FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Extraction of linguistic information from Wikipedia
Student:	Andriy Nazim
Supervisor:	Ing. Milan Dojčinovski
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2019/20

Instructions

DBpedia is a crowd-sourced community effort which aims at extraction of information from Wikipedia and publishing this information in a machine-readable format. Currently, DBpedia is primarily derived for semi-structured sources such as Wikipedia infoboxes. However, vast amount of information is still hidden in the Wikipedia article texts. The ultimate goal of the thesis is to increase the knowledge in DBpedia with lexical information extracted from Wikipedia.
Guidelines:
- Get familiar with the DBpedia NIF dataset, which provides Wikipedia article texts.
- Analyze existing approaches for extraction of lexical information.
- Design and implement a method for extraction of lexical information (e.g. synonyms, homonyms, etc.) from Wikipedia.
- Apply the method on several Wikipedia languages and provide language-specific lexical datasets using Ontolex model.
- Evaluate the quality of the created lexical datasets.
- Implement a simple user interface for browsing/querying the dataset.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 24, 2019

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Master's thesis

Extraction of linguistic information from Wikipedia

Bc. Andriy Nazim

Supervisor: Ing. Milan Dojchinovski, Ph.D.

6th May 2019

Acknowledgements

I would like to thank my supervisor Ing. Milan Dojchinovski, Ph.D for valuable consultations during writing this thesis.

Also I'd like to thank Visegrad Fund for financial and grant support during my studying in Master's study programme

supported by

Visegrad Fund

Declaration

In Prague on 6th May 2019 . . . .

Citation of this thesis

Nazim, Andriy. *Extraction of linguistic information from Wikipedia.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

DBpedia je komunitní úsilí, jehož cílem je získávání informací z Wikipedie a poskytování těchto informací ve strojově čitelném formátu. V současné době jsou informace obsažené v DBpedii primárně odvozeny pro polostrukturované zdroje, jako jsou infoboxy Wikipedia. V textech článku Wikipedie je však stále skryto obrovské množství informací.

V této práci prezentuji přístupy k extrahování lingvistických informací z DBpedie, které jsou založeny na kombinování a analýze zdrojů DBpedie - datasetů a výsledky magisterského projektu jsou datové sady jazykových informací: synonyma, homonyma, sémantické vztahy a synonyma mezi jazyky . Můj projekt také věnuje zvláštní pozornost čištění, filtrování vytvořených datových souborů a jeho vyhodnocení bylo provedeno taky vytvořením jednoduché webové aplikace pro dotazování výsledků.

Klíčová slova DBpedia, NLP, lingvistika, synonyma, homonyma

Abstract

DBpedia is a crowd-sourced community effort which aims at the extraction of information from Wikipedia and providing this information in a machine-readable format. Currently, the information contained in DBpedia is primarily derived for semi-structured sources such as Wikipedia infoboxes. However, vast amount of information is still hidden in the Wikipedia article texts.

In this Thesis, I present approaches for extracting linguistic information from DBpedia, which are based on combining and parsing DBpedia sources - datasets and the results of the Master Project are datasets of linguistic information: synonyms, homonyms, semantic relationships, and inter-language synonyms. My project also pays special attention to cleaning, filtering of produced datasets, and its evaluation was carried out also by developing a Simple Web-Application for querying results.

Keywords DBpedia, NLP, lingustics, synonyms, homonyms

Contents

List of Figures

List of Tables

Introduction

Motivation

The amount of data produced every day is rapidly growing up. Just over the years 2016 - 2018 90% of the world data was created [4]. In Computer Science two concepts exist: data and information, data is unstructured information, which has its own disadvantages. Data is senseless and it can't be used by people, but it takes storage place. One of the challenges facing the scientists is how to make this data organized, structured and useful. This will solve many problems - structured data could be used in computing processing to get answers to scientific and social tasks. Extracted information will make data work for people.

One of the modern fields in Computer Science where data can be used is NLP (Natural Language Processing). NLP is a relatively new field which includes areas such as text summarization, named entity disambiguation, Question Answering, text categorization, coreference resolution, sentiment analysis, and plagiarism detection. Wide-coverage structured lexical knowledge is expected to be beneficial for areas other than text processing, e.g., grounded applications such as Geographic Information Systems and situated robots.

One of the fields of NLP is the extraction of linguistic data. Linguistic data include words definitions, synonyms, homonyms, translations, semantically close words. This data can be used by scientists to create richer vocabularies, people who are interested in linguistics or just by users who need some linguistic information. This data can be organized into datasets. Datasets can be represented in different formats (see 1.1.2 RDF).

There already exist projects which focus on extraction and structuring of linguistic data. These projects are the following WordNet (see 1.2.1 WordNet), Dbnary (see 1.2.3 Dbnary) or BabelNet (see 1.2.2 BabelNet). These projects have their own advantages and disadvantages (see Chapter State-of-the-art). The motivation of the Master Project is to provide additional linguistic datasets to DBpedia (see 1.1.4 DBpedia) and both to analyze and to create own extraction methods.

Objectives

The main concern of the thesis is to extend the existing results solutions in NLP linguistic data extraction field by creating of new linguistic datasets. Results should have relatively good quality and should be extracted in optimum amount of time.

Master Thesis objective is to make analysis and solve practical tasks for extraction of linguistic information from Wikipedia. It should extend the results of DBpedia. Wikipedia was chosen as one of the biggest resources of the open data, as the basis of the research already structured datasets of DBpedia such as links, page-structures and inter-language links will be used (see 2.1 Input data). The result of the research should be generated datasets of synonyms, homonyms, semantically close words, and inter-language synonyms (see 1.1.1 Synonyms, homonyms, semantic relationships).

The next problem is to efficiently structure and store results. Data structuring means the organizing data into datasets. The efficiency of data organizing can be compared by the quality of data sets and using efficient formats of storing data like RDF (see 1.1.2 RDF). Quality of data set depends on the clearness of data, how sufficient data is for the given dataset.

A solution of these issues could be found in proper algorithms, parallel computing and efficient analyzing and filtering of output results (see Experimental Evaluation Chapter).

One of the objectives is to provide for the convenience of users website with GUI (see 2.9 Simple Web-Application) which enables querying and browsing of the results of Master Thesis.

- Get familiar with the DBpedia NIF dataset, which provides the underlying Wikipedia article content.
- Analyze existing approaches for extraction of lexical information from texts.
- Design and implement a method for extraction of lexical information (e.g. synonyms, homonyms, etc.) from Wikipedia article texts.
- Apply the method on several Wikipedia languages and provide language-specific lexical datasets
- Implement a simple user interface for browsing/querying the dataset.
- Evaluate the quality of the developed lexical datasets.

Chapter 1

State-of-the-art

1.1 Background

It's necessary to give definitions of common concepts used in Master Thesis.

1.1.1 Synonyms, homonyms, semantic relationships

In Objectives, it has been described that Master Thesis is mostly focused on the extraction of such linguistic information as synonyms, homonyms, semantically close words, and inter-language synonyms.

Synonyms. A word or phrase that means exactly or nearly the same as another word or phrase in the same language, for example, shut is a synonym of close [5].

Homonyms. Each of two or more words having the same spelling or pronunciation but different meanings and origins, for example, rock - a genre of music / a stone [6].

Semantic is used to describe things that deal with the meanings of words and sentences. Semantically close words are words which are often used together and relate to the same field [7].

Inter-language synonyms are synonyms of word or phrase in different languages.

Also one of the objectives is to efficiently present the results and data. Here one of the common approaches is Linked Data.

1.1.2 RDF

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model [8]. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. It is also used in knowledge management applications. RDF files could differ in formats like TTL (turtle) or HDT - they differ in compression levels. HDT uses more efficient compression mechanisms. In the project both formats - HDT and TTL were used.

1.1.3 Linked Data

In the Computer Science field, linked data is a method of publishing structured data so that it can be interlinked and become more useful through semantic queries. Linked Open Data-Cloud currently contains 1,239 datasets with 16,147 links (as of March 2019) [1]. One of the biggest datasets is DBpedia which contains about 4.6 million concepts described by more than 1 billion triples, including abstracts in 11 different languages. DBpedia has been chosen as a base of the Master Project. In the following picture, it's possible to see the place of DBpedia in Linked Open Data Cloud.

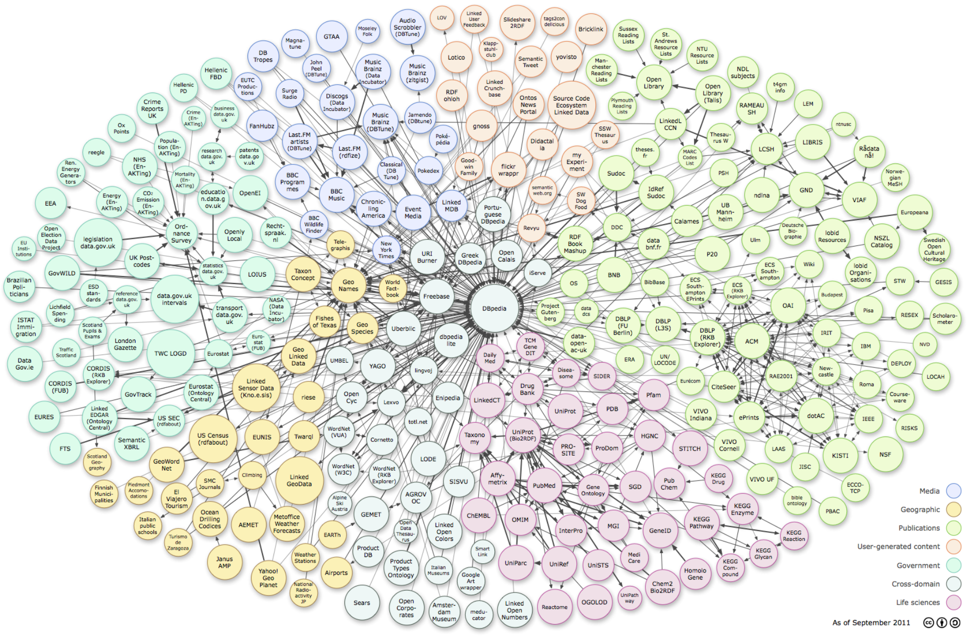


Figure 11: Linked-Open Data Cloud [1]

1.1.4 DBpedia

DBpedia was noted by Tim Berners-Lee (inventor of the World Wide Web) as one of the most famous examples of the implementation of the concept of related data [9].

The project was initiated by a group of volunteers from the Free University of Berlin and the University of Leipzig, in collaboration with OpenLink Software, the first dataset being published in 2007. Since 2012, the University of Mannheim has been an active participant in the project.

As of the date of September 2014, DBpedia describes more than 4.58 million entities, of which 4.22 million are classified according to ontology, including 1.445 million personalities, 735 thousand geographical objects, 123 thousand music albums, 87 thousand films, 19 thousand video games, 241 thousand organizations, 251 thousand taxa, and 6 thousand diseases. DBpedia contains 38 million tags and annotations in 125 languages; 25.2 million links to images and 29.8 million links to external web pages; 50 million external links to other RDF-format (see 1.1.5 RDF) databases, 80.9 million Wikipedia categories.

The project uses the Resource Description Framework (RDF) to present the extracted information. As of the date of September 2014, the bases consist of more than 3 billion RDF triples, of which 580 million were taken from the English section of Wikipedia and 2.46 billion extracted from sections in other languages.

One of the problems in extracting information from Wikipedia is that the same concepts can be expressed in templates in different ways, for example, the concept of "place of birth" can be formulated in English as "birthplace" and as "place of birth". Because of this ambiguity, the query passes through both variants to obtain a more reliable result. To facilitate the search while reducing the number of synonyms, a special language was developed - DBpedia Mapping Language and DBpedia users have the opportunity to improve the quality of data extraction using the Mapping service.

The goal of DBpedia community is to extract structured information from the data created in various Wikimedia projects. This structured information resembles an open knowledge graph (OKG) which is available for everyone on the Web. A knowledge graph is a special kind of database which stores knowledge in a machine-readable form and provides a means for information to be collected, organized, shared, searched and utilized. Google uses a similar approach to create those knowledge cards during search [10]. Further given above knowledge graph will be mentioned as a dataset. DBpedia NIF datasets are stored in NIF content format [11].

1.1.5 NIF

The NLP Interchange Format (NIF) is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources, and annotations. NIF consists of specifications, ontologies, and software. NLP basically inter-operates RDF files of different formats [12].

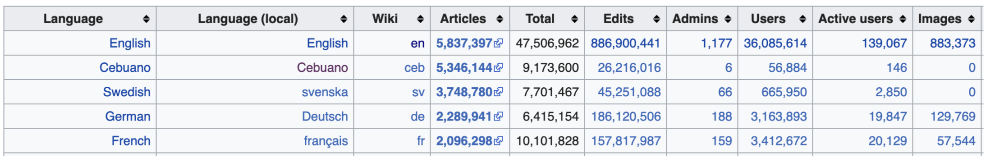
The core of NIF consists of the vocabulary, which can represent Strings as RDF resources. A special URI Design is used to pinpoint annotations to a part of a document. These URIs can then be used to attach arbitrary annotations to the respective character sequence. Based on these URIs, annotations can be interchanged between different NLP tools [13].

An example of NIF:

```
<http://dbpedia.org/resource/Anderida> <http://dbpedia.
    org/ontology/wikiPageRedirects> <http://dbpedia.org/
    resource/Anderitum> .
<http://dbpedia.org/resource/Adrian_I> <http://dbpedia.
    org/ontology/wikiPageRedirects> <http://dbpedia.org/
    resource/Pope_Adrian_I> .
```

1.1.6 DBpedia datasets

DBpedia gives the list of datasets in different formats for different proposes for users. Also, datasets are presented in dozens of languages including the most rare ones like Saha and of course the most popular one - English. In this research paper just two languages will be used : English and German, but the methods described will be working for all languages. English and German have been chosen as the most commonly used ones and having the biggest number of articles in Wikipedia.



Language	Language (local)	Wiki	Articles	Total	Edits	Admins	Users	Active users	Images
English	English	en	**5,837,397**	47,506,962	886,900,441	1,177	36,085,614	139,067	883,373
Cebuano	Cebuano	ceb	**5,346,144**	9,173,600	26,216,016	6	56,884	146	0
Swedish	svenska	sv	**3,748,780**	7,701,467	45,251,088	66	665,950	2,850	0
German	Deutsch	de	**2,289,941**	6,415,154	186,120,506	188	3,163,893	19,847	129,769
French	français	fr	**2,096,298**	10,101,828	157,817,987	159	3,412,672	20,129	57,544

Figure 12: Wikipedia Languages

As shown in the picture above the second and the third most commonly used languages in Wikipedia are Swedish and Cebuano, but these Wikipedias almost 100 percent were created by an automatic bot - Lsjbot. There exist a lot of criticism of usage of this Bot, because articles become poor and such project as Extraction of linguistic information from Wikipedia requires not just number of articles, but the volume of content is also important. The volume of content is much more important than a number of articles because in Master Thesis methods based on the extraction information from the context are used.

It's possible to find lots of datasets in turtle notations like ttl (Terse RDF Triple Language) on the DBpedia download web page. Turtle (ttl): provides data in n-triple format (subject, predicate, object) as a subset of turtle serialization (Turtle) and tql: quad-turtle (tql): the quad turtle serialization (subject, predicate, object, graph/context.) adds context information to every triple, containing the graph name and provenance information.

Size of datasets is huge. Just dataset of text links in archived format .bz2 is about 6GB, but after extraction, it could be even more than 80GB. Such big files require lots of memory and computational power. So, there has been made a decision to search for less resource-consuming datasets.

