FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Automatic detection of topics in poetic texts
Student:	Bc. Martin Bendík
Supervisor:	Ing. Karel Klouda, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

This work is part of a cooperation with the Institute of Czech Literature (ICL), which involves processing more than 1,300 digitized poetry collections from the 19th and early 20th centuries. The aim is to explore the possibilities of identifying topics through unsupervised and supervised learning.
1) Explore the data of the Corpus of Czech Verse [1].
2) Explore and survey NLP techniques for data clustering methods, focusing on visualization and evaluation possibilities.
3) Apply the selected method to the data and evaluate and visualize the results.
4) Explore methods for supervised topic identification – use data from point 1 annotated by colleagues from CAS. Apply the selected method and evaluate the results.
5) Try to compare the results of the methods from points 3 and 4.

[1] https://github.com/versotym/corpusCzechVerse

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 13 February 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Automatic detection of topics in poetic texts

Bc. Martin Bendík

Department of Applied Mathematics
Supervisor: Ing. Karel Klouda, Ph.D.

May 3, 2023

Acknowledgements

I would like to thank the thesis supervisor Ing. Karel Klouda, Ph.D. for his helpful and useful comments and his dedicated time in supervising the thesis. I would also like to thank the Institute of Czech Literature at the CAS for their help in defining the classes for the annotated data. Last but not least, I would like to thank my family and friends, especially my fiancée, for their support during the writing of this thesis.

Declaration

In Prague on May 3, 2023 …………………

Citation of this thesis

Bendík, Martin. *Automatic detection of topics in poetic texts.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

Táto práca sa zaoberá detekciou tém v Korpuse českého verša, ktorý obsahuje desaťtisíce básní z 19. a počiatku 20. storočia. Na efektívne spracovanie veľkého množstva dát využíva metódy strojového učenia. Výstupom týchto algoritmov je množina detekovaných tém a zaradenie jednotlivých básní do týchto tém. To môže pomôcť pri ďalšej analýze diel, sumarizovaní a skúmaní, čomu sa jednotlivé diela venujú.

Práca prezentuje súčasný výskum v oblasti detekcie tém v poetických textoch v rôznych jazykoch a s využitím rôznych technológií. Súčasťou práce je aj vytvorenie niekoľkých modelov, ktoré slúžia na pridelenie tém jednotlivým básniam. Na tento účel boli využité nesupervizované, supervizované a semi–supervizované algoritmy. Všetky vytvorené modely detailne vyhodnocujeme, vizualizujeme, poukazujeme na ich silné a slabé stránky, špecifické vlastnosti a v neposlednom rade modely navzájom porovnávame. Keďže Korpus českého verša neobsahuje anotácie tém básní, pre potreby supervízie učenia bol vytvorený anotovaný dataset, ktorý tvorí podmnožina básní z pôvodného datasetu.

Klíčová slova detekcia tém, modelovanie tém, clusterovanie textu, klasifikácia textu, spracovanie prirodzeného jazyka, poézia

Abstract

This thesis studies the detection of topics in the Corpus of Czech Verse, which contains tens of thousands of poems from the 19th and early 20th centuries. It uses machine learning methods to efficiently process the large amount of data. The output of these algorithms is a set of detected topics and the classification of individual poems into these topics. This can help in further analysis of the artworks, summarizing and exploring what each poem addresses.

This thesis presents current research in the area of detecting topics in poetic texts in different languages and using different technologies. The thesis also includes the development of several models that are used to assign topics to individual poems. Unsupervised, supervised and semi–supervised algorithms have been used for this purpose. We evaluate all the created models in detail, visualize them, point out their strengths and weaknesses, specific features and last but not least compare the models with each other. Since the Corpus of Czech Verse does not contain annotations of poem topics, for the purpose of supervised learning, an annotated dataset was created, which consists of a subset of poems from the original dataset.

Keywords topic detection, topic modeling, text clustering, text classification, natural language processing, poetry

Contents

List of Figures

List of Tables

Introduction

Machine learning algorithms are constantly coming to the fore in a wide range of industries. They are automating processes that were previously done by hand or with multiple single–purpose tools, making them more effective, faster or more precise. There are also areas where the amount of data is so large that scaling these processes by adding human labor is not effective, sometimes even impossible.

Topic detection is an example of a machine learning application in areas where there is a huge amount of data and it does not make sense to process it manually. The task of a topic detection model is to identify what a given natural language text or audio recording is about. Topic models are often used in an environment where there is a need to process a lot of data very quickly in order to minimize the time it takes to gain valuable insights from the data.

The knowledge extracted from the data using the topic model can be diverse. This could include discovering hot news from broadcasts, sentiment mining from user reviews or simply sorting documents into groups based on underlying themes.

Undoubtedly, books are also an interesting source of knowledge. Not only factual or prose, but also poetry. Writers use poetry to express their feelings, their perception of the outer or inner world or to decoratively describe past or present events. In addition to the subjective perception of the author, we can also find social problems or public sentiments in the poems. In combination with the metadata of literary works, we can use topic detection to study, for example, which topics individual authors have dealt with, what topics were frequent in a certain period of time or what are the topics discussed in a corpus of texts.

Objective

The aim of this thesis is to study the data of the Corpus of Czech Verse, to propose a strategy for its preprocessing and to develop a system for topic detection with the help of algorithms. This includes the study of related work and the state of the art in supervised and unsupervised methods, evaluation and visualization. Finally, the ultimate goal was to apply selected machine learning methods to the data in order to assign individual data points to the topics, evaluate their performance and compare their results.

Structure of the thesis

The thesis consists of seven chapters. Chapters 1 to 4 focus on the theory, Chapters 5 to 7 are the practical part of the work.

The first chapter is devoted to the task of topic detection in general and discusses the current state of topic detection research in poetry using examples of related work. Chapters 2, 3 and 4 cover unsupervised, supervised, and semi–supervised methods for topic detection, respectively. In addition, Chapter 2 focuses on evaluation and visualization techniques for unsupervised topic detection algorithms. The third chapter also presents the steps of the NLP pipeline and the techniques used in each step.

The fifth chapter is divided into 2 parts. The first part is a detailed presentation of the Corpus of Czech Verse. The second part is devoted to the creation of an annotated dataset for a subset of data from the Corpus of Czech Verse. The core of the thesis is Chapter 6. It is devoted to the application of selected unsupervised, supervised and semi–supervised algorithms for topic detection. It introduces a unified system of evaluation and interpretation of different topic models, presents their results, and describes their strengths and weaknesses. Finally, the last chapter provides a detailed comparative analysis of the created topic models together with an example of topic detection and interpretation on a poem from the dataset. It also includes possibilities for future work.

Chapter 1

Topic detection

1.1 Natural Language Processing

With the rise of computers, internet, mobile devices, IoT and especially with social networks, more and more unstructured and textual data is being generated on a daily basis. The period in human history characterized by the widespread adoption and use of digital technologies to create, store, process and share information is referred to as the digital age. The digital age has fundamentally transformed the way people live, work and interact. In the digital age, it is natural for people to rely heavily on technology in various aspects of their lives. This includes using their devices for work, entertainment, communication and other activities. With the increased accessibility and convenience of technology, people are also becoming more connected than ever before, with the ability to communicate and share information with friends, family and even strangers from around the world. While digital transformation has made many activities easier and faster, the vast majority of them still rely on human language. This is because language is a highly complex and sophisticated system that allows us to express a wide range of ideas and emotions in a way that other forms of communication, such as images or symbols, simply cannot match.

This has given rise to new industries that store and analyze different types of data. Companies and organizations process emails, requests, user reviews, surveys and many other sources of unstructured textual data. They even digitize printed and recorded materials such as books, old magazines and newspapers or magnetic tapes in order to extract valuable information. This can be used to expand knowledge and support decision making.

It would be impossible for humans to manually process and analyze the ever–increasing volume of data in a timely and cost–effective manner. Fortunately, many language–related tasks that would require extensive human intervention can be replaced by NLP techniques. NLP, or **Natural Language Processing**, is a field of computer science and artificial intelligence that focuses on the processing of human language. It involves the development of algorithms and models that can process and analyze natural language text, speech and other forms of human communication.

Similar to other areas of machine learning, NLP tasks can be divided into these three fundamental groups in terms of supervision:

- **Supervised learning** – the model is trained on a labeled dataset, where the inputs and corresponding outputs are explicitly provided. The model learns to generalize the patterns in the training data and can predict outputs for new inputs. Some of the tasks that can be solved by supervised learning methods are text classification, sentiment analysis, part–of–speech tagging, named entity recognition, machine translation and question answering.

- **Unsupervised learning** – the model is trained on an unlabeled dataset, where the inputs are given but the outputs are not. The model learns to find patterns and structure in the data without explicit guidance. Text clustering, topic modeling, word embedding and text generation are examples of tasks that can be modeled using unsupervised learning methods.

- **Semi–supervised learning** – the model is trained on a combination of labeled and unlabeled data. Typically, a small amount of labeled data is used to train a model, while the remaining unlabeled data is used to improve the accuracy and robustness of the model.

The accuracy of the learning method depends on the specific problem being solved and on the quality and quantity of data available. In general, supervised learning tends to be more accurate in cases where labeled data is available. Unsupervised learning can be useful in cases where labeled data is not available or when the goal is to discover inherent patterns or structure in the data. Semi–supervised learning can combine the strengths of both supervised and unsupervised learning and can be more accurate than either method alone when labeled data is scarce.

Some NLP tasks can be approached using multiple paradigms or a combination of them. For example, text classification can be done using both supervised and unsupervised methods. In supervised learning, we can use a labeled dataset to train a model to classify new texts. In unsupervised learning, we can cluster similar texts together and assign labels to each cluster, which can be used for classification. However, we can also use semi–supervised learning to improve the clustering by using a small set of labeled data to guide the clustering process. The choice depends on the specific task, the available data and on the required accuracy of the results.

1.2 Topic detection

One of the tasks in NLP is **topic detection**. It aims to identify the main themes or topics present in a collection of unstructured textual data. Topic detection can be used to help in understanding the main ideas and trends in a particular domain, such as customer feedback or news articles. By identifying the main topics and associated keywords, one can gain insight into the interests, preferences and opinions of individuals or groups.

Topic detection is mostly used in situations where we face a large collection of documents, often without a title or caption that would describe the content of the text, so there is no other way for a human to find out what the document is about without reading it. While this approach is poorly scalable, over time several NLP approaches have been developed that are able to process a large number of documents in the blink of an eye.

The most straightforward paradigm for topic detection is unsupervised learning with topic modeling and text clustering methods. However, it is not the only option. Topic detection is one of the tasks that can be approached using different methods. If there is a way to create labels for some of the data, we can shift from unsupervised methods to supervised text classification using classifiers such as Naive Bayes (NB), Support Vector Machines (SVMs), or more sophisticated ones based on neural networks. If it is too difficult to obtain labels for a sufficient amount of the data for supervised methods, but we are able to acquire at least some, we can still use that information, albeit limited, for guidance in semi–supervised methods.

In this work we study the possibilities of topic detection on texts from a specific domain. We are dealing with Czech poetic texts from the 19th and the beginning of the 20th century. This setting raises a number of challenges. Poems can be very subjective and abstract, so it is not easy to determine or to evaluate the topic assigned by an algorithm. They often contain archaic and obscure words that are no longer in common use, which can make it difficult to apply standard NLP techniques that rely on large corpora of current language, mostly downloaded from the Internet. Poetry often relies heavily on metaphor, allusion and other forms of figurative language that can be difficult to interpret. Ambiguity is also common in poetry, which can be a challenge for topic detection. Another characteristic of poetry is that it does not fully respect grammatical rules and often employs non–standard grammar and syntax to achieve a particular effect. Another big obstacle to building robust models is data scarcity and sparsity. Even in a relatively large dataset, individual data points may be too different and distant from each other, making it difficult to group them into classes.

Although the original dataset contains only unlabeled data, thanks to the collaboration with experts in the field, we were able to obtain a small sample of data in which topics have been assigned to the texts. This allows us not only to experiment with unsupervised techniques. Labeled data also allows us to use supervised learning techniques. However, the key will be to choose appropriate methods that can robustly generalize the learned knowledge based on a very limited sample of annotated data.

1.3 Related work

Computational poetry refers to the use of computer algorithms and techniques to create or analyze poetry. Researchers explore ways to apply NLP, machine learning (ML) and other computational methods. The tasks of computational poetry vary depending on the specific goals and applications, but some common tasks include rhyme and meter analysis, style transfer, poetry generation or classification. Classification or clustering can be used in a variety of contexts. For example, to determine style, genre, emotional state, to identify authors or to detect topics.

In this work, we will focus on topic detection and classification of poems based on underlying themes. One of the main properties that distinguishes topic detection from other NLP tasks in poetry is that the analysis of poem topics and the classification of poems require semantic features rather than syntax, orthography, phonology or rhyme.

There are many methods for extracting features and building topic models based on them. However, it is still an under–explored area. And that is mainly because poetry collections have a different character – they were written at a different time, in a different place and in a different language. Therefore, a careful methodology is required not only for the application, but especially for the evaluation of topic models.

We can illustrate a variety of ML methods that can be applied to poetry using the example of Persian poetry by Hafez, a 14th century Persian poet, and a series of research papers analyzing his poetic texts. The task was to group the poems from the collection into categories based on the period of his life – youth, middle age and old age. The hypothesis was that based on the extracted topics of the poems, it would be possible to assign them to a time period. This hypothesis is based on the fact that an author's view of the world changes over time. This change is reflected in the topics he addresses in his work.

The work of Rahgozar et al. [1] had a similar setting and goal to ours. They applied several NLP techniques to the poetry of Hafez to classify his poems into three categories based on the period of his life. The study works with only 248 labeled poems, but they are used without labels when training a model, applying unsupervised learning techniques. The poems are clustered in order to to group similar poems based on their underlying themes. The purpose of this step is to explore the semantic structure of Hafez's poetry. A special feature of this work is the use of the quasi–semi–supervised method of so–called anchors. Anchors are hand–picked, expert–labeled poems that serve as representatives of each class and help to guide the clustering process towards the classes defined by the expert. The authors evaluate the quality of the clustering by its consistency with expert–labeled data. Although they were able to produce compact clusters with decent properties, such as silhouette score of up to 95%, it proved that it is quite difficult to achieve a significant consistency with expert classification.

The same authors in [2] used the same dataset to solve the same task using supervised learning. They used similar semantic features based on Bag of Words (BoW) and Term Frequency–Inverse Document Frequency (TF–IDF) and then transformed these representations into the Latent Dirichlet Allocation or Latent Semantic Indexing vector space. They then used these transformed representations to train the SVM classifier achieving almost 85% accuracy.

Research on this dataset continues. Recent work by Ruma et al. [3] employs deep learning (DL) methods for both feature extraction and for classification, achieving state of the art performance. This work is an example of the fact that DL methods are beginning to be successfully applied in the field of poetry processing. The authors also provide an extensive list of related work on poetry classification in several languages, such as English, Indian, Bengali, Arabic, Hindi, Malay and Punjabi. Traditional ML methods are used in most of the works. The most common one is SVM. In addition to the listed languages, research is also being conducted in English [4], Spanish [5] and Czech [6], for example. This only confirms that poetry processing is a current and not yet fully explored subject.

Chapter 2

Unsupervised methods

Topic detection in the context of unsupervised learning is known as topic modeling and text clustering. It refers to the process of automatically identifying the underlying topics present in a collection of documents or text data without the need for pre–existing labels or categories. It involves analyzing latent features of the text, such as word occurrences, patterns of word co–occurrence, language usage or style. The goal is to group together documents or chunks of text that are likely to discuss similar topics or themes.

Topic modeling algorithms are typically based on statistics. By analyzing word co–occurrence patterns and other statistical features of the texts, they aim to uncover the underlying themes present in a set of documents. They assume that the content of each document in the collection is a combination of multiple topics and that these topics are determined by specific words. The goal of the topic modeling approach is to find topics present in the collection and assign each document a set of topics that define the content of the document. A set of topics assigned to a document is typically weighted. These topic weights reflect the likelihood that a topic is present in the document. The output of the application based on topic modeling depends on the use case. In a single–label application, the topic with the highest likelihood is typically assigned to the document. In a multi–label application, the document is typically assigned a fixed number of topics with the highest probability or topics with a probability above a predefined threshold. Due to the nature of the topic representation as a mixture of words, these representations can also be used for text summarization or opinion mining.

Some of the most popular topic modeling methods include Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Non–Negative Matrix Factorization (NMF), Top2Vec and BERTopic. Some of these models are discussed later.

Text clustering, on the other hand, does not explicitly identify topics. Instead, it aims to group documents based on the similarity of their content. It divides a collection of documents into clusters, so that documents in the same cluster are more similar to each other than to documents in other clusters. The output of text clustering is a set of clusters and the documents assigned to them. A common approach to deriving topics from the clustering of documents is to analyze the most frequent words in each cluster and try to infer the underlying topics based on these words. Additionally, various visualization tools such as word clouds can help to interpret the results of clustering. Extracting relevant features from text is a crucial step in text clustering. If the feature extraction step produces a poor representation of the documents, the resulting clustering may be inaccurate, even with the most sophisticated clustering methods.

Algorithms such as K–means, DBSCAN (Density–Based Spatial Clustering of Applications with Noise) and a family of hierarchical clustering algorithms are the most common for text clustering.

2.1 Evaluation

In contrast to supervised methods, unsupervised methods do not use labeled data, which makes the evaluation even more challenging. The absence of "ground truth" labeled test data forces us to use metrics that assess the internal quality and other intrinsic properties of the clustering or topic modeling. The main idea of these metrics is to evaluate how similar documents within a group – cluster or topic – are to each other and how different they are from documents in other groups.

