, (2) transactional history or other financial interaction of a client with the bank, and (3) publicly available third-party information that can be linked with the internal client data. In theory, especially the last option seems to provide valuable information so that the identified concerns of banks can be tackled. In practice, however, there are various limitations, technical as well as legal (data and privacy protection or other regulations), which might not permit to use this approach in its full extent.

There exists a number of data mining approaches that try to solve some of the indicated problems [31]. Most of the solutions use some kind of a supervised method [12, 16, 21], where we can predict the future behavior of clients based on a learning data set where the corresponding targets were well known or identified manually. Unsupervised approaches [23] use characteristics of clients in order to group them into clusters on the basis of their mutual similarity, trying to maximize the difference between them.

In this paper, we propose a basic concept and principles of a novel solution inspired by both of these approaches based on the idea of analyzing a social network of clients. Of course, also in this context the idea of utilizing social networks was not only already proposed [2] (e.g., for fraud detection [14, 19, 27] or peer-to-peer lending [17]), but this need is also confirmed by bank representatives themselves. However, in our approach, we go much further. First, we do not assume an existence of such a network (which is a strong, often unrealistic assumption), but based on the available characteristics and history of financial transactions we construct an *inferred* social network. For this purpose, we exploit not only direct facts when describing client relationships, but also features inferred from the available data indirectly using various similarity methods and statistical approaches, all that with additional measures of *reliability* and time-limited *validity* of the information.

Contemporary solutions deployed in banks are often based on techniques such as traditional data processing and querying (projection, selection, and aggregation), integration of externally available information, processing of strings (substrings, n-grams, common subsequences or regular expressions), natural language processing, logistic regression, decision trees, or neural networks. These can help us to solve partial problems such as household, salary, or installment detection. For this purpose, both client and transactional data are exploited: counterparty account numbers are matched to well-known collection accounts (e.g., big sup-

pliers, tax offices, etc.), text comments or other payment symbols accompanying bank transfers analyzed, or descriptions of merchants associated with card payment transactions categorized. All in all, these techniques primarily use rather obvious, direct and only factual kind of information. Having implemented our vision, a new set of possibilities opens, simply because we will be able to apply behavioral patterns observed on clients who just tend to be similar (i.e., clients who would otherwise not be related to each other at all, let alone because they have absolutely no direct or obvious relationships), and simulate propagation of such information, bank decisions and offers, their consequences and impact through the network. For example, two clients can be considered similar, just because they have similar distribution of monthly card payments based on different *Merchant Category Codes* (MCC), money withdrawals carried out abroad, or unexpectedly dwindled overall account balance normalized to the average achieved salary.

As indicated so far, such envisioned networks will permit the banks to view and analyze clients, their characteristics, behavior, and mutual relationships from a new perspective, and provide otherwise impossible insights. However, the outlined vision cannot be attained straightforwardly, since it poses several technical as well as research challenges. In this paper we (1) describe the first results of our project, i.e., the process of inference of the social network from information in bank transactions (see Sect. 2) and (2) envision the open problems and challenges of its analysis and exploitation (not only) for the financial sector (see Sect. 3). We believe that these two contributions will trigger a new research direction for the information retrieval community applicable in many other domains.

## 2 Inferred Social Network

In our first use case from the financial sector, we assume the possession of the following input (anonymized) data:

1. client characteristics (e.g., name, addresses, education, etc.),
2. history of financial transactions (e.g., credit/debit card payments, payments with cash back service, cash withdrawals, transfers, permanent transfer orders, regular or overdue loan installments, etc.), and, eventually,
3. third-party information related to the non-person entities from the previous two data sets (e.g., shops, institutions, etc.).

In all the cases, only data belonging to a particular monitored time interval  $I = [t_{start}, t_{end}]$  is covered. In the following text, we provide a more precise definition of the target social network and the process of its inference from the input data.

### 2.1 Social Network Graph

We construct the social network of bank clients at two different levels of granularity, representing a dual view of the same problem from different perspectives:

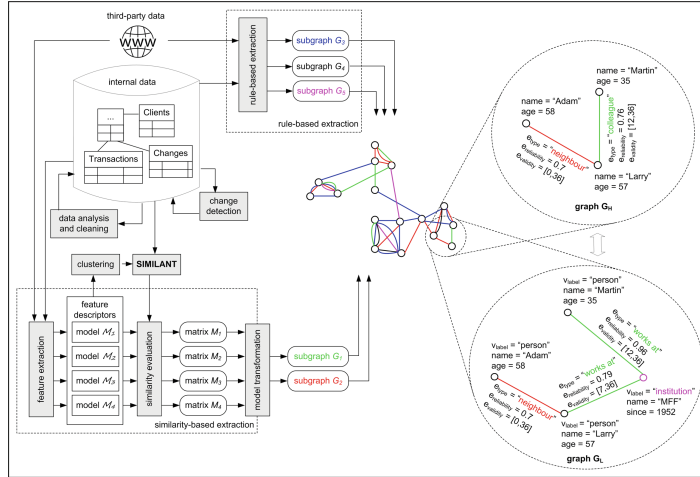


Fig. 1. The process of inference of a social network

(1) *high-level* view, where the social network only consists of vertices for bank clients, and (2) *low-level* view, where there are vertices also for different kinds of real-world entities (e.g., institutions, companies, etc.). Both the views are depicted in Fig. 1 as graphs  $G_H$  and  $G_L$  on the right.

**High-Level Network.** We define a *high-level network* to be a multigraph  $G_H = (V_H, E_H)$ , where the set of vertices  $V_H$  represents individual bank clients and the set of edges  $E_H$  relationships among the clients. Each client  $v \in V_H$  has a set of properties  $\{p_1, p_2, \dots, p_n\}$ , each of which is modeled as a tuple with the following components:

- name  $p_{name}$  (e.g., *age*),
- value  $p_{value}$  (e.g., *58*),
- reliability  $p_{reliability} \in [0, 1]$  (e.g., *0.85* for information certain for 85%), and
- validity  $p_{validity} = [t_{start}, t_{end}]$  representing the time interval of validity of the information.

Each edge  $e \in E_H$  has the following components:

- relationship type  $e_{type}$  (e.g., *colleague*),
- reliability  $e_{reliability} \in [0, 1]$ , and
- validity  $e_{validity} = [t_{start}, t_{end}]$ .

The high-level view  $G_H$  has the advantage of having a simple data structure to work with. It enables to perform general analyses without the need to distinguish between types of vertices and to know and understand technical details of the reality. For example, we are 85% sure that two clients work in the same company, but we do not (need to) know which company.

**Low-Level Network.** In case we are interested in more details (e.g., we want to know particulars about a given company), we can use the low-level view  $G_L$ , where we also work with vertices for different real-world entities, possibly enriched by third-party publicly available information about them.

Formally, we model this *low-level network* as a multigraph  $G_L = (V_L, E_L)$ , where the set of vertices  $V_L$  involves  $V_H$  plus vertices for new kinds of entities, each one of them newly associated also with a vertex type  $v_{label}$ , as well as properties  $\{p_1, p_2, \dots, p_n\}$  with the unchanged structure as in the original high-level network  $G_H$ . The edges in  $E_L$  represent relationships among clients and institutions, locations, etc., also having  $e_{type}$ ,  $e_{reliability}$ , and  $e_{validity}$ .

## 2.2 Inference Process

As usual with real-world data sets, the data first need to be pre-processed using a *data analysis and cleaning* module. In this step, we can already identify information whose reliability is  $<1$  due to lower data quality. However, this feature will mainly be adjusted by the inference process. Similarly, from the input data, we already know the basic time intervals of validity to be further refined.

The process of inference of graph  $G_H$  (or  $G_L$ ) consists of two orthogonal approaches which we denote as *rule-based* extraction and *similarity-based* extraction (see Fig. 1 on the left).

**Rule-Based Extraction.** The more straightforward inference path is based on the idea of the definition of a set of domain-specific rules, each of which defines a way of constructing a particular type (or types) of relationships in the target graph  $G_H$  (or  $G_L$ ). So far, we distinguish the following classes:

- *factual relationships* directly present in the input data (e.g., a married couple),
- *aggregated relationships* derived from the input data using various information aggregations (e.g., co-workers defined as people working concurrently in the same company), and
- *transitive relationships* derived from the input data on the basis of statistically significant amount of occurrences of specific events (e.g., people living in the same household defined as people with the same home address, frequent mutual money transfer, frequent payments in similar shops, etc.). In this case usually  $e_{reliability} < 1$ .

**Similarity-Based Extraction.** The process of data extraction in this inference path is more complex and, to the best of our knowledge, also unique in the given context, so we describe it in more detail. We first define a set of similarity models  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$ , each of which expresses a particular similarity of users defined by certain selected *features*. These features can be of various types, such as based on single fixed values, time-varying series, accumulated values (e.g., per week or month), etc., or their combinations. With regards to the monitored time interval



$I$ , we can focus on the whole  $I$ , or just its part (e.g., since the moment a client entered a contract with the bank).

Within a model  $\mathcal{M}_I$ , values of such features for a given particular client  $c$  are represented using a descriptor  $d_c^I$ ; mostly a vector or time series where individual coordinates/elements represent values in the selected features (e.g., a normalized overall volume of performed debit card payment transactions, converted into the base currency, all that separately for every month of interval  $I$ ). Having a set of descriptors for all the clients, standard approaches can be exploited to calculate similarities of the clients using the selected metrics, configured weights, or other parameters [30].