One of the solutions was HDT documents RDF HDT stands for Header-Dictionary-Triples. This is a format based on binary encoding and it is used in storing large RDF files, their publishing and exchange. The idea of RDF HDT is to store RDF graph in a compact manner, by splitting RDF graphs into several chunks. Design of RDF HDT also allows archiving high compression rates. There exists an approach on how to do it, by decomposing RDF graph into two main components as Triple structures and Dictionary. The Dictionary in this case works as an index for high-speed searching and allows high compression ratios. The second triples component allow storing pure graph data in a compressed way. Also, there exists an additional Header, which is recommended to store metadata about RDF graph and it's organization.

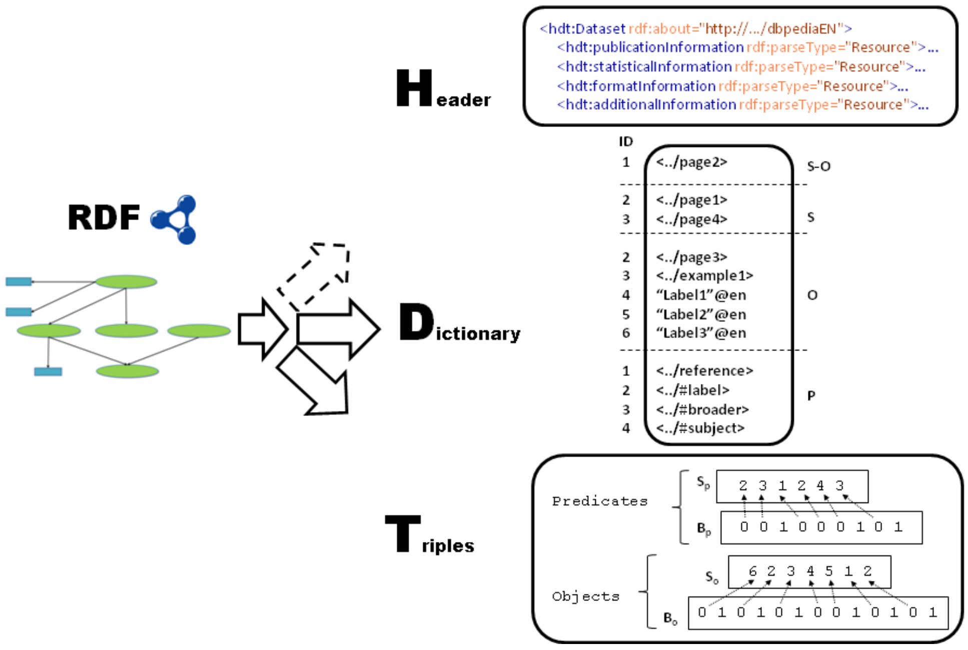
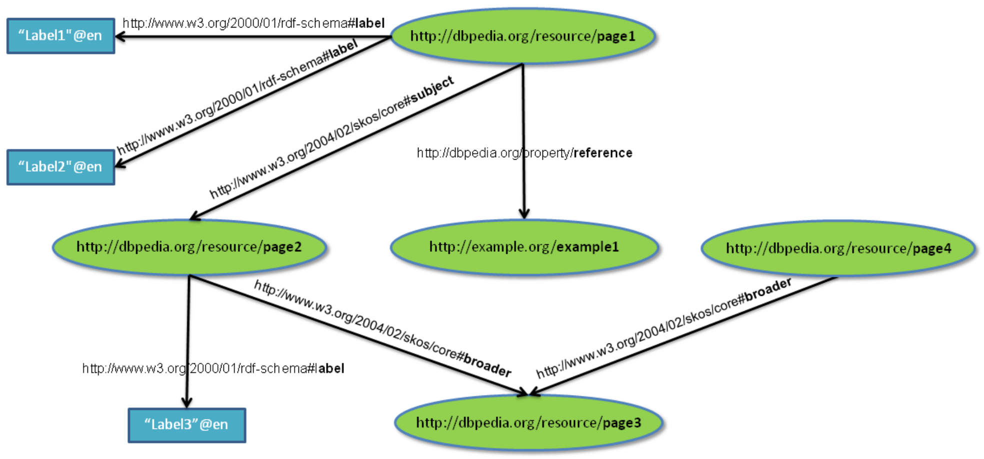

Header
<hdt:Dataset rdf:about="http://.../dbpediaEN">
<hdt:publicationInformation rdf:parseType="Resource">...
<hdt:statisticalInformation rdf:parseType="Resource">...
<hdt:formatInformation rdf:parseType="Resource">...
<hdt:additionalInformation rdf:parseType="Resource">...
RDF
Dictionary
ID
1 <../page2> S-O
2 <../page1>
3 <../page4> S
2 <../page3>
3 <../example1>
4 "Label1"@en
5 "Label2"@en
6 "Label3"@en O
1 <../reference>
2 <../#label>
3 <../#broader>
4 <../#subject> P
Triples
Predicates
Sp 2 3 1 2 4 3
Bp 0 0 1 0 0 0 1 0 1
Objects
So 6 2 3 4 5 1 2
Bo 0 1 0 1 0 1 0 0 1 0 1 0 1

Figure 13: RDF to HDT Comparison [2]



"Label1"@en
http://www.w3.org/2000/01/rdf-schema#label
http://dbpedia.org/resource/page1
http://www.w3.org/2000/01/rdf-schema#label
"Label2"@en
http://www.w3.org/2004/02/skos/core#subject
http://dbpedia.org/property/reference
http://dbpedia.org/resource/page2
http://example.org/example1
http://dbpedia.org/resource/page4
http://www.w3.org/2004/02/skos/core#broader
http://www.w3.org/2004/02/skos/core#broader
http://www.w3.org/2000/01/rdf-schema#label
"Label3"@en
http://dbpedia.org/resource/page3

Figure 14: HDT Example [2]

Comparing to usual RDF N-triples notations HDT has higher compression levels and higher speed of queries with less delayed times. HDT is also compatible with SPARQL queries as all RDF based formats. In the pictures above the example and structure of RDF HDT files with illustrating linking in DBpedia are shown.

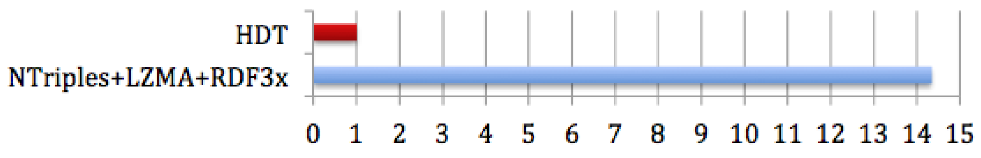


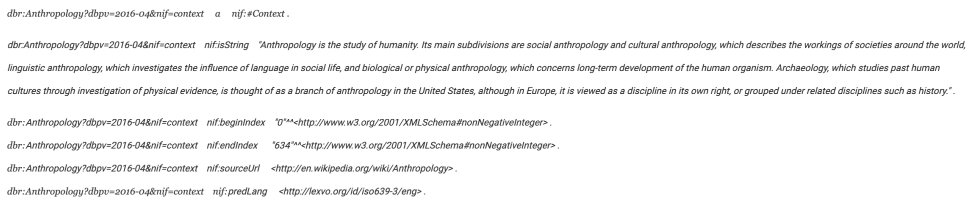
Figure 15: Comparison of HDT against with traditional techniques regarding the time to download and start querying a dataset [3]

.

Comparing to TTL files the same dataset in HDT will take more than 3 times less storage space. For example, downloaded unarchived TTL file for text links was taking almost 90 GB, but in HDT format it's just 26 GB. HDT also was provided lots of API. As the main programming language for the whole project Python has been chosen, which allows quick data structures operations. Also, there exist lots of tools available for Python as for programming language. In an open access HDT files are available just in English, TTL files were used for inter-language links.

Overview of the most frequently used datasets in DBpedia.

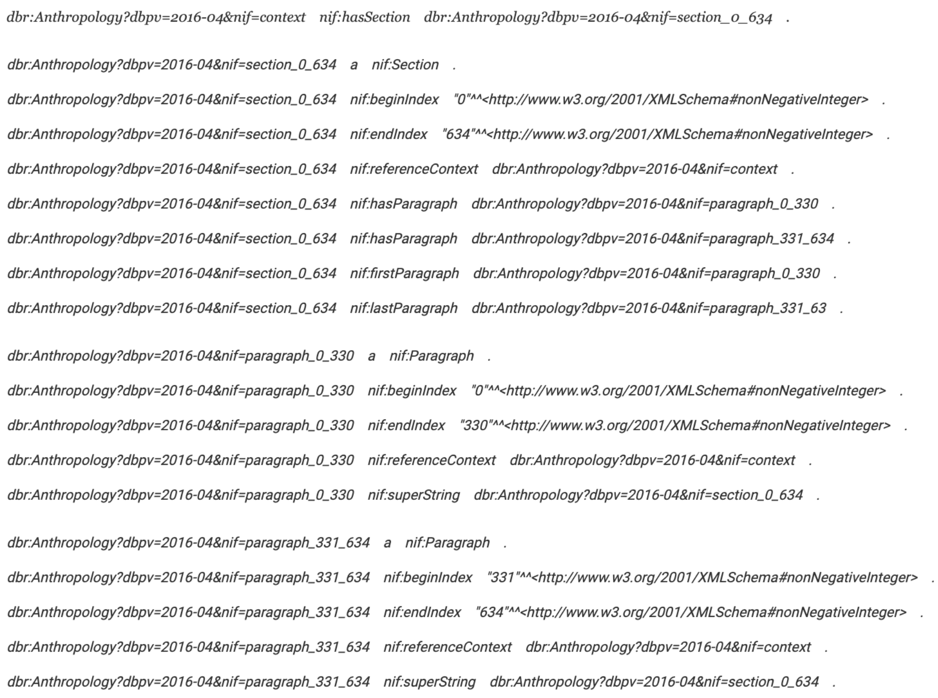
The context dataset is used to describe full texts of articles. It doesn't cover links, just text of part of article (entity). Each entity is described with 6 triples: type of triple (context), entity text, link to it, language and begin and end index of the given part of the article.



```
dbr:Anthropology?dbpv=2016-04&nif=context   a   nif:#Context .

dbr:Anthropology?dbpv=2016-04&nif=context   nif:isString   "Anthropology is the study of humanity. Its main subdivisions are social anthropology and cultural anthropology, which describes the workings of societies around the world,
linguistic anthropology, which investigates the influence of language in social life, and biological or physical anthropology, which concerns long-term development of the human organism. Archaeology, which studies past human
cultures through investigation of physical evidence, is thought of as a branch of anthropology in the United States, although in Europe, it is viewed as a discipline in its own right, or grouped under related disciplines such as history." .

dbr:Anthropology?dbpv=2016-04&nif=context   nif:beginIndex   "0"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=context   nif:endIndex   "634"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=context   nif:sourceUrl   <http://en.wikipedia.org/wiki/Anthropology> .
dbr:Anthropology?dbpv=2016-04&nif=context   nif:predLang   <http://lexvo.org/id/iso639-3/eng> .
```

Figure 16: NIF Context

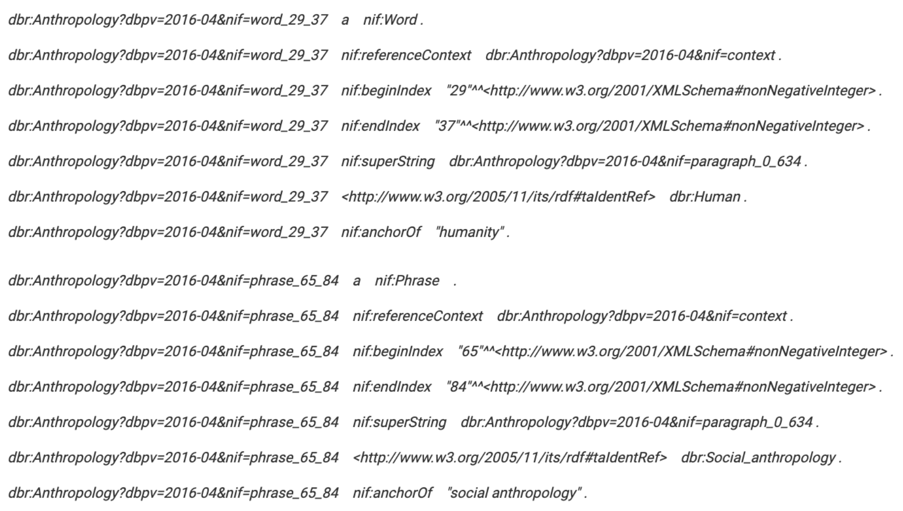
Page structure dataset describes such entities as Section, Paragraph, and Title. It also has properties like begin index and end index, Section triples also describe each paragraph that it contains separately by begin and end index.



```
dbr:Anthropology?dbpv=2016-04&nif=context   nif:hasSection   dbr:Anthropology?dbpv=2016-04&nif=section_0_634   .

dbr:Anthropology?dbpv=2016-04&nif=section_0_634   a   nif:Section   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:beginIndex   "0"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:endIndex   "634"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:referenceContext   dbr:Anthropology?dbpv=2016-04&nif=context   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:hasParagraph   dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:hasParagraph   dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:firstParagraph   dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   .
dbr:Anthropology?dbpv=2016-04&nif=section_0_634   nif:lastParagraph   dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_63   .

dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   a   nif:Paragraph   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   nif:beginIndex   "0"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   nif:endIndex   "330"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   nif:referenceContext   dbr:Anthropology?dbpv=2016-04&nif=context   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_330   nif:superString   dbr:Anthropology?dbpv=2016-04&nif=section_0_634   .

dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   a   nif:Paragraph   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   nif:beginIndex   "331"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   nif:endIndex   "634"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   nif:referenceContext   dbr:Anthropology?dbpv=2016-04&nif=context   .
dbr:Anthropology?dbpv=2016-04&nif=paragraph_331_634   nif:superString   dbr:Anthropology?dbpv=2016-04&nif=section_0_634   .
```

Figure 17: NIF Page structure

Text Links Dataset describes links Words and Phrases by their references, anchors, representing in which article and paragraph this link is situated. This dataset is the most usable one in the given project.



```
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   a   nif:Word .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   nif:referenceContext   dbr:Anthropology?dbpv=2016-04&nif=context .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   nif:beginIndex   "29"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   nif:endIndex   "37"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   nif:superString   dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_634 .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   <http://www.w3.org/2005/11/its/rdf#taIdentRef>   dbr:Human .
dbr:Anthropology?dbpv=2016-04&nif=word_29_37   nif:anchorOf   "humanity" .

dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   a   nif:Phrase   .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   nif:referenceContext   dbr:Anthropology?dbpv=2016-04&nif=context .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   nif:beginIndex   "65"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   nif:endIndex   "84"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger> .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   nif:superString   dbr:Anthropology?dbpv=2016-04&nif=paragraph_0_634 .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   <http://www.w3.org/2005/11/its/rdf#taIdentRef>   dbr:Social_anthropology .
dbr:Anthropology?dbpv=2016-04&nif=phrase_65_84   nif:anchorOf   "social anthropology" .
```

Figure 18: NIF Text Links

1.1.7 OntoLex-Lemon

Ontology-lexicon interface (ontolex). The aim of the lexicon model for ontologies (lemon) is to provide rich linguistic grounding for ontologies. Rich linguistic grounding includes the representation of morphological and syntactic properties of lexical entries as well as the syntax-semantics interface, i.e. the meaning of these lexical entries with respect to an ontology or vocabulary [14]. Ontolex helps to create well-defined RDF structures.