The most common measures used to evaluate topic models are topic coherence and topic diversity. [7, 8] Both measures evaluate the quality of topics, which are represented as discrete distributions over words.

Topic coherence evaluates a topic, represented as a set of words, for being human–interpretable. For a topic to be interpretable, the words it contains should be semantically similar to each other or used in similar contexts. Several different formulations have been proposed for calculating the metric. Most formulations compute the overall coherence between all combinations of pairs of top words from the topic, estimated from word co–occurrences in a reference corpus. [8] A proven similarity measure is cosine similarity. [9] Individual similarities between pairs can then be aggregated into the resulting value.

Röder et al. compared several formulations of coherence measures, including C_{UCI}, C_{UMass}, C_V and many others, and found that the C_V measure performed well compared to other coherence measures in terms of correlation with human judgments of topic coherence. [9] All of these coherence measures are used to evaluate the quality of topics generated by topic models, but the difference between them lies in the way they compute the relatedness between pairs of words in the topics. While C_{UCI} and C_{UMass} are based on statistical measures of the association between the words, C_V focuses on the semantic similarity between the words.

In [10], the calculation of the C_V measure is based on the pairwise pointwise mutual information (PMI) between the words in the topic, and it computes the coherence score by aggregating the PMI values between all possible word pairs in the topic. In general, the C_V coherence score ranges from 0 to 1, with higher scores indicating a higher degree of coherence and interpretability of the topic.

Topic diversity measures the dissimilarity between topics, i.e. how different topics are from each other. A topic model should generate a diverse set of topics that are not redundant or overlapping. The percentage of unique words in the fixed number of top words can be used to compute the metric. In the work by Dieng et al. [11], the authors define topic diversity as the percentage of unique words in the collection that contains the top 25 words from each topic. The topic diversity measure ranges from 0 to 1, with values close to 0 indicating redundant topics and values close to 1 indicating diverse, more varied topics.

Computing topic coherence and topic diversity is fairly straightforward once we have topics represented as word distributions. However, text clustering does not output topics directly. It groups similar documents into common clusters based on their content similarity. And, as with the creation of these clusters, a variety of similarity metrics are used to evaluate them as well.

Silhouette Coefficient is one of the most commonly used metrics to evaluate clustering. [12] It combines the cohesion and separation metrics. Cohesion measures how similar the documents within a cluster are to each other, while separation measures how different the documents in different clusters are from each other.

There are several other options, such as the Davies–Bouldin index, the Calinski–Harabasz index or the Dunn index. Although there are many metrics that can be used to automatically evaluate clustering, it is important to summarize the result. Metrics for clustering methods are not always definitive or universally agreed upon and may act more subjective. Quality is often assessed by human experts who try to interpret and make sense of the discovered patterns or clusters, which can be even more difficult and time–consuming than the modeling itself.

Metrics for the evaluation of clustering can be used to evaluate the quality of the clusters from a general point of view, but not directly from a topic modeling point of view. Nevertheless, there are ways to automate the extraction of topics from clusters. If we select the most frequent words for each cluster, or the most important ones, we get the same representation as in the case of topic modeling methods. This allows us to use the same evaluation methods for both paradigms.

2.2 Visualization

A common way to inspect the output of a topic model is to print the most important words for each topic, along with their associated probabilities. This can provide a quick overview of the discovered topics and the words that define them. However, this approach does not provide a comprehensive understanding of the relationships between topics or their coherence, which is where visualizations can be helpful. Visualizations provide a more intuitive way to analyze the topic model. They can also help to quickly explore the overall structure of the corpus and identify issues that need further refinement, such as splitting or merging topics. In addition, visualizations can help to communicate the results to non–experts. There are typically two levels of visualization for topic models.

Overall visualization of extracted topics

This level of visualization provides an overview of the topics found in the corpus, their composition, the number of assigned documents and the relationships between topics. The visualization can help to understand the major themes and patterns present in the corpus.

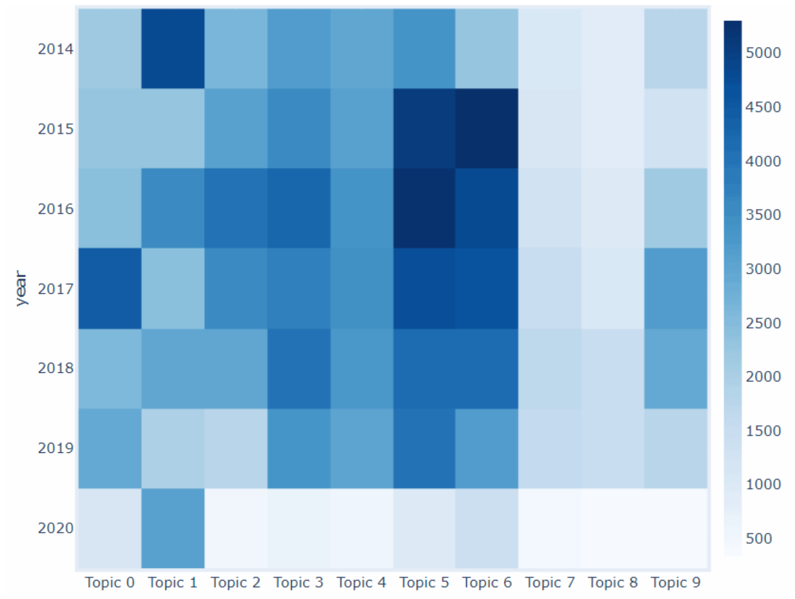


Figure 2.1: An example of visualizing topic representations over years using heatmap. [13]

Histograms can be used to display the distribution of the assigned topics across the documents in the corpus, as shown in Figure 5.1 on page 39. If we find a significant imbalance in the distribution, we can inspect the topics that are underrepresented or overrepresented and consider merging or splitting them.

Heatmaps typically capture a quantitative relationship between two attributes using color gradients. In the context of topic modeling, heatmaps can represent topic density over time, the evolution of topics that each author focuses on, the analysis of topics present in the book, the most common topics by publisher and many more.

The technique of **word coloring** is useful for a deeper study of the distribution of topics throughout the document. By coloring the words of the documents according to their association with the topic, we can study how different topics are represented. Especially in cases where the document is assigned to the topic that is not expected, it can help to identify the words that are the cause.

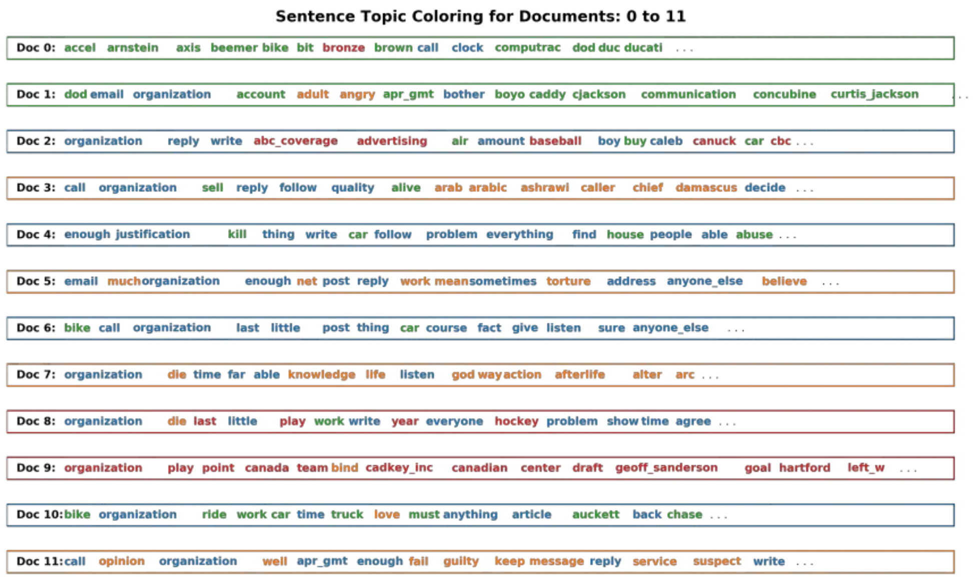

Sentence Topic Coloring for Documents: 0 to 11
Doc 0: accel arnstein axis beemer bike bit bronze brown call clock computrac dod duc ducati ...
Doc 1: dod email organization account adult angry apr_gmt bother boyo caddy cjackson communication concubine curtis_jackson ...
Doc 2: organization reply write abc_coverage advertising air amount baseball boy buy caleb canuck car cbc ...
Doc 3: call organization sell reply follow quality alive arab arabic ashrawi caller chief damascus decide ...
Doc 4: enough justification kill thing write car follow problem everything find house people able abuse ...
Doc 5: email muchorganization enough net post reply work meansometimes torture address anyone_else believe ...
Doc 6: bike call organization last little post thing car course fact give listen sure anyone_else ...
Doc 7: organization die time far able knowledge life listen god wayaction afterlife alter arc ...
Doc 8: organization die last little play work write year everyone hockey problem show time agree ...
Doc 9: organization play point canada team bind cadkey_inc canadian center draft geoff_sanderson goal hartford left_w ...
Doc 10:bike organization ride work car time truck love must anything article auckett back chase ...
Doc 11:call opinion organization well apr_gmt enough fail guilty keep message reply service suspect write ...

Figure 2.2: An example of word coloring visualization technique. [14]

Detailed visualization of a single topic

This level allows for a deeper exploration of a specific topic, showing the words that make up the topic and their relative importance within the topic. It can help in understanding the nuances and finer details of a topic and in identifying potential issues with the topic model.

A **word cloud** provides an intuitive overview of the top words in the topic. It displays a collection of words, where the size of each word corresponds to its importance within the topic. Figures 6.3, 6.7, 6.10 and 6.13 in Chapter 6 show examples of word cloud visualizations.

A **bubble chart** is a scatter plot that uses bubbles to represent words, with the size of each bubble corresponding to its importance. Moreover, it can also capture spatial information – the relative distances between words – in the 2D or 3D projection of the original multi–dimensional vector space. We can see an example of topic visualization using a bubble chart in Figure 2.3. The numbers in the bubbles correspond to words and the diameter of a bubble represents the relative importance of the word within a topic.

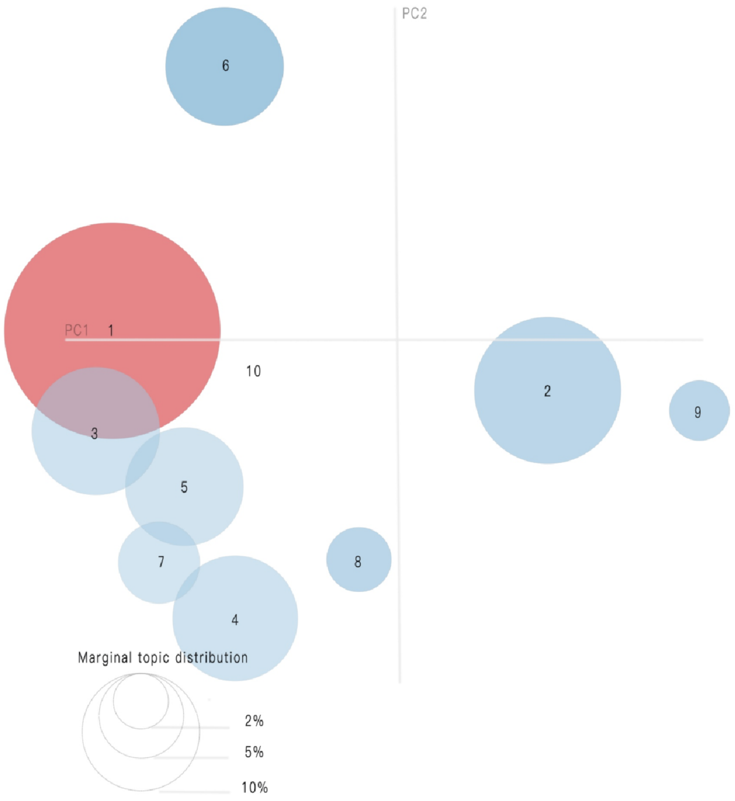


Figure 2.3: An example of bubble chart visualization. [15]

We can also use **histograms** on the topic level to represent word counts and their respective importance for the topic.

2.3 State of the art and related work

This thesis is a part of the research effort on topic detection for the Corpus of Czech Verse. It is a direct continuation of Tesaříková's thesis *Topic Modeling for the Corpus of Czech Verse.* [16] She studied the possibilities of unsupervised topic detection. The work provides an overview of multiple widely used topic modeling algorithms divided into groups, and an overview of their application to similar tasks of topic modeling for poetic texts in different languages.

The first group of algorithms is a non–probabilistic or algebraic group. It includes Latent Semantic Analysis (LSA) [17] and Non–Negative Matrix Factorization (NMF) [18] approaches. Both of the methods are based on matrix factorization. In LSA, a term–document matrix is created from the corpus. In order to factorize this matrix into $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$, it applies Singular Value Decomposition (SVD). The $\mathbf{U}$ and $\mathbf{V}$ matrices represent the relationships between terms and topics and between documents and topics, respectively. The $\mathbf{\Sigma}$ matrix contains the singular values and determines the importance of each topic. NMF provides a different way of term–document matrix decomposition. Given a non–negative matrix $\mathbf{A}$, NMF aims to find two non–negative matrices $\mathbf{W}$ and $\mathbf{H}$ such that $\mathbf{A} \approx \mathbf{W}\mathbf{H}^T$. The columns of the matrix $\mathbf{W}$ represent the topics using terms, and the rows of the matrix $\mathbf{H}$ represent the distribution of topics in each document.

The second group includes probabilistic generative models. Probabilistic generative methods for topic modeling assume that the documents in a corpus are generated from a set of underlying topics, with each topic being a probability distribution over a fixed vocabulary of terms. The goal of these methods is to learn the topic mixture proportions for each document and the word distribution for each topic. Probabilistic Latent Semantic Analysis (PLSA) [19] is a probabilistic modification of LSA. The model assumes that each word in a document is generated by a topic, with the topic's probability of generating that word. The probability of a document is then calculated as the product of the probabilities of each word given the topics. The model is trained using the Expectation–Maximization (EM) algorithm, which updates the topic–word and document–topic probabilities in iterations until convergence. One of the most widely used methods for the topic modeling – Latent Dirichlet Allocation (LDA) [20] – works under the same assumptions. LDA's generative process uses a Dirichlet distribution to choose a distribution over topics for a document. Then, for each word in the document, it generates a topic from the chosen distribution over topics and a word from the distribution over words associated with the chosen topic. It is trained by iteratively updating the probability distributions for topics and words based on the words in the documents, until convergence.

The last group, and the most dynamic in terms of research, is a group of neural approaches. Neural–based models use (pre–trained) neural network models to generate word/document embeddings that aim to capture the semantic meaning of documents. The ability of neural models to extract complex and informative relationships between words from documents can lead to more accurate topic models. In the work [16], the author uses a neural–based approach with the BERT (Bidirectional Encoder Representations from Transformers) model. [21] It uses document embeddings produced by the BERT model in the clustering using Hierarchical Density–Based Spatial Clustering of Applications with Noise (HDBSCAN) to obtain clusters of documents with the similar topic. Before that, Uniform Manifold Approximation and Projection (UMAP) is used to reduce the dimensionality of the embeddings for better clustering performance. Another example from this group is the Top2Vec [22] model. A special feature of this method is that it creates jointly embedded topic, document and word vectors so that similar documents are placed close to each other and also close to the most distinctive words. It also uses UMAP and HDBSCAN to get clusters of documents, but it goes a few steps further. First, it computes the centroids of the clusters – topic vectors. Thanks to the common vector space of the words, documents and topics, it finds the words closest to the centroid, which are the words that describe the topic. An advantage of Top2Vec over the BERTopic [23] is that it can handle the classification of outliers detected by HDBSCAN by simply assigning them to the cluster with the closest centroid.

Tesaříková also summarizes several studies of topic models proposed for poetic texts in English, German, Russian, Spanish and Czech. Applied methods range from algebraic, such as NMF, all the way to neural–based, using BERT embeddings. But the most common method among them is LDA. A nice example of topic modeling using LDA has been done by Plecháč et al. [6] It studies the evolution of poetic traditions across several languages over the time using topic extraction. The authors extracted topics from poetry corpora of four languages – German, English, Russian and Czech – and illustrated the similarities and disparities between different poetic traditions based on how certain topics emerged, merged, or diverged. They picked several topics and found that some are consistent across different languages, some are delayed, whereas other topics are not as extensively discussed than in other languages. The evolution of several topics is discussed in more detail from a literary–historical point of view, which is in agreement with the extracted topics that poets addressed at certain times. This work was also a pioneer work on topic modeling for the corpus we use, since the Czech corpus used in this work was the Corpus of Czech Verse.

Finally, the work [16] provides a comparison of methods applied to the Corpus of Czech Verse, namely LSA, LDA, BERT with UMAP and HDBSCAN, BERTopic and Top2Vec, using topic coherence measure. The best results were achieved using Top2Vec and LDA topic models with topic coherence of 0.454 and 0.432 respectively. The other methods achieved performance similar to each other, but significantly worse than the top two. The work also focuses on the Czech language specifics from an NLP perspective, describes a preprocessing routine and suggests tools for Czech language preprocessing.

One of the goals of this thesis is to follow up on work done by applying unsupervised methods to unlabeled data and to compare these methods with results obtained by supervised methods on newly acquired labeled data.

Chapter 3

Supervised methods

Topic detection in the context of supervised learning is known as topic classification or topic labeling. It is a subtask of general text classification task. Text classification, also known as text categorization, is a common task in NLP. The goal of the task is to assign predefined categories or labels to text based on its content. It works by training a machine learning model to learn patterns and features in the text that distinguish between different categories. All text classification subtasks process a text from the input and output a discrete label, but they differ in what the output label represents.