For each model  $\mathcal{M}_I$ , this step thus outputs a dense square matrix  $M_I$ , where for each two vertices  $v_i$  and  $v_j \in V_H$  (or  $V_L$ ), the measure of their mutual similarity  $sim_I(v_i, v_j) \in [0, 1]$  with regards to the corresponding descriptors is stored at  $M_I[i, j]$ . The efficient matrix computation<sup>1</sup> could be implemented by similarity self-join on  $V_H$  (or  $V_L$ ), e.g., using the Hadoop MapReduce algorithms [6].

Next, the set of dense matrices  $M_1, M_2, \dots, M_k$  is transformed to a set of sparse matrices  $M'_1, M'_2, \dots, M'_m$  ( $k$  and  $m$  do not necessarily need to be equal), each representing one type of relationships to be added into the target graph, each particular edge  $e$  with the value of reliability  $e_{reliability}$  corresponding to  $M'_i[i, j]$ . The “sparsing” transformation can involve application of similarity thresholds, combination of multiple similarity results (e.g., using a weighted sum), clustering of clients into categories, etc. In general, we want to restrict the result just to the most interesting information with a reasonable size.

To summarize, the similarity-based extraction is a method for indirect detection of relationships between clients (or entities, in general). While the rule-based extraction can reach  $e_{reliability} = 1$  and well-defined semantics of the relationships, there is usually a limited amount of data available. The similarity-based extraction, on the other hand, provides weaker semantics and  $e_{reliability} < 1$  of relationships (in fact, it is correlated with the similarity scores), but it still allows to identify relationships in situations where the rule-based extraction is not applicable (due to, e.g., a small amount of data, a small number of direct relationships, etc.).

**SIMILANT.** For the analysis of promising similarity models to be used in the similarity-based extraction, we have designed and implemented an analytics tool called SIMILANT. It enables effective browsing of data based on the chosen similarity measure. It consists of three independent parts: clustering, visualization, and browser. During the extraction process, we create a complete weighted graph (i.e., the dense similarity matrix  $M_I$ ), which is, however, hard to interpret. Instead of its visualization, we visualize only a small number of related clusters. The browser part of SIMILANT provides simple analytics of similarity models at different levels of detail (e.g., different numbers of these clusters). Their semantics can be further validated using various *targets* (i.e., labeled ground truth), if available.

<sup>1</sup> Instead of fully materialized matrices approximations can be used.

**Dynamics of the Input Data.** An important feature of the input data set is its dynamics. One aspect is that new data continuously appear. Another one is the event-driven nature of the problem domain. For example, there are various life situations (e.g., the loss of a job or a notable increase in salary) highly influencing the financial behavior of bank clients and, consequently, the structure of the graph representing their financial behavior. In order to prevent skewed data, a *change detection* module determines points of such changes using a set of predefined domain-specific rules combined with statistical analyses. When such a change connected with a given client is identified at time  $t_{change}$ , instead of working with the entire interval  $I = [t_{start}, t_{end}]$ , we can technically split a given client behavior into two sub-intervals  $I_1 = [t_{start}, t_{change}]$  and  $I_2 = [t_{change}, t_{end}]$  – before and after the change – and study a given client within the two (or more) intervals.

### 2.3 Social Network Analysis

Having the inferred (high-level or low-level) social network, in the next phase of our project, we will focus on its thorough analysis. Currently, there exist several verified approaches for social network analysis which primarily need to be utilized for validity and reliability of the information. In addition, since the inferred network is not built by people themselves, quite probably its features will not correspond to features of traditional social networks. As a consequence, verified approaches may need to be optimized, modified, or even identified as inapplicable for inferred networks.

In general, our main near-future target areas are as follows:

- *Visualization*: We will focus on the visualization of the inferred network with filters related to both validity and reliability. In addition, we need to visualize the dynamics of the graph.
- *Structural Analysis*: We will utilize traditional methods [14], like, e.g., analysis of density, centrality, structural holes, clustering coefficient, communities, etc. Due to the specifics of inferred networks, we assume that the characteristics will not correspond to usual observations for social networks [24].
- *Information Propagation*: Inspired by sociology, we will deal with how information is propagated [8] in inferred networks, primarily using *independent cascade* or *linear threshold* models, both progressive stochastic diffusion models.

These steps will lead to the main target: to identify the most relevant information for the financial sector in order to solve the three key issues and optimize customer services.

## 3 Challenges

The envisioned idea of inferring a social network from information describing people and their interactions can be used in many other domains. Regardless of

the domain, such inferred social networks will probably have different features and thus will require novel approaches for their processing and analysis. We believe that this idea opens a new, challenging, and highly practically applicable research area for the information retrieval community. Besides the previous list of target research areas of our project, we summarize other related challenges of inferred networks as follows:

- *Big Graph Data*: Even a small bank can have hundreds of thousands of clients and hundreds of transactions per a client and month and the inferred social network can be very large. Approaches, such as *network embedding* [28], that allow us to compact the information necessary to describe its structure can increase the efficiency of its processing. Even if this should lead to approximations. Other challenges are observed in big graphs with high-degree vertices, randomness in graph structure or other irregularities, different types of edges, or in graphs with dynamic changes, causing sharding, data locality or load balancing issues [20, 25, 29]. Data quality issues, although studied for decades, also require attention in the context of Big Data processing [3, 13, 15].
- *Time-Varying Data*: Various time aspects of features and behavior of people are natural. Hence, management of data with a temporal dimension has already been addressed in a number of fields, from relatively old *temporal databases* [9, 22] for relational data to newer *time-varying networks* [11] for which there exists an extensive amount of specific approaches [4], including dedicated data stores [5]. Inferred social networks have the time aspect too, however, its processing needs to be further adjusted with regards to their other specific features, such as (time-varying) reliability.
- *Feature Selection*: The similarity-based extraction path brings a challenge in exclusion of redundant or irrelevant descriptors whose number can be extremely large. While the *supervised* approaches (which require additional information) are well explored [1, 7, 18], the *unsupervised* ones are more suitable in the considered context, but also more challenging and thus less represented. For example, several criteria and unsupervised approaches are discussed in [10], whereas paper [26] proposes how to measure the validity of feature selection objectively. Since the target inferred social network cannot be extremely dense, the choice of the most relevant data is a crucial issue in general.

## References

1. Ang, J.C., Mirzal, A., Haron, H., Hamed, H.N.A.: Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**(5), 971–989 (2016). <https://doi.org/10.1109/TCBB.2015.2478454>
2. Baesens, B., Vlasselaer, V.V., Verbeke, W.: *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*, 1st edn. Wiley, Hoboken (2015)

## 7. ADVANCED BEHAVIORAL ANALYSES USING INFERRED SOCIAL NETWORKS: A VISION

---

218 I. Holubová et al.

3. Cai, L., Zhu, Y.: The challenges of data quality and data quality assessment in the big data era. *Data Sci. J.* **14**(2), 1–10 (2015). <https://doi.org/10.5334/dsj-2015-002>
4. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. In: Frey, H., Li, X., Ruehrup, S. (eds.) *ADHOC-NOW 2011*. LNCS, vol. 6811, pp. 346–359. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22450-8\\_27](https://doi.org/10.1007/978-3-642-22450-8_27)
5. Cattuto, C., Quaggiotto, M., Panisson, A., Averbuch, A.: Time-varying social networks in a graph database: a Neo4j use case. In: *First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013*, pp. 11:1–11:6. ACM, New York (2013). <https://doi.org/10.1145/2484425.2484442>
6. Čech, P., Maroušek, J., Lokoč, J., Silva, Y.N., Starks, J.: Comparing mapreduce-based k-NN similarity joins on hadoop for high-dimensional data. In: Cong, G., Peng, W.-C., Zhang, W.E., Li, C., Sun, A. (eds.) *ADMA 2017*. LNCS (LNAI), vol. 10604, pp. 63–75. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-69179-4\\_5](https://doi.org/10.1007/978-3-319-69179-4_5)
7. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. *Comput. Electr. Eng.* **40**(1), 16–28 (2014). <https://doi.org/10.1016/j.compeleceng.2013.11.024>
8. Chen, W., Lakshmanan, L.V., Castillo, C.: Information and Influence Propagation in Social Networks. *Synthesis Lectures on Data Management*, vol. 5, no. 4, pp. 1–177 (2013). <https://doi.org/10.2200/S00527ED1V01Y201308DTM037>
9. Date, C.J., Darwen, H., Lorentzos, N.A.: *Temporal Data and the Relational Model*. Elsevier, Amsterdam (2002)
10. Dy, J.G., Brodley, C.: Feature subset selection and order identification for unsupervised learning. In: *Proceedings of the Seventeenth International Conference on Machine Learning*, October 2000
11. Holme, P., Saramäki, J.: Temporal networks. *Phys. Rep.* **519**(3), 97–125 (2012)
12. Islam, S.R., Eberle, W., Ghafoor, S.K.: Mining bad credit card accounts from OLAP and OLTP. *CoRR abs/1807.00819* (2018). <http://arxiv.org/abs/1807.00819>
13. Katal, A., Wazid, M., Goudar, R.H.: Big data: issues, challenges, tools and good practices. In: *2013 Sixth International Conference on Contemporary Computing (IC3)*, pp. 404–409, August 2013. <https://doi.org/10.1109/IC3.2013.6612229>
14. Kirchner, C., Gade, J.: *Implementing social network analysis for fraud prevention* (2011)
15. Kwon, O., Lee, N., Shin, B.: Data quality management, data usage experience and acquisition intention of big data analytics. *Int. J. Inf. Manag.* **34**(3), 387–394 (2014). <https://doi.org/10.1016/j.ijinfomgt.2014.02.002>
16. Lessmann, S., Baesens, B., Seow, H., Thomas, L.C.: Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. *Eur. J. Oper. Res.* **247**(1), 124–136 (2015). <https://doi.org/10.1016/j.ejor.2015.05.030>
17. Lin, M., Prabhala, N.R., Viswanathan, S.: Judging borrowers by the company they keep: friendship networks and information asymmetry in online peer-to-peer lending. *Manag. Sci.* **59**(1), 17–35 (2013). <https://doi.org/10.1287/mnsc.1120.1560>
18. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **4**, 491–502 (2005)
19. Lookman, S., Nurcan, S.: A framework for occupational fraud detection by social network analysis. In: *Proceedings of the CAiSE 2015 Forum at the 27th International Conference on Advanced Information Systems Engineering co-located with (CAiSE 2015)*, Stockholm, Sweden, 10 June 2015, pp. 221–228 (2015). <http://ceur-ws.org/Vol-1367/paper-29.pdf>