1.2 Related work

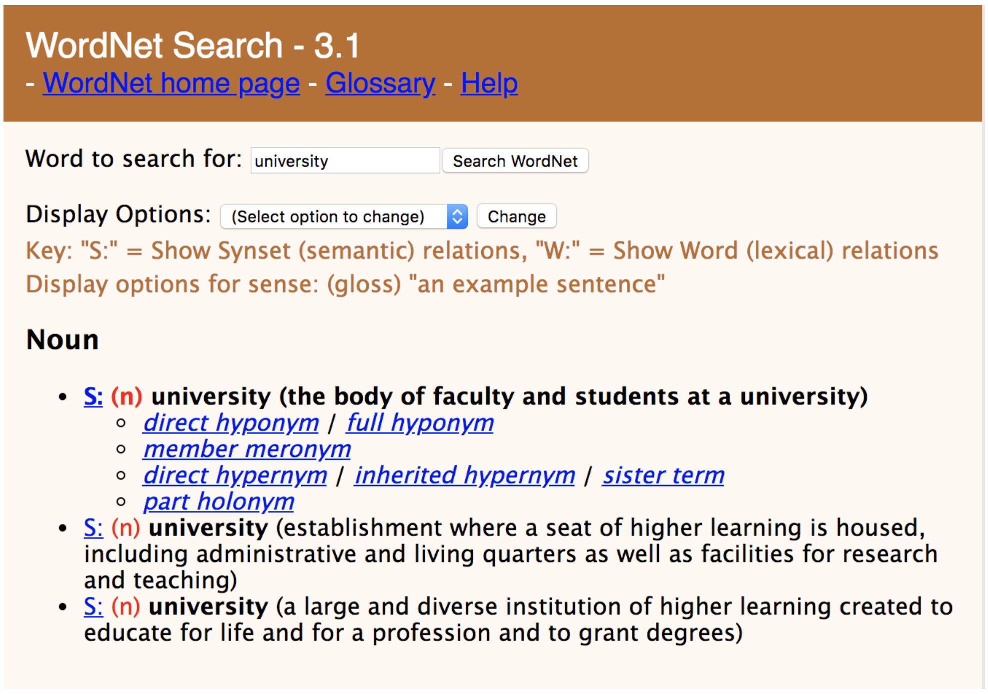
Lexical information is possible to be represented in a large amount of distinct forms, ranging from unstructured terminologies such as a list of terms to glossaries such as Web-derived domain glossaries, machine-readable dictionaries like LDOCE, thesauri like Roget's Thesaurus and full-fledged computational lexicons and ontologies as WordNet and Cyc. However it's almost impossible to build such datasets manually - it could take dozens of years, additionally, it's not scalable, this will require the extra work to do the same in different languages. In addition, there must be completed a job to connect all the entities across languages. Besides, for lots of works there exists a problem for covering rare languages. There is a huge distance in covering and research of the resource-rich languages like English and others. There are many resources already done such as BalkaNet [15], MultiWordNet [16], EuroWordNet [17] which focus on some particular languages.

Recently, the number of online open-source projects is increasing. It includes lots of researching by communities of Artificial Intelligence and Open Linked Data and such universities as Princeton University (WordNet) and Leipzig University and University of Mannheim (DBpedia). Furthermore, a huge amount of enthusiasts and open-source communities help to develop projects like these. A sort of resources contain semi-structured information, mainly in textual, possibly hyperlinked, form. In this case and from the view of multilingual resources Wikipedia is the largest and the most popular multilingual and collaborative work of lexical information. There have already been done a lot of work based on Wikipedia like DBpedia and BabelNet. A lot of work on the extraction and structuring information, such as extracting lexical and semantic relations between concepts, factual information, and transforming the Web encyclopedia into a full-fledged semantic network has already been conducted. One major feature of Wikipedia is its richness of explicit and implicit semantic knowledge, mostly about named entities (e.g., Apple as a company).

In the following sections the most popular works will be more precisely over viewed. The most famous related projects in field are WordNet, BabelNet, Dictionary.com.

1.2.1 WordNet

WordNet is one of the biggest linguistic linked databases which has its own implementations by Princeton University.


WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: university Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) university (the body of faculty and students at a university)
  - direct hyponym / full hyponym
  - member meronym
  - direct hypernym / inherited hypernym / sister term
  - part holonym
- S: (n) university (establishment where a seat of higher learning is housed, including administrative and living quarters as well as facilities for research and teaching)
- S: (n) university (a large and diverse institution of higher learning created to educate for life and for a profession and to grant degrees)

Figure 19: WordNet Query Screenshot

On the image above is shown a result of search query on WordNet search web-page [18].

This database links English nouns, verbs, adjectives, and adverbs to sets of synonyms that are in turn linked through semantic relations that determine word definitions.

WordNet is one of the biggest English databases of lexical information. It consists of lots of speech parts like adverbs, adjectives, and verbs which are grouped into synsets. Synset is a set of cognitive synonyms. Every synset means a definite construct. These sets are mutually linked with support of conceptual-semantic lexical relations. As a result, WordNet is a network of close words by meanings and concepts which can be navigated in web-browser. It's a structured network which helps it to be used as an instrument for natural language processing and in computations for linguistics.

The base of WordNet is a thesaurus which was resembled into words groups. These groups are made upon definitions of these words. One of the features of Wordnet is that it connects words, based not just on letters or words similarity, but also based on senses of the words. As a result, WordNet connects words which have not a lot in common at first sight. If there are no any words meaning linking, WordNet will label these words and follow standard rules of linking words itself.

Structure

The highest dependencies among words in WordNet are synonyms, such words as task and assignment Synonyms are words which represent the same meaning and could be used instead of each other in many cases. In WordNet, they are matched in unordered sets (synsets). There are 117 000 linked synsets, which are connected by definitions of a small number of “conceptual relations.” Extra words contain a short description (“gloss”) and also a few examples illustrating applications of usages. Each word form can have several different meanings and each one is stored in a separate synset. So each pair in WordNet is unique.

Relations

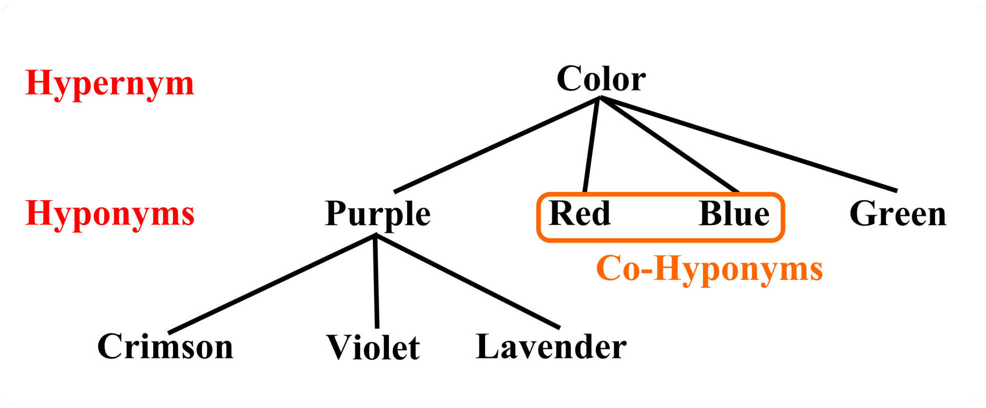


Figure 110: Hyponyms

WordNet also represents super-subordinate relation - hyperonymy, hyponymy. These relations connect more general synsets like Red and Color. In such a way WorldNet describes that the general field color includes Red, which in turn includes Blue. Conversely, meanings like Red and blue create a category Color. There exist roots and hierarchy which goes up like in a node. Hyponyms relation is transitive: if Violet is a kind of purple and purple is a color, then violet is a color too. In WordNet there are also Types and Instances. Types describe common nouns like Violet is kind of purple. Instances are terminal nodes in the hierarchies. In the case of colors, it’s difficult to give a good example of ”terminal color”, because colors could mix infinitely. Instances could be some specific things like countries, persons and geographic entities.

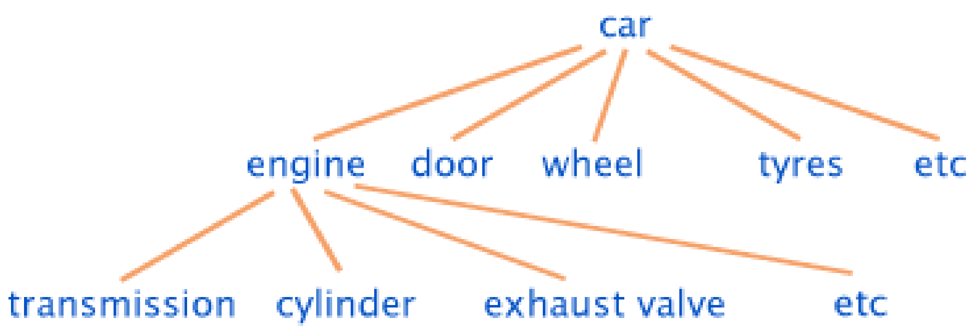


Figure 111: Meronyms

Meronyms, are words describing parts of a synset. For example, the chair has legs. There exist strong inheritance and as was shown in hyperonyms meronyms also follow the hierarchy and a node system. But in comparison to hyperonyms - there isn't "upward" inheritance, because lower characteristics describe just some specific kind of thing, rather than a class as a whole. All kinds of chairs have legs, but not all types of furniture have legs.

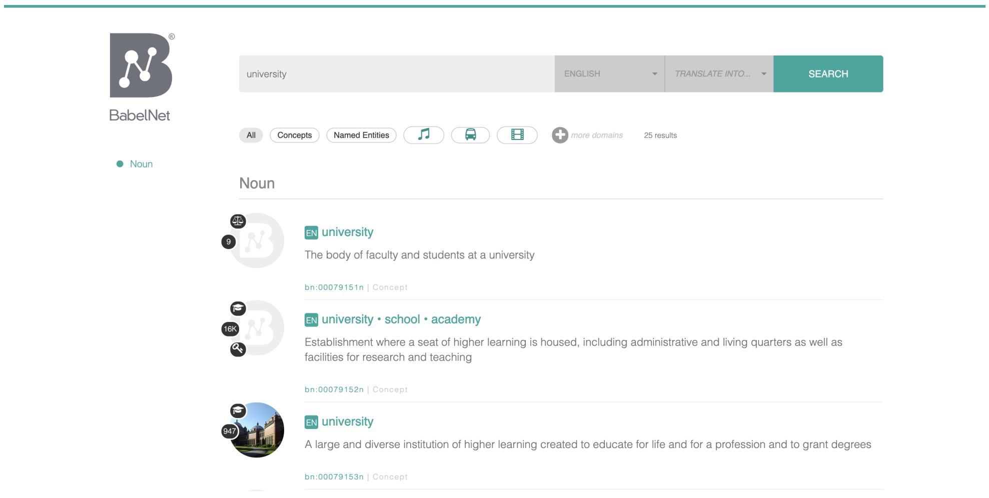
WordNet collects also verbs, which are also organized in trees (troponyms) as nouns. A verb is describing events and in the same way, has links between each other. For instance, communicate - talk - whisper. The way how words are connected depends on their semantic meaning. It could be a manner of doing something as described in the example above, so it could have direction and level of action or it could be just undirectional synset like buy - pay

WordNet also stores such part of speech as antonyms. They are stored as pairs - pairs of words which have opposite meanings. For example dry and wet. This pair has a strong relationship. But to expand results WordNet connected in turn number of "semantically similar" words. For instance: dry is a synonym of arid, bare, barren, dehydrated, dusty, parched, stale. In such a way given words are "indirect antonyms" of synonyms of wet: dank, foggy, humid, misty, muggy, rainy, slippery.

WordNet also consists of adverbs, but this amount is not high. It happens because the majority of English adverbs are straightforwardly derived from adjectives via morphological affixation (surprisingly, strangely, etc.)

Basically, WordNet connects just words of the same part of speech (POS). These connections were build depending on similarity of words in writing. observe (verb), observant (adjective) observation, observatory (nouns) [19].

1.2.2 BabelNet



EN university
The body of faculty and students at a university
bn:00079151n | Concept

EN university • school • academy
Establishment where a seat of higher learning is housed, including administrative and living quarters as well as facilities for research and teaching
bn:00079152n | Concept

EN university
A large and diverse institution of higher learning created to educate for life and for a profession and to grant degrees
bn:00079153n | Concept

Figure 112: BabelNet

BabelNet is another open-source project which is trying to implement different methods and implementations. BabelNet is trying to extend and combine the results of WordNet and DBpedia. It's presenting wide-coverage multilingual knowledge resource. BabelNet presents also enrichment and integration methodology which creates a large multilingual semantic network.

BabelNet is created by linking the largest multilingual Web encyclopedia - Wikipedia - to the most popular computational lexicon - WordNet [20]. The integration is performed via automatic mapping and by filling in lexical gaps in resource-poor languages by means of Machine Translation.

As a result, BabelNet is an extended "encyclopedic dictionary" which contains concepts and named entities connected with a large number of semantic relations and in many languages.

The BabelNet methodology of linking data could be described in three statements.

1. BabelNet has a lightweight methodology to map encyclopedic entries to a computational lexicon. The given methodology has different approaches to estimate mapping probabilities such as graph representations bag-of-words methods. BabelNet provides methods to map tens of thousands of Wikipedia pages and corresponding Wordnet synsets. The quality of results is about 78% F1 measure (see 3.1 Evaluation Metric).

2. BabelNet also provides translations in different languages. At the beginning six languages were chosen. The translation is made by combining two methods. Human-edited translations, inter-language links and state of the art statistical Machine Translations for filling gaps are used. Machine Transla-

tions help to translate millions of sense-tagged sentences from Wikipedia and SemCor [21]. As a result, it's possible to cover the biggest part of the existing WordNet senses and to provide many novel lexicalizations.

The SemCor corpus is an English language dictionary with annotated texts in an annotated way. SemCor has 352 texts from Brown corpus. SemCor also has provided semantic analysis done manually with WordNet 1.6 senses (SemCor version 1.6). Later it was automatically mapped to WordNet 3.0 (SemCor version 3.0).

The Brown corpus (full name Brown University Standard Corpus of Present-Day American English) was the first text corpus of American English. The original corpus was published in 1963-1964 by W. Nelson Francis and Henry Kučera at Department of Linguistics, Brown University Providence, Rhode Island, USA.

The corpus consists of 1 million words (500 samples of 2000+ words each) of running text of edited English prose printed in the United States during the year 1961 and it was revised and amplified in 1979.

3. BabelNet is using knowledge encoding to perform graph-based and knowledge rich Word Sense Disambiguation in both a multilingual and monolingual setting. The given results indicate that associative ones from Wikipedia can complement each other and enable to receive state of the art performance when they are combined with a wide-coverage semantic network [22].

1.2.3 Dbnary

Dbnary is one more project in Linked Open Data. Dbnary uses Wiktionary, which is part of Wikipedia project. Wiktionary is a multilingual, web-based project to create a free content dictionary of terms (including words, phrases, proverbs, etc.) in all natural languages and a number of artificial languages. These entries may contain definitions, pronunciation guides, inflections, usage examples, related terms, images for illustration, among other features.

The goal of Dbnary is not to extensively reflect wiktionary data, but to create a lexical resource that is structured as a set of monolingual dictionaries + bilingual translation information. Such data are already useful for several application, but it is merely a starting point for a future multilingual lexical database.

The monolingual data are always extracted from its own wiktionary lexical edition. For instance, the French lexical data is extracted from French language edition (the data available on http://fr.wiktionary.org). Hence, we completely disregard the French data that may be found in other language editions. Dbnary also filtered out some parts of speech in order to produce a result that is closer to the existing monolingual dictionaries. For instance, in French, were disregarded abstract entries that are prefixes, suffixes or flexions.

Lexical Entries: an instance of lemon:LexicalEntry corresponds more or less to a "part of speech" section in a wiktionary page. This means that it is defined by unique canonical written form, a part of speech and a number (in case of homonymy). When wiktionary data allows for it, Dbnary try to distinguish between lemon:Word and lemon:Phrase that are defined as specific lexical entries.