Sentiment Analysis aims to analyze a text to determine the emotional tone, attitude or opinion of the writer towards a particular topic, product, service or event. The output label represents the sentiment, which can be positive, negative or neutral. Sentiment analysis has many applications in various fields, including marketing, social media, customer service and public opinion analysis.

Question Answering consists of two parts – extractive and generative. Extractive part is a classification task. It determines which candidate answers from the answer pool are reasonable and which are not. Generative part then takes a set of possible answers to produce a final answer.

Topic Classification aims to categorize a text into predefined classes, that represent possible topics, based on the content of the text. The goal is to automatically identify the topics discussed in the text, which can help in organizing and summarizing large volumes of textual data, as well as in extracting insights and trends from them.

Natural Language Inference predicts whether the meaning of one text can be inferred from another. In other words, given a premise and a hypothesis, the task is to determine whether the hypothesis is entailed by the premise, contradicted by the premise, or neither, which represent the three possible categories.

Basic classification process is single–label, which means that each instance is assigned to one and only one class. However, this can be extended to multi–

label classification where each instance can be assigned to multiple classes at the same time. Multi–label classification has especially use in text classification, where analyzed document may belong to several different categories. For example, social media post from a restaurant visited on vacation is relevant to both the "travel" and "food" categories simultaneously.

3.1 State of the art

With the pace of machine learning research it can be challenging to keep up with the latest developments, models and techniques being proposed on a regular basis and understand the nuances and trade–offs between different models. However, not all new models are necessarily relevant to every application or problem.

A good way for individuals to stay up–to–date with the latest developments in machine learning are survey papers. These papers often offer a thorough analysis of the state of the art in a particular field of study and they can assist readers in becoming familiar with the most recent methods and trends. Recently, several survey articles have been published regarding text classification. The work by Li et al. [24] provides a comprehensive overview of the text classification task and its applications, comparing both traditional and deep learning–based approaches and their strengths and weaknesses. Kowsari et al. [25] review a deconstruction of text classification systems into four modules – feature extraction, dimension reduction, classifier selection and evaluations – and explore methods, algorithms and techniques used for each of them. Similar to these works, Minaee et al.'s [26] survey focuses specifically on deep learning. They study more than 150 deep learning models that were introduced in the last few years and significantly improved state of the art on different text classification tasks. In addition, they present an analysis of the performance of these models on more than 40 publicly available benchmark datasets.

Publicly available benchmarks and datasets are essential for evaluating the performance of machine learning models. They provide a standardized way to compare the performance of different models on a given task. This allows researchers to select the currently best performing model for their needs. There are also several benchmarks dedicated for topic classification such as 20 Newsgroups [27], AG News [28] or DBpedia [29]. Even multi–task benchmarks are being proposed, such as the GLUE (General Language Understanding Evaluation) benchmark [30] and SuperGLUE [31]. The GLUE benchmark aims to evaluate and compare the performance of models on nine different tasks with majority of text classification. Multi–task benchmarks reward models that are able to generalize their understanding of language regardless of the task at hand and penalize models that perform well on only some tasks.

Papers With Code[1] is a platform that aggregates research resources – papers, codes, datasets – and provides overview of actual state of the art solutions in various fields of ML. It also exposes a leaderboard of methods and models for particular datasets including topic classification datasets [27], [28] and [29] among others. In the meantime, majority of benchmarks in NLP, similar to other ML fields, were dominated by DL models. Minaee et al. in [26] emphasize that DL models started to consistently outperform traditional ML models with use of large embedding models trained on huge amounts of data. The breakthrough work was done by Mikolov et al. in 2013 with their introduction of Word2vec model [32]. Another big step forward was achieved in 2017 thanks to the new neural network architecture called Transformer [33]. Transformers use attention mechanism to better capture context between words. BERT [21], introduced in 2018, and other BERT–based models such as alBERT or roBERTa still occupy state of the art positions for many NLP tasks.

Shortly after BERT was introduced, researchers at Carnegie Mellon University and the Google AI Brain Team presented XLNet [34] in 2019, a language model that beat BERT on 20 tasks, including text classification. XLNet uses a permutation–based approach to pre–training that allows it to model all possible permutations of the input sequence. This technique allows it to capture the context of each word in all possible relations. Additionally, XLNet introduces the idea of "auto–regressive" pre–training, which means that the model can use its own predictions from previous words to inform its predictions for the current word.

Starting from models like BERT with 340 million parameters, there has been a trend towards using larger models. GPT–3 [35] model developed in 2020 contains 175 billion parameters, GShard [36], also from 2020, contains 600 billion parameters and Switch Transformers [37] from 2021 scale to giant size of 1.6 trillion parameters.

We can expect many new models and improvements in the future. However, it is always necessary to consider how significant the change is and whether it is worth for our particular case.

Despite progress, DL models also have their limitations. Wahba et al. [38] developed a simple SVM linear classifier with TF–IDF vectorized text that holds up with BERT–based pre–trained language models on number of well–established publicly–available datasets in terms of performance. Not to mention the computational requirements associated with running large pre–trained language models.

This is especially the case when there is not enough data available. Xu et al. [39] empirically compare DL models and Random Forests on tabular, vision and auditory data. They focus on datasets with at most 10 000 samples

[1]https://paperswithcode.com

and found that forests excel in scenarios with small sample sizes, whereas deep neural networks perform better with larger sample sizes.

However, the problems in ML are very diverse and so are the data we work with. Therefore, it is good not only to apply the most sophisticated and latest methods available, but also to test simpler and quickly applicable methods that can bring satisfactory results for specific applications or can serve as a baseline solution. Models like NB or SVM may not always provide state of the art performance, but they can be effective for many applications and may have advantages over more complex models, such as faster training, inference time and greater interpretability.

3.2 Text classification pipeline

The process of text classification consists of several consecutive steps specific to the processing of textual data. As shown in Figure 3.1, Li et al. [24] divided the process into modules with specific functions and techniques used in each of the modules. They distinguish between traditional ML methods and DL methods, as only traditional methods require explicit Feature Extraction step in the process. Some works, such as [25], add an optional Dimensionality Reduction step applied after Feature Extraction and before Classification. They also offer a detailed overview of the methods used throughout the pipeline, which is a source of information for the next sections. In the following sections, we will present the role of each module and some of the most commonly used techniques associated with each module.

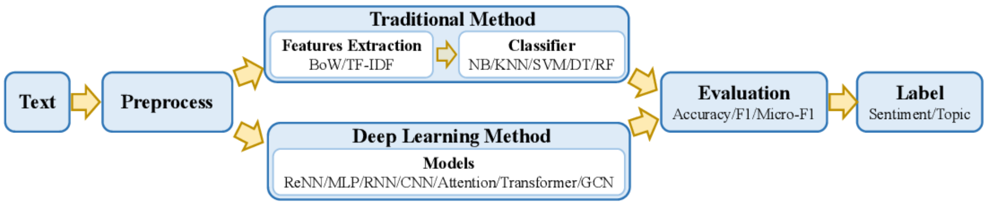


Figure 3.1: Pipeline of the text classification with common methods in each module. [24]

3.2.1 Preprocessing

Preprocessing is crucial for the effectiveness of the methods. Textual data are inherently unstructured and messy, with lots of noise, irregularities and inconsistencies that make it difficult for ML models to learn patterns and make accurate predictions. Several steps of preprocessing aim to clean and convert raw text into structured format that can be used for analysis and learning.

The first step of preprocessing is **tokenization**. It is a method of breaking down the text into individual tokens – the basic unit of NLP analysis. Tokens are most often represented by words, but they can also be phrases or other meaningful elements.

Natural text contains lots of words that have no meaning or do not contain any meaningful information. These are called stop words. We typically want to **remove stop words** because they are not unique and do not carry any specific information about the meaning of a document. Other elements, such as special characters and **punctuation**, that are important for human, can be misleading for algorithms and are usually removed as noise.

The text is most often formed into sentences. The same words can appear at the beginning of a sentence, but also in the middle of a sentence, which means that they are written with a **capital letters** in one case and a small letters in the other. But from the computer's point of view, these are two different words. Therefore, it is necessary to convert all words to the same format, most often to lowercase. But this can lead to other inconsistencies. For example, if we transform the word "US" (United States of America) into "us" (pronoun), we change the meaning of the word. Therefore, we should be careful and use more advanced parsers. Similarly with **abbreviations and slang**, we need to unify the same concepts that are expressed in different ways.

Human–written texts can also contain many **typographical errors**. The ideal is to find these errors, clearly determine whether it is a real error and fix it.

Finally, stemming and lemmatization are text normalization techniques that reduce words to their base or root form. **Stemming** transforms an inflected, affixed, suffixed or otherwise adapted word into a so–called stem or root by "cutting off" parts of the word that are not common to all words that come from the same root. The root of a word is not necessarily an actual word. **Lemmatization** is a more complex technique that transforms different forms of a word into a lemma. A lemma is always an actual word in its basic form. Although lemmatization is more computationally intensive, it usually produces more comprehensible results.

Here is a simple example of text preprocessing on a single sentence using the presented methods.

- Original sentence:

  `The dogs are barking loudly outside.`

- Tokenization:

  `["The", "dogs", "are", "barking", "loudly", "outside", "."]`

- Transformation to lowercase:

  `["the", "dogs", "are", "barking", "loudly", "outside", "."]`

- Stop words removal:

  `["dogs", "barking", "loudly", "outside", "."]`

- Punctuation removal:

  `["dogs", "barking", "loudly", "outside"]`

- Stemming:

  `["dog", "bark", "loud", "outsid"]`

- Lemmatization:

  `["dog", "bark", "loudly", "outside"]`

Czech text preprocessing is specific because of the complex grammar, flexibility and morphology system. Czech has seven grammatical cases and distinguishes between three genders, which results in a sophisticated system of adapting word forms in sentences. This is accomplished by inflection – prefixes and suffixes added to words – and conjugation. Inflection and conjugation also allow for flexible word order because they determine the meaning of the sentence rather than the position of the words in the sentence. Due to these characteristics, Czech texts require specific preprocessing tools to ensure accurate and effective analysis. One of the tools designed for Czech language processing is MorphoDiTa: Morphological Dictionary and Tagger. [40] It is an open–source tool for morphological analysis of natural language texts. It is distributed with trained linguistic models and provides morphological analysis, morphological generation, tokenization and tagging.

3.2.2 Feature Extraction

Feature extraction is the process of transforming textual data into a structured, most often numerical representation that can be used as input to a ML model. The goal of feature extraction is to capture and preserve as much relevant information as possible that will be used to distinguish between different classes or to make predictions. It is important to realize that while the performance of traditional methods depends strongly on the features we prepare for them, DL models can extract them on their own. Their initial layers are capable of learning a set of nonlinear transformations to extract features. This principle is well illustrated by Convolutional Neural Networks, where it is obvious that the initial layers capture simple features such as edges and lines, which are then combined into more complex features.

Weighted Words

The most basic way to extract features from text is to count words that occur in documents. First, a vocabulary is created. This vocabulary contains all the words that appear in the collection of all documents, and each word is assigned a number. Words, which can be further weighted, are then used to represent documents.

Bag of Words (BoW) is a simple model that counts the number of occurrences of each unique word in a document. In this representation, a document is seen as a bag of words – a set of words with repetitions. This means that the order of the words is not preserved.

Here is an example of representing a sentence using the BoW model.

- Original sentence:

  `The cat chased the mouse, but the mouse escaped and hid.`

- Preprocessed text:

  `["cat", "chase", "mouse", "mouse", "escape", "hide"]`

- BoW representation:

  `{"cat": 1, "chase": 1, "mouse": 2, "escape": 1,`

  `"hide": 1}`

The **n-gram** technique splits the text of a document into a set of sequences consisting of n terms that occur in the text in that order. This allows to capture the local context of the words. BoW is a special case using 1–gram. In practice, 2–grams and 3–grams are commonly used.

- Original sentence:

  `The cat chased the mouse, but the mouse escaped and hid.`

- 2–gram representation:

  `["the cat", "cat chased", "chased the", "the mouse",`

  `"mouse but", "but the", "the mouse", "mouse escaped",`

  `"escaped and", "and hid"]`

Term Frequency – Inverse Document Frequency (TF–IDF) combines Term Frequency (TF) and Inverse Document Frequency (IDF) components. TF expresses the number of occurrences of term t in document d relative to the total number of terms in the document. IDF measures how much information the word carries. It solves the problem of common words that appear in the corpus D of all documents. IDF component reduces the weight of words that appear in many documents and, on the contrary, increases the weight of words that are specific to a particular document. Therefore, words that are frequent in the particular document, but also not frequent in other documents, should dominate the representation. In contrast, generic words have smaller weights in the representation because they do not describe the content of the document.

The most common definition is

$$\text{TF–IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D) =$$

$$= \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \cdot \log \frac{|D|}{|d' \in D : t \in d'|},$$

where $f_{t,d}$ is the frequency of the term t in document d, $\sum_{t' \in d} f_{t',d}$ is the total number of terms in the document, $|D|$ is the number of documents in the corpus and $|d' \in D : t \in d'|$ is the number of documents in which the term t appears.

Word Embeddings

Weighted Words methods have several shortcomings. Probably the biggest of them is the inability to capture the semantics of words. For example, synonyms are considered as two completely different words in the feature space if we do not match them during preprocessing. Although n–grams attempt to capture the context of words, they can only do so locally. Word embeddings, on the other hand, can learn richer representations of words based on their usage patterns in a large corpus of text and can capture both local and global context thanks to pre–training on huge corpora of billions of words. Word Embeddings methods map words from a vocabulary to vectors of real numbers in an multi–dimensional space.

Word2vec is an approach proposed by Mikolov et al. [32] in 2013. It uses newly developed CBOW (Continuous Bag–of–Words) and Skip–Gram models to create high–dimensional vectors for words.

The **CBOW** architecture is designed to predict the current word based on the surrounding words (context). The past and future words are used to train a log–linear classifier to classify the current word. Like BoW, COBW does not depend on the order of the input words. The vectors of all input words are averaged in the projection layer. The CBOW model is based on the NNLM (Neural Network–based Language Model) [41]. However, it removes the non–linear hidden layer and connects the projection layer for all words.

The architecture of the continuous **Skip–Gram** model is similar to CBOW but uses the opposite approach. Based on the current word, it predicts the surrounding words – past and future. By optimizing the probability of context words given a current word, it learns a distributed representation of words. In this case, the projection layer is used to project the current word into a high–dimensional space. This projection is used to predict context words.

One year after the introduction of Word2vec, the **GloVe** [42] word embedding technique outperformed it in word analogy, word similarity and named entity recognition tasks. GloVe combines the advantages of the global matrix factorization and local context window methods. While Word2vec relies on a prediction model, which learns to predict words from their context, GloVe

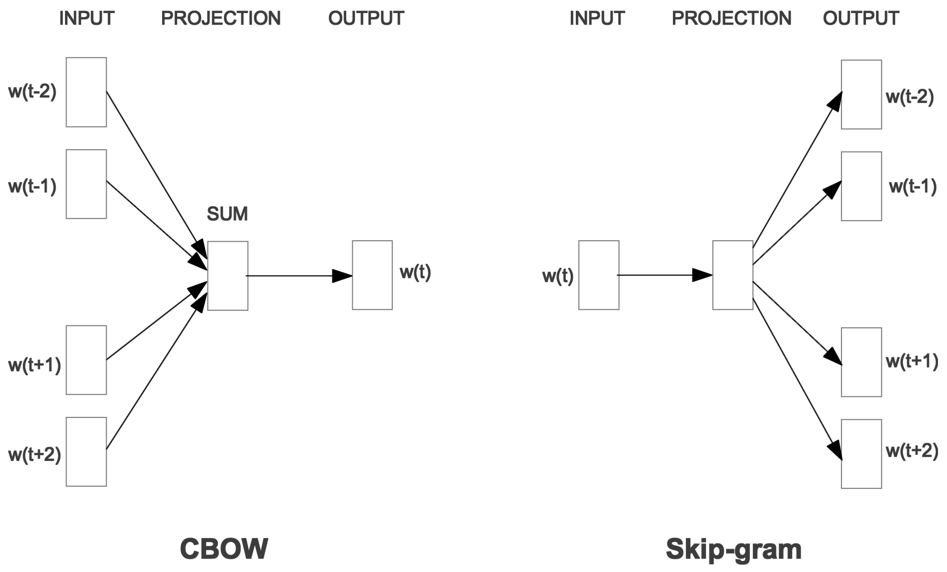


Figure 3.2: Illustration of CBOW and Skip–Gram model architectures. [32]

is a count–based model, which learns word embeddings by analyzing the co–occurrence statistics of words in a corpus. To obtain a low–dimensional vector space where words with similar co–occurrence patterns are closer to each other, it factorizes the co–occurrence matrix of word pairs containing the frequency of their co–occurrence in a given window of words. Despite the different learning process, GloVe embeddings are similar to Word2Vec embeddings in the way they capture semantic relationships between words.

FastText [43] developed by Facebook AI Research lab in 2016 is an extension of the Skip–Gram model with sub–word information. Words are represented as a bag of character n–grams instead of just whole words. Each character n–gram is associated with a vector representation, and the word representation is computed as the sum of these partial representations. This allows the method to better compute word representations for words that did not appear in the training data and rare out–of–vocabulary words. Pre–trained 300–dimensional word vectors for nearly 300 languages have been published and are available for use.