20. Nai, L., Xia, Y., Tanase, I.G., Kim, H., Lin, C.: GraphBIG: understanding graph computing in the context of industrial solutions. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, pp. 1–12, November 2015. <https://doi.org/10.1145/2807591.2807626>
21. Ngai, E.W.T., Hu, Y., Wong, Y.H., Chen, Y., Sun, X.: The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. *Decis. Support Syst.* **50**(3), 559–569 (2011). <https://doi.org/10.1016/j.dss.2010.08.006>
22. Ozsoyoglu, G., Snodgrass, R.T.: Temporal and real-time databases: a survey. *IEEE Trans. Knowl. Data Eng.* **7**(4), 513–532 (1995). <https://doi.org/10.1109/69.404027>
23. Quah, J.T.S., Sriganesh, M.: Real-time credit card fraud detection using computational intelligence. *Expert Syst. Appl.* **35**(4), 1721–1732 (2008). <https://doi.org/10.1016/j.eswa.2007.08.093>
24. Santoro, N., Quattrociochi, W., Flocchini, P., Casteigts, A., Amblard, F.: Time-varying graphs and social network analysis: temporal indicators and metrics. In: 3rd AISB Social Networks and Multiagent Systems Symposium (SNAMAS), United Kingdom, pp. 32–38, May 2011. <https://hal.archives-ouvertes.fr/hal-00854313>
25. Singh, D.K., Patgiri, R.: Big graph: tools, techniques, issues, challenges and future directions. In: Sixth International Conference on Advances in Computing and Information Technology (ACITY 2016), pp. 119–128 (2016)
26. Tang, J., Alelyani, S., Liu, H.: Feature selection for classification: a review. In: *Data Classification: Algorithms and Applications* (2014)
27. Vlasselaer, V.V., et al.: APATE: a novel approach for automated credit card transaction fraud detection using network-based extensions. *Decis. Support Syst.* **75**, 38–48 (2015). <https://doi.org/10.1016/j.dss.2015.04.013>
28. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234. ACM (2016)
29. Xia, Y., et al.: Graph analytics and storage. In: 2014 IEEE International Conference on Big Data (Big Data), pp. 942–951, October 2014. <https://doi.org/10.1109/BigData.2014.7004326>
30. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search - The Metric Space Approach. *Advances in Database Systems*, vol. 32. Kluwer, Dordrecht (2006). <https://doi.org/10.1007/0-387-29151-2>
31. Zhou, J.: Data mining for individual consumer credit default prediction under e-commerce context: a comparative study. In: Proceedings of the International Conference on Information Systems - Transforming Society with Digital Innovation, ICIS 2017, Seoul, South Korea, 10–13 December 2017 (2017)



---

## **SIMILANT: An Analytic Tool for Similarity Modeling**

**Bernhauer, D. (20 %); Skopal, T. (20 %), Holubová, I. (20 %); Peška, L. (20 %); Svoboda, M. (20 %)** SIMILANT: An Analytic Tool for Similarity Modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, ACM, 2019, <https://doi.org/10.1145/3357384.3357852>doi: 10.1145/3357384.3357852 [A.8]

## SIMILANT: An Analytic Tool for Similarity Modeling

David Bernhauer  
bernhdav@fit.cvut.cz  
Faculty of Information Technology,  
Czech Technical University in Prague  
Prague, Czech Republic

Tomáš Skopal  
skopal@ksi.mff.cuni.cz  
Faculty of Mathematics and Physics,  
Charles University  
Prague, Czech Republic

Irena Holubová  
holubova@ksi.mff.cuni.cz  
Faculty of Mathematics and Physics,  
Charles University  
Prague, Czech Republic

Ladislav Peška  
peska@ksi.mff.cuni.cz  
Faculty of Mathematics and Physics,  
Charles University  
Prague, Czech Republic

Martin Svoboda  
svoboda@ksi.mff.cuni.cz  
Faculty of Mathematics and Physics,  
Charles University  
Prague, Czech Republic

### ABSTRACT

We present SIMILANT, a data analytics tool for modeling similarity in content-based retrieval scenarios. In similarity search, data elements are modeled using black-box descriptors, where a pair-wise similarity function is the only way how to relate data elements to each other. Only these relations provide information about the dataset structure. Data analysts need to identify meaningful combinations of descriptors and similarity functions effectively. Therefore, we proposed a tool enabling a data analyst to systematically browse, tune, and analyze similarity models for a specific domain.

### KEYWORDS

similarity modeling and analytics; similarity visualization

### ACM Reference Format:

David Bernhauer, Tomáš Skopal, Irena Holubová, Ladislav Peška, and Martin Svoboda. 2019. SIMILANT: An Analytic Tool for Similarity Modeling. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3357384.3357852>

### 1 INTRODUCTION

The similarity search model is a popular retrieval model for unstructured data, such as multimedia, text, time series, and other complex unstructured data (often sensory data). By definition, in similarity search, the data objects are modeled by descriptors that are to be compared by a similarity function. Both descriptors and similarity functions (i.e., the similarity model) are not specified for the database operations such as indexing or retrieval algorithms (i.e., being a black box), while their implementation is left to the domain-specific modules of the retrieval system. This separation of the general retrieval logic and the domain-specific similarity modeling provides a great degree of extensibility, as only the domain-specific

modules need to be implemented as plug-ins to a general similarity search engine. The most popular similarity search model is the metric space model [10] allowing to index data by similarity for efficient (fast) retrieval.

Software tools for data analytics mostly focus on structured (relational) data, providing aggregations and data mining over individual attributes. Unlike this traditional approach that provides access to data at the level of domain-specific attributes, in our tool called SIMILANT (SIMILarity ANalytics Tool) we support to plug in arbitrary black-box similarity models, yet providing data analytics functionalities. SIMILANT aims at visualization and aggregation of similarity relationships between data objects at the level of atomic elements and their clusters. The main benefit of this approach is that relationships between database objects are based purely on their overall similarities, freeing the data analyst from the domain-specific data structure and enabling analysis at a higher level of abstraction.

The main highlights of SIMILANT can be described using the following two use cases:

- *Discovering Implications:* As we do not know the semantics of the data source, we try to discover interesting implications based on the descriptors and similarity functions. The same descriptor can provide different information for different similarity functions and vice versa. SIMILANT provides an ability to generate and process a large volume of combinations of a descriptor and a similarity function, as well as visualization and browsing of the pre-processed data. The user can navigate through different combinations and evaluate the quality of the proposed models. The discovered implications can then be selected for further processing in different similarity models (as new descriptors). An example from the bank domain is, e.g., to identify accounts of governmental institutions based on their incoming transfers pattern.
- *Cluster-based Validation:* Every classification task has its limits, at least at the noise level. Due to the component architecture of SIMILANT, we can replace the clustering algorithm for a classifier. We can observe the quality of the classifier with respect to a chosen descriptor and find the particular problematic entries. Using the descriptor chart, we can justify the decision of the classifier and identify a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
CIKM '19, November 3–7, 2019, Beijing, China  
© 2019 Association for Computing Machinery.  
ACM ISBN 978-1-4503-6976-3/19/11...\$15.00  
<https://doi.org/10.1145/3357384.3357852>



group of similar poorly classified entries (and its structure). As a result, we can design an improvement of an existing classifier by considering additional data features (feature selection) based on the particular similarity model.

In the presentation, we will demonstrate the functionality and advantages of SIMILANT using two real-world scenarios that result from two distinct research projects we are participating in. By choosing these two different domains we want to also show the extensibility and general usability of SIMILANT.

### 1.1 Related Work

As mentioned above, data analytics tools (such as, e.g., Rapid-Miner [4] or R [8]) mostly focus on structured data. A different approach was considered in [2], proposing a visual analytics tool for high dimensional data. The tool mostly focuses on visualizing individual clusters and provides users (domain experts) the capability to manually update the clusters. In contrast, SIMILANT focuses on evaluation and validation of multiple possible clusterings induced by the combination of descriptors and similarity functions and also finding implications towards relevant labels.

In fact, after a thorough review of related work we found only a single closely related application called SIMG-VIZ [6]. SIMG-VIZ is a system for identifying semantically similar entities. As the authors mentioned, the entity resolution part can be replaced for any pre-computed similarity graph. SIMG-VIZ visualizer targets the graph representation of elements, whereas the graph can be dense and, hence, the visualization unclear. SIMILANT aims to visualize (a reasonable amount of) clusters as groups of similar entities with a common property. Furthermore, SIMG-VIZ is presented as the system for entity resolution, so the analyst must have some domain knowledge. SIMILANT uses the visualization for discovering and browsing the data without any prior knowledge of the domain.

## 2 PIPELINE

Currently, tasks of SIMILANT pipeline are implemented in Python Notebooks. As we discuss in Section 2.3, our module implementation is not critical since the final result of the whole pipeline is a static file. Python Notebooks are perfect for demonstration, but in the production environment, we can switch them for more efficient scripting frameworks.