Lexical Forms: lexical entries are connected, through the lemon:canonicalForm property to a lexical form that gathers a written form and a pronunciation (when available). They may also be connected to alternative spelling through lemon:lexicalVariant property.

Lexical Senses: an instance of lemon:LexicalSense corresponds to one definition in the wiktionary page. It is the target of the lemon:sense property of its containing Lexical Entry. Each lexical sense is associated with a dbpedia:senseNumber property (that contains the rank at which the definition appeared in the wiktionary page) and a lemon:definition property.

Part Of Speech part of speech properties are available in the wiktionary data in 2 distinct properties that are attached to lexical entries:

• dbnary:partOfSpeech is a data property whose value is a string that contains the part of speech as it was defined in wiktionary

• lexinfo:partOfSpeech is a standard property that is bound to isocat data categories and which value is a correct isocat data category.

This property is only available when the mapping between wiktionary part of speech and isocat part of speech is known.

Vocable: the main unit of data in wiktionary is a wiktionary page that may contain several lexical entries. Many lexical data is represented as links to a page. Most of the time, there is not enough data to know to which lexical entry (or lexical sense) these links point to. Hence if we want to keep these underspecified relations, we need to define units that represent wiktionary pages. This is the role of the dbnary:Vocable class. Instances of this class are related to their lexical entries through the dbnary:refersTo property.

Nyms: most wiktionary language editions do provide ”nym” relations (mainly synonym, antonym, hypernym, hyponym, meronym and holonym). This legacy data is not representable using LEMON model, unless Dbnary know for sure the source and target lexical sense of the relation. In order to cope with this legacy data, 6 new ”nym” properties (in dbnary name space). Additionaly, Dbnary defined a class called dbnary:LexicalEntity that is defined as the union of LEMON lexical entries and lexical senses. The ”nym” properties domain and range are lexical entities. Most of these properties do link a lexical entry to a vocable, as there is not enough information in wiktionary to promote this relation to a full class sense to sense relation. Some of these properties are however promoted to a Lexical Sense to Vocable relation when the lexical entry is unambiguous (contains only one sense).

Translations: As there is no way to represent bilingual translation relation in LEMON, Dbnary introduced the dbnary:Equivalent class that collects translation information contained in wiktionary [23]..

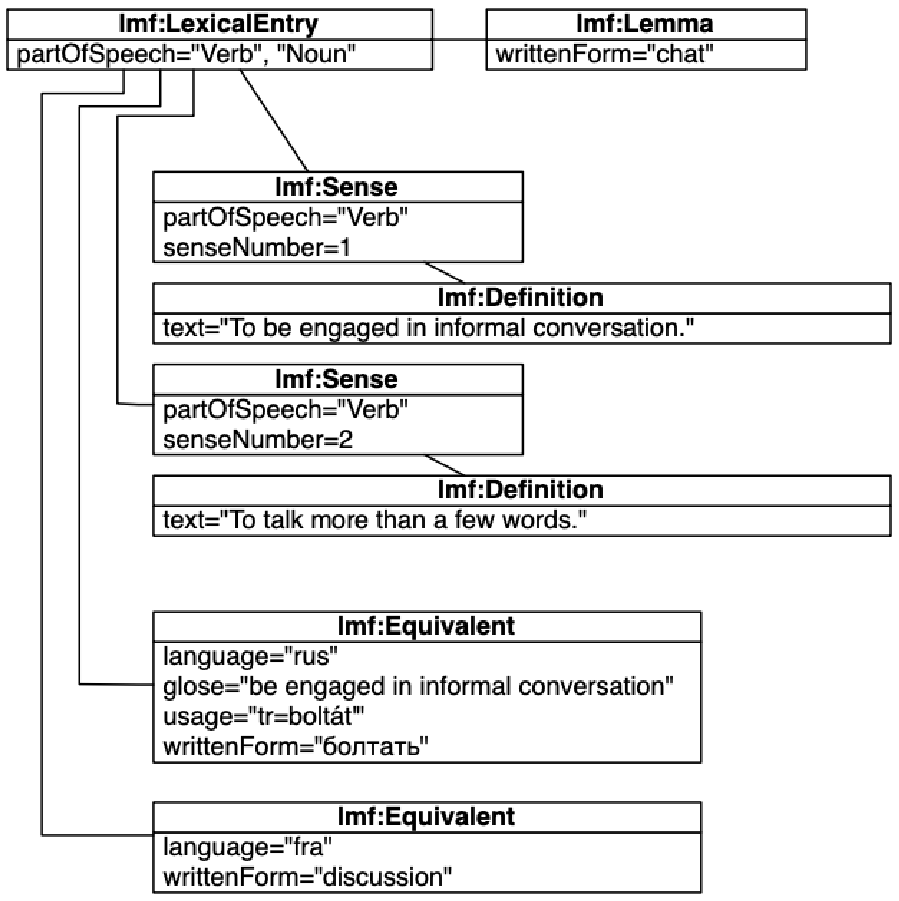
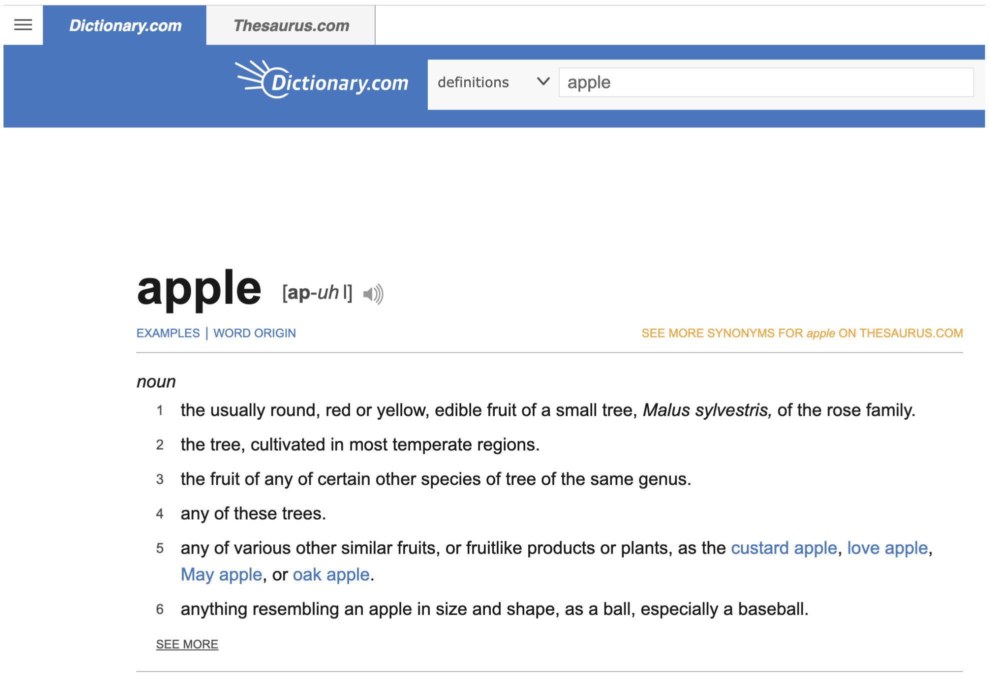


Figure 113: Dbnary

1.2.4 Dictionary.com

Dictionary.com is the world's leading digital dictionary. It provides millions of English definitions, spellings, audio pronunciations, example sentences, and word origins. Dictionary.com's main, proprietary source is the Random House Unabridged Dictionary, which is continually updated by Dictionary.com team of experienced lexicographers and supplemented with trusted, established sources including American Heritage and Harper Collins to support a range of language needs. Dictionary.com also offers a translation service, a Word of the Day, a crossword solver, and a wealth of editorial content that benefit the advanced word lover and the English language student alike [24].



apple [ap-*uh* l]

EXAMPLES | WORD ORIGIN

SEE MORE SYNONYMS FOR *apple* ON THESAURUS.COM

noun

1 the usually round, red or yellow, edible fruit of a small tree, *Malus sylvestris,* of the rose family.

2 the tree, cultivated in most temperate regions.

3 the fruit of any of certain other species of tree of the same genus.

4 any of these trees.

5 any of various other similar fruits, or fruitlike products or plants, as the custard apple, love apple, May apple, or oak apple.

6 anything resembling an apple in size and shape, as a ball, especially a baseball.

SEE MORE

Figure 114: Dictionary.com

Dictionary.com is not an open-source project, so it's not possible to estimate mechanisms used in its work. But judging by the information from the Dictionary.com web-site, it's possible to predict that it uses just manual extension of database and no advanced algorithms of data linking.

Dictionary.com also has its own mobile application.

1.2.5 Non-Wikipedia based methods

In the book S̈peech and Language Processingb̈y Daniel Jurafsky Stanford University and James H. Martin University of Colorado at Boulder [25] were described other automated methods of extraction lexical information from texts. They are based on generating the so-called semantic vectors.

Let's see an example illustrating this distributionalist approach. Suppose it's unknown what the Cantonese word ongchoi means, but it is possible to see it in the following sentences or contexts:

(6.1) Ongchoi is delicious sauteed with garlic.

(6.2) Ongchoi is superb over rice.

(6.3) ...ongchoi leaves with salty sauces...

And furthermore there are many of these context words occurring in contexts like:

(6.4) ...spinach sauteed with garlic over rice...

(6.5) ...chard stems and leaves are delicious...

(6.6) ...collard greens and other salty leafy greens

The fact that ongchoi occurs with words like rice and garlic and delicious and salty, as do words like spinach, chard, and collard greens might suggest to the reader that ongchoi is a leafy green similar to these other leafy greens.

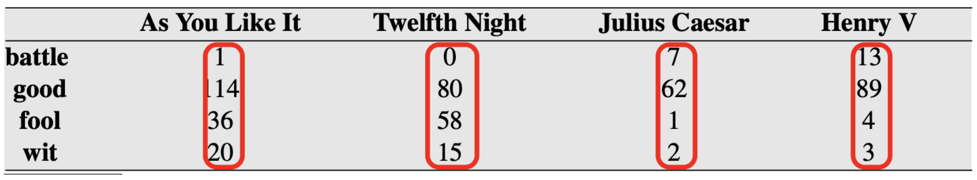
It's possible to do the same thing computationally by just counting words in the context of ongchoi; we'll tend to see words like sauteed and eaten and garlic. The fact that these words and other similar context words also occur around the word spinach or collard greens can help us discover the similarity between these words and ongchoi.

Vector semantics thus combines two intuitions: the distributionalist intuition (defining a word by counting what other words occur in its environment), and the vector intuition of Osgood et al. (1957): defining the meaning of a word w as a vector, a list of numbers, a point in Ndimensional space. There are various versions of vector semantics, each defining the numbers in the vector somewhat differently, but in each case the numbers are based in some way on counts of neighboring words.

The idea of vector semantics is thus to represent a word as a point in some multidimensional semantic space. Vectors for representing words are generally called embeddings, because the word is embedded in a particular vector space.

Notice that positive and negative words seem to be located in distinct portions of the space - antonyms. This suggests one of the great advantages of vector semantics: it offers a fine-grained model of meaning that lets us also implement word similarity.

For example:



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 115: Vector Table

This table represents occurrences of words together in the same context. It's possible to build a graph based on this table .

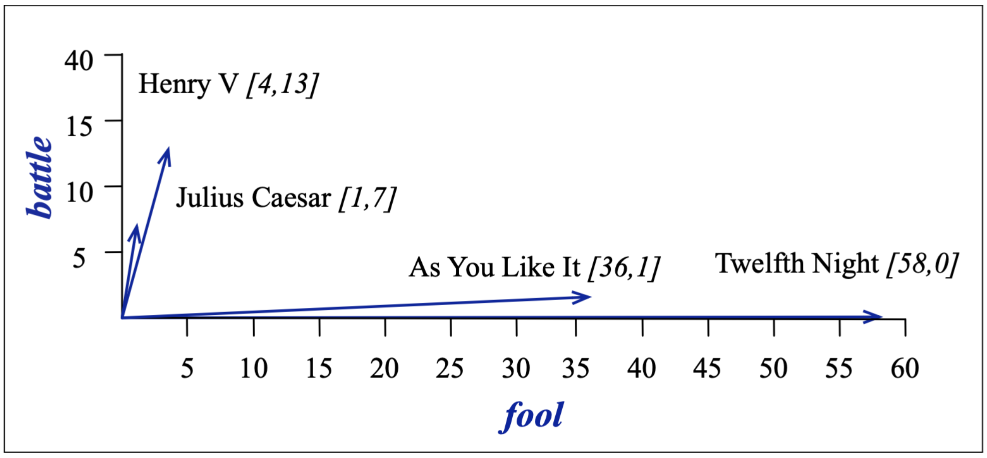


Figure 116: 2D Vector

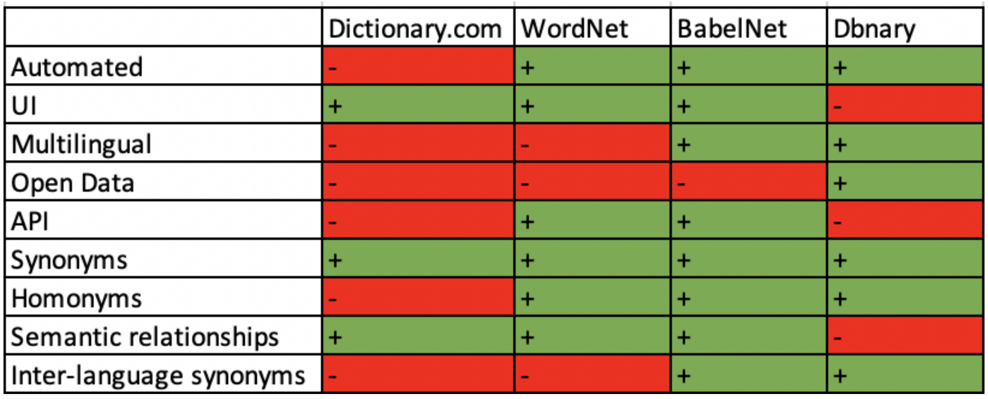
The vectors for the comedies As You Like It [1,114,36,20] and Twelfth Night [0,80,58,15] look a lot more like each other (more fools and wit than battles) than they do like Julius Caesar [7,62,1,2] or Henry V [13,89,4,3] [25].

1.3 Summary

Aspects of qualifying different projects are:

- Automated
  All extraction must be done automatically using scripts
- UI
  Project has UI for easy querying / navigation results
- Multilingual
  Results of linguistic data extraction should be presented in multiple languages
- Open Data
  Datasets should be open and downloadable
- API
  API for querying results should be presented
- Synonyms
  synonyms dataset should be presented
- Homonyms
  homonyms dataset should be presented

- Semantic relationships
  semantic relationships dataset should be presented
- Inter-language synonyms
  inter-language synonyms dataset should be presented



	Dictionary.com	WordNet	BabelNet	Dbnary
Automated	-	+	+	+
UI	+	+	+	-
Multilingual	-	-	+	+
Open Data	-	-	-	+
API	-	+	+	-
Synonyms	+	+	+	+
Homonyms	-	+	+	+
Semantic relationships	+	+	+	-
Inter-language synonyms	-	-	+	+

Figure 117: Comparison

As it is possible to see in a picture, all projects have their own pros and cons.

For example, DBpedia is a good source of free datasets to use. But it's not complete and can be extended by new datasets like synonyms, homonyms, related semantic meaning, and inter-language words. Basically, it's just Wikipedia in a structured form with lots of different datasets which is good to use.

BabelNet is a big project which allows using some API functions, but it doesn't allow to get full datasets. API of BabelNet is also limited to use. This project is developing by a team of university researchers (Sapienza University of Rome) and basically, it's not open-source.