Pre–trained language models (PLMs), like BERT, RoBERTa, ALBERT, XLNet, T5 or GPT–3 can also be used for feature extraction. The hidden layer representations of the deep PLMs can serve as features to train models for various downstream tasks. To generate a high–dimensional vector representation, the input text is first tokenized into individual tokens or subwords. The tokenized input is then passed through the PLM, which is typically a neural network–based architecture that has been pre–trained on large amounts of text data using learning methods like masked language modeling or next sentence prediction. During the pre–training process, the model learns to encode the contextual meaning of each token based on the surrounding context in the text. This results in a rich representation of each token that captures not only its surface form but also its semantic meaning. This is an example of transfer learning techniques, where models are pre–trained on a large corpus of text to learn general language representations and these learned representations can then be used as input features for specific tasks. This can be especially beneficial when working with limited amounts of task–specific training data, as the pre–trained features can help to capture a more general understanding of the language used in the task.

3.2.3 Model Selection

There are several considerations when choosing a model for text classification. One key decision is whether to use traditional machine learning or deep learning models, depending on the specific requirements of the task and the available data.

Traditional machine learning models, such as NB [44] classifiers and SVMs [45], are often considered as baseline models for text classification tasks. The NB classifier is a simple probabilistic model that assumes independence among features and it is particularly useful for small datasets with limited training data. SVM is a powerful model that can handle high–dimensional data and capture complex patterns, making it suitable for text classification tasks.

On the other hand, DL models have shown significant success in text classification tasks, especially when dealing with large datasets. DL models are capable of automatically learning complex feature representations from raw text data, without relying on hand–crafted features, which can be an advantage in tasks where feature engineering is challenging or time–consuming.

When selecting a model, it is important to consider the characteristics of the dataset and the specific requirements of the text classification task. Factors such as the size of the dataset, the complexity of the text data, the availability of labeled data and the desired balance between accuracy and interpretability should be considered. It is also important to evaluate the models using appropriate metrics.

Another important aspect of model selection is training multiple models and then selecting the final model based on their performance. We typically start with the simplest and fastest solutions – baseline models. If they do not deliver the required performance, we move on to more complex models.

3.2.4 Evaluation

When training a machine learning model, it is common to split the data into separate sets for training and evaluation. The most common approach is to use three sets: a training set, a validation set and a test set. The training set is used to train the model. The validation set is used to tune the hyperparameters and to perform model selection, where the best performing model is chosen from a set of candidate models. The test set is used to evaluate the performance of the final model and to estimate its generalization performance on unseen data. This method is called "hold–out" evaluation.

An alternative evaluation method is called "cross–validation". The original dataset is split into two sets – training and test. The training set is divided into multiple folds and the model is trained and validated on different subsets of these folds. This allows for a more robust evaluation as the model is trained and validated on different combinations of data. The purpose of the test set remains the same.

Accuracy and F1 score are the most used metrics to evaluate text classification methods. [24] Accuracy is a measure of the proportion of correctly classified instances out of the total number of instances in the dataset. It is calculated as the ratio of the number of correctly predicted instances to the total number of instances. Accuracy is a commonly used metric when classes are balanced. However, accuracy can be misleading when classes are imbalanced, as it can be high even if the model performs poorly on the minority class.

The F1 score, on the other hand, is a measure of the trade-off between precision and recall. Precision is the proportion of true positives (correctly predicted positive instances) out of the sum of true positives and false positives (incorrectly predicted positive instances), while recall is the proportion of true positives out of the sum of true positives and false negatives (incorrectly predicted negative instances). The F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance in terms of both false positives and false negatives.

Metrics such as sensitivity, specificity, and other metrics derived from the confusion matrix are often used to examine model error in more detail.

Chapter 4

Semi–supervised methods

Semi–supervised learning is a machine learning approach that combines both labeled and unlabeled data to train the model. It is typically used in cases where the dataset contains large amounts of unlabeled data, but acquiring labels for them is expensive. Unlike supervised learning, where the model is trained only on labeled data, semi–supervised learning utilizes a small subset of available labeled data for training and uses remaining unlabeled data to make the model more robust and improve the model's accuracy in predicting the labels of new data points. Semi–supervised learning can achieve higher accuracy than supervised learning with limited labeled data. Natural language processing, computer vision and speech recognition are common areas where semi–supervised learning is used because labeling such data is extremely time–consuming.

4.1 Recursive K–means

The authors in [46] proposed a semi–supervised text clustering method called **Recursive K–means clustering**. The method works with a small set of labeled data and a larger set of unlabeled data. First, it creates a training set from the labeled data and a small portion of unlabeled data, about the size of the labeled data. Then it performs the initial K–means clustering where K is equal to the number of classes in the labeled set. It will divide a training set into K clusters. The generated clusters may contain data from multiple classes. This is where the recursive clustering comes in. Each cluster with data from more than one class is divided using the K–means algorithm, where K is the number of unique data classes within the cluster. This process is applied recursively to each cluster until the entire training collection is divided into a number of small clusters, each containing labeled samples from only one class. Finally, the remaining unlabeled data is assigned to the cluster with the nearest centroid.

4.2 Class–based TF–IDF

Another method of semi–supervised learning, or semi–supervised modeling, is built into the BERTopic algorithm. BERTopic [23] uses a simple approach to to extract the most informative words per each cluster in the computed partitioning. These words are used as cluster representatives for topic interpretation. It is a variant of the standard TF–IDF called **class–based TF–IDF** (c–TF–IDF). Instead of computing term weights across the documents, c–TF–IDF transforms all documents from a class (or cluster) into a single document and computes TF–IDF weights of terms across these aggregated documents per class. It is defined as:

$$\text{c-TF-IDF}(t, i, I, D) = \frac{f_{t,i}}{\sum_{t' \in i} f_{t',i}} \cdot \log \frac{|D|}{\sum_{j \in I} f_{t,j}},$$

where $f_{t,i}$ is the frequency of a term t in a class document i and $\sum_{t' \in i} f_{t',i}$ is the total number of terms in the class document. In the IDF expression, total number of the original, unjoined, documents are divided by the total frequency of the term t across all class documents I.

The c–TF–IDF method can be used for various applications other than informative word extraction. Some of these applications include reducing the number of classes and semi–supervised modeling. [47] By creating c–TF–IDF vectors for a small amount of labeled data, it is possible to capture the content of the classes in the semantic space. Then, without the need to train a predictive model, we can directly perform semi–supervised modeling of unlabeled documents using the distance between the TF–IDF vector of the new document and the vectors of the classes.

4.3 Conclusion

The methods presented in the previous sections are just a few examples of semi–supervised methods. The semi–supervised learning principle offers freedom in the handling of labeled and unlabeled data and in managing learning and classification of large amounts of unlabeled data. However, it is important to note that the semi–supervised learning heavily depends on the specific domain, dataset and problem at hand. To make the most of the limited labeled data available, distilling the maximum information is a crucial task. The supervised part does not even need to use traditional ML algorithms at all. A simple solution, like c–TF–IDF, can often be effective if it can capture the information needed for a specific problem.

Chapter 5

Data

5.1 Corpus of Czech Verse

The dataset used in this work is a subset of the Corpus of Czech Verse [48], which is publicly available through the Github repository[2]. It contains 1 305 out of 1 689 books of poetry with a total of 66 428 out of 76 699 poems. The remaining 384 books are still protected by copyright.

The Corpus of Czech Verse is a corpus of Czech poetry of the 19th century and of the beginning of the 20th century. It is maintained by the Versification Research Group of the Institute of Czech Literature at the Czech Academy of Sciences.

Each book is encoded in a separate JSON file. The data contain unique identifiers of the book and the poem, metadata about the author or editor of the book, the author of the poem, the publication and the poem itself. Furthermore, poems are lemmatised, phonetically, morphologically, metrically and strophically annotated.

Each book file consists of a list of JSON objects representing poems. A poem is described by key–value pairs. These are the keys that each poem contains:

- **book_id**: a unique identifier of a book,
- **poem_id**: a unique identifier for a poem or part of a poem within a book, since some long poems are split up,
- **p_author**: details about an author of a poem,
- **b_author**: details about an author of a book or about an editor if a book contains poems by several authors,
- **biblio**: bibliographic information about the issue of a book,

[2]`github.com/versotym/corpusCzechVerse`

- **body**: the text of a poem with extensive tagging and linguistic preprocessing.

Information about an author includes years of birth and death, the real name under the `identity` key and a name as it appears in the book under the `name` key. It may differ from the `identity` if an author used a pseudonym.

```
"p_author": {
    "born": 1880,
    "died": 1932,
    "name": "Lilia, Hermor",
    "identity": "Bíbl, František"
}
```

Listing 5.1: An example of the author metadata.

The `biblio` key holds details about the the publication. Specifically

- **p_title**: a title of a poem,
- **b_title**: a title of a book,
- **b_subtitle**: a subtitle of a book,
- **publisher**: a publisher of a book,
- **place**: a place of publication,
- **year**: a year of publication,
- **pages**: number of pages of a poem,
- **dedication**: the dedication of a book,
- **motto**: a motto of a book,
- **motto_aut**: an author of the motto,
- **edition**: an edition description,
- **signature**: a library information.

```
"biblio": {
    "motto_aut": "Ovidius Naso, Publius  (Ovid.)",
    "b_subtitle": "Satirické verše o lásce a ženách",
    "publisher": "Mikuláš Boleslavský, Josef",
    "edition": "[1.]",
    "motto": "Zamilovati se jest tolik, jako při zdravém rozumu
   blázniti.",
    "p_title": "Dívka-li tě oklamala,",
    "place": "Praha",
    "dedication": null,
    "b_title": "Trny a trnky",
    "pages": "52",
    "year": "1881",
    "signature": "Národní knihovna ČR, Praha; 54 K 244"
}
```

Listing 5.2: An example of the biblio metadata.

The body of a poem is further structured into a list of stanzas and stanzas into a list of lines. The text of the line is split into a list of words and these are well processed and tagged by

- **token**: a word as it appears in the text,
- **token_lc**: a token in lower case,
- **lemma**: a lemmatised token,
- **morph**: a morphological tag in the Prague positional tag system [49],
- **xsampa**: a phonetic transcription in the X-SAMPA system [50],
- **phoebe**: a simplified phonetic transcription.

From an NLP point of view, we are mainly interested in the `lemma`, `token_lc` and `morph` fields. Using lemmatised tokens is more common in ML applications, as it helps to reduce the dimensionality of the data, remove inflectional variations and improve the consistency of the data. However, the use of lemmatised tokens can also lead to a loss of information about the original form of the words and reduce the interpretability of the data. Rather than relying solely on their lemma or base form, today's state of the art systems trained on vast amounts of data are able to take advantage of the inflected words by considering the context in which they occur.

The 16–character sequence under the `morph` key reveals the characteristics of the word from a morphological perspective. Each position of the sequence corresponds to a different category. In terms of preprocessing the data and removing stop words, part of speech is a very valuable category for us.

Lines that rhyme all share the same value under the `rhyme` key. All punctuation marks that appear in the line text and their respective indexes are

Position	**Description**
1	Part of Speech
2	Detailed Part of Speech
3	Gender
4	Number
5	Case
6	Possessor's Gender
7	Possessor's Number
8	Person
9	Tense
10	Degree of comparison
11	Negation
12	Voice
13	Unused
14	Unused
15	Variant, Style, Register, Special Usage
16	Aspect

Table 5.1: List of morphological categories in the *morph* field.

summarised in the dictionary of the `punct` field. Syllable stress – the emphasis or prominence given to certain syllables in a word when it is pronounced – is encoded as a sequence of bits, where a value of 1 means that the syllable is stressed and a value of 0 means the opposite.

Finally, field `meter` describes the rhythmic structure, or the pattern of beats in a line of verse. It is a systematic arrangement of stressed and unstressed syllables and it creates a musical and rhythmic quality in the poem. The meter of a poem determines the number of syllables in a line, as well as the pattern of stress in those syllables. If it is ambiguous, more metre could be assigned to a single line. Four properties describe the meter, namely

- **type**: a specific named type of meter,
  - *J*: iamb – an unstressed syllable followed by a stressed syllable,
  - *T*: trochee – a stressed syllable followed by an unstressed syllable,
  - *D*: dactyl – a stressed syllable followed by two unstressed syllables,
  - *A*: amphibrach – an unstressed syllable followed by a stressed syllable and then another unstressed syllable,
  - *X*: dactylotrochee – alternating dactyls and trochees. The first foot (a unit of metrical structure, see below for more information) is a dactyl and the second foot is a trochee,
  - *Y*: dactylotrochee with

    * *anacrusis*: an unstressed syllable or group of syllables placed at the beginning of a line of verse, before the main meter begins,
    * *hexameter*: a meter consisting of six feet per line,
    * *pentameter*: a meter consisting of five feet per line,
  - *N*: not recognised.

- **clause**: type of line ending,
  - *f*: feminine,
  - *m*: masculine,
  - *a*: actalectic.
- **foot**: number of feet. A foot is a unit of metrical structure in a line of verse, an arrangement of stressed and unstressed syllables, used to define its rhythm,
- **pattern**: a pattern of strong (s) and weak (w) positions.

```
"text": "Tvá loď jde po vysokém moři,",
"words": [
    {
        "token": "Tvá",
        "token_lc": "tvá",
        "lemma": "tvůj",
        "morph": "PSFS1-S2------1-",
        "xsampa": "tva:",
        "phoebe": "tvA"
    }, {
        "token": "loď",
        "token_lc": "loď",
        "lemma": "loď",
        "morph": "NNFS1-----A-----",
        "xsampa": "loc",
        "phoebe": "loT"
    }, ...
],
"rhyme": 1,
"stress": "011100010",
"metre": [
    {
        "type": "J".
        "clause": "f",
        "foot": "4",
        "pattern": "WSWSWSWSW"
    }
],
"punct": {"6": ","}
```

Listing 5.3: An example of line data.

5.2 Annotations

The Corpus of Czech Verse does not contain any information regarding the topic of poems. Since unsupervised learning does not require such information to create a model, we could use the dataset without any further modifications. However, it is inevitable to extend the data with annotations to provide input–output example pairs for supervised learning methods, as we aim to compare unsupervised learning methods with supervised ones.

There are several ways to create annotations for data, including machine and expert approaches. The machine approach uses algorithms to automatically infer labels for data points in a dataset. They range from trivial rule-based methods, where a set of predefined rules are applied to data points, to complex methods based on ML, such as clustering, where the goal is to find patterns in the data and group similar examples into a common group. On the other hand, the expert approach involves having human experts manually assign labels to examples in a dataset. This is usually done by having experts observe the examples and assign labels based on their knowledge and expertise.

While machine annotation is typically much faster and cheaper than expert annotation, it may not be as accurate and reliable. This is particularly important when the data is sensitive, such as in healthcare or finance, where errors in predictions can have serious consequences.

In some cases, a combination of both approaches can be used to leverage the best of both worlds. For example, using a machine annotation system to pre–label a dataset, and then having experts review and correct any mistakes in the labels. This can help to reduce the time and cost of expert annotation while still ensuring high accuracy and reliability of the labeled dataset.

5.2.1 Creating annotations

Since we directly collaborate with the Institute of Czech Literature of the CAS, we took advantage of their expert knowledge to create an annotated dataset. The first step was to define a list of topics that appear in the corpus. It is crucial to include all possible topics that could describe every single poem in the dataset, but also to keep the number of topics reasonably small. It is therefore important to choose proper level of granularity in order to group poems with similar topic in the same cluster but do not create clusters for too few poems. There is no rule to how specific or how general we should be when creating a list of topics. It truly requires a domain–specific knowledge and highly depends on the application and its purpose. For example, what should be the topic of a poem about a story from World War II? If we work with a dataset of war poems, it may be described by a “World War II” topic but if we utilize a general dataset of poems from a longer period of time, a topic “War” might be specific enough.

The list of topics shown in the Table 5.2 was created thanks to this cooperation.

ID	**Name of the topic**
1	Childhood/Motherhood/Parenthood
2	Home
3	Exotics/Travel
4	Poverty/Oppression
5	Religion/Faith
6	Progress/Technology
7	Politics/Society
8	Work
9	Nature
10	Revolution/Liberation
11	Loneliness/Alienation
12	Death/Dying/Old age
13	Body/Sport
14	Tradition/Folklore
15	Art/Literature, Poetry/Creation
16	War
17	Education/School
18	Entertainment/Leisure
19	Crime/Punishment
20	Love/Romantic relationship
21	City
22	Countryside
23	Nation/Homeland
24	History
25	Self–reflection/Inner life

Table 5.2: List of topics.

The poem annotation process itself involves reading the poem and assigning 25 values – one for each topic – to each poem. These values express the relationship of the poem to each topic. We have designed a 3–value annotation system. The values and their meanings are

- **+1**: a major topic,
- **0**: a minor topic,
- **-1**: a topic is not present.

Due to the fact that annotating the whole dataset would be extremely costly in terms of time, we decided to annotate a random sample of 500 (out of 66 428) poems from the original dataset. This is about 0.75% of the total number of poems.

Data annotation is a difficult task that requires human expertise and careful attention to details. Annotating data comes with several challenges. One of the main challenges in data annotation is ambiguity and subjectivity. Some data points may be difficult to label because the labels may not be well–defined or the data itself is ambiguous. Poets often use figurative language such as metaphors, similes, personification and symbolism. This leaves room for the reader's imagination and subjectivity. Thus, it can easily happen that one text is annotated differently by different annotators. To make the system more robust, instead of having just one annotator, we can ask more experts to annotate the poems. The final set of topic labels for each poem could then be determined by aggregating labels from all annotators. The most common way to do that is using majority vote scheme.

We asked 7 experts from the Institute of Czech Literature of the CAS to annotate the poems. Unfortunately, annotating the data took more time than expected and at the time of our experiments there were only a few dozen annotated data. Therefore, we decided to create the annotations ourselves. The annotated dataset used further in the thesis was created by one annotator – the author of this thesis. Later, just before submitting the thesis, there was more data annotated by the experts, but we did not have time to run and evaluate the experiments again. However, the created source codes allow to run experiments with new data and evaluate the results.