The pipeline of similarity analysis in SIMILANT consists of several steps as depicted in Figure 1. For every combination of a descriptor and a similarity function, we can choose the optimal clustering algorithm and a suitable visualization of clusters. This fact indicates that the potential space of combinations is enormous. In this section, we introduce in more detail each part of SIMILANT. The data browser, the last step of the pipeline, is described in Section 3.

### 2.1 Descriptor Identification

During our original similarity task targeting bank clients, we have found out that without a more in-depth knowledge of the banking domain we can create even hundreds or thousands descriptors, not knowing which will (not) be usable. Moreover, we can design tens of similarity functions for each descriptor, while this problem is not specific to bank clients only. The aim of SIMILANT is to minimize

the necessary effort and to discover the promising similarity models efficiently using a semi-automatic process.

In the first step, we pre-compute similarity matrices for all the combinations. The computation can be done automatically on a server, either sequentially or in parallel.

### 2.2 Clustering & Visualization

Having all the possible similarity models computed, in the second step SIMILANT enables to cluster and visualize the data respectively, together with information about the quality of the model.

The similarity is not just about a relationship of elements. We perceive similarity as a group of similar data elements that share a certain property (or even a semantic category). As the visualization of every data element can be confusing, we utilize clustering algorithms. In particular, different variants of hierarchical clustering are suitable for the task due to its natural ability to quickly produce arbitrary volumes of clusters. In the current implementation, agglomerative hierarchical clustering with average linkage [5] is used; however, the choice of the clustering algorithm is independent of the other components, and thus can be replaced if necessary.

In the current implementation, PCA [1], MDS [7] or t-SNE [9] visualization approaches are used to layout the sample of objects or directly the clusters (see Figures 2 and 3 described in detail in Section 3). In case of cluster visualization, the size of the dots represents the logarithmic size of a cluster (i.e., the number of its elements). The shade of a dot represents the distance span of elements within the particular cluster (cluster radius). In order to evaluate the applicability of a particular set of descriptors and a clustering method, we can validate the resulting clusters against the selected target attributes. For this purpose, the *information gain* [3] (i.e., the amount of information gained about a target variable by separating data entries assigned to the current cluster from others) is calculated w.r.t. each cluster and target. The information gain is currently displayed as a record in cluster's statistics; however, in the future work, we plan to implement a target-specific cluster visualization depicting target attribute's distribution as RGB channels of the cluster's color, while the information gain of the cluster will correspond with the alpha channel.

### 2.3 Modularity

One of the important requirements resulting from the use case analysis is that every part of the pipeline should be replaceable. As a result of a clearly defined interface between the pipeline components, the proposed tool is highly modular and it is not limited just to the presented clustering and visualizing algorithms, neither specific types of descriptors or similarity functions.

## 3 DATA BROWSER

The data browser is a web-based GUI entry point to SIMILANT. The current layout consists of three main parts; however, the architecture also allows for an extension of each part, or even a change of the whole layout structure.

### 3.1 Layout

Figure 2 shows the layout of SIMILANT. The left part is the control panel of the whole tool. In the top-left part (marked as #1), the

# 8. SIMILANT: AN ANALYTIC TOOL FOR SIMILARITY MODELING

Session: Demo - Demo Session 1

CIKM '19, November 3-7, 2019, Beijing, China

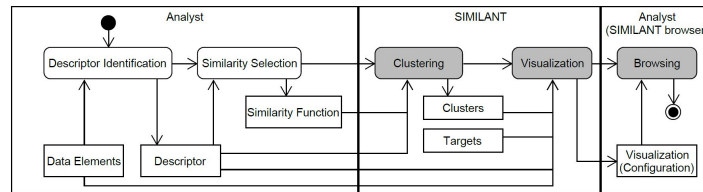


Figure 1: SIMILANT pipeline

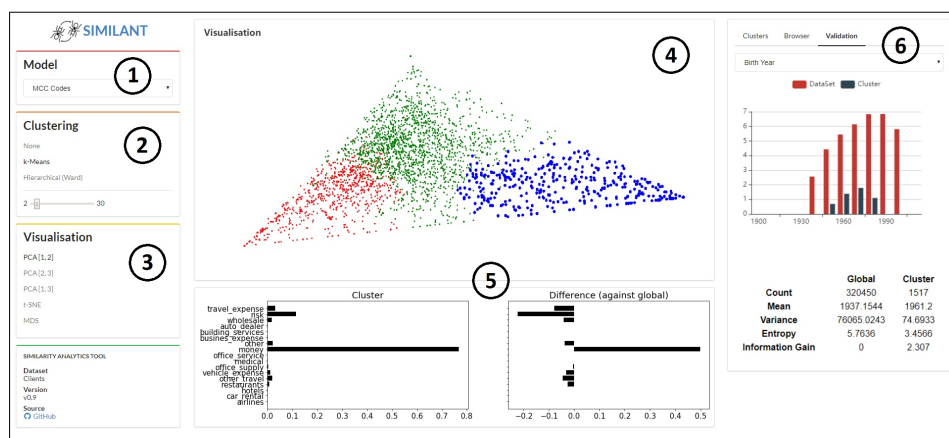


Figure 2: Screenshot of SIMILANT GUI for bank clients

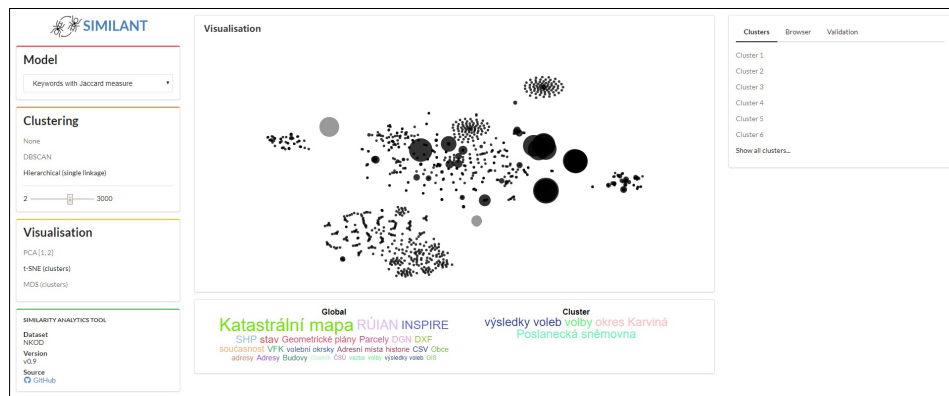


Figure 3: Screenshot of SIMILANT GUI for Linked Data

analyst can select the desired combination of a descriptor and a similarity function. In the next step (#2), the analyst can choose the clustering algorithm and its parameters (usually the number of clusters). The last step (#3) is to select the visualization algorithm. Currently, SIMILANT enables to visualize a sample of objects (shown in Figure 2), or clusters only (shown in Figure 3).

After the initial configuration, the analyst focuses on the central part (#4), since it visualizes the data in 2D and gives information about the partitioning of the dataset. The analyst can easily verify whether the clustering makes sense, while tab *Validation* (#6) provides additional statistical information. We can select a particular cluster by a mouse click or select it from list *Clusters*, which is shown in Figure 3 on the right. The color of an individual object represents its cluster (shown in Figure 2), the shade of the cluster determines the radius of the cluster (shown in Figure 3). In the bottom part, there is a summary of a descriptor (#5) of the chosen cluster or the whole dataset. Visualization of descriptors is also modular: currently, we are able to visualize histograms, tag clouds (shown in Figure 3), or time series.

In the right part (#6), we can select a particular cluster, browse the data set (if there is additional information about particular objects), or validate clusters against a chosen target. As we can see in the example, the descriptor suggests that the selected cluster represents clients who often perform cash withdrawals from ATMs (represented by a visible peak of the money category in the bottom-right histogram, #5). The validation panel indicates that these people are mostly middle-aged. Finally, the table below the chart (a graphical representation of the target) shows statistical indicators. In this example, we can accept this model and mark the clustering as appropriate for further processing.

### 3.2 Considerations

One of the controversial parts of SIMILANT is the offline mode of the browser. First, we need to generate the necessary files using the pipeline, only then we can visualize them. The opposite approach presented in [6] is an application communicating with an independent server. The disadvantage of this method are time-consuming operations during which the application does not display anything useful. We believe that the response time is a crucial aspect of every application, and therefore should be considered wisely.

Another issue is Big Data, that require a large volume of data transfer, which can be a source of high network traffic and other problems. Since we work only with clusters, we can reduce the size of the files. In the case of bigger clusters we can encounter a longer processing time. The large clusters cause many DOM model changes (during the preparation of element list), which can be difficult for the browser to handle.

## 4 DEMONSTRATION OUTLINE

In the demonstration, we will present individual parts of SIMILANT pipeline, step by step. Sample descriptors and similarity functions will be processed by the clustering and visualization algorithm. The results will be imported into the SIMILANT browser.

We will use two real-world industry-based datasets. Firstly, we will present how to utilize SIMILANT in case of bank clients. In particular, we will present a similar scenario, as shown in Figure

1. We will work with anonymized data about bank clients and divide them into several groups based on a transactional history. The targets (e.g., the histogram in *Validation* Tab) will be used to indicate that the clients in the selected cluster are well separated. In our example, we can see that the information gain for the selected cluster is about 40% of the original entropy, which is a sign of a good partitioning regarding the age of clients. The descriptor summary shows that these clients are more likely to withdraw from an ATM than an average client. So we can estimate the age category of clients based on their card payment behavior.