WordNet is not an open-source project either. It's being developed by Princeton University and it covers just English words. WordNet doesn't provide much information about mechanisms of computing data and algorithms of receiving results either.

Dbnarry is similar to BabelNet as it uses data based on Wikipedia, but the number of covered languages is significantly less: 20 in Dbnary and more than 250 in BabelNet.

Dictionary.com is just a presentation of white-papers dictionaries without using any algorithms or advanced computing. Also, it's a commercial project which doesn't allow any community changes.

All the rest of the projects are quite similar to Dictionary.com - they are not using any algorithms and just copying data from dictionaries. This approach is not efficient and can't give any new results for scientists, researchers, and the community.

The idea of the project is to develop a fully open-source tool for further researching, this will also create a new datasets for DBpedia which can be used in the future such as synonyms, homonyms, semantic relations and inter-language synonyms. Another point is to research clearness of data and results, because even such big and old projects as BabelNet and WordNet not always show a perfect result.

Chapter 2

Extraction of linguistic information from Wikipedia

This chapter discusses methods and approaches used in work for Extraction linguistic information from Wikipedia, such as choice of initial datasets for project (section Datasets analyzing), describing process of generating further datasets (sections surface, synonyms, homonyms, close semantic, interlanguage synonyms generating), methods of their filtering and cleaning (section Limitation datasets). In some tasks a combination of different datasets and setting up thresholds for limitation results and storage final datasets in most suitable formats was used. Creating of user interface website (section Web-application) with user-friendly design and designing of properly formed database queries was also part of the research.

At the end (Experimental Evaluation Chapter) statistics and evaluation to similar projects have been done.

All these steps were done to fulfill Master Project objectives. Practical part steps could be described as follows:

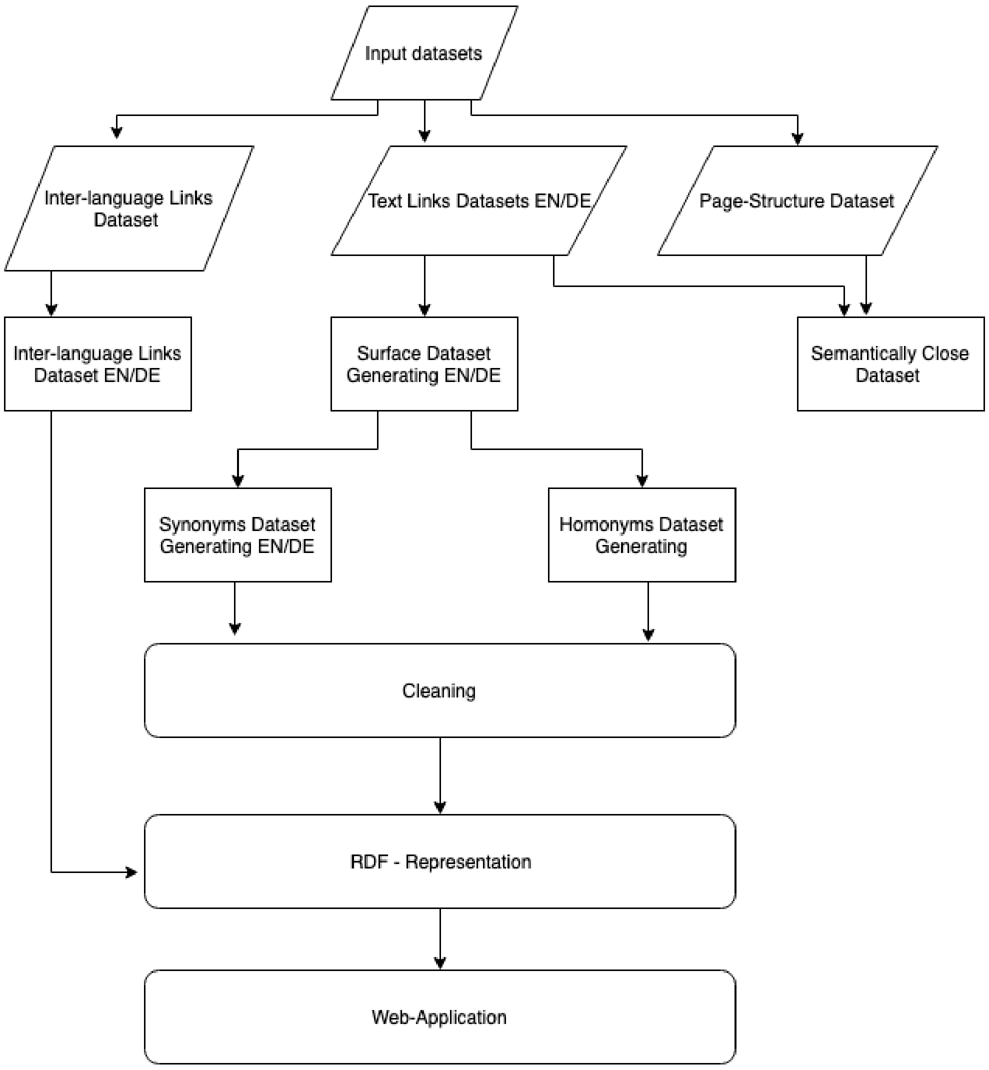

Input datasets
Inter-language Links Dataset
Text Links Datasets EN/DE
Page-Structure Dataset
Inter-language Links Dataset EN/DE
Surface Dataset Generating EN/DE
Semantically Close Dataset
Synonyms Dataset Generating EN/DE
Homonyms Dataset Generating
Cleaning
RDF - Representation
Web-Application

Figure 21: Steps

Generally, the picture above can be described by 5 steps.

Step 1. Preparation.

Preparation step consists of downloading proper datasets and creating surface dataset Link - Anchor - Count. The surface dataset was created using text links NIF dataset.

Step 2. Datasets generating. CSV.

The next step is generating datasets of synonyms, homonyms, semantic relationships and inter-language synonyms in the CSV format. For synonyms and homonyms previously generated surface dataset was used. Semantic relationships dataset was created using text links and page structure dataset.

For inter-language synonyms inter-language links dataset was also used.

Step 3. Cleaning and Filtering

The next steps are focused on cleaning and filtering produced datasets of synonyms and homonyms. Methods are based on Levenshtein distance and frequency of generated synonyms and homonyms.

Step 4. RDF - Representation

To efficiently store and navigate in datasets RDF graphs in triple format are used.

Step 5. Simple Web-Application.

The last step is based on creating simple web-application for easy and user-friendly navigating through the dataset.

2.1 Input data

After analysis of the previous projects and research and making a comparison it was made a decision to make DBpedia datasets as a basis of the Master Project. They are free to use and are one of the most complete for open linked data.

Links, page-structure and inter-language links datasets were used as an input data.

2.2 Surface forms dataset generation

One of the main datasets used in the Master Project is the so-called called "Surface" dataset. This dataset is built from links dataset. The idea of creating a Surface dataset is like to create a base dataset from which all other datasets will be generated. This dataset is a basis for generating synonyms, homonyms and semantic meaning datasets.

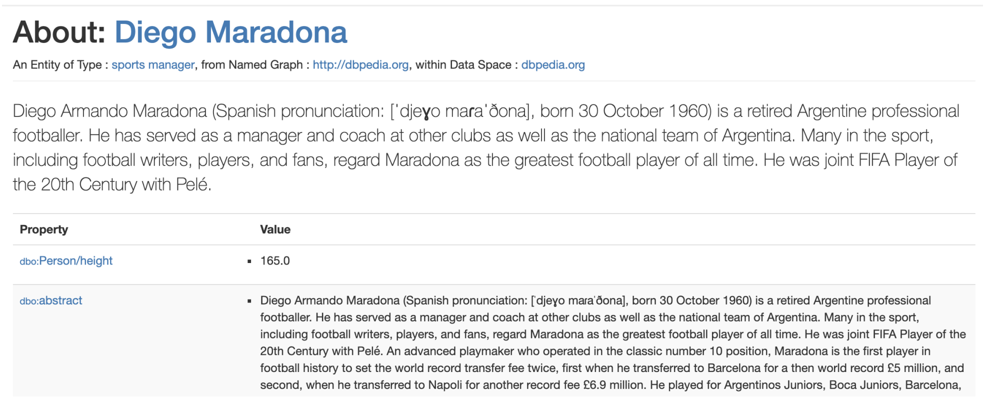
Structure of this dataset will look like:

Anchor - Link - Count

Anchor, in this case, is a text of the link or reference. That means that Anchor should have the same meaning or sense as page, article or entity it refers to. It could be a word, a phrase, a list, a number, a date, etc.

Link is a reference to some other page, article or entity. A link has the following form: ('http://dbpedia.org/resource/Apple) dbpedia.org/resource prefix could be easily replaced with wikipedia.org/wiki and it's easily possible to see a real page at wikipedia.org and not just DBpedia entity.

DBpedia entity is the same as the Wikipedia article page, but it also consists of parts of structured data.



About: Diego Maradona

An Entity of Type : sports manager, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Diego Armando Maradona (Spanish pronunciation: [ˈdjeɣo maɾaˈðona], born 30 October 1960) is a retired Argentine professional footballer. He has served as a manager and coach at other clubs as well as the national team of Argentina. Many in the sport, including football writers, players, and fans, regard Maradona as the greatest football player of all time. He was joint FIFA Player of the 20th Century with Pelé.

Property	Value
dbo:Person/height	▪ 165.0
dbo:abstract	▪ Diego Armando Maradona (Spanish pronunciation: [ˈdjeɣo maraˈðona], born 30 October 1960) is a retired Argentine professional footballer. He has served as a manager and coach at other clubs as well as the national team of Argentina. Many in the sport, including football writers, players, and fans, regard Maradona as the greatest football player of all time. He was joint FIFA Player of the 20th Century with Pelé. An advanced playmaker who operated in the classic number 10 position, Maradona is the first player in football history to set the world record transfer fee twice, first when he transferred to Barcelona for a then world record £5 million, and second, when he transferred to Napoli for another record fee £6.9 million. He played for Argentinos Juniors, Boca Juniors, Barcelona,

Figure 22: DBpedia Article

It could have lots of forms depending on the type and topic of the article. For an example of Diego Maradona - it consists of information as about person height, birthplace, birth date, etc. If it's someplace then this will have parameters like coordinates etc..

Count. The count is a column which represents a number of occurrences of the same Anchor - Link pairs. The count is some representation of the weight of pairs, which is also important for our investigations and is widely used in the project. It also gives user relevant information.

In the beginning, all results are stored in .csv files. It's necessary to convert them after to RDF based files.

To create such dataset Python script was developed.

In the beginning, it's necessary to import special packets and libraries like CSV and DateTime.

After that, it's required to read HDT file. HDT file requires also special index file, for fast searching. In the following code example, we are searching for all triples.

```

document = HDTDocument("nif_text_links_en.hdt")
writer = csv.writer(open("surface_non_group.csv","w"))
(triples, cardinality) = document.search_triples("", "", "")
```

Each link in the NIF file is described with seven triples. For the purpose of surface creation, we need just the first and the last, triples which describe anchor and link accordingly. The following example shows the first and the seventh triple of the first link in NIF text links dataset. Each triple has the same triplet as it describes the same link.

Anchor:

```
http://dbpedia.org/resource/!!!?dbpv=2016-10&nif=phrase&char
    =1258,1284

```

```
http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#
    anchorOf

Gold Standard Laboratories
```

Link:

```
http://dbpedia.org/resource/!!!?dbpv=2016-10&nif=phrase&char
    =1258,1284

http://www.w3.org/2005/11/its/rdf#taIdentRef

http://dbpedia.org/resource/Gold_Standard_Laboratories
```

It's necessary to catch anchor and link, the following example shows the code of writing link and anchor to the CSV file.

```
 1
 2 db_prefix="http://dbpedia.org/resource/"
 3 i=0
 4 j=0
 5 anchor=""
 6 ref=""
 7 print(str(datetime.datetime.now())+" Start")
 8 for s, p, o in triples:
 9  if "#anchorOf" in p or (db_prefix in o and "#taIdentRef" in p):
10   if i==0:
11    anchor=o
12    i=1
13   elif i>0 and db_prefix in o:
14    ref=o.replace(db_prefix,"")
15    i=0
16    writer.writerow((anchor,ref,"1"))
17    j=j+1
18    if j%100000==0:
19     print(str(datetime.datetime.now())+" "+str(j))
20   else:
21    i=2
22    anchor=o
```

It has been shown in line 8 that searching of triples with anchorOf or taIdentRef part in the predicate has been made. Internal Wikipedia links are concerned in the research, so it's also required to check that link consists of prefix http://dbpedia.org/resource/.

After these operations the surface dataset is created. But as the previous code example shows it's not complete. All Count columns are filled just with ones. The next step is to group rows by the first two columns - anchor and link. This task was done using pandas Python library and package.

```
import datetime
import pandas as pd
print(str(datetime.datetime.now())+" Start")
df = pd.read_csv('semantic_non_group.csv', names=['col1', 'col2',
    'col3'])
```

```
5 (df.groupby(['col1', 'col2'])['col3'].sum()).to_csv('surface_group
      .csv')
6 print(str(datetime.datetime.now())+" Finish")
```

The operation above groups surface rows by the first two columns and sum values of the third column (line 4).

The result of the given code is the new Surface dataset with grouped rows.

```
"""'Aglow Koto'""",Nepenthes_'Aglow_Koto',1
"""'Agnes Hopkins'""",Hibiscus_'Agnes_Hopkins',1
"""'Aichi'""",Nepenthes_'Aichi',2
"""'Ajax'""",Alcantarea_Ajax,1
"""'Akaba'""",Nepenthes_'Akaba',1
"""'Al Jolson'""",Neoregelia_Al_Jolson,1
"""'Alaka'i'""",Billbergia_Alaka'i,1
"""'Alaya'""",Aechmea_Alaya,1
"""'Alba'""",Buddleja_davidii_var._nanhoensis,3
"""'Alba'""",Ludisia_discolor_'Alba',1
"""'Alba'""",Nepenthes_'Alba',4
"""'Alba'""",Ulmus_'Alba',1
"""'Albertii'""",Billbergia_Albertii,1
"""'Albertii'""",Vriesea_Albertii,1
```

The task for creating surface is done. The next one is to extract information about synonyms and homonyms.

2.3 Synonyms dataset generation

Synonyms are words which have the same meaning but have different writing. From the view of Wikipedia links and created surface dataset, synonyms - are anchors which refers to the same link.

Form of synonyms dataset should be as follows:

Link - Anchor1 - Count1 - Anchor2 - Count2 - .... - AnchorN - CountN

This task has quite a big complexity on the unordered list. It means that each link should be compared with each one following and in the worst case. And it means that the worst complexity would be N*sqrt(N). For such dataset as a surface with almost 20 million element, this task is too time and resource consuming.

The preliminary task is to sort surface dataset by references.

Again this task was done quite fast by high-performance Pandas library. On the top there is importing of Pandas and DateTime libraries, then new data frame is assigned and Pandas perform sorting with writing the result to surface group order reference file.

```
import datetime
import pandas as pd

print(str(datetime.datetime.now())+" Start")
```

```
df = pd.read_csv('../csv/surface_group.csv', names=['col1', 'col2'
    , 'col3'])
(df.sort_values("col1")).to_csv('../surface_group_order_ref.csv')
print(str(datetime.datetime.now())+" Finish")
```

After that surface dataset will look like:

```
124816,"""'t Haantje""","'t_Haantje,_Overijssel",1
124817,"""'t Haantje""","'t_Haantje,_Rijswijk",1
124818,"""'t Haantje""",'t_Haantje_(Coevorden),1
17145224,"""t'Haantje""",'t_Haantje_(North_Brabant),1
124819,"""'t Haantje""",'t_Haantje_(disambiguation),1
124821,"""'t Harde""",'t_Harde,17
124822,"""'t Harde""",'t_Harde_railway_station,2
124823,"""'t Heem""",'t_Heem,1
124825,"""'t Hof""",'t_Hof,1
```

With sorted Surface dataset task of generating synonyms datasets will take not many resources - this will N time complexity task.