This allows us to shift from unsupervised methods to semi–supervised or fully supervised learning methods. By creating labels for a fraction of the original unlabeled data, we can still take advantage of the benefits of unsupervised learning, such as dealing with large amounts of data and more efficient use of resources and time, but with the added accuracy of supervised learning. The combination of labeled and unlabeled data can provide a more complete representation of the data distribution, leading to better generalisation performance and improved accuracy compared to using only one of the two types of data.

5.2.2 Processing annotations

Figure 5.1 shows a huge class imbalance of the annotations. Even if 500 data points are too few to estimate the distribution of the whole dataset, it is certain that some topics will have much more members than others. It is interesting to note that the distribution of minor topic assignments does not match the distribution of major topic assignments. For most topics, the number of minor assignments is much smaller than the number of major assignments. This also applies to the total number of major and minor assignments. This is a sign that most of the poems belong to one or more major topics and do not contain minor topics. Fortunately, at least one poem was assigned to each topic, which is extremely important from a modeling point of view.

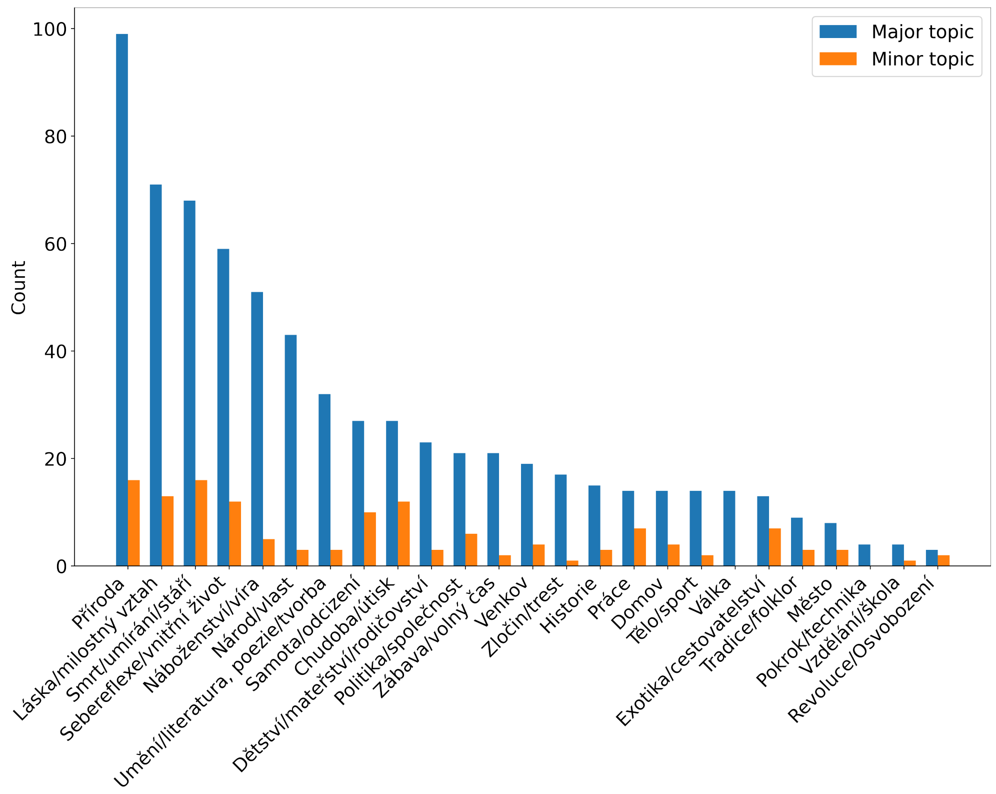
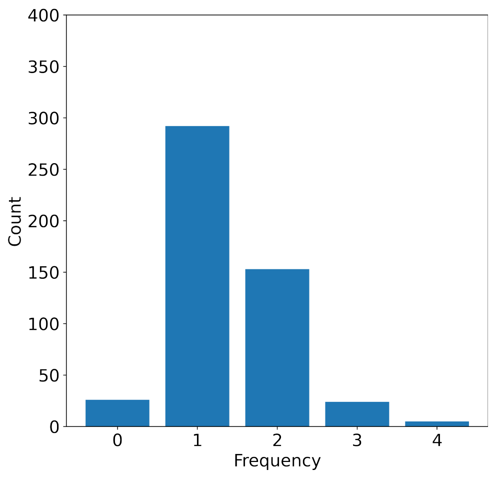
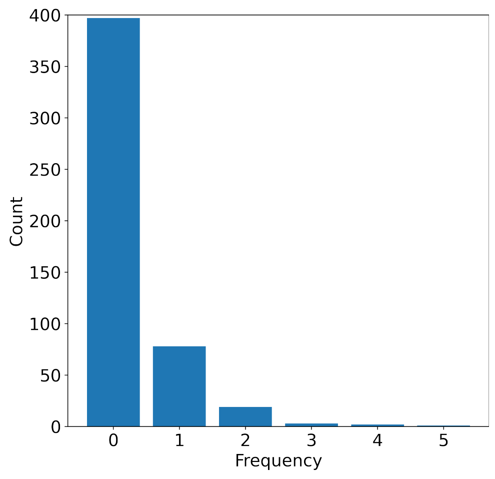


Figure 5.1: Counts of major and minor annotations per topic.

The assignment of a poem to a topic is not exclusive. A poem may belong to none or to multiple major and minor topics. Figure 5.2 shows the distribution of major, minor and combined major and minor topic assignments. Individual histograms show counts of poems to which a certain number of topics have been assigned. It is clear from Figure 5.2a that most of the poems were assigned exactly one major topic, but there are also some poems without a major topic. These are poems that are too abstract or too general, mak-


(a) Major.


(b) Minor.

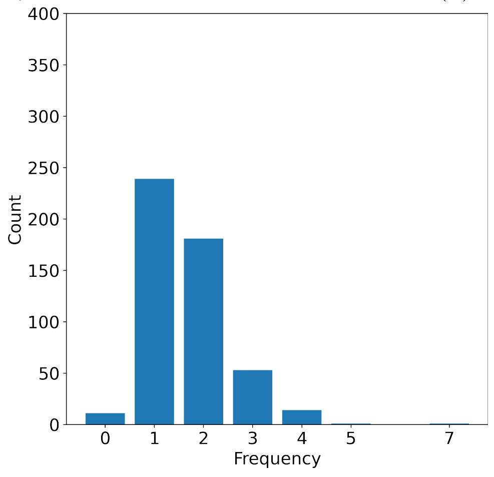

(c) Major + minor.

Figure 5.2: Distribution of topic assignments.

ing it hard to determine what the topic is. The distribution of minor topic assignments in Figure 5.2b reveals that almost 400 poems out of 500 do not contain a minor topic. On the other hand, there are some poems with up to 5 minor topics. Finally, Figure 5.2c shows the combined counts of assigned major and minor topics together. There is a small number of poems that were not assigned to either major or minor topic. This de facto means that they remain unlabeled and cannot be used for supervised learning. On the one hand, we can see this fact as an imperfection of the proposed set of topics. On the other hand, we can see these poems as outliers. From the point of view of topic clarity and compactness, it does not make sense to create separate topics for a very small number of poems.

Since we use topic coherence and topic diversity as metrics for comparing models throughout the work, we decided to compute the values of these metrics for labeled data as well. The coherence of the labeled data reaches the value of 0.4241, the diversity score is at the value of 0.7440. These numbers will help us to compare manually annotated data with classified data obtained using different supervised and unsupervised learning algorithms. We will monitor whether the algorithms achieve the same results as the manual annotations, or whether the performance decreases or even increases.

Chapter 6

Experiments

In this chapter, I will describe the experiments performed using selected unsupervised and supervised methods. This includes data preprocessing procedure, model parameter selection, model training, evaluation and examples of detected topics.

6.1 Data preprocessing

Thanks to the excellent processing and tagging of the data in the Corpus of Czech Verse, the data preprocessing requires only a fraction of the steps of the normal text preprocessing pipeline. Texts of poems are tokenized, tokens are lowercased and lemmatized. This allows us to skip the most complex preprocessing steps and use the lemmas directly. All we have to do is remove the stop words. Similar to the work of Plecháč et al. [6] on the same corpus, we filter out all the parts of speech except for nouns, adjectives and verbs, because these are considered to be the most meaningful parts of speech in terms of conveying the main ideas and concepts of a text. Nouns typically represent objects, people or concepts that are central to a text, while verbs represent actions or states of being. Adjectives describe the properties or characteristics of nouns and can help to provide more detail about them. Other parts of speech, such as prepositions, conjunctions and articles are generally considered to be less informative in terms of the overall meaning of a text.

We also proposed a short additional list of stop words that should be discarded. It consists of verbs that occur frequently in the corpus and have little semantic information for describing topics, or occur in poems regardless of topic. These words are “být” (to be), “mít” (to have), “jít” (to go), “dát” (to give), “moci” (to can). This list was derived from the list of the most frequent lemmas in the corpus. The top 30 most frequent lemmas are depicted in Table 6.1.

There are also other frequent verbs, such as “chtít” (to want) or “stát” (to stand), which may be considered as stop words in a general context but can carry important semantic information and contribute to the meaning of the poem. They capture feelings, desires, attitudes or emotional states. We did not remove nouns with frequent occurrences, such as “srdce” (heart) or “duše” (soul), despite the fact that they appear in a large number of poems, they are fundamental to certain largely represented topics, such as *love*. However, defining a stop words list for poetry can be subjective, depends on the specific corpus and the task.

	Lemma	**Frequency**
1	být	370 467
2	mít	64 415
3	srdce	43 562
4	jít	36 091
5	duše	34 324
6	chtít	33 677
7	oko	29 828
8	svět	29 604
9	láska	29 444
10	dát	27 525
11	bůh	26 782
12	země	26 460
13	ruka	26 083
14	hlava	24 241
15	den	23 920
16	stát	22 923
17	vědět	20 687
18	život	20 229
19	sen	20 172
20	noc	20 048
21	člověk	19 463
22	květ	19 072
23	celý	18 320
24	vidět	17 019
25	čas	16 602
26	velký	16 420
27	slunce	16 406
28	píseň	16 265
29	moci	16 242
30	bílý	16 144

Table 6.1: Top 30 most frequent noun/adjective/verb lemmas in the corpus.

Finally, we removed rare words that occur less than 20 times in the entire corpus. By removing the rare words, we can reduce the dimensionality of the data, remove noise, speed up the processing time and potentially improve the accuracy of our models. This entire preprocessing routine yields 16 786 unique tokens.

For LDA, we also investigated the effect of adding bigrams to the dictionary. Adding bigrams with a frequency greater than 20 to the dictionary increased the size of the dictionary to 17 978 unique tokens.

In the work, we use the expessions "term" and "word" interchangeably to refer to the basic unit of meaning. In the dictionary we used, each term was exactly one word, except for the experiment with bigrams.

The preprocessed data was then vectorized using several methods. Namely, BoW, TF–IDF, word embeddings (FastText [43]) and sentence embeddings (`paraphrase-multilingual-mpnet-base-v2` [51]).

6.2 Evaluation

We used the topic coherence and topic diversity metrics to evaluate the detected topics. Topic coherence expresses how interpretable a resulting topic is. It measures the semantic coherence of a topic by quantifying how closely the top words in a topic are related to each other. Topic diversity measures the diversity of topics by quantifying the uniqueness of the topics. Using topic coherence and topic diversity as the main evaluation metrics for topic modeling ensures that the resulting topics are both interpretable and comprehensive. It can also help in selecting the best topic model for a given application.

The top 10 words from each topic were selected for the coherence and diversity computation. The choice of the number of top words depends on several factors such as the size of the corpus, the number of topics, the length of the documents, and the number of unique tokens in the corpus. However, the interpretation of the topics is also a very important factor. The number 10 was chosen as a compromise between ease of interpretation and comprehensiveness.

In order to evaluate all tested models uniformly, we had to extract a topic representation as a distribution over words for each of them. While LDA and Top2Vec models do this implicitly, text clustering models, supervised and semi–supervised models require an additional processing step to extract the most significant words from each cluster or class of documents. To achieve this, we used the class–based TF–IDF vectorization as described in Chapter 4. We used `gensim`[3] implementation for the C_V coherence computation. It is the implementation of the four stage topic coherence pipeline from the paper [9].

[3] `https://radimrehurek.com/gensim/`

6.3 Unsupervised methods

We picked two best performing models from the related work [16]. Namely, the LDA and the Top2Vec models. We followed the similar process of model training and tried to improve the results using hyper–parameter tuning. In addition to the topic modeling approaches, we implemented a K–means model as a representative of clustering approaches.

6.3.1 LDA

We have trained an LDA topic model using the `LdaMulticore` implementation from the `gensim` library. The model was trained for a number of topics ranging from 5 to 120 with a step size of 5, on both unigram and bigram BoW representations. For each setting, the coherence and diversity of the topics were computed. The number of passes was fixed at 100, which proved to be a reasonable setting to obtain distinctive topics. [6]

Figure 6.1 illustrates the relationship between the number of topics and the topic coherence score of the model. We can see that, except for small deviations, the results obtained using unigrams and bigrams do not differ significantly. The highest coherence values are obtained for the number of topics from about 25 to 45. The best coherence scores are 0.4513 and 0.4512 for the model using unigrams and bigrams, respectively.

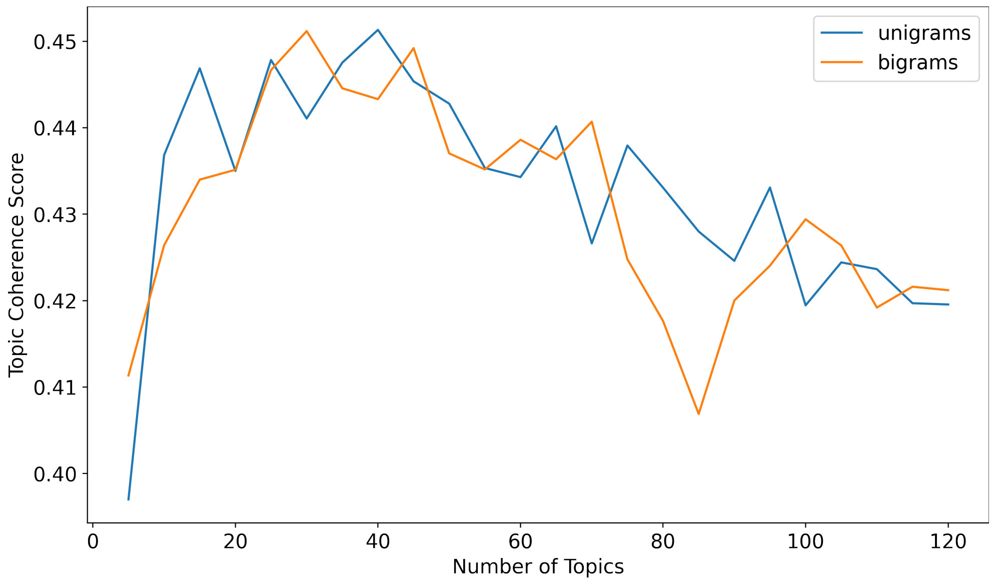


Figure 6.1: Topic coherence score of LDA model depending on number of topics.

Topic diversity decreases with the increasing number of topics up to the value of 80 topics. After that, we observe a subtle increase. This is due to the fact that the model starts to generate more specific and unique topics instead of general topics represented by frequent words.

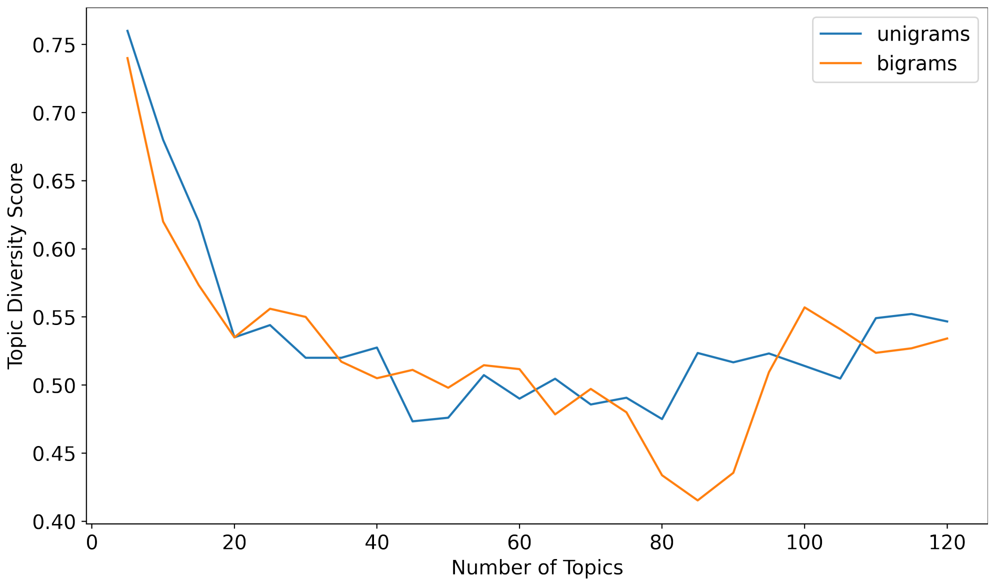
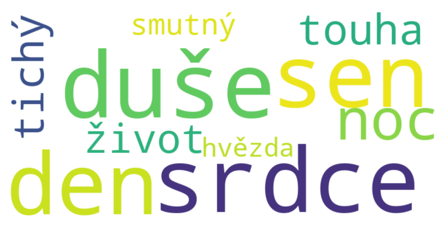
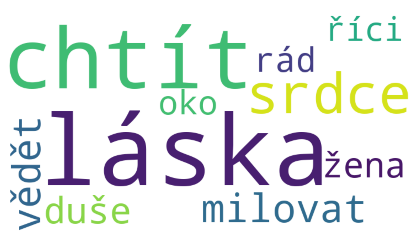


Figure 6.2: Topic diversity score of LDA model depending on number of topics.