The second data set involves linked open data repositories. Since these are unstructured data, too, Figure 3 shows a trivial clustering based on the identity. In this case, we will demonstrate the importance of the visualization part. As it is evident, some clusters are similar to the others. However, the targets are not available, and so we will have to utilize a cluster validity index, such as cohesion and separation. The cohesion and separation indices can suggest the optimal size of the clustering. Therefore, in this case, we will show how to find a more coarse-grained distribution.

For both examples, other similarity models, types of clustering, and visualizations will be available during the demonstration (and to be played with by the audience).

### 4.1 Intended Audience

This demonstration is intended for a wide scope of audience, ranging from researchers looking both for a possible way of solving their research issues and for new research directions in general, to practitioners who can utilize SIMILANT for their particular use cases. Since the ideas can be applied universally regardless of the chosen domain, SIMILANT represents a unique tool with an extremely wide area of well-motivated applications.

### Acknowledgments

This research has been supported in part by Czech Science Foundation (GAČR) project number 19-01641S and Technology Agency of the Czech Republic (TAČR) project number TH03010276.

### REFERENCES

- [1] H. Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* (1933).
- [2] E. J. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. 2007. ClusterSculptor: A Visual Analytics Tool for High-Dimensional Data. In *2007 IEEE Symposium on Visual Analytics Science and Technology*. 75–82. <https://doi.org/10.1109/VAIST.2007.4388999>
- [3] J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (01 Mar 1986), 81–106. <https://doi.org/10.1007/BF00116251>
- [4] Inc. RapidMiner. 2006–2019. RapidMiner. <http://rapidminer.com/>.
- [5] Lior Rokach and Oded Maimon. 2005. *Clustering Methods*. Springer US, Boston, MA, 321–352. [https://doi.org/10.1007/0-387-25465-X\\_15](https://doi.org/10.1007/0-387-25465-X_15)
- [6] M. Ali Rostami, Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Interactive Visualization of Large Similarity Graphs and Entity Resolution Clusters. In *Proceedings of the 21st International Conference on Extending Database Technology (EDBT 2018)*. OpenProceedings, 690–693. <https://doi.org/10.5441/002/edbt.2018.01>
- [7] Heng Tao Shen. 2009. *Multidimensional Scaling*. Springer US, Boston, MA, 1784–1784. [https://doi.org/10.1007/978-0-387-39940-9\\_548](https://doi.org/10.1007/978-0-387-39940-9_548)
- [8] R Core Team. 1993–2019. The R Project for Statistical Computing. <https://www.r-project.org/>.
- [9] L. van der Maaten and G. E. Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Research* (2008).
- [10] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. 2006. *Similarity Search - The Metric Space Approach*. Advances in Database Systems, Vol. 32. Kluwer. <https://doi.org/10.1007/0-387-29151-2>



---

## Non-metric Similarity Search Using Genetic TriGen

**Bernhauer, D. (50 %); Skopal, T. (50 %)** Non-metric Similarity Search Using Genetic TriGen. In *Similarity Search and Applications: 13th International Conference, SISAP 2019*, Newark, NJ, USA: Springer, 2019, doi:10.1007/978-3-030-32047-8\_8. [A.9]

The paper has been cited in:

- Connor, R.; Dearle, A.; Mic, V.; Zezula, P. On the application of convex transforms to metric search. *Pattern Recognition Letters*, volume 138, 2020, doi: 10.1016/j.patrec.2020.08.008.
- Beecks, C.; Berns, F.; Schmidt, K. W. Ptolemaic indexing for managing and querying internet of things (IoT) data. In *2019 IEEE International Conference on Big Data, (Big Data)*. IEEE, 2019, doi: 10.1109/BigData47090.2019.9005725.



## Non-metric Similarity Search Using Genetic TriGen

David Bernhauer<sup>1,2</sup> and Tomáš Skopal<sup>2</sup>(✉)

<sup>1</sup> Faculty of Information Technology,  
Czech Technical University in Prague, Prague, Czech Republic  
[bernhdav@fit.cvut.cz](mailto:bernhdav@fit.cvut.cz)

<sup>2</sup> Faculty of Mathematics and Physics,  
Charles University, Prague, Czech Republic  
[skopal@ksi.mff.cuni.cz](mailto:skopal@ksi.mff.cuni.cz)

**Abstract.** The metric space model is a popular and extensible model for indexing data for fast similarity search. However, there is often need for broader concepts of similarities (beyond the metric space model) while these cannot directly benefit from metric indexing. This paper focuses on approximate search in semi-metric spaces using a genetic variant of the TriGen algorithm. The original TriGen algorithm generates metric modifications of semi-metric distance functions, thus allowing metric indexes to index non-metric models. However, “analytic” modifications provided by TriGen are not stable in predicting the retrieval error. In our approach, the genetic variant of TriGen – the TriGenGA – uses genetically learned semi-metric modifiers (piecewise linear functions) that lead to better estimates of the retrieval error. Additionally, the TriGenGA modifiers result in better overall performance than original TriGen modifiers.

**Keywords:** Approximate similarity search · Semi-metric space · Genetic TriGen

### 1 Introduction

The similarity search models stand in the center of methods for content-based retrieval in datasets of multimedia and other unstructured data. For decades, the metric space model [7] has been widely accepted as the standard model for similarity search applications. The metric space model is both extensible (supporting black-box descriptors and similarities) as well as indexable (due to metric properties), thus providing efficient similarity search by metric access methods (MAMs) [2, 7].

In the era of Big data – with the increasing diversity and complexity of data and algorithms for entity matching – there is a need for more general schemes of similarity modeling. The metric space properties could be too restrictive in many fields [6], for example in bioinformatics/cheminformatics. At the same time, the datasets grow to sizes that are not possible to query without indexing.

© Springer Nature Switzerland AG 2019  
G. Amato et al. (Eds.): SISAP 2019, LNCS 11807, pp. 86–93, 2019.  
[https://doi.org/10.1007/978-3-030-32047-8\\_8](https://doi.org/10.1007/978-3-030-32047-8_8)

Hence, it is extremely challenging to provide scalability in both retrieval aspects – the effectiveness (retrieval quality) and efficiency (system performance). There have been many approaches developed, trading effectiveness for efficiency, e.g., approximate search methods. However, most of the results were based on the metric space model. Only a few approaches considered a more general non-metric case, one of which is the TriGen algorithm [5]. In this paper, we build on the idea of TriGen-based modification of non-metric space into approximate metric space, that enables metric indexing of (initially) non-metric data models for approximate search. As a contribution, we introduce a variant of TriGen based on genetic algorithm, that produces more robust semimetric-to-metric modifiers and better efficiency-effectiveness tradeoffs.

## 2 Non-metric/Approximate Similarity Search by TriGen

When talking about non-metric similarities, we usually consider distance functions that do not satisfy some of the metric axioms (reflexivity, non-negativity, symmetry, triangle inequality). Most of the practical non-metric distances actually miss just one of the axioms, like pseudo-metrics (reflexivity), quasi-metrics (symmetry), or semi-metrics (triangle inequality). As the lack of reflexivity and symmetry can be solved easily in the design of indexing/query algorithms, the real challenging problem is the semi-metric case; the lack of triangle inequality.

The *TriGen* algorithm [5] was developed to transform a semi-metric space (dataset- and distance-specific) into an equivalent (approximate) metric space. The idea behind TriGen is to use an increasing modifying function  $f : \mathbb{R} \rightarrow \mathbb{R}$  that preserves query-induced similarity ordering when applied to a semi-metric distance function  $\delta$ . Having a query object  $q \in \mathbb{U}$  and database objects  $x_i \in \mathbb{S} \subset \mathbb{U}$ , then ordering/ranking of the database objects based on  $\delta(q, x_i)$  is the same as when based on  $f(\delta(q, x_i))$ . Whereas all modifying functions behave the same with regard to the similarity ordering (thus to sequential similarity search), they are dramatically different in terms of the degree of triangle inequality exhibited by  $f(\delta(\cdot, \cdot))$ . Consider three objects  $x_1, x_2, x_3 \in \mathbb{U}$  and the distances  $\delta(x_i, x_j)$  among them – a distance triplet  $\delta(x_1, x_2), \delta(x_2, x_3), \delta(x_1, x_3)$ . In semi-metric spaces, some triplets form triangles and some do not (one distance is larger than the sum of the others). It is easy to show that concave modifiers increase the degree of triangle inequality by turning more distance triplets into triangles. A concave function magnifies short distances more than large distances, so that any distance triplet can be  $f$ -modified to a triangle if  $f$  is concave enough. On the other hand, convex modifiers do the opposite (break triangles).

From practical point of view, concave modifiers increase the degree of triangle inequality, hence eventually turn the semi-metric space into a metric space (indexable by MAMs). Unfortunately, they also increase the intrinsic dimensionality [2] of the space by decreasing the variance of distance distribution (up to equilateral triangles). Convex modifiers decrease the intrinsic dimensionality but also decrease the degree of triangle inequality, increasing thus retrieval error when such a semi-metric space is indexed by a MAM.

The TriGen algorithm finds the optimal level of concavity/convexity of a modifier in order to minimize intrinsic dimensionality of the resulting space, while keeping the expected retrieval error (degree of triangle inequality violation) below user-defined threshold. Hence, TriGen not only provides a solution for efficient (approximate) search in semi-metric spaces, but also fast approximate search in metric spaces. Specifically, TriGen utilizes a set of similarity-preserving T-base modifiers with convexity/concavity parameter  $w$  (see Fig. 1). Using binary search on  $w$ , such T-base  $f$  and  $w$  is found that exhibits the lowest intrinsic dimensionality for a given T-error threshold (where T-error is the proportion of non-triangles in all sampled distance triplets).