On the following code the process of generating synonyms dataset based on grouped and ordered surface is shown:

```
import csv
import datetime

writer = csv.writer(open("../csv/synonyms.csv","w"))
reader = csv.reader(open("../csv/surface_group_order_ref.csv"))
writeString = []
currentWord=""
print(str(datetime.datetime.now())+" Start")
for row in reader:
 if currentWord!=row[2]:
  if len(writeString)!=0:
   writer.writerow(writeString)
   writeString.clear()
  currentWord=row[2]
  writeString.append(currentWord)
 writeString.append(row[1])
 writeString.append(row[3])
print(str(datetime.datetime.now())+" Finish")
```

The process is going in reading the third column and writing it first in a row for new synonyms dataset. The first is always a link, than references with their counts are appending to a new row. When new link appeared string is written to file and the process begins from the start.

```
$20K_House,"""$20K House""",5
$20k_House,"""$20k House""",1
$21_a_Day_(Once_a_Month),"""$21 a Day (Once a Month)""",2
$24_in_24,"""$24 in 24""",5
"$25,000_Pyramid","""$25,000 Pyramid""",4
$25_Million_Dollar_Hoax,"""$25 Million Dollar Hoax""",3
$2_Pistols,"""$2 Pistols""",1
$2_Wonderfood,"""$2 Wonderfood""",1
```

```
9 $2_billion_arms_deal ,"""$2 billion arms deal""",2
```

As it is possible to see from the example above, results are not clear. Some Anchors differ just by one letter or even quotation marks.

Cleaning

The next step of dealing with extract synonyms task is to analyze produced dataset and develop proper filtering and cleaning methods. For this purpose more than one hundred surface rows were taken randomly.

The first thing that was seen is that words could have not relevant synonyms. This could happen when people edited links wrongly.

The second thing is that Wikipedia comprises separate articles for Lists which should also be removed.

The third is that some anchors and links are just dates, numbers without any sense.

Furthermore, it's necessary to strike a golden mean - not to clean dataset too much, but also to remove all unrelated data.

For this purpose Python script was written.

It consists of 3 steps.

1. Clean

```
 1 def clean(experiment):
 2     writer = csv.writer(open("../csv/experiments/experiment_"+
          experiment+"/clean.csv", "w"))
 3     reader = csv.reader(open("../csv/synonyms.csv"))
 4     writeString = []
 5     for row in reader:
 6         writeString.clear()
 7         if len(row) > 3 and is_valid(row[0]):
 8             writeString.append(row[0])
 9             for index in range(1, len(row)):
10                 if index % 2 == 1:
11                     non_quotes = row[index].strip(<list of quote
                          marks>)
12                     non_quotes = non_quotes.strip()
13                     if is_valid(non_quotes):
14                         writeString.append(non_quotes)
15                         writeString.append(row[index + 1])
16         if len(writeString) > 1:
17             writer.writerow(writeString)
```

At this step, the script removes all rows which have less than three elements (line 6 of code above). This means that there is just one link and one anchor in a row.

```
1 def is_date(string):
2     try:
3         parse(string)
4         return True
5     except ValueError:
6         return False
```

```

def is_valid(string):
    if string.startswith("List_"):
        return False
    if string.isdigit():
        return False
    if is_date(string):
        return False
    return True
```

The given script removes all the quote marks, Lists, check if a given anchor is a date or digit and remove it if it is.

2. Group

After cleaning it's necessary to group by produced new rows because after quote marks removing it could happen that two rows are similar.

```
def group(experiment):
    writer = csv.writer(open("../csv/experiments/experiment_"+
        experiment+"/group.csv", "w"))
    reader = csv.reader(open("../csv/experiments/experiment_"+
        experiment+"/clean.csv"))
    writeString = []
    for row in reader:
        writeString.clear()
        writeString.append(row[0])
        for x in range(1, len(row)):
            if x % 2 == 1 and not row[x] in writeString:
                writeString.append(row[x])
                writeString.append(row[x + 1])
                for y in range(x + 2, len(row)):
                    if row[x] == row[y]:
                        index = writeString.index(row[x])
                        writeString[index + 1] = int(writeString[
                            index + 1]) + int(row[y + 1])
        writer.writerow(writeString)
```

After that similar rows are grouped and here comes the most interesting part - Limitation of non-relevant synonyms.

3. Threshold

In threshold task, two approaches were used: Valid/Invalid cases and Levenshtein Distance.

Valid/Invalid cases assume 4 situations. In the case of the threshold, it's very similar to the F1 score problem. Limitation is based on two parameters, so there exist 4 cases.

These parameters are a similarity of words to the link, which can be calculated by Levenshtein Distance and Frequency of occurrences of given synonym for given link. Experiments are described in 2.7 Cleaning and Filtering

The basic algorithm of limitations (synonyms):

```
Valid
if currentSimilarity >= similarity and currentFreq >= freq:
 writeStringValid.append(row[x])
 writeStringValid.append(row[x + 1])
Valid
elif currentSimilarity >= similarity and currentFreq < freq:
 writeStringValid.append(row[x])
 writeStringValid.append(row[x + 1])
Valid
elif currentSimilarity < similarity and currentFreq >= freq:
 writeStringValid.append(row[x])
 writeStringValid.append(row[x + 1])
Invalid
elif currentSimilarity < similarity and currentFreq < freq:
 writeStringNonValid.append(row[x])
 writeStringNonValid.append(row[x + 1])
```

After that relatively clean synonyms are generated.

2.4 Homonyms dataset generation

Process of generating homonyms is similar to the process of generating synonyms. It's necessary to create a dataset according to the following scheme:

Homonyms are words which have similar pronunciation, but different meaning. From the view of Wikipedia anchor represents similar pronunciation and references - different meanings.

Anchor - Link1 - Count1 - Link2 - Count2 - ... - LinkN - CountN

Steps for generating homonyms dataset.

1. Sort Surface by anchors 2. Generate Homonyms dataset 3. Cleaning 4. Grouping 5. Limitations

There are just minor changes in the algorithm due to a different order of references and anchors, except :

1) Grouping part. For homonyms, grouping should be done twice. At first in a vertical axis by columns and then in a horizontal axis in a row. It happens because after quotations mark removing it could happen that some Anchors are similar, after grouping them - it's necessary group also their links values, which also could be the same after the first grouping.

2) Algorithm of attaching words to valid and non-valid groups is different from synonyms.

```
Valid
if currentSimilarity >= similarity and currentFreq >= freq:
Invalid
elif currentSimilarity >= similarity and currentFreq < freq:
Valid
elif currentSimilarity < similarity and currentFreq >= freq:
Valid
elif currentSimilarity < similarity and currentFreq < freq:
```

2.5 Semantic relationships dataset generation

One more task was to create a dataset of words close in semantic meaning.

Two approaches have been used to find it.

Approach #1

Document - Link Connection

The idea is to create a dataset in the following form:

Source Article URL - Destination Article URL - Count

To implement this approach HDT file text links was used again. The following sample of code shows the extraction process.

```
document = HDTDocument("nif_text_links_en.hdt")
(triples, cardinality) = document.search_triples("", "", "")
db_prefix="http://dbpedia.org/resource/"
j=0
source=""
destination=""
print(str(datetime.datetime.now())+" Start")
for s, p, o in triples:
 if "#taIdentRef" in p and db_prefix in o:
   source=s.split("?dbpv=")[0]
   destination=o
   writer.writerow((source,destination,"1"))
   j=j+1
   if j%100000==0:
     print(str(datetime.datetime.now())+" "+str(j))
```

After computing dataset with the following values was obtained:

```
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Khamis_Mushait,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Dhahran_Aljanoub,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Sarat_Ubaida_Governorate,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Rijal_Alma_Asir,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Rijal_Al-Hajr,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Khamis_Mushayt,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Oil_boom,1
```

After that the similar similar approach was used as in Synonyms and Homonyms Generation. It's necessary to group values by the first two columns. As in Synonyms and Homonyms, datasets Pandas library was used. Grouping showed the following results:

```
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Bariq,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Barley,1
```

```
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Basket,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Bisha,1
http://dbpedia.org/resource/'Asir_Region,http://dbpedia.org/
    resource/Coffee,1
```

It's possible to see minor changes. The number of occurrences counted more than once was seen as extremely low. To check and to be sure in this assumption is was created a script for statistics analyzing.

```
 if maxCount < len(row)-2:
  maxCount = len(row)-2
print(maxCount)
statList = [0]*(maxCount+1)
for row in reader:
 totalCount = totalCount + 1
 statList[len(row)-1]=statList[len(row)-1] + 1
elemIndex=0
for elem in statList:
 writer.writerow((str(elemIndex), str(elem)))
 elemIndex = elemIndex + 1
print(totalCount)
```

After receiving results from the statistics it becomes clear that 90,97769872% of pairs have been counted just once. And basically, it corresponds to Wikipedia organizational rules to link words or phrases just once in one article.

After receiving statistical results it was made a decision to search for other methods of finding semantically close words. But still produced results can be used in the future.

Approach #2

The second approach is based on the paragraph method. DBpedia allows finding a paragraph inside the article where the link is situated. To archive that it is required to use another DBpedia NIF dataset - a page-structure. This dataset allows finding an index of the symbol where each paragraph starts and ends.

In this task a combination of two datasets text links dataset and page structure dataset was used.

In the beginning, it's created a list of paragraphs for each article. Paragraph is represented as two indexes: start index and end index. After that all links from the article link with the corresponding paragraph. This connection is based on the start and end indexes of paragraph and reference.

```
 for paragraph in entityParList:
  k=k+1
  if startIndex>paragraph[0] and endIndex<paragraph[1]:
   entityParList[k].append(o1)
```

After that, it produces a list of paragraphs, where each paragraph is represented already as list of links. Now it's required to make combinations of the given references inside one paragraph. Combinations are made as grouping links each other into pairs.

The formula for calculating a number of such combinations is combinations without repetitions.

$$N = \frac{n!}{r! \cdot (n-r)!}$$

For example, there exist 4 references inside one paragraph. Then there will be $24/4 = 6$ combinations.

Combinations are found using combinations package from the itertools library.

At the end such results are produced:

```
Yoko_Ono,Teknolust,1
Cindy_Sherman,Barbara_Kruger,1
Cindy_Sherman,B._Ruby_Rich,1
Cindy_Sherman,Ingrid_Sischy,1
Cindy_Sherman,Carolee_Schneemann,1
Cindy_Sherman,Miriam_Schapiro,1
Cindy_Sherman,Marcia_Tucker,1
Cindy_Sherman,Lynn_Hershman_Leeson,1
Cindy_Sherman,Strange_Culture,1
Cindy_Sherman,Sleater-Kinney,1
Cindy_Sherman,Teknolust,1
Barbara_Kruger,B._Ruby_Rich,1
```

2.6 Inter-language synonyms dataset generation

As it has already been described in the previous chapters, German as the most complete language in Wikipedia by humans after English was chosen as a second language for generating Inter-language Synonyms . It covers most articles after English. But still, it has almost three times fewer entities comparing to English Wikipedia.

Unfortunately, there doesn't exist HDT file for German Wikipedia and it's essential to use .ttl files. There exist online tools for converting TTL to HDT, but they are not appropriate for huge files. TTL have the same structure but have a much bigger size and it's needed to use different APIs to parse these files. One of the solutions for parsing .ttl files is rdflib. But the problem in the usage of it is that at first, it's building full RDF graph and just after that it is doing operations on it.

```
result = g.parse(location="../datasets/nif_text_links_de.ttl",
    format='nt')
```

This operation is highly resource consuming and it's difficult to predict how much time it can take to build a full graph. In addition, this operation takes lots of RAM memory. After analyzing these factors it was taken a decision to create own methods of parsing .ttl files.

The solution is based on combinations of different methods - simple readying .ttl files using python and rdflib methods of splitting triples in a dataset.

```
    print(fp.readline())
    for line in fp:
        rdfxml = line.strip()
        if rdfxml:
            g = rdflib.Graph()
            g.parse(data=rdfxml, format='nt')
```

Using these methods .ttl file is readying line by line without requiring the consumption of a big amount of resources. And it's easy to estimate completion time of producing new dataset.

The next operations are similar to creation of surface and synonyms for the English version. There are just small changes in the code, related to specific character of German links in Wikipedia.

Cleaning, grouping, and Limitations operations are similar to operations in English.

After all these operations a new, similar dataset, but for the German language is created.

```
Adam_Hunt_(Dartspieler),Adam Hunt (Dartspieler),1.0,Adam Hunt,2.0
Adam_Hunt_(Schachspieler),Adam Hunt (Schachspieler),1.0,Adam Hunt
    ,2.0
```

The next step is to create a links dataset for English and German. For this purpose inter-language links dataset for German references was taken. It consists of triples - German link - sameAs - English link (or other languages links). Links for other languages except German and English were removed, also dataset was cleaned from external links.

```
<http://de.dbpedia.org/resource/Migingo> <http://www.w3.org
    /2002/07/owl#sameAs> <http://dbpedia.org/resource/
    Migingo_Island> .
<http://de.dbpedia.org/resource/Francis_Dana> <http://www.w3.org
    /2002/07/owl#sameAs> <http://dbpedia.org/resource/Francis_Dana
    > .
<http://de.dbpedia.org/resource/Little_Marton_Mill> <http://www.w3
    .org/2002/07/owl#sameAs> <http://dbpedia.org/resource/
    Little_Marton_Mill> .
```

2.7 Cleaning and filtering

The task is to find a proper value for such parameters as similarity and frequency.

The similarity is calculated by Levenshtein Distance. The code could be found in Appendix B. It can take values from 0 - absolutely different, to 1 - absolutely similar.

Frequency is a ratio of occurrences of the word in synonyms dataset to the total number of synonyms in a given dataset. The dataset in a given case is a row with anchors in synonyms file. Frequency also can take values more than 0 - rare occurrences and 1 - absolutely just only one in a dataset.

Synonyms and homonyms have different nature, so finding limitations for these two datasets is different.

Examples of Experimental evaluation:

2.7.1 Post-processing synonyms

Table 21: Threshold Synonyms Table

Valid currentSimilarity $\geq$ similarity currentFreq $\geq$ freq	**Valid** currentSimilarity $\geq$ similarity currentFreq $<$ freq
Valid currentSimilarity $\geq$ similarity currentFreq $<$ freq	**Invalid** currentSimilarity $<$ similarity currentFreq $<$ freq

The table above describes valid and invalid categories for finding proper synonyms values.

Absolutely valid synonyms are those which have high symbols similarity and high frequency. Absolutely invalid ones are absolutely different, this group has low similarity and low frequency.

Concerning the other two groups - they also correspond as valid synonyms, because if any values appear quite often and have a low similarity - they can also be synonyms.

In synonyms values in one dataset - one row should be as much close as it is possible.

All experiments are made on a small dataset with 109 rows. All input and output files could be found on a CD.

Experiment Name,001. Let's start with low limitations.

Similarity,0.1

Frequency,0.1

True Positive,230

False Positive,176

False Negative,11

True Negative,20

Observations: Neckar, Neckar River Valley,1, Neckar river,4. Neckar is a river, so Neckar River Valley is not a synonym for it.

The next experiments were based on step by step increasing limitations. Just on parameters of similarity 0.5 and frequency 0.5, Neckar River Valley disappeared from the valid results.

Experiment Name,002
Similarity,0.5
Frequency,0.5
True Positive,81
False Positive,146
False Negative,19
True Negative,191
Observations: Visitors_(V_TV_series),Visitor,1,The Visitors,4,Visitors,4. The given values were marked as non-valid. Let's decrease a bit frequency and similarity limitations.
Experiment Name,003
Similarity,0.47
Frequency,0.4
True Positive,90
False Positive,142
False Negative,21
True Negative,184
The given parameters satisfy all previous observations.