In fact, the generative process tends to be more sensitive to frequent words and includes them in many topic representations. The best performing unigram model, the one with 40 topics, includes the word “srdce” (heart) in 5 topics and “duše” (soul) even in 10 topics. Figure 6.3 shows two topics from the model where the words “srdce” and “duše” are represented. They are relevant for both topics, but they are not distinctive words when it comes to comparing and interpreting the topics.



(a) Topic 1.



(b) Topic 11.

Figure 6.3: LDA: top 10 words for selected topics.

6.3.2 Top2Vec

As a second topic model, a Top2Vec model was applied. We used an implementation of the `top2vec`[4] library. Since the input to the model is a continuous text, the preprocessed lemmas were concatenated into a single string. By default, the model's tokenizer removes accent marks. We replaced the default tokenizer with a custom tokenizer that preserves accents, since we are working with the Czech language, which is rich in accents. This step is necessary for the coherence calculation, which works with a dictionary built from the preprocessed data with accents. We also filtered out words with a frequency of less than 20 when training the model, just as we did when building the dictionary.

Top2Vec uses the UMAP method for dimensionality reduction and the HDBSCAN method for clustering of documents. These methods have their own parameters that can be tuned, but we used the default settings for simplicity. We used 6 different embedding models supported by the library to vectorize the data and compared their performance. Finally, the `speed` parameter was set to `deep-learn` to learn the best quality vectors.

The highest coherence was achieved using the `doc2vec` embedding model, as we can see in Figure 6.4. It is interesting that `doc2vec` is the only one of the embedding models used that is not pre–trained. The lower performance of the pre–trained models may be due to the fact that they were trained on texts in current language, while poems use many archaic words and figurative language concepts. We can see that when the same model is used in both the English and the multilingual versions, the multilingual version naturally achieves a better result. This is exactly the case with the `universal-sentence-encoder` and its multilingual version, the `universal-sentence-encoder-multilingual` model. The Czech language did not even appear among the training languages. This indicates that it is possible to achieve better performance by fine–tuning the pre–trained models with specific data in the target language.

Topics detected using the `doc2vec` embeddings achieve not only the highest coherence, but also a very high diversity score of 0.84. Figure 6.4 also shows, that only one model achieved a higher diversity than `doc2vec`. However, this model achieved a low coherence score. The resulting coherence and diversity scores indicate a clear dominance of the `doc2vec` embeddings.

The Top2Vec method does not allow to define a desired number of topics at the output. Instead, it is determined by the underlying clustering algorithm. To some extent, the number of resulting topics can be controlled by setting the UMAP and HDBSCAN parameters, but it is not that straightforward. Figure 6.5 reveals huge differences in the number of topics detected using different embedding models. Intuitively, we would expect that if we divide the corpus into a larger number of topics, very specific topics with specific words will

[4]`https://github.com/ddangelov/Top2Vec`

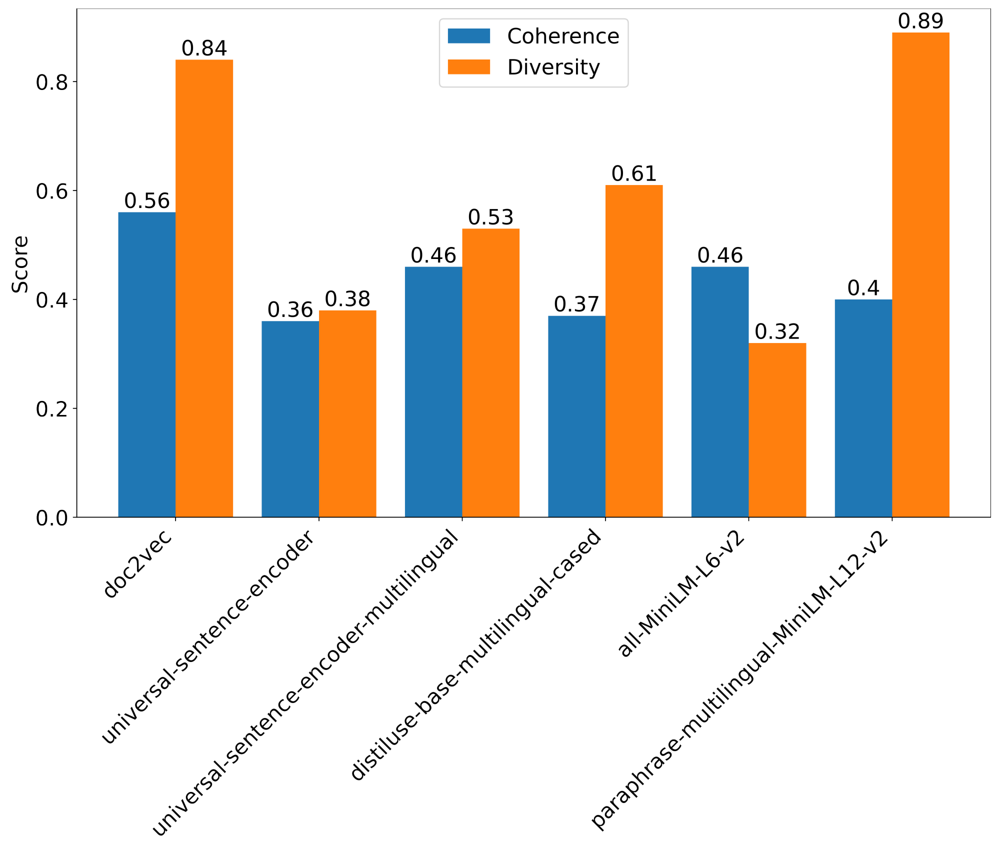


Figure 6.4: Topic coherence score of Top2Vec model depending on used embedding model.

emerge, and thus coherence and diversity should increase. However, in our case, the number of topics does not correlate with coherence or diversity.

The model using the `doc2vec` embeddings detected 168 topics. That may be too much when it comes to interpretation. By closely examining the detected topics, we found that some of the topics are too specific and form a hierarchy. While the Top2Vec model does not offer the possibility to directly define the number of topics at the beginning, it does offer the possibility to reduce the number of topics of the trained model. Hierarchical topic reduction aims to retrieve the most representative topics of the corpus by iteratively merging the smallest topic with the most similar topic until the desired number of topics is reached.

We reduced the original model with 168 topics to a smaller number of topics from 165 to 5 with a step of 5 and monitored the metrics, as shown in Figure 6.6. As the number of topics decreases, we observe an increase in diversity, which even reaches 100% when the number of topics is less than 15. This illustrates the process of merging redundant topics. Even the value of the coherence increases slightly up to a certain point and it does not fall below the

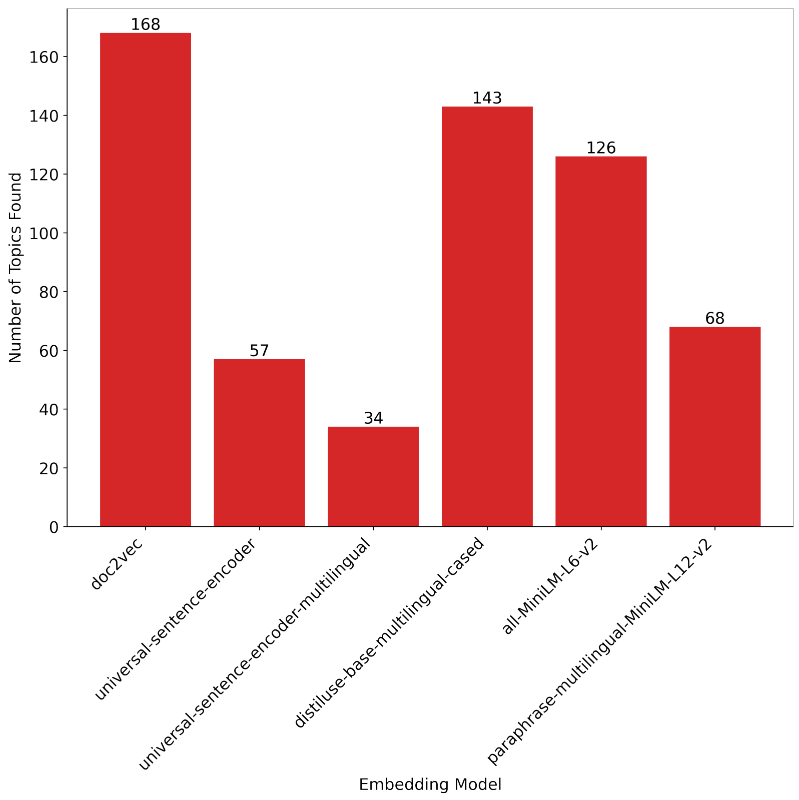


Figure 6.5: Number of topics found by Top2Vec model depending on used embedding model.

level of the original model for a reasonable number of topics. This indicates that the merged topics contain words that occur together in the texts.

Figure 6.7 illustrates an example of merging topics. Topic 132, represented by the words "akiba", "rabbi" and "Izrael" (Israel), can be interpreted as a Judaism/Israel topic. Topic 149 presents a biblical story of Noah's Ark and The Flood using the words "archa" (ark), "Noe" (Noah) and "potopa" (flood). It can be thought of as a sub–topic of topic 132, since it deals with the history of the people of Israel. The combination of the two topics results in a topic that is mainly represented by the words from topic 132 and generalizes the content of the two topics at a higher level of granularity.

This way of merging topics may not be optimal. In fact, it would be possible to iteratively merge the two closest topics, regardless of their size. This is a matter of implementation, but the key is that the joint embedding space allows us to compute such relationships easily.

In addition, we have not noticed the problem with frequent words that LDA suffers from. Representing words in a semantic vector space prevents frequent words from being present in many topic representations because they are equally distant from all documents.

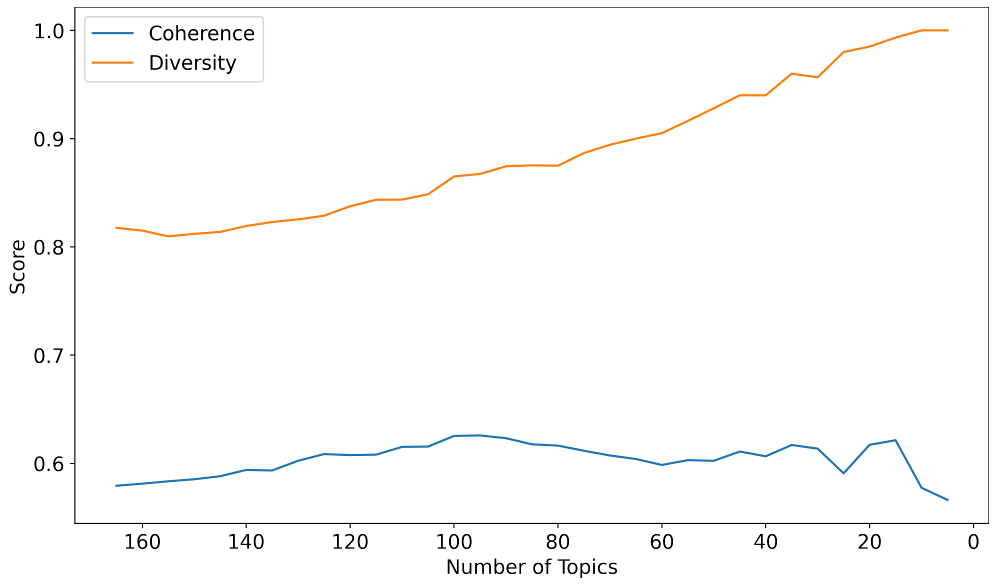
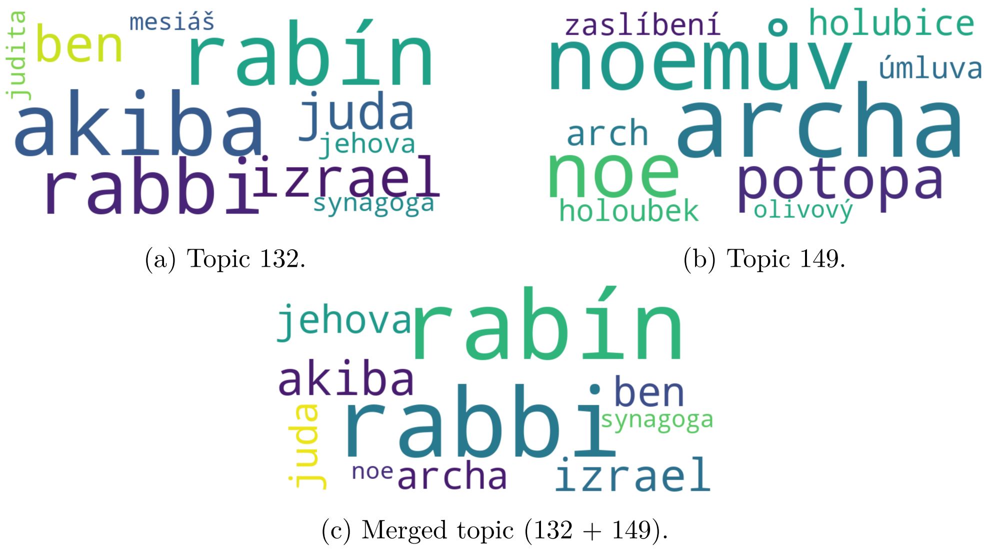


Figure 6.6: Topic coherence and diversity score of the reduced Top2Vec model depending on defined number of topics.



(a) Topic 132.

(b) Topic 149.

(c) Merged topic (132 + 149).

Figure 6.7: Top2Vec: top 10 words for selected topics and their combination.

6.3.3 K–means

The last of the implemented unsupervised algorithms is K–means. K–means is a clustering algorithm that partitions a set of data points into K clusters, where K is a user–defined parameter, based on their similarity. We used the `KMeans` implementation from the `sklearn`[5] library with default parameter settings. We only varied the number of clusters from 5 to 120 with a step of 5. We repeated the process for the 4 computed vectorizations.

Figure 6.8 demonstrates that, except for a small number of clusters, the BoW vectorization achieves the best coherence score. Interestingly, sentence embeddings and TF-IDF vectors perform similarly, while word embeddings have the worst performance.

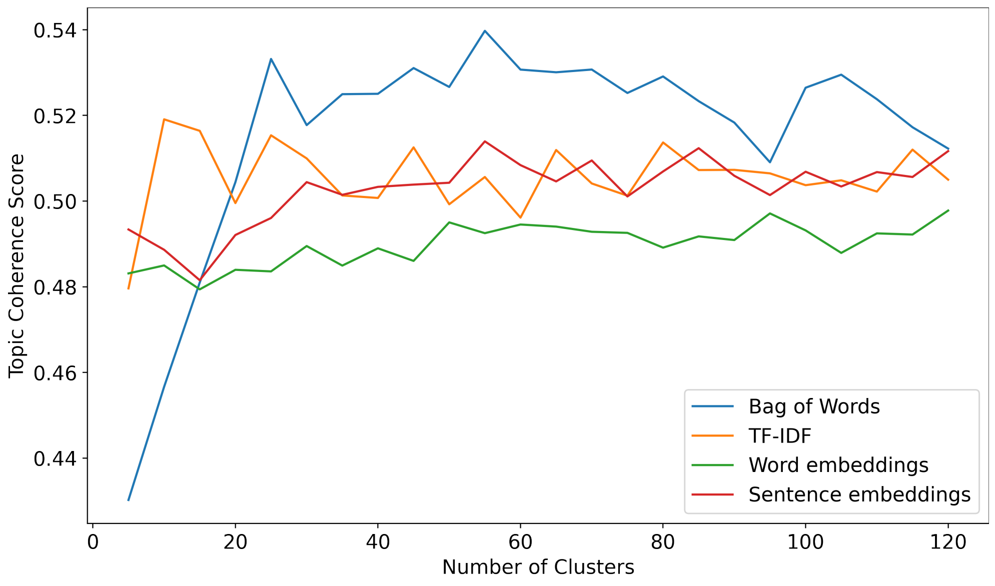


Figure 6.8: Topic coherence score of K–means model depending on the number of clusters.

In the case of diversity, we observe a similar situation. Moreover, compared to coherence, TF–IDF vectorization achieves slightly better results than sentence embeddings.

Although the coherence score is not much worse than in the case of the Top2Vec model, our subjective opinion is that the quality of the topics is lower. Figure 6.10 shows an example of two topics from the K–means model with 55 clusters that achieved the best coherence score. We find them difficult to interpret. Topic 34 is mainly represented by verbs and could therefore be abstract or general. On the other hand, topic 50 is represented almost entirely by nouns, but they are so diverse that it is difficult to find connections between them.