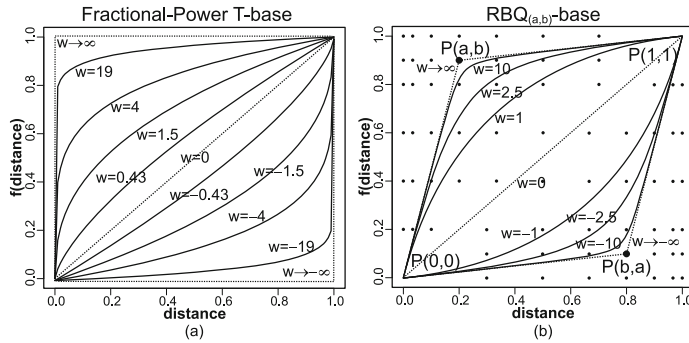


Fig. 1. T-bases of TriGen [5], parameterized by convexity/concavity weights  $w$ .

### 3 Genetic TriGen

In this paper, we present the TriGenGA (developed from an experiment [1]), a variant of TriGen that replaces the binary-search algorithm of finding modifiers by a genetic algorithm. The original TriGen algorithm finds just one T-base parameter  $w$  determining the concavity/convexity weight of the respective modifier function. In the genetic variant, we have implemented a new modifier  $g_v : \langle 0, 1 \rangle \mapsto \langle 0, 1 \rangle$  represented by a piecewise linear function (Eq. 1, Fig. 3). As in TriGen, the modifier is a strictly increasing continuous function with  $g_v(0) = 0$ ,  $g_v(1) = 1$ . However, instead of a predefined pool of T-base functions (i.e., FP-base, RBQ-bases), each parameterized by  $w$ , the genetic modifier  $g_v$  is composed by  $n$  linear segments, given  $n - 1$  parameters ( $n$  is defined by user). The starting/ending points  $(x, y)$  of the  $i$ -th/ $(i - 1)$ -th segment are defined as  $(i/n, v_i)$ , where vector  $v = [v_1, v_2, \dots, v_{n-1}]$  stores the parameters. The genetic algorithm is then used to find the  $n$  parameters of the modifier, given a dataset  $\mathbb{S}$  and a distance function  $\delta$ . Unlike the original TriGen where modifiers are strictly concave or convex (due to the single-parameter  $w$  optimization), the multi-parameter optimization provided by genetic TriGen is able to



generate locally convex/concave modifiers. We anticipate such modifiers could better control the degree of triangle inequality (achieving lower T-error for the same or lower intrinsic dimensionalities), resulting in better precision/efficiency tradeoff exhibited by MAMs when searching.

$$g_v(x) = \begin{cases} \mathbf{v}_1(n-1)x & 0 \leq x \leq \frac{1}{n} \\ \mathbf{v}_1 + (\mathbf{v}_2 - \mathbf{v}_1)((n-1)x - 1) & \frac{1}{n} \leq x \leq \frac{2}{n} \\ \vdots & \vdots \\ \mathbf{v}_i + (\mathbf{v}_{i+1} - \mathbf{v}_i)((n-1)x - i) & \frac{i}{n} \leq x \leq \frac{i+1}{n} \\ \mathbf{v}_{n-1} + (1 - \mathbf{v}_{n-1})(n-1)(x-1) & \frac{n-1}{n} \leq x \leq 1 \end{cases} \quad (1)$$

### 3.1 The Algorithm

The genetic algorithm (GA) consists of an evolution cycle described by Algorithm 1. The population is a set of vectors representing the modifiers (piecewise linear functions). For the selection of potentially successful individuals, we implemented the Tournament selector with variable size of the tournament  $k$ , because it performed better than other selectors. The selector randomly samples  $k$  individuals, and the best individual (with the highest fitness) is chosen in the selection. The one-point crossover of two parents (potentially successful individuals  $\mathbf{v}$  and  $\mathbf{u}$ ) randomly generates a breakpoint  $b_p$ . The new individual (child) is generated as

$$\mathbf{child}_i = \begin{cases} \mathbf{v}_i & \text{for } 1 \leq i < b_p, \\ \mathbf{u}_i & \text{for } b_p \leq i \leq n-1. \end{cases}$$

Mutation with probability  $p_M$  randomly chooses one parameter and moves it down or up randomly (still keeping the modifier  $g_v$  monotonous).

As there were several cases when the fitness of the population degenerated quickly, we have implemented the catastrophic scenario. When the best fitness score has not changed for several generations, part of the generation is replaced by randomly generated. This process should bring a different kind of genomes into the population. If the catastrophic scenario is repeatedly not successful (the best fitness score is the same) or the maximum number of generations is reached, the genetic algorithm terminates. Such modifier  $g_v$  is selected from the final population for which the intrinsic dimensionality of  $(\mathbb{S}, g_v(\delta))$  is minimal.

### 3.2 The Fitness Function

The most important part of TriGenGA is the fitness function, which is the optimization criterion. As it is not possible to optimize  $(\mathbb{U}, \delta)$  space globally, we have taken over the idea of triplet sampling from TriGen where a random subset of the dataset  $\mathbb{S}^* \subset \mathbb{S}$  is used. However, we sample the triplets  $(x, y, z)$  in a different way. First, a distance matrix on  $\mathbb{S}^*$  is computed, i.e., all distances  $\delta(x, y)$   $x, y \in \mathbb{S}^*$ . Then, a fraction of pairs are selected for construction of the

---

**Algorithm 1.** TriGenGA
 

---

```

Result: modifier  $g_v$  with the best fitness score
Population  $\leftarrow$  GenerateRandom( $pop\_size$ )
while UnsuccessfulCatastrophe()  $\leq$  max. # of catastrophes do
    Parents  $\leftarrow$  TournamentSelector $_k$ (Population)
    for  $\forall$ pair of Parents do
        /* combine modifiers */
        Child  $\leftarrow$  Crossover(pair)
        if mutation probability succeeded then
            /* modify modifier */
            Mutate(Child)
        Population  $\leftarrow$  Population  $\cup$  {Child}
    keep only  $pop\_size$  best individuals in Population
    if best fitness score does not change for last  $l$  iterations then
        CatastropheScenario(Population)
return best individual in Population
    
```

---

sample. For every selected pair  $(x, y)$ , the third object  $z \in \mathbb{S}^*$  is found that maximizes  $\frac{\delta(x, z)}{\delta(x, y) + \delta(y, z)}$  (with  $\delta(x, z)$  maximal). This way we obtain as many non-triangle triplets  $(x, y, z)$  in the sample as possible.

The fitness function  $fit(\mathbf{v})$  consists of two parts, it takes into account the T-error (the proportion of non-triangle triplets in all triplets) as well as the indexability (e.g., intrinsic dimensionality). In preliminary experiments, we found that modifiers  $g_v$  with small number of alternating concave and convex segments perform better. Based on that observation, we have proposed the ConFactor indicator (Eq. 2) which is part of the fitness function. The number of concave segments is defined as  $c_v^+ = |\{i | 2\mathbf{v}_i > \mathbf{v}_{i-1} + \mathbf{v}_{i+1}\}|$ , and number of convex segments is defined  $c_v^- = |\{i | 2\mathbf{v}_i < \mathbf{v}_{i-1} + \mathbf{v}_{i+1}\}|$ , for both  $1 < i < n - 1$ . We utilized the idea of TriGen, which assumes more triangle-preserving modifiers imply worse indexability (higher intrinsic dimensionality), so the current implementation of  $fit$  is described by Eq. 3, where  $\epsilon_T$  is the T-error and  $\epsilon_{threshold}$  is the T-error threshold (expected retrieval error specified by user at query time).

$$\text{ConFactor}(\mathbf{v}) = \frac{|c_v^+ - c_v^-|}{c_v^+ + c_v^-} \quad (2)$$

$$fit(\mathbf{v}) = \begin{cases} 1 - \epsilon_T(\mathbf{v}) & \text{for } \epsilon_T(\mathbf{v}) > \epsilon_{threshold}, \\ 1 + \epsilon_T(\mathbf{v}) \cdot \text{ConFactor}(\mathbf{v}) & \text{otherwise} \end{cases} \quad (3)$$

## 4 Experimental Results

We have experimented with  $k$ NN queries, where pivot tables (LAESA [4]) were used with TriGen and TriGenGA, as well as the sequential search. There were

randomly sampled 400 10NN queries from the respective dataset and efficiency and retrieval error was measured. The efficiency was defined as the proportion of distance computations needed with respect to sequential search. The real retrieval error was defined as  $\frac{|E \cap O|}{\max\{|E|, |O|\}}$ , where  $E$  were the expected objects (a result of sequential search) and  $O$  were observed objects (result of LAESA).

For all experiments, we have used the same configuration. The size of the database subset used by TriGen/TriGenGA was  $|\mathbb{S}^*| = 1000$ . Triplet sample size was 25000. 15%, 30%, 45%, 60%, 75%, and 90% triplets were selected by the algorithm maximizing the ratio of erroneous triplets; the other part was sampled randomly. The size of the GA population was 150 individuals. Probability of mutation was set to 5%. The catastrophe scenario was invoked after 10 iterations without the best fitness score improvement. The algorithm terminates after 1000 iterations or ten catastrophe scenarios, without improvement of fitness score. The dimensions  $n = 5$  and  $n = 7$  were used for piecewise linear modifiers.