2.7.2 Post-processing homonyms

Homonyms are words which sound alike and are written alike but have a different meaning. So in contrast to synonyms values in one dataset - one row should be as much different as possible.
Experiment Name,001
Similarity,0.2
Frequency,0.5
True Positive,104
False Positive,251
False Negative,11
True Negative,88
Observations: Economy of North Dakota,
Economy_of_North_Dakota,1,North_Dakota#Economy,1 - given values marked as valid as basically, they mean the same.

In a given case, it's required to increase the frequency parameter just a little bit to illuminate given values.
Experiment Name,002
Similarity,0.2
Frequency,0.51
True Positive,70
False Positive,285
False Negative,5
True Negative,94

Observations: Economy of North Dakota,North_Dakota#Economy,1 is still marked as valid. Let's decrease the similarity parameter.

Experiment Name,003
Similarity,0.13
Frequency,0.51
True Positive,73
False Positive,320
False Negative,2
True Negative,59

Observations: The given parameters fully satisfy previous observations.

2.8 Converting to RDF graph

For better search methods with good performance, it's good to store datasets into graphs. RDF graphs allow faster search algorithms using SPARQL queries or using other libraries search.

In the project it was used rdflib to convert CSV to TTL files. TTL files were chosen as the most suitable for the given project, as they store files in triples and produced on previous steps CSV files can be easily converted to graphs with triples.

Structure of RDF TTL files is similar to Ontolex Lemon, but it also includes counted weights for synonyms and homonyms to give results in more precise forms.

```
reader = csv.reader(open("../csv/experiments/synonyms/
    experiment_003/valid.csv"))
namespace_manager = NamespaceManager(Graph())
namespace_manager.bind('ns1', nif, override=True)
g=Graph()
for row in reader:
    for x in range(1,len(row)):
        if x%2==1:
            g.add([rdflib.term.URIRef("http://dbpedia.org/resource
                /"+row[0]+"?dbpv=2016-10&nif=synonyms"),
                    rdflib.term.URIRef("http://dbpedia.org/
                        resource/?dbpv=2016-10&nif=anchors"),
                    rdflib.term.Literal(row[x])])
            g.add([rdflib.term.URIRef("http://dbpedia.org/resource
                /"+row[0]+"?dbpv=2016-10&nif=synonyms"),
                   rdflib.term.URIRef("http://dbpedia.org/resource
                       /?dbpv=2016-10&nif=counts"),
                   rdflib.term.Literal(row[x+1])])
g.bind("nif",nif)
g.serialize(format="turtle")
g.serialize(destination="synonyms.ttl",format="turtle")
```

On the sample above it is shown a code example for generating a .ttl file for synonyms. For homonyms procedure is similar.

As namespaces used DBpedia ontology.

On the following sample the result of the given code described:

```
<http://dbpedia.org/resource/Mario_Kempes?dbpv=2016-10&nif=
    synonyms>
ns1:anchors
 "Kempes",
 "Mario Kempes" ;
ns1:counts
 "2",
 "60".

<http://dbpedia.org/resource/Mikea_Forest?dbpv=2016-10&nif=
    synonyms>
ns1:anchors
 "Mikea",
 "Mikea Forest";
ns1:counts
 "6",
 "8".
```

2.9 Simple Web-Application

Another part of the research was to create a website with a user friendly interface. As the whole project was made in Python, so it was taken a decision to make website back-end using also Python to have better compatibility of products.

As a basis for the back-end part Flask Framework was chosen. This framework was chosen because it's a light-weight micro-framework. Comparing to another Python Framework Django, it's easy to install, learn and start working.

It's easily installed, just using one command:

pip install Flask

Front-end part was done using HTML, CSS. As base CSS bootstrap styles were taken.

For asynchronous communication and data updates JQuery was taken, to update user output after changing input info. JQuery is a powerful tool based on JavaScript. Using it - there is no need to refresh web-page each time when it's necessary to receive results.

2.9.1 Front-End

Front-end part is simple and it consists of HTML, CSS and JavaScript code.

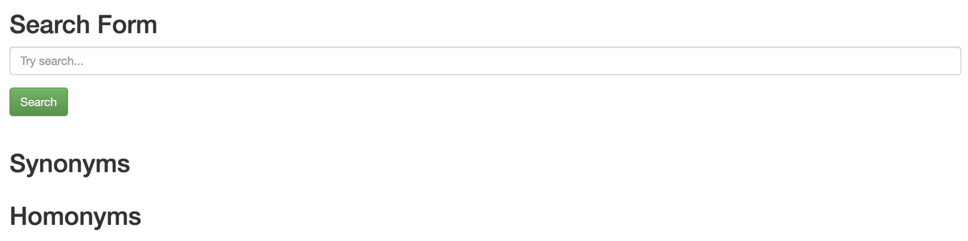


Figure 23: Front-end

Each time user clicks on the Search button, JQuery sends a request to Back-End function to update results from the datasets without refreshing the web-page.

```
            $('#search').bind('click', function() {
                $.getJSON($SCRIPT_ROOT + '/_search', {
                    name: $('input[name="name"]').val(),
                }, function(data) {
                    $("#synonyms").text(data[0]);
                    $("#homonyms").text(data[1]);
                });
                return false;
            });
        });
```

It receives results in the data array and updates the text attribute of each field. Div fields of results are described by id.

2.9.2 Back-End

Back-end part was developed in Python. It allows search in both formats (CSV and RDF).

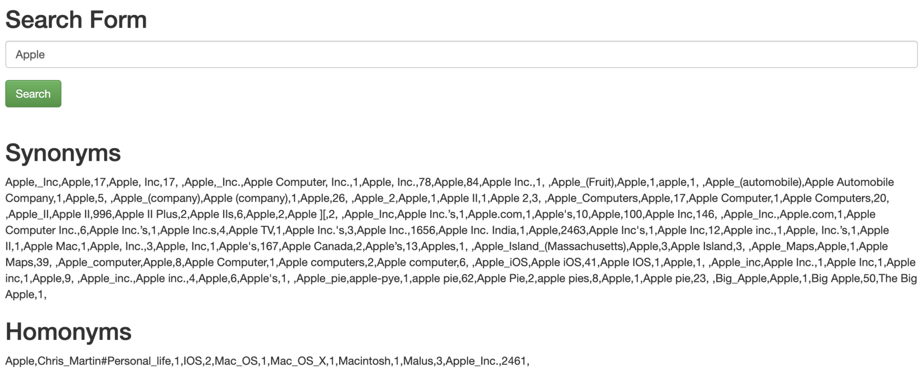
RDF method requires loading RDF file to an RDF data structure when the application deploying on the server.

```
def index():
    synG.parse("../ttl/synonyms.ttl", format="turtle")
    homG.parse("../ttl/homonyms.ttl", format="turtle")
    return render_template('index.html')
```

After that the search mechanism looks like that:

```
        for s,p,o in synG:
            if levenshtein_distance(search,o)> <value from 0 to
               1>:
                result.append(synG)
                break
    else:
        reader = csv.reader(open("../csv/synonyms_fin.csv"))
        for row in reader:
            for x in range (1,len(row)):
                if x%2==1:
                    if levenshtein_distance(search,row[x])> <value from 0 to 1>:
                        result.append(row)
                        break
```

On the following picture is shown an example of the result of the search:



Search Form

Apple

Search

Synonyms

Apple,_Inc,Apple,17,Apple, Inc,17, ,Apple,_Inc.,Apple Computer, Inc.,1,Apple, Inc.,78,Apple,84,Apple Inc.,1, ,Apple_(Fruit),Apple,1,apple,1, ,Apple_(automobile),Apple Automobile Company,1,Apple,5, ,Apple_(company),Apple (company),1,Apple,26, ,Apple_2,Apple,1,Apple II,1,Apple 2,3, ,Apple_Computers,Apple,17,Apple Computer,1,Apple Computers,20, ,Apple_II,Apple II,996,Apple II Plus,2,Apple IIs,6,Apple,2,Apple][,2, ,Apple_Inc,Apple Inc.'s,1,Apple.com,1,Apple's,10,Apple,100,Apple Inc,146, ,Apple_Inc.,Apple.com,1,Apple Computer Inc.,6,Apple Inc.'s,1,Apple Inc.s,4,Apple TV,1,Apple Inc.'s,3,Apple Inc.,1656,Apple Inc. India,1,Apple,2463,Apple Inc's,1,Apple Inc,12,Apple inc.,1,Apple, Inc.'s,1,Apple II,1,Apple Mac,1,Apple, Inc.,3,Apple, Inc,1,Apple's,167,Apple Canada,2,Apple's,13,Apples,1, ,Apple_Island_(Massachusetts),Apple,3,Apple Island,3, ,Apple_Maps,Apple,1,Apple Maps,39, ,Apple_computer,Apple,8,Apple Computer,1,Apple computers,2,Apple computer,6, ,Apple_iOS,Apple iOS,41,Apple IOS,1,Apple,1, ,Apple_inc,Apple Inc.,1,Apple Inc,1,Apple inc,1,Apple,9, ,Apple_inc.,Apple inc.,4,Apple,6,Apple's,1, ,Apple_pie,apple-pye,1,apple pie,62,Apple Pie,2,apple pies,8,Apple,1,Apple pie,23, ,Big_Apple,Apple,1,Big Apple,50,The Big Apple,1,

Homonyms

Apple,Chris_Martin#Personal_life,1,IOS,2,Mac_OS,1,Mac_OS_X,1,Macintosh,1,Malus,3,Apple_Inc.,2461,

Figure 24: Front-end Result

2.10 Used Technologies

Open-Linked Data is a modern field in computer science. Most of the tools developed for Open-Linked Data are open-source and developed by communities. When conducting the project a lot of tools for working with big data have been investigated, for making computations on big data. It was important to choose tools which perfectly correspond to the tasks. They shouldn't be too large for a given task, but at the same time they should correspond to the requirements facing them. Also, there should be a compromise and the golden mean between consumption of RAM and processor load.

2.10.1 Programming Language

First of all, there was a task to choose the proper and the most suitable language. Most of the tools operating with big data converse with Python, Java or R.

R language. R language is a powerful data analyzing tool, which has a rich history, lots of libraries and packages. But it also has its disadvantages like hard integration with other tools, relatively high entry-level and Slow High Learning curve Dependencies between library.

Java. Java presents quite huge and complex frameworks mostly from Apache, such as Apache Spark, Apache Hadoop, Apache Mahout and Java JFreechart. These frameworks allow to build highly scalable and distributed computing. They are good for solving really huge problems for companies like Google, Facebook and etc. For usage in a project of extraction of linguistic information from Wikipedia it could be too much, all computations were done on one laptop, so there are not so many opportunities to scale it to distributed or parallel computing.

Python. Python basically has fewer libraries and packages to work with data structures compared to R or it's not so powerful as Java frameworks and tools. But it has lots or light-weight libraries which can be used easily without any huge installations, it's easily integrated with other technologies and could be used in lots of fields. As it has already been shown Python can be also used in website development as back-end language, but it has also lots of data manipulating libraries, such as pandas, numpy, etc. Besides, Python can be integrated and used in the same tools as Java as Apache Spark and Hadoop.

After analyzing this programming languages Python has been selected as the most suitable programming language for project needs. It's easily integrated and could be used in a variety of fields.

2.10.2 Working with HDT

As it has already been described HDT is a compressed form of RDF files. There exists an open-source project which provides a simple tool for manipulating with HDT triples - pyHDT. It has also simple installation through pip command.

```
from hdt import HDTDocument
document = HDTDocument("test.hdt")

# Fetch all triples that matches { ?s ?p ?o }
# Use empty strings ("") to indicates variables
(triples, cardinality) = document.search_triples("", "", "")

print("cardinality of { ?s ?p ?o }: %i" % cardinality)
for triple in triples:
  print(triple)
```

Basically, there's all functionality which pyHDT provides. It just searches for all triple in a file and prints them. In given example search_triples() showed that we are searching for all triples. A disadvantage of this API is that it's necessary to input the full name of a predicate, subject or object to quotation marks to find it dataset. PyHDT can't find triples by partial values of a predicate, subject or object.

PyHDT read RDF HDT files line by line, so it doesn't require a lot of RAM and speed of computation depends just on processor speed. The given computational power speed of reading HDT is about 12500 triples per second.

It's not possible to parallel or distribute computations with the current library, all computation is done linearly.

This is the only found tool for working with HDT files for Python, tools for Java and C++ were also found.

2.10.3 Working with TTL

TTL files have higher volume and require more resources in computing. TTL files were used in computations for inter-language synonyms because HDT files are available just for an English version of DBpedia.

One of the tools which can read and work with RDF TTL files is a rdfLib library. This library at the begging stores all the values inside the graph and just after that it's possible to work with the data. As German language text-links file size is more than 80GB, it's very resource consuming process for such task. So working with RDF TTL files was done using a combination of rdfLib and simple python reading of files.

The speed of reading such TTL files line by line is 7800 triples per second.

2.10.4 Datasets generation

During Datasets Generating there was a need to find an easy, lightweight tool with high performance.

When conducting the project a bunch of tools and methods was experienced. One of the most resource consuming operations in the project was the Grouping operation.

To perform these operations the following tools and methods were experienced.

1) Simple Script.

To write a simple script for grouping, it takes a lot of time to produce simple surface dataset. With N = 128 million in the worst case it would take years without any hashing mechanisms. And hashing also require time.

2) Database.

The second approach was to use databases to handle complicated operations. One of the databases under experience was Oracle MySQL database. The advantage of this method is that it requires relatively not much time - about 30 minutes for grouping 128 million elements. But it has disadvantages as well: it has a very long installation process and requires a lot of space on PC. One of the focus thing in the research was to develop tools using just one programming language and inter-connected technologies.

3) Dask

Dask is a parallelization library for Python. The advantage is that it can do tasks in parallel. The problem is that not everything can be parallelized and pure number of methods to work with.

4) Other Methods

Some other Python methods and tools were also used. For example, dividing a file into small pieces and grouping them by parts. After that they were combined and grouped again. But this method is too complicated, it doesn't give much performance win and the program has failed due to RAM lack.

5) Apache Spark

Apache Spark has also been used. The advantage of it is a big bunch of tools and methods. The disadvantage is a big size of the package and lot of computations, it has also failed on RAM luck. When making the project Python version of Apache Spark - PySpark has been used.

6) Pandas

Pandas is a small open-source library which provides simple, but powerful tools to work with datasets. During experiments with different tools and methods, it has been found that it's the best choice for such task as the extraction of lexical information from Wikipedia.

2.10.5 Web-Site Tools

Back-end framework for website Python Flask framework has been used. It's easier and simpler than Django, also it's Python framework which makes it possible to use just one programming language while making the project.

For front-end have been used Bootstrap styles, HTML, CSS, and JQuery - a library based on JS.

2.11 Statistics

Finally have been made 6 datasets: Synonyms English, Synonyms German, Homonyms, Semantics on a Paragraph level, Semantics on an article level and Inter-language links. Four of them have been made in both formats: RDF and CSV.

Table 22: RDF Datasets

RDF	**Size, triples**	**Size, Mb**
Synonyms EN	*5873468*	*285.2*
Synonyms DE	*3104034*	*151.1*
Homonyms	*2721239*	*149.9*
Inter-language links	*1139523*	*168.7*

Semantic Relationships Dataset on Paragraph level in CSV format has also been created.

Size of dataset is 45.03 GB and 1,145,099,724 relationships.

All computations were done on PC with given parameters.