[5]`https://scikit-learn.org/stable/`

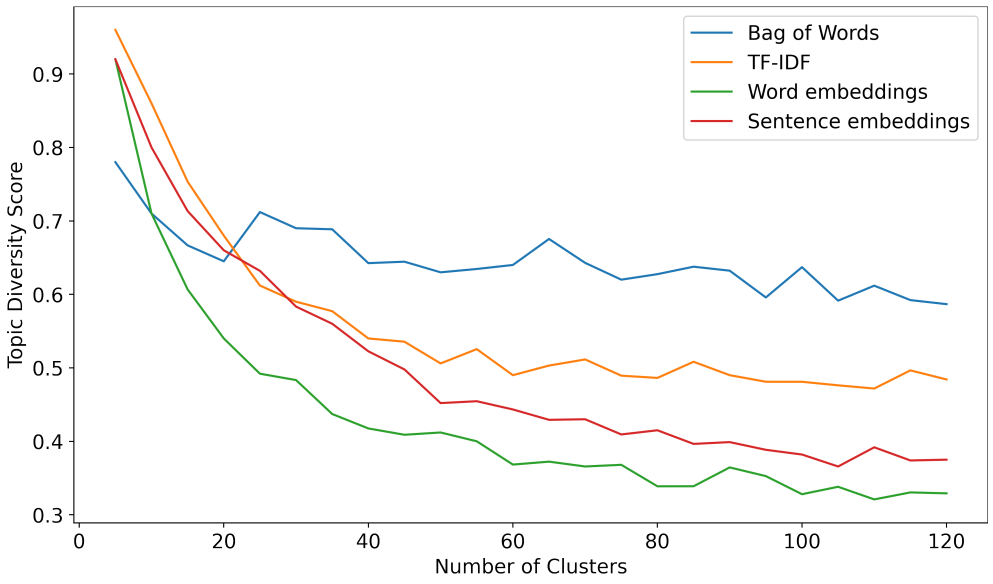
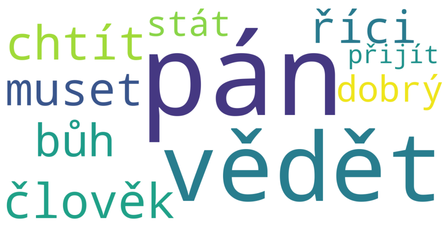
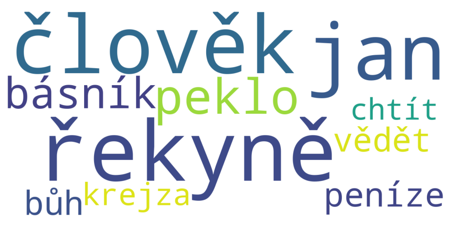


Figure 6.9: Topic diversity score of K–means model depending on the number of clusters.



(a) Topic 34.



(b) Topic 50.

Figure 6.10: K–means: top 10 words for selected topics.

Another problem that the K–means model suffers from is the creation of small clusters. Of the 55 clusters, 29 have less than 10 members. This leads to a huge imbalance. For example, the largest cluster contains 38 070 poems, which is more than 57% of the total size of the corpus. The solution might be to create more clusters and then merge them based on similarity, similar to what Top2Vec can do. Or just consider small clusters as outliers.

6.4 Supervised methods

One of the main advantages of using supervised learning for classification over unsupervised learning is the use of labeled data itself. Thus, we not only know the number of classes in advance, but also their names and descriptions. Labeled data also gives us the ability to evaluate using traditional metrics such as accuracy and F–score. For this part, we decided to use Naive Bayes classifier and Support Vector Machines, because these methods can handle a small amount of training data. Similar to the previous algorithms, we used several text vectorization methods as input for these methods.

We divided the labeled data into training and test sets in a 70:30 ratio. We resampled the training data using `RandomOverSampler` from the `sklearn` library. Since all classes in the training data contained the same number of data points after resampling, we used accuracy for the evaluation. We also used K–fold cross–validation with `K=5`. Figures 6.11 and 6.12 show the average cross–validation accuracy score.

6.4.1 Naive Bayes classifier

We applied two implementations of the NB classifier from the `sklearn` library. `Multinomial NB` models the conditional probability distribution of the class using a multinomial distribution, which is suitable for discrete count data like word counts. `Complement NB` is a variation of Multinomial NB that is designed to address the problem of unbalanced datasets. It applies the same probability calculation as Multinomial NB, but uses a complement function to balance the contribution of each class.

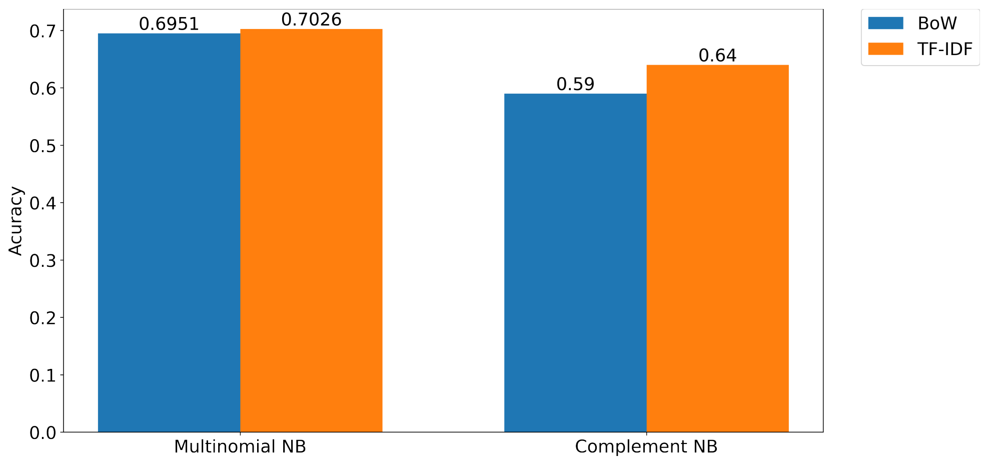


Figure 6.11: Performance of Naive Bayes classifiers.

Since the NB classifier works with discrete data expressing the number or importance of words in documents, we used BoW and TF–IDF text vectorizations. The NB classifier also requires features to be independent of each other. This assumption is not met for dense vectors such as word embeddings and sentence embeddings, where the dimensions of the vector are related and each dimension contributes to the overall meaning of the vector.

Figure 6.11 shows that the Multinomial NB classifier performs better on the training set, reaching slightly over 70% of the average cross–validation accuracy score. Overall, the TF–IDF text vectorization provides better results than the BoW vectorization.

6.4.2 Support Vector Machines

Similar to NB, we also applied two implementations of the SVM classifier – `SVC` and `NuSVC`. The implementations differ in the way they handle misclassified examples. The `SVC` uses the regularization parameter `C`. It controls the trade–off between maximizing the margin of the decision boundary and minimizing the classification error on the training data. It represents the penalty parameter for misclassified samples. On the other hand, the `NuSVC` implementation uses the parameter `nu` to control the number of support vectors. The parameter `nu` is a floating point number between 0 and 1 and represents an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors. Thus, interpreting `nu` may be easier than interpreting `C`.

For each combination of model implementation and data vectorization, we tuned the `C` or `nu` parameter using the grid search method. Figure 6.12 shows the average cross—validation accuracy scores obtained by the `SVC` and `NuSVC` implementations using different text vectorization methods. We were able to achieve a decent training accuracy of over 75%, but the performance differences between different configurations are almost negligible.

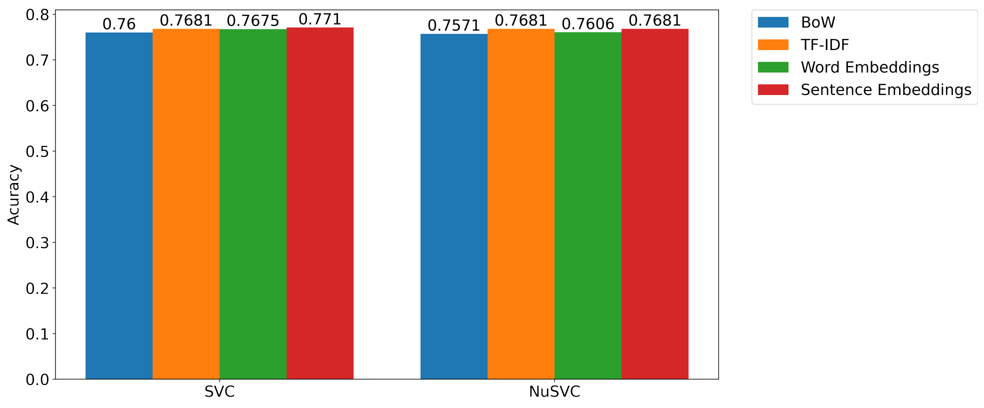
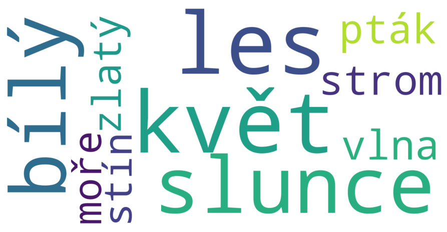
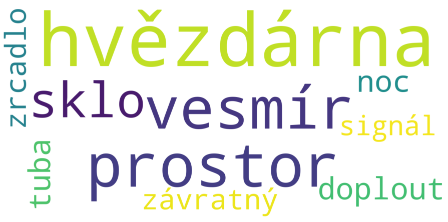


Figure 6.12: Performance of SVM classifiers.



(a) Topic Nature.



(b) Topic Revolution/Liberation.

Figure 6.13: SVM: top 10 words for selected topics.

6.4.3 Model selection and evaluation

It is clear from the previous two sections that the SVM model achieves better results. We chose `SVC` with the use of sentence embeddings as the final model based on the observed results. We trained the model using the entire training set with the tuned value of the parameter `C=8.5`. Unfortunately, the small number of labeled data and the imbalance of the classes affected the performance on the test set. The final model achieved an accuracy of 23.67% and an F1–score of 16.52%.

Nevertheless, we used the trained model to classify the entire dataset. We again used c–TF–IDF to extract the top words for each class. The model obtained a coherence score of 0.4510, which is about the same level as the LDA model. The diversity score reached a value of 0.64.

The interpretability of individual topics depends strongly on the number of data points in the training set. Figure 6.13 shows the top 10 words for selected topics. These are the topics with the highest and lowest number of data points in the training set. While the topic Nature had 69 data points in the training set, the topic Revolution/Liberation had only 2. And their interpretability also corresponds to this fact. From the words shown in Figure 6.13a, it is not difficult to deduce that the topic is Nature. On the other hand, among the words shown in Figure 6.13b, we can hardly find any related to the topic Revolution/Liberation.

6.5 Semi–supervised methods

We have seen an application of several algorithms that were purely unsupervised or supervised. However, the nature of the available data leads us to use semi–supervised methods. In addition to the unsupervised and supervised methods, we wanted to see if using a small amount of labeled data together with a large amount of unlabeled data could improve the performance compared to the methods used before. We chose a semi–supervised modeling method with class–based TF–IDF because of its simple idea and implementation.

6.5.1 Semi–supervised modeling using class–based TF–IDF

The ability of the c–TF–IDF method to extract the top words from the group of documents is useful for feature extraction because it can distill the most significant words from a class document. And this is a property that is well suited for semi–supervised modeling.

The first step is to create a representative vector for each class. This can be done by grouping the labeled data by topic and merging all texts from a class into a single text string. This gives us a class document for each class. Applying the c–TF–IDF vectorization, we get a vector that represents the class content. This approach is easily scalable. The more labeled data we have, the more robust representatives we can create. Figure 6.14 illustrates the representative class vectors in two–dimensional space. The original 8 166 dimensions were reduced using the UMAP method.

There are several interesting relationships that we can observe. For example, topics “Historie” (History), “Politika/společnost” (Politics/society), “Národ/vlast” (Nation/homeland), “Revoluce/osvobození” (Revolution/liberation), “Válka” (War) and “Domov” (Home) are close to each other. Together, these topics capture social events, political life, social sentiments, patriotism and much more. Topics with negative sentiment, including “Smrt/umírání/stáří” (Death/dying/old age), “Chudoba/útisk” (Poverty/oppression) and “Zločin/trest” (Crime/punishment), are also located together. And we could continue with the pairs “Venkov” (Countryside) and “Práce” (Work), “Město” (City) and “Pokrok/technika” (Progress/technology), “Vzdělání/škola” (Education/school) and “Dětství/mateřství/rodičovství” (Childhood/motherhood/parenthood). It is clear that the vector space created in this way is capable of capturing the semantic relations between the topics.

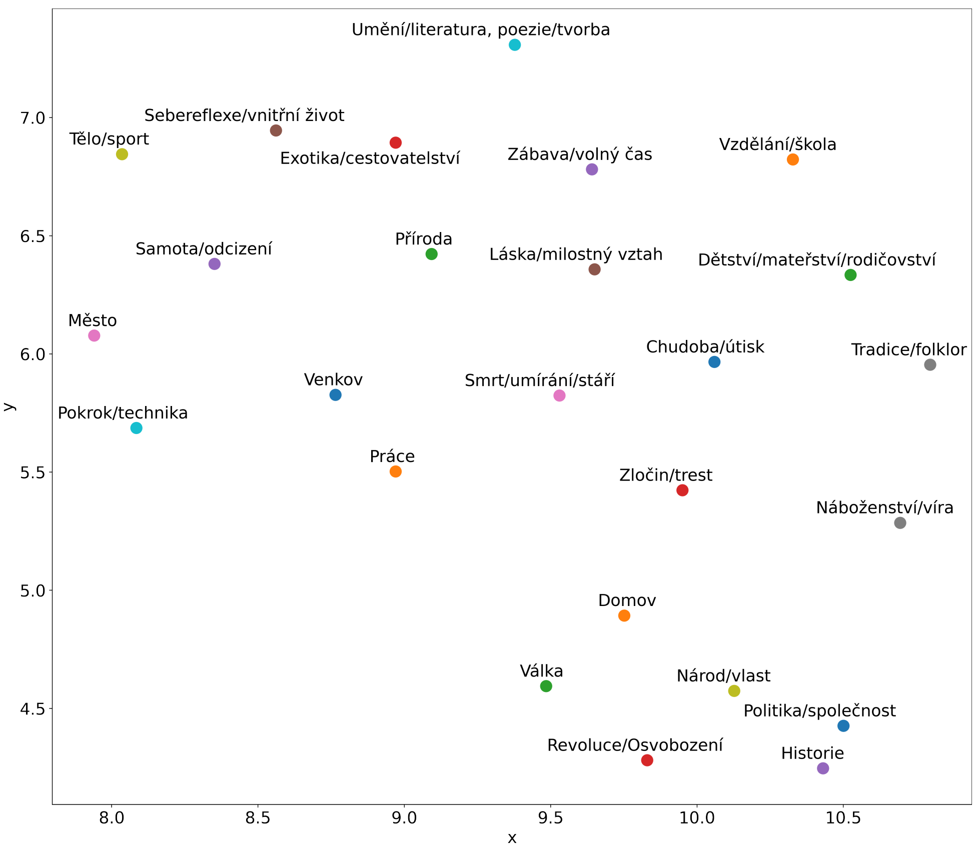


Figure 6.14: Representative class vectors constructed from labeled data using the c–TF–IDF method.

The next step is to create the TF–IDF vector for each unlabeled data point. The resulting vectors were compared with representative class vectors using cosine similarity. The unlabeled data points were then classified using the most similar class vector. It is also possible to assign a fixed number of classes according to the similarity or to set a similarity threshold to assign multiple most likely classes.

The last step is the evaluation. As in the supervised methods, we again merged the texts of the poems according to the computed classification and used c–TF–IDF to extract the top words for each class. First, we used only major topic assignments to create representative class vectors. Then, we used both major and minor topic assignments to represent the classes. The obtained results, shown in Figure 6.15, indicate an insignificant change. Coherence increased slightly, but diversity decreased.

The very high diversity demonstrates that using even a small portion of data to represent the class can be sufficient and separate topics well. Intuitively, by using a larger set of labeled data in the supervised part, we would

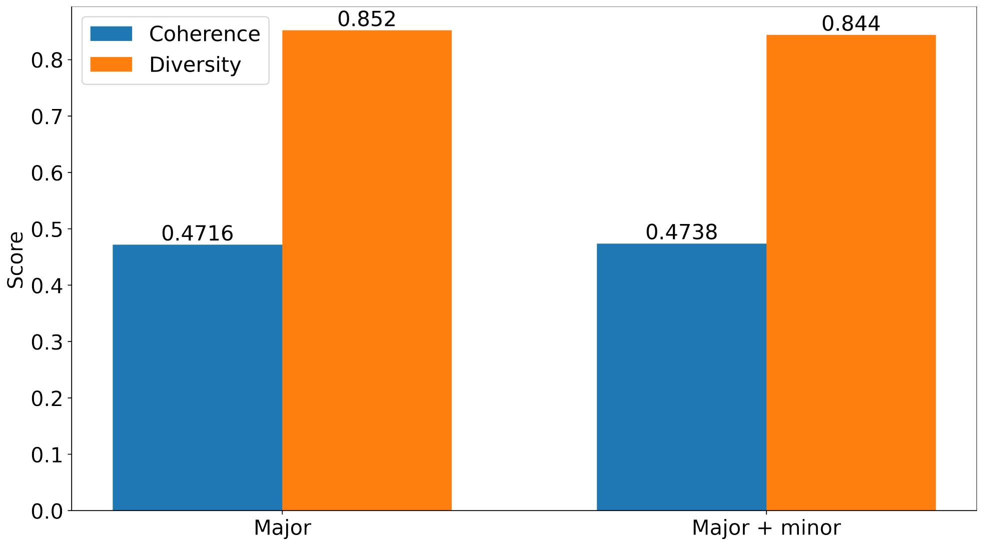


Figure 6.15: Performance of the c–TF–IDF–based model using different data in the supervised part.

be able to build more robust representations and, hopefully, achieve better performance. But the experiment with adding minor topic assignments reminds us that it is not just the quantity of data that matters. First of all, the data must be of good quality. In our case, adding minor topic assignments does not change performance much, but it can be misleading for semantic modeling in general. If we are sure that minor topic assignments contain useful information for our model, we could consider a weighting system. Minor topic assignments could be included in the modeling, but would have a lower weight than major assignments.

Chapter 7

Results and discussion

In this chapter, we present a comparative analysis of the topic detection methods presented in the previous chapter. For each of the methods, we selected the version with the parameterization that gave the best result. The resulting coherence and diversity scores were measured after dividing the entire dataset into separate topics.