#### 4.1 Datasets

SISAP NASA dataset [3] (40150 objects with 20 dimensions) was used for vector-based descriptors with metric Minkowski  $L_p$  and semi-metric Fractional  $L_p$  distances ( $L_3, L_2, L_{0.75}, L_{0.5}, L_{0.25}$  and  $L_{0.125}$ ), SISAP English dictionary [3] (69069 English words) for string-based descriptors with metric Levenshtein distance, and an industrial dataset of ATM withdrawal time series (5985 ATM's time-series with 168 dimensions) with semi-metric dynamic time warping (DTW) distance bounded by Sakoe-Chiba band of size 5. We have tested the T-error threshold  $\epsilon_{threshold}$  in five different ranges (0.0, 0.025, 0.05, 0.1 and 0.2).

#### 4.2 Results

Figure 2 summarizes the results. Note that TriGen with FP-modifier has a smooth progress as only one parameter defines the modifier. In contrast, the non-linear and non-deterministic optimization in TriGenGA can generate two modifiers with the same retrieval error but different efficiency. However, the standard deviation of repeated experiments with different random seed was 1% for both the retrieval error and distance computations. TriGenGA performs better than original TriGen for DTW distance on ATM dataset, as well as for metric Minkowski and most of the semi-metric Fractional distance measures (NASA dataset). On the other hand, for the Levenshtein distance and  $L_{0.125}$  the TriGen algorithm dominates TriGenGA.

We evaluated not only the error/efficiency tradeoff but also the T-error threshold vs. real retrieval error dependency. Figure 3 shows this difference – the x coordinate of a circle shows the T-error threshold, while left endpoint of the connected line shows the real retrieval error (y coordinate is the efficiency). Hence, TriGenGA behaves better in terms of real retrieval error estimation.

The distance computation cost is the same for both approaches. The distance matrix on  $\mathbb{S}^*$  is precomputed (see Sect. 3.2). In terms of complexity, both

92 D. Bernhauer and T. Skopal

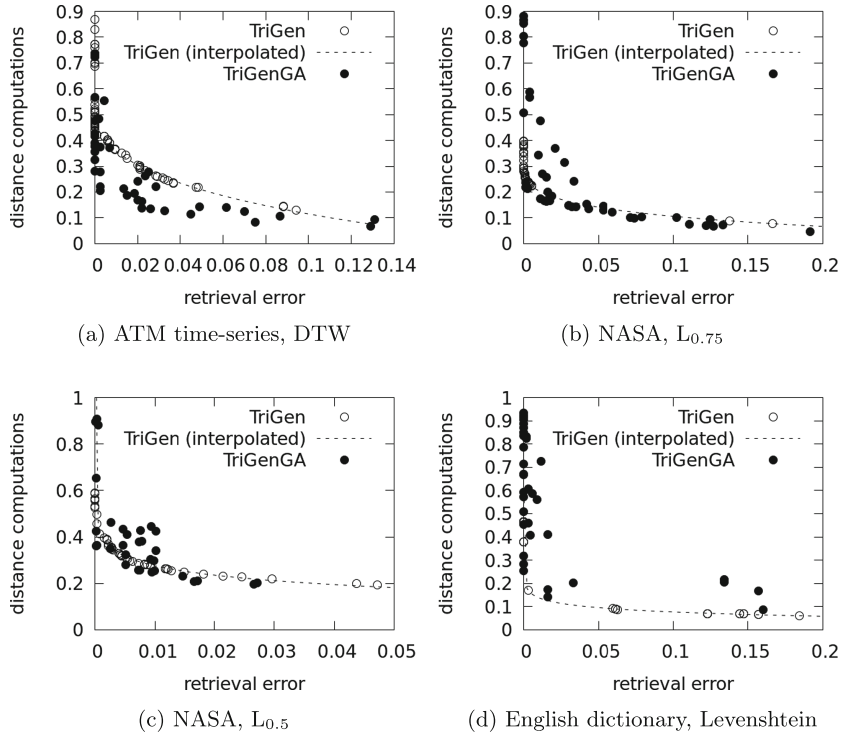


Fig. 2. Comparison of TriGen and TriGenGA on various datasets/distances

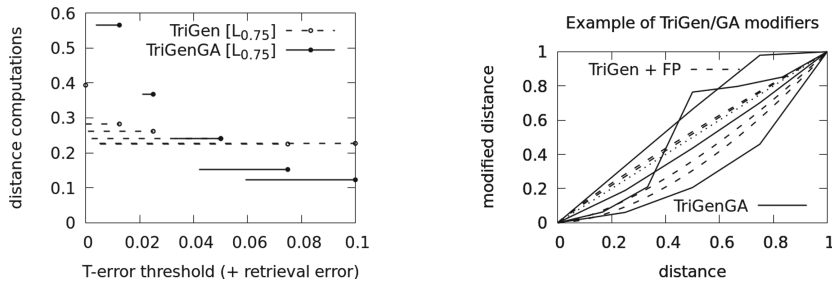
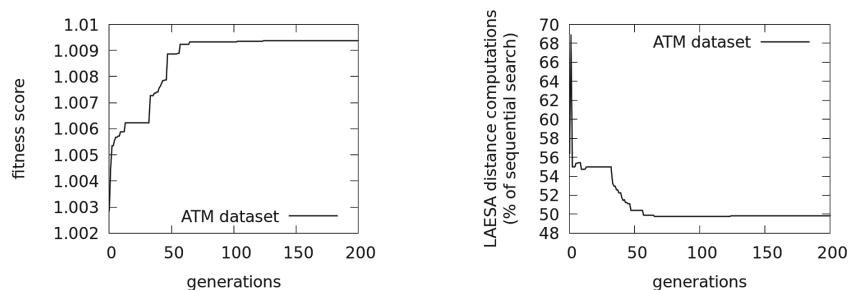


Fig. 3. Left figure shows difference between T-error threshold and retrieval error. Right figure illustrates examples of modifiers generated by TriGen/GA.

approaches can limit the number of iterations. The difference is only in the type of algorithm, the binary search versus the genetic algorithm. Intuitively, the genetic variant has larger requirements for learning parameters because the binary search has only one parameter with only one optimum. However, these



**Fig. 4.** Effects of TriGenGA early termination (limited number of generations).

computations are done only once before the querying. In Fig. 4, see the evolution convergence and the effects of sub-optimal modifier obtained by early termination (i.e., fitness achieved and distance computations of LAESA search).

## 5 Conclusion

We proposed a genetic variant of the TriGen algorithm for approximate similarity search in semi-metric spaces. Experiments proved that the piecewise linear modifier generated by the genetic algorithm can outperform the original TriGen algorithm with FP-modifier, as TriGenGA in some cases provides modifiers exhibiting both lower retrieval error and lower number of distance computations (LAESA-based search). The TriGenGA-generated modifiers also provide better retrieval error estimation given a user-specified T-error threshold.

**Acknowledgments.** This research has been supported in part by the Czech Science Foundation (GAČR) project Nr. 17-22224S.

## References

1. Bernhauer, D., Skopal, T.: Approximate search in dissimilarity spaces using GA. In: GECCO, pp. 279–280. ACM (2019)
2. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. *ACM Comput. Surv.* **33**(3), 273–321 (2001)
3. Figueroa, K., Navarro, G., Chávez, E.: Metric spaces library (2007). <http://www.sisap.org/Metric.Space.Library.html>
4. Mico, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.* **15**, 9–17 (1994)
5. Skopal, T.: Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Trans. Database Syst.* **32**(4), 29 (2007)
6. Skopal, T., Bustos, B.: On nonmetric similarity search problems in complex domains. *ACM Comput. Surv.* **43**(4), 34:1–34:50 (2011)
7. Zezula, P., Amato, G., Dohnal, V., Batko, M.: *Similarity Search: The Metric Space Approach*. Springer, New York (2005). <https://doi.org/10.1007/0-387-29151-2>



---

## Approximate Search in Dissimilarity Spaces using GA

**Bernhauer, D. (50 %); Skopal, T. (50 %)** Approximate search in dissimilarity spaces using GA. In *GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion*, Prague, Czech Republic: ACM, 2019, doi:10.1145/3319619.3321907. [A.10]

The paper has been cited in:

- Rezazadeh Hamedani, A.; Moattar, M. H.; Forghani, Y. Dissimilarity space reinforced with manifold learning and latent space modeling for improved pattern classification. *Journal of Big Data*, volume 8, no. 135, 2021, doi: 10.1186/s40537-021-00527-6.

## Approximate Search in Dissimilarity Spaces using GA

David Bernhauer

Faculty of Information Technology, Czech Technical  
University in Prague, Czech Republic  
bernhdav@fit.cvut.cz

Tomáš Skopal

Faculty of Mathematics and Physics, Charles University  
Prague, Czech Republic  
skopal@ksi.mff.cuni.cz

### ABSTRACT

Nowadays, the metric space properties limit the methods of indexing for content-based similarity search. The target of this paper is a data-driven transformation of a semimetric model to a metric one while keeping the data indexability high. We have proposed a genetic algorithm for evolutionary design of semimetric-to-metric modifiers. The precision of our algorithm is near the specified error threshold and indexability is still good. The paper contribution is a proof of concept showing that genetic algorithms can effectively design semimetric modifiers applicable in similarity search engines.

### CCS CONCEPTS

• Information systems → Search engine indexing;

### KEYWORDS

Genetic algorithm, Similarity search, Content-based retrieval

### 1 INTRODUCTION

Modern searching engines provide not only standard keyword search but also content-based retrieval. The content-based search aims at finding similar objects, e.g., in multimedia databases and generally in datasets of unstructured data. Similarity function (distance, respectively) measures the similarity (dissimilarity) of two database objects. To achieve applicability in different domains, the search engines have to be able to employ various similarity models.