OS: MacOS Mojave version 10.14.4

Processor Name: Intel Core i5

Processor Speed: 3,3 GHz
Number of Processors: 1
Total Number of Cores: 2
L2 Cache (per Core): 256 KB
L3 Cache: 4 MB
Memory: 16 GB
Computational time to build the build the biggest dataset - synonyms 3 hours, 24 minutes.

Chapter 3

Experimental Evaluation

While conducting the project a series of experiments and evaluations have been made.

3.1 Evaluation Metric

For analyzing produced results certain actions should be taken. For this purpose there exists a F1 score.

To measure test accuracy F1 score, or also known as F-score or F-measure is used in the statistical analysis of binary classification. It takes in advance the number of correct positive results divided by the number of all positive results returned by the classifier. and r is the number of correct positive results divided by the number of all relevant samples. Relevant samples - are the samples that should have been identified as positive. The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and the worst at 0 [26].

The F1 score is also known as the Sørensen–Dice coefficient or Dice similarity coefficient (DSC) .

$$F1 = \frac{2TP}{2TP + FP + FN}$$

3.2 Evaluation

To determine quality of results was done two series of experiments.

First series of experiments has goal to determine F1 Score. To determine F1 Score it's required to have some dataset to be sure that we determine synonyms and homonyms correctly. For such purpose was chosen Oxford Dictionary.

Second series of experiments has goal to determine quality of results comparing to similar project - BabelNet.

For both series of experiments dataset of words was chosen randomly. Dataset contains 20 words. Experiments were conducted manually by browsing BabelNet and our dataset with the same queries.

3.2.1 F1 Score Evaluation

Evaluation Example:

Tree

Our Dataset:

Synonyms (tree as plant):trees, saddle

Homonyms: Tree (as plant), BOPM, Crataegus, MM tree, Porphyrian tree, Quadtree, Saddle tree, Strappado, Tree Computing, Tree Data Structure, Tree (descriptive set theory), Tree (graph theory), Tree and hypertree networks, Peach.

Errors in Synonyms: trees. Trees is not synonym for tree. Saddle is a synonym of tree (as Oxford Dictionary says)

Errors in Homonyms: Crataegus and Peach are trees, but it just more specific meaning of tree as plant. Binomial options pricing model (BOPM) is a model and also can't be named as tree. Everything else can be counted as homonyms.

F1 Score (synonyms) = 1/2 = 66.7 % F1 Score (Homonyms) = 11/14 = 78.6 %

Following table shows results of series of experiments and average value of F1 Score.

1 - Word

2 - Synonyms F1 Score, %

3 - Homonyms F1 Score, %

Table 31: F1 Score

1	**2**	**3**
Tree (as plant)	*50*	*78.6*
Table (as furniture)	*100*	*88.9*
Car	*66.7*	*16.6*
Window	*100*	*87.5*
Wall	*66.7*	*57.1*
Apple (Inc.)	*84.2*	*100*
Diego (as Ribas da Cunha)	*100*	*100*
Bottle	*100*	*77.8*
Water	*50*	*48.3*
Road	*66.7*	*13.5*
Screen	*100*	*66.7*
Socket	*100*	*100*
Coat (as clothing)	*0*	*100*
Bag	*66.7*	*88.9*
Stairs	*87.5*	*100*
Door	*66.7*	*60*
Tool	*50*	*75*
Handle	*66.7*	*85.8*
Stone	*100*	*62.5*
Universe	*75*	*71.4*
Average	*77.1*	*90.7*

Lots of words has synonyms F1 score 66.7 - it mean that it was found 3 synonyms where 2 of them were correct. Also for example car handle and door handle are not homonyms, they describe sub-types of the same object.

Average F1 score for synonyms is 77.1 percent, which is relatively good. BabelNet has 78 percent.

Average F1 score for homonyms is 90.7 percent, probably it happens because of very strict filtering.

3.2.2 Comparison to BabelNet

Below are presented examples of evaluation our datasets to BabelNet.

Synonyms

Comparing to BabelNet the number of synonyms is less than 8,855,224 vs 5,873,468

Apple (Inc.)

BabelNet gave the following synonyms - Apple Inc., Apple, Apple Computer.

Produced synonyms in Master Project for Apple (Inc.): Apple.com,1,Apple Computer Inc,6,Apple Inc.'s,1,Apple Inc.s,4,Apple TV,1,Apple Inc.'s,3,Apple Inc.,1656,Apple Inc. India,1,Apple,2463,Apple II,1,Apple Mac

Observations: It is possible to note that produced synonyms have higher variety. Some of them should be invalid. For example: Apple II. Apple II is a computer, but not company. If we take into account the low count value of such invalid synonyms (Apple TV, Apple II) most of values are valid.

Diego Ribas da Cunha

BabelNet: 3 synonyms: Diego (footballer, born 1985), Diego Ribas, Diego Ribas da Cunha

Our dataset: 3 synonyms. Diego,59,Diego Ribas da Cunha,3,Diego Ribas,1

Observations: Diego (footballer, born 1985) was not included into our produced dataset

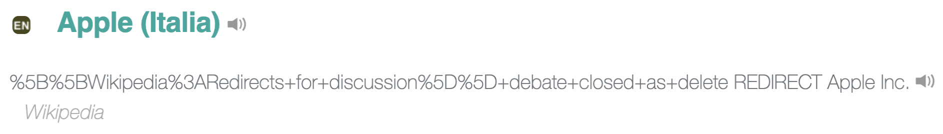
Homonyms

Comparing to BabelNet the number of homonyms is less than 22,922,522 vs 2,721,239

A disadvantage of our homonyms dataset is too strong cleaning, so during experiments on homonyms also was counted links from synonyms, which also refer as homonyms.

Apple

BabelNet gave 19 homonyms for Apple, such as Apple (Fruit), Apple (Tree), Apple (Company), Apple (Store)... Some of them like Apple (Tree) are repeated. Or Apple (Italia) is an empty entity.


EN Apple (Italia)

%5B%5BWikipedia%3ARedirects+for+discussion%5D%5D+debate+closed+as+delete REDIRECT Apple Inc.
Wikipedia

Figure 31: BabelNet Apple(Italia)

Produced homonyms in Master Project for Apple:

Apple,IOS,2,Mac_OS,1,Mac_OS_X,1,Macintosh,1,Malus,3,Apple_Inc.,2461

Observations: in this case Malus is a Apple tree, so all homonyms are valid.

Diego

BabelNet: 21 homonyms.

Our dataset: 15 homonyms

All Homonyms are valid

Following table shows results of series of experiments with BabelNet:

Syn = Synonyms Hom = Homonyms

1 - Word

2 - Synonyms Common

3 - Extra Synonyms BabelNet
4 - Extra Synonyms our Dataset

Table 32: BabelNet Synonyms Comparison

1	**2**	**3**	**4**
Tree (as plant)	*1*	*0*	*0*
Table (as furniture)	*1*	*4*	*1*
Car	*2*	*3*	*2*
Window	*1*	*2*	*1*
Wall	*1*	*0*	*1*
Apple (Inc.)	*3*	*0*	*13*
Diego Ribas da Cunha	*3*	*1*	*0*
Bottle	*1*	*0*	*1*
Water	*1*	*1*	*0*
Road	*2*	*1*	*0*
Screen	*2*	*1*	*1*
Socket	*1*	*0*	*0*
Coat (as clothing)	*0*	*1*	*0*
Bag	*2*	*0*	*0*
Stairs	*1*	*2*	*6*
Door	*2*	*1*	*0*
Tool	*1*	*1*	*1*
Handle	*2*	*1*	*0*
Stone	*2*	*0*	*0*
Universe	*2*	*0*	*0*

Comparing to BabelNet results are relatively close. In some cases, BabelNet was giving better results, in some of our datasets was better. For improvements could be also made removing some plural words which has the same meaning as singular.

1 - Word
2 - Homonyms Common
3 - Extra Homonyms BabelNet
4 - Extra Homonyms our Dataset

Table 33: BabelNet Homonyms Comparison

1	**2**	**3**	**4**
Tree (as plant)	*7*	*15*	*6*
Table (as furniture)	*5*	*5*	*3*
Car	*2*	*25*	*0*
Window	*3*	*7*	*4*
Wall	*3*	*8*	*1*
Apple (Inc.)	*6*	*12*	*1*
Diego	*15*	*21*	*0*
Bottle	*3*	*3*	*4*
Water	*13*	*36*	*1*
Road	*3*	*14*	*2*
Screen	*7*	*14*	*1*
Socket	*2*	*8*	*3*
Coat	*3*	*0*	*0*
Bag	*8*	*6*	*0*
Stairs	*1*	*4*	*0*
Door	*3*	*2*	*0*
Tool	*2*	*4*	*1*
Handle	*2*	*1*	*4*
Stone	*5*	*3*	*0*
Universe	*13*	*5*	*2*

Comparing to BabelNet our dataset was giving in some cases much fewer homonyms. It happens because of strict limitations, but the quality of results is better. In some cases, BabelNet was just returning the same entities twice or some empty entities.

3.2.3 Summary

After conducting a series of experiments it was found that the F1 score for synonyms is 77.1 and homonyms 90.7 percents. In some cases, results of produced datasets were better than BabelNet.

A disadvantage of produced datasets was too strong cleaning for homonyms dataset, so during experiments on homonyms also was counted links from synonyms, which also refer as homonyms.

Conclusion and Future Work

Master Project was conducted to extract useful linguistic information from Wikipedia.

By parsing and combining DBpedia datasets was created such datasets like synonyms, homonyms, semantic relationships and inter-language synonyms. Also was presented simple web-application for querying datasets results.

Results are 5 datasets, which can be navigated using web-application.

Results:

- DBpedia datasets were fully analyzed.
- Existing methods based on extraction linguistic information was analyzed, but unfortunately, not all projects provide detailed methods of extracting information. Also were analyzed methods of extraction linguistic information from non-Wikipedia sources.
- Methods for extraction linguistic information from Wikipedia was developed. As improvements could be used interlinking of produced datasets with similar projects and non-Wikipedia sources. Besides, could be improved filtering methods.
- Developed methods could be used with all DBpedia languages. In Master Project also was produced a dataset of German synonyms.
- Simple web-application for querying and navigating results was developed. As improvements could be used better search mechanisms.
- Developed linguistic datasets has the close quality to similar projects, such as DBpedia. In some cases, our datasets are better.

As for further works it is possible to implement more efficient search algorithms using hash tables and to link produced results with other sources of Open Linked Data, also can be implemented better filtering methods.

Bibliography

[1] Cloud, T. L. O. D. The Linked Open Data Cloud [online]. April 2019, [Cited 2019-04-28]. Available from: `https://lod-cloud.net/`

[2] W3C. Binary RDF Representation for Publication and Exchange (HDT) [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.w3.org/Submission/2011/SUBM-HDT-20110330/`

[3] rdfhdt.org. What is HDT [online]. April 2019, [Cited 2019-04-28]. Available from: `http://www.rdfhdt.org/what-is-hdt/`

[4] Marr, B. How Much Data Do We Create Every Day? [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read`

[5] Dictionary, O. Definition of synonym in English [online]. April 2019, [Cited 2019-04-28]. Available from: `https://en.oxforddictionaries.com/definition/synonym`

[6] Dictionary, O. Definition of homonym in English [online]. April 2019, [Cited 2019-04-28]. Available from: `https://en.oxforddictionaries.com/definition/homonym`

[7] Dictionary, C. Definition of 'semantic' [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.collinsdictionary.com/dictionary/english/semantic`

[8] W3C. RDF [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.w3.org/RDF/`

[9] Miller, P. Sir Tim Berners-Lee Talks with Talis about the Semantic Web [online]. April 2019, [Cited 2019-04-28]. Available from: `https://web.archive.org/web/20130510134842/http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html`

[10] Prateek Jain, A. P. S., Pascal Hitzler. Ontology Alignment for Linked Open Data. *Springer-Verlag*, 2010: pp. 402–416.

[11] DBpedia. DBpedia [online]. April 2019, [Cited 2019-04-28]. Available from: `https://wiki.dbpedia.org/`

[12] University, L. NLP Interchange Format (NIF) 2.0 - Overview and Documentation [online]. April 2019, [Cited 2019-04-28]. Available from: `http://persistence.uni-leipzig.org/nlp2rdf/`

[13] AKSW. NLP Interchange Format (NIF) [online]. April 2019, [Cited 2019-04-28]. Available from: `http://aksw.org/Projects/NIF.html`

[14] W3C. Final Model Specification [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.w3.org/community/ontolex/wiki/Final_Model_Specification`

[15] BalkaNet. BalkaNet - Design and Development of a Multilingual Balkan WordNet [online]. April 2019, [Cited 2019-04-28]. Available from: `http://www.dblab.upatras.gr/balkanet/index.htm`

[16] MultiWordNet. MultiWordNet [online]. April 2019, [Cited 2019-04-28]. Available from: `http://multiwordnet.fbk.eu/english/home.php`

[17] EuroWordNet. Building a multilingual database with wordnets for several European languages. [online]. April 2019, [Cited 2019-04-28]. Available from: `http://projects.illc.uva.nl/EuroWordNet/`

[18] WordNet. WordNet Search [online]. April 2019, [Cited 2019-04-28]. Available from: `http://wordnetweb.princeton.edu/perl/webwn`

[19] George A. Miller, C. F., Richard Beckwith; Miller, K. *Introduction to WordNet: An On-line Lexical Database.* Princeton University Press, 1993.

[20] Maud Ehrmann, D. V., Francesco Cecconi; Navigli, R. *Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0.* European Language Resources Association (ELRA), 2014, 401–408 pp.

[21] sketchengine.eu. SemCor: semantically annotated English corpus [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.sketchengine.eu/semcor-annotated-corpus/`

[22] Roberto Navigli, S. P. P. *BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network.* Elsevier Science Publishers Ltd. Essex, UK, 2012, 217–250 pp.

[23] Gilles, S. DBnary: Wiktionary as a Lemon-Based Multilingual Lexical Resource in RDF. *Semantic Web Journal*, 2014.

[24] Dictionary.com. Dictionary.com [online]. April 2019, [Cited 2019-04-28]. Available from: `https://www.dictionary.com/e/about/`

[25] Daniel Jurafsky, J. H. M. *Speech and Language Processing.* Oxford, 2018, 101–123 pp.

[26] Van Rijsbergen, C. J. *Information Retrieval.* University of Glasgow, 1979.

Appendix **A**

Acronyms

GUI Graphical user interface

NLP Natural Language Processing

NIF The NLP Interchange Format

RDF Resource Description Framework

HDT Header, Dictionary, Triples

TTL Terse RDF Triple Language

TQL Terse RDF Quad-Turtle Language

W3C World Wide Web Consortium

API Application Programming Interface

Appendix B

Code

B.1 Levenshtein distance

```
    if a == b:
        return 1
    if len(a) < len(b):
        a, b = b, a
    if not a:
        return len(b)
    if not b:
        return len(a)
    previous_row = range(len(b) + 1)
    for i, column1 in enumerate(a):
        current_row = [i + 1]
        for j, column2 in enumerate(b):
            insertions = previous_row[j + 1] + 1
            deletions = current_row[j] + 1
            substitutions = previous_row[j] + (column1 != column2)
            current_row.append(min(insertions, deletions,
                substitutions))
        previous_row = current_row
    dis = previous_row[-1]
    if len(a) > len(b):
        return ((len(a) - dis) / len(a))
    else:
        return ((len(b) - dis) / len(b))
```

Appendix C

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
├── experiments ....... the directory of experiments of cleaning and filtering
├── src ..................................... the directory of source codes
│   ├── scripts ................................... implementation sources
│   ├── thesis ............. the directory of LaTeX source codes of the thesis
│   └── wep-app ........................... the directory of web-app sources
└── text .......................................... the thesis text directory
    └── DP_Nazim_Andriy_2019.pdf ........... the thesis text in PDF format
```