The efficiency is the most crucial part of every search engine today. The naïve algorithm provides precise results, but the sequential scan is time-consuming. As the size of databases is growing, a suitable indexing is necessary. Some generic algorithms, such as LAESA [4], have additional requirements on the distance function  $\delta$  used. Most of them require distance functions satisfying the metric properties. Metric properties, especially the triangle inequality, are an essential part of the indexing process. The basic idea is a construction of computationally cheap lower bound to the original expensive  $\delta$  using the triangle inequality and some pre-selected objects  $P_i$  called pivots. As we know the distances  $\delta(Q, P_i)$  (computed) and  $\delta(X, P_i)$  (fetched from index), where  $Q$  is a query object and  $X$  is a database object, we can compute lower-bound distance as  $\delta_{LB}(Q, X) = |\delta(Q, P_i) - \delta(X, P_i)|$ . If  $\delta_{LB}(Q, X) \geq r_Q$ , then we can

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6748-6/19/07...\$15.00

<https://doi.org/10.1145/3319619.3321907>

say that  $X$  is not within query range  $r_Q$ . However, many distance functions, *semimetrics*, do not satisfy the triangle inequality.

In this paper, we present an alternative approach for approximate search in dissimilarity spaces using a genetic algorithm. We took the idea of TriGen [5] algorithm and extended it to a new way of generating metric distance from semimetric. We also propose new methods of triplet sampling necessary for our algorithm.

### 2 RELATED WORK

One group of non-metric approaches use a different kind of lower-bounding instead of triangle inequality. In [2] authors define relaxed triangle inequality as  $\rho(\delta(O_i, O_j) + \delta(O_j, O_k)) \geq \delta(O_i, O_k)$  which can be used as normal triangle inequality. In [3] usage of Ptolemy's inequality for lower bounds is presented. In [1] the authors present an evolutionary algorithm to generate artificial inequalities for a particular database and a distance function.

The TriGen algorithm [5] idea is to modify a semimetric distance to satisfy the triangle inequality. The assumption is that the original distance can be normed to  $[0, 1]$ . From the database, there is chosen a sample of objects to form triplets. The *triplets* are then distances between triplet objects that form a triangle in metric space. TriGen is implemented as a binary search to find the best triangle-generating (TG) modifier with parameter  $w$  (concavity). The best TG modifier should satisfy the triangle inequality for at least  $\omega$  triplets and should have the as low intrinsic dimensionality ( $iDim = \mu^2/2\sigma^2$ ) as possible (the indexability indicator).

### 3 EXPERIMENT

We have proposed the new kind of semimetric-to-metric modifier and designed a genetic algorithm to learn its parameters. Our Point Modifier (PMod) is represented as a sorted vector  $\vec{P}$  of real values from  $[0, 1]$ . The  $\vec{P}$  dimension  $D$  is variable, but for purpose of our experiment we used  $D = 10$ . The  $\vec{P}$  values represent PMod as piecewise linear function. The  $\vec{P}$  values must be strictly increasing as we want to preserve similarity orderings [5].

As the exact computation of real error and efficiency is time-consuming, we take over TriGen validation of modifier using the triplet error  $\epsilon_t$  (ratio of triplets not satisfying the triangle inequality) and  $iDim$  as the indexability indicator.

#### 3.1 Genetic Algorithm

Our genetic algorithm consists of the basic cycle that selects  $2pop$  candidates for recombination and creates a new population of size  $pop = 100$ . We have implemented Tournament selector with adaptive  $K$  parameter that is increasing in time, represented as the percentage of a population from  $1/3$  to  $2/3$  at the end of the algorithm. The algorithm ends after 1000 iterations or after  $Cat_m = 15$



consecutive catastrophic operator (see below) invocation. We have implemented the following operators.

**Crossover**, that randomly selects a point in Point Modifier representation and split both modifiers into two pieces. The first piece from the first modifier and the second piece from the second modifier are merged again in sorted sequence. **Mutation**, with probability  $p_m = 0.05$  randomly choose a point in Point Modifier and move them up or down without breaking the order of parameters. **Catastrophic operator**, with adaptive tournament selector solves the problem with early population degeneration. The catastrophic operator is invoked after  $Cat_n = 10$  generations without fitness improvement. In that case, the best 10% of the population is kept, and the rest of the population is newly randomly generated.

The most important part of our genetic algorithm is the fitness function (Eq. 1). During the pre-experiments, we have found that functions with few inflection points show better precision compared to function with a large number of inflection points. Based on that observation we have proposed the  $ConFactor(\vec{P})$  as the weighted ratio  $\frac{\max\{2|C^+|, |C^-|\}}{2|C^+| + |C^-|}$  that prefers concave function and less inflection points. The  $C^+$  ( $C^-$ , resp.) is the number of points in  $\vec{P}$  where PMod is concave (convex).

$$f(\vec{P}) = \begin{cases} 1 - \epsilon_T(\vec{P}), & \text{for } \epsilon_T(\vec{P}) > \epsilon_t, \\ 1 + \frac{ConFactor(\vec{P})}{\sqrt{iDim(\vec{P})}}, & \text{otherwise.} \end{cases} \quad (1)$$

### 3.2 Database & Distance

We have tested our algorithm on Minkowski fractional  $L_p$  distances (semimetrics) with  $p$  coefficients  $p_1 = 0.25$ ,  $p_2 = 0.125$  and  $p_3 = 0.0625$ . As the database, we have used NASA vector database<sup>1</sup> with 40150 entries described as 20-dim. vectors. As the search index we have used LAESA with 5 randomly chosen pivots.

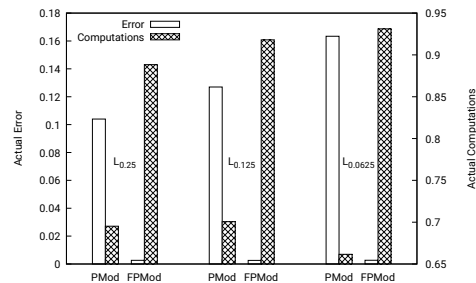
For learning the parameters we sampled 1000 entries of the original database and generated 25000 triplets. The TriGen algorithm [5] prefers anomalous (corner case) triplets in the sample, while such triplets have high variance in order to almost break the triangle inequality (i.e.,  $\min\{a+b/c\}$ , where  $c$  is the biggest side of a triangle). In TriGen only top  $p_a\%$  triplets with the biggest  $c$  value are considered. However, in experiments it was observed that our approach break triangle inequality in lower  $c$  values. So, we have proposed uniform anomalous triplet sampling, i.e., we sample  $p_a^u\% = 0.05$  triplets of uniformly distributed  $c$  values (so we have smaller triangles). The remaining triplets are generated randomly.

### 3.3 Validation

We have compared our proposed genetic algorithm method with iDim-based TriGen algorithm, where as TG-modifiers the Fractional Point (FPMod) modifier and several Rational Bézier Quadratic modifiers (RBQ) modifiers were used. Since the RBQ modifier performed similarly to FPMod, we considered only the FP modifier. Both algorithms used triplet error  $\epsilon_t$  and iDim for learning its parameters.

The effectivity of methods was evaluated by the query error function  $\epsilon_{QE}(E, O) = |E \cap O| / \max\{|E|, |O|\}$  where  $E$  are expected objects (a result of sequential search), and  $O$  are observed objects (LAESA result). The efficiency was measured as the number of distance computations (ratio). Expected error threshold was  $\theta = 0.1$ .

<sup>1</sup><http://www.sisap.org/library/metricSpaces/dbs/vectors/nasa.tar>



**Figure 1: Comparison of average precision (left bar) and efficiency (right bar) of PMod (GA) and FPMod (TriGen).**

Validation queries were range queries generated randomly for different ranges  $\mu = \{0.1, 0.2, \dots, 0.9\}$ . For each query range 0.25%  $\approx$  100 query objects were sampled from the original database.

## 4 RESULTS

We have ran our GA algorithm five times and took the average error and efficiency. During the tests, the FPMod (TriGen) was far below the threshold, but the ratio of distance computation was high. In Figure 1 see that our proposed PMod (GA) is nearly at the error threshold  $\theta$  while the computation ratio is lower than using FPMod. We observe that different PMods have different query range optima that indicate we can construct PMods per group of queries.

## 5 CONCLUSIONS

We have shown that there is a possibility to generate triangle generating functions by genetic algorithms. PMod can be effectively used in the approximated similarity search. GA trained PMod needs just 75% of distance computation against TriGen. As the fitness function does not directly correspond with final results, the future work should aim to tune the fitness function. Also, the experiments with other non-metric distances are needed to confirm our hypothesis, in particular similarities on Linked Data datasets and their contexts.

**Acknowledgments.** This research has been supported by Czech Science Foundation (GAČR) project Nr. 19-01641S.

## REFERENCES

- [1] Tomáš Bartoš, Tomáš Skopal, and Juraj Moško. 2013. Efficient Indexing of Similarity Models with Inequality Symbolic Regression. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, New York, NY, USA, 901–908.
- [2] Ronald Fagin and Larry Stockmeyer. 1998. Relaxing the Triangle Inequality in Pattern Matching. *International Journal of Computer Vision* 30, 3 (01 Dec 1998), 219–231.
- [3] Magnus Lie Hetland, Tomáš Skopal, Jakub Lokoč, and Christian Bieceks. 2013. Ptolemaic access methods: Challenging the reign of the metric space model. *Information Systems* 38, 7 (2013), 989 – 1006.
- [4] Francisco Moreno, Luisa Mico, and Jose Oncina. 2002. Extending LAESA Fast Nearest Neighbour Algorithm to Find the k Nearest Neighbours. 718–724.
- [5] Tomáš Skopal. 2007. Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces. *ACM Trans. Database Syst.* 32, 4, Article 29 (Nov. 2007).