Table 7.1 shows the coherence and diversity metrics of the applied ML algorithms. In addition, we also present the coherence and diversity scores for the annotated dataset, which we compare to the classification of the whole dataset based on the applied algorithms.

Method	**Number of topics**	**Coherence**	**Diversity**
Annotated data	25	0.42	0.74
	Unsupervised methods		
LDA (unigrams)	40	0.45	0.53
LDA (bigrams)	30	0.45	0.55
Top2Vec	168	0.56	0.84
Top2Vec (reduced)	35	**0.62**	**0.96**
K–means	55	0.54	0.63
	Supervised methods		
SVM	25	0.45	0.64
	Semi–supervised methods		
c–TF–IDF	25	0.47	0.85

Table 7.1: Performance comparison of applied topic detection methods.

We can see that the annotated dataset has the lowest coherence. This is probably due to the small number of annotated topics and the significant class imbalance. When a class contains a small number of members, noise and diverse vocabulary can be much more pronounced. Even if poems in a class share words that are characteristic of a particular theme, they may be so different or used in different contexts that coherence is compromised. This is due to the definition of coherence as a metric that is typically based on the co–occurrence of words. More data in a topic can suppress this noise. On the other hand, the diversity of the annotated data is relatively high, because in most cases it is easy for the annotator to distinguish the topics from each other.

The best results are found among the unsupervised methods, but they vary significantly between individual algorithms. LDA achieves significantly lower performance compared to others. As we have shown in Figure 6.3, it suffers from frequent words because it tends to include them in a large number of topics. This leads to inconsistencies, lack of clarity and redundancy.

Slightly better results than LDA were obtained with the K–means algorithm. While a higher number of topics improved the metrics, the resulting clustering was enormously unbalanced. In fact, it resulted in the creation of a few large clusters and many small clusters. The largest of these clusters contained more than 57% of all data, and the top 5 clusters contained more than 96% of all data. This setting creates inconsistencies and is a huge problem when it comes to interpretation. This can also be seen in Table 7.2.

The situation is completely different with the Top2Vec algorithm. Initially, the algorithm detected 168 topics. However, unlike K–means, they were very specific and in most cases easy to interpret, which fully correlates with the resulting values of the metrics. However, since 168 topics is a too fine grouping, we created a new grouping by merging similar topics, which gave the best results globally. By merging very similar and redundant topics, the coherence value increased slightly and the diversity approached 100%.

The performance of the SVM, the supervised algorithm, was strongly influenced by the amount of training data. Although the overall results are not overwhelming, this is mainly due to the fact that the distribution of the annotated data was imbalanced. The algorithm had problems learning the representation for classes with a small number of training samples, less than about 20. However, classes with a higher number of training samples were represented by words that formed coherent sets and their interpretation was relatively easy. In our case, about 60 samples were sufficient.

The semi–supervised algorithm based on c–TF–IDF performed better using the annotated data, particularly in terms of diversity. It handled underrepresented classes much better than the SVM. Even for topics with less than 10 annotated samples, this algorithm was able to create a compact and understandable representation. Another good sign is that the results of the overall classification improved the results of the annotated data itself. This is a confirmation of the hypothesis that unlabeled data can help improve the model.

In the appendix, we present the top 10 words for each topic in the form of word clouds for selected methods. We have chosen the Top2Vec algorithm as a representative of the unsupervised algorithms, since it was the one that achieved the best results. Topic representations can be found in Appendix B. Appendix C contains topic representations of the annotated data itself. Finally, Appendix D is dedicated to the semi–supervised c–TF–IDF method. The performance of this method was the best when using annotated data.

It is worth noting that unsupervised algorithms detected many topics that can be clearly mapped to the topics defined for the annotated dataset. As an example, we present the results of topic detection for *Jaroslavu Vrchlickému* by Eduard Albert using different methods. The following is the text of the poem as written.

“Tvá loď jde po vysokém moři,
v ně brázdu jako stříbro reje,
svou přídu v modré vlny noří
a bok svůj pěnné do peřeje.

Tvá lana sviští, plachty duní
a třepe vlajka. V noční chvíli
zříš magický svit mořských tůní,
a ve snu, Albatros jak pílí.

Já samotním, jsem na ostrově,
ohýnek topím, rybku lově
zasedám na břeh za večera.

Dým v kotoučích se modrých krade,
kdes písklo ptáče, ještě mladé,
tma na mne hrozí z pološera.” [52]

This poem depicts scenes of a sea voyage, with an emphasis on ships, waves, sails and night on the island, with an air of nostalgia and mystery. Table 7.2 shows the detected topics, respectively their representation by the top 10 words. Unsupervised methods lack topic names because they require an additional step to interpret topics from top words.

The detected top words are very similar for all methods. However, we can note a few details. Besides obvious words like “loď” (ship) and “vlna” (wave), different methods detected some specific words. For example, LDA captures the words “skála” (rock), “hora” (mountain) and “slunce” (sun). We can see this as a form of generalization. These words are not directly related to the topic of the sea and sailing, but belong to the more general topic of travel.

On the other hand, the topics detected by Top2Vec are very specific. Among the top 10 words, we cannot find a single one that is not related to the topic of the sea and sailing. Not even in the reduced model. This is an example of how this model achieves a high degree of coherence and diversity.

Using the K–means model, this poem fell into the largest cluster. Its representation is very incoherent.

Both the SVM and c–TF–IDF algorithms assigned the topic Exotics / Travel to this poem. In fact, they use the same 9 out of 10 words in the representation. It is interesting that the word "vlak" (train) is included. This again is a sign that these models can generalize, i.e. represent several very specific sub–topics under one topic.

Method	**Topic name**	**Top 10 words of the topic**
	Unsupervised methods	
LDA (unigrams)	-	moře, vlna, loď, břeh, voda, plout, bouře, mořský, skála, veslo
LDA (bigrams)	-	moře, vlna, břeh, loď, voda, hvězda, hora, slunce, zlatý, plout
Top2Vec	-	loď, plavec, plachta, přístav, koráb, člun, příď, paluba, stožár, stěžeň
Top2Vec (reduced)	-	loď, plavec, člun, plachta, přístav, koráb, příď, plout, stožár, vlna
K–means	-	člověk, rád, nebe, vědět, svět, čas, píseň, život, rok, bůh
	Supervised methods	
SVM	Exotics/Travel	moře, loď, vlna, břeh, plout, voda, vlak, loďka, noc, dálka
	Semi–supervised methods	
c–TF–IDF	Exotics/Travel	loď, vlna, moře, břeh, plout, koráb, dálka, noc, vlak, loďka

Table 7.2: Topic detection for the poem *Jaroslavu Vrchlickému* using different methods.

As in Tesaříková's work [16] on the same dataset, the Top2Vec model performed best. In addition, we were able to significantly increase the coherence score by tokenizing with an emphasis on accents. We also added a new metric to the evaluation – topic diversity – which makes it easier to compare and evaluate different topic models.

7.1 Future work

There are several directions for future research and development in the area of topic modeling and classification of poems in the Corpus of Czech Verse.

The availability of annotated data plays a crucial role in the performance of supervised methods. In the current work, the supervised methods showed relatively lower performance compared to the unsupervised and semi–supervised methods, which could be attributed to the limited availability of annotated data. Therefore, a possible future direction would be to create more annotated data, which could help improve the performance of supervised methods and provide a more robust topic models.

Additionally, we only performed a single–label classification, where each poem was assigned to a single topic. However, poems can belong to multiple topics simultaneously. Therefore, an interesting direction for future work would be to explore multi–label classification techniques where poems can be assigned to multiple topics simultaneously. This could provide a more nuanced and realistic approach to topic modeling.

Future work could also involve implementing or proposing new semi–supervised methods that utilize both labeled and unlabeled data. The current work explored one semi–supervised method, but there are several other techniques that could be explored. This solution can still be interesting especially if a larger annotated dataset becomes available. As we have seen, semi–supervised algorithms can challenge supervised ones in this setting.

Conclusion

The main objective of this thesis was to explore the possibilities of topic detection in a digitized poetry collection, the Corpus of Czech Verse, using unsupervised and supervised learning and to compare the two approaches.

In the first part of the thesis we investigated several unsupervised and supervised machine learning algorithms for topic detection, with a focus on visualization and evaluation possibilities. Across different parts of the thesis, we focused on related work of topic detection in poetic texts. We also extensively explored the data from the Corpus of Czech Verse in order to preprocess the data as efficiently as possible using richly tagged poetic texts.

As part of the data preparation, we also created an annotated dataset intended for training and evaluation of supervised methods. We manually assigned a random sample of poems from the dataset to the classes we defined in cooperation with the Institute of Czech Literature of the CAS.

In the next part, we selected several methods and applied them to the data. The results were evaluated and visualized, providing insights into the identified topics. In the last chapter, we summarize and compare the applied methods, pointing out not only their performance metrics, but also their characteristic features, strengths and weaknesses using sample texts from the dataset.

The results of the unsupervised and supervised methods differed significantly in favor of the unsupervised methods. The best performing method overall was the Top2Vec method. Not only did it achieve the results that strongly dominated the other methods, but also the topics detected were specific and subjectively consistent and distinct. In addition, this method offers a simple way of post–processing. It makes it possible to merge similar topics and thus control the granularity of the resulting partitioning and the number of topics.

We have successfully completed all the assigned tasks, contributing to the advancement of topic identification in Czech poetry collections. In addition, we also explored and applied a semi–supervised approach, which extended our research to include both labeled and unlabeled data together to improve the accuracy, robustness and effectiveness. As a result, we were able to extract even more from a small amount of labeled data than with the use of supervised methods and achieve better results.

Overall, this thesis represents a step towards the identification of topics in Czech poetry collections. The findings and insights gained from this project can contribute to building a reliable topic model that can be used for automatically sorting a corpus of documents into classes based on underlying topics.

Bibliography

[1] RAGHOZAR, Arya and Diana INKPEN. Semantics and homothetic clustering of Hafez poetry. In *Proceedings of the 3rd Joint SIGHUM workshop on computational linguistics for cultural heritage, social sciences, humanities and literature*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 82–90. [cit. 2023-02-13]. Available from: doi:10.18653/v1/W19-2511

[2] RAGHOZAR, Arya and Diana INKPEN. Bilingual chronological classification of Hafez's poems. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*. San Diego, California, USA: Association for Computational Linguistics, June 2016, pp. 54–62. [cit. 2023-02-13]. Available from: doi:10.18653/v1/W16-0207

[3] RUMA, Jannatul F., et al. A deep learning classification model for Persian Hafez poetry based on the poet's era. *Decision Analytics Journal*. September 2022, **4**. [cit. 2023-02-13]. ISSN 2772-6622. Available from: `https://doi.org/10.1016/j.dajour.2022.100111`

[4] LOU, Andres, Diana INKPEN and Chris TANASESCU. Multilabel subject–based classification of poetry. *The Florida AI Research Society*. April 2015. [cit. 2023-02-14]. Available from: `https://www.site.uottawa.ca/~diana/publications/flairs_2015_paper.pdf`

[5] NAVARRO–COLORADO, Borja. On poetic topic modeling: extracting themes and motifs from a corpus of Spanish poetry. *Frontiers in Digital Humanities*, June 2018, **5**(15). [cit. 2023-02-14]. ISSN 2297–2668. Available from: doi:10.3389/fdigh.2018.00015

[6] PLECHÁČ, Petr, and Thomas N. HAIDER. Mapping topic evolution across poetic traditions. *arXiv preprint arXiv:2006.15732*. June 2020. [cit. 2023-02-14]. Available from: `https://arxiv.org/pdf/2006.15732.pdf`

[7] ABDELRAZEK, Aly, et al. Topic modeling algorithms and applications: A survey. *Information Systems.* February 2023, **112**(C), pp. 102–131. [cit. 2023-02-14]. ISSN 0306–4379. Available from: `https://doi.org/10.1016/j.is.2022.102131`

[8] ZHAO, He, et al. Topic modelling meets deep neural networks: A survey. *arXiv preprint arXiv:2103.00498.* February 2021. [cit. 2023-02-15]. Available from: `https://arxiv.org/pdf/2103.00498.pdf`

[9] RÖDER, Michael, Andreas BOTH and Alexander HINNEBURG. Exploring the Space of Topic Coherence Measures. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* [online]. New York, NY, USA: ACM, February 2015, pp. 399–408 [cit. 2023-02-20]. ISBN 9781450333177. Available from: doi:10.1145/2684822.2685324

[10] SYED, Shaheen and Marco SPRUIT. Full-Text or Abstract? Examining Topic Coherence Scores Using Latent Dirichlet Allocation. In: *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* [online]. IEEE, October 2017, pp. 165–174 [cit. 2023-02-21]. ISBN 978-1-5090-5004-8. Available from: doi:10.1109/DSAA.2017.61

[11] DIENG, Adji B., Francisco J. R. RUIZ and David M. BLEI. Topic Modeling in Embedding Spaces. *Transactions of the Association for Computational Linguistics* [online]. January 2020, **8**, pp. 439–453 [cit. 2023-02-21]. ISSN 2307-387X. Available from: doi:10.1162/tacl_a_00325

[12] PALACIO–NIÑO, Julio–Omar and Fernando BERZAL. Evaluation metrics for unsupervised learning algorithms. *arXiv preprint arXiv:1905.05667.* May 2019. [cit. 2023-03-10]. Available from: `https://arxiv.org/pdf/1905.05667.pdf`

[13] BICA, John. Tweet Topic Modeling: Visualizing Topic Modeling Results with Plotly. *Medium.com* [online]. January 2021 [cit. 2023-02-26]. Available from: `https://pub.towardsai.net/tweet-topic-modeling-part-4-visualizing-topic-modeling-results-with-plotly-66d5dbaaf7fb`

[14] PRABHAKARAN, Selva. Topic modeling visualization — How to present the results of LDA models? *Machinelearningplus.com* [online]. Bengaluru, December 2018 [cit. 2023-02-26]. Available from: `https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models`

[15] LIU, Qian, et al. Data Analysis and Visualization of Newspaper Articles on Thirdhand Smoke: A Topic Modeling Approach. *JMIR Medical Informatics* [online]. January 2019, **7**(1) [cit. 2023-03-10]. ISSN 2291-9694. Available from: doi:10.2196/12414

[16] TESAŘÍKOVÁ, Anna. *Topic Modeling for Corpus of Czech Verse* [online]. Praha, 2022. [cit. 2023-02-14]. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, Department of Applied Mathematics. Supervisor Ing. Magda Friedjungová, Ph.D. Available from: `https://dspace.cvut.cz/handle/10467/101806`

[17] LANDAUER, Thomas K, Peter W. FOLTZ and Darrell LAHAM. An introduction to latent semantic analysis. *Discourse Processes* [online]. January 1998, **25**(2-3), pp. 259–284 [cit. 2023-03-30]. ISSN 0163-853X. Available from: doi:10.1080/01638539809545028

[18] SHI, Tian, et al. Short–Text Topic Modeling via Non-negative Matrix Factorization Enriched with Local Word-Context Correlations. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18* [online]. New York, New York, USA: ACM Press, April 2018, pp. 1105–1114 [cit. 2023-03-30]. ISBN 9781450356398. Available from: doi:10.1145/3178876.3186009

[19] BRANTS, Thorsten, Francine CHEN and Ioannis TSOCHANTARIDIS. Topic–based document segmentation with probabilistic latent semantic analysis. In: *Proceedings of the eleventh international conference on Information and knowledge management* [online]. New York, NY, USA: ACM, November 2002, pp. 211–218 [cit. 2023-03-30]. ISBN 1581134924. Available from: doi:10.1145/584792.584829

[20] BLEI, David M., Andrew Y. NG and Michael I. JORDAN. Latent Dirichlet Allocation. *The Journal of Machine Learning Research.* March 2003, **3**, pp 993—1022. [cit. 2023-02-14]. ISSN 1532-4435. Available from: `https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf`

[21] DEVLIN, Jacob, et al. BERT: Pre–training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, **1**, pp. 4171–4186. [cit. 2023-02-15]. Available from: doi:10.18653/v1/N19-1423

[22] ANGELOV, Dimitar. Top2Vec: Distributed Representations of Topics. *arXiv preprint arXiv:2008.09470.* August 2020. [cit. 2023-02-15]. Available from: `https://arxiv.org/pdf/2008.09470.pdf`

[23] GROOTENDORST, Maarten. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794.* March 2022. [cit. 2023-02-15]. Available from: `https://arxiv.org/pdf/2203.05794.pdf`

[24] LI, Qian, et al. A Survey on Text Classification: From Traditional to Deep Learning. *ACM Transactions on Intelligent Systems and Technology* [online]. April 2022, **13**(2), pp. 1–41 [cit. 2023-02-21]. ISSN 2157-6904. Available from: doi:10.1145/3495162

[25] KOWSARI, Kamran, et al. Text Classification Algorithms: A Survey. *Information* [online]. April 2019, **10**(4) [cit. 2023-02-21]. ISSN 2078-2489. Available from: doi:10.3390/info10040150

[26] MINAEE, Shervin, et al. Deep Learning–based Text Classification. *ACM Computing Surveys* [online]. April 2022, **54**(3), pp. 1–40 [cit. 2023-02-21]. ISSN 0360-0300. Available from: doi:10.1145/3439726

[27] RENNIE, Jason. 20 Newsgroups. *Qwone.com* [online]. 2007 [cit. 2023-02-26]. Available from: `http://www.qwone.com/~jason/20Newsgroups`

[28] GULLI, Antonio. AG's corpus of news articles. *di.unipi.it* [online]. 2005 [cit. 2023-02-26]. Available from: `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

[29] LEHMANN, Jens, et al. DBpedia — A large–scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* [online]. 2015, **6**(2), pp. 167–195 [cit. 2023-02-26]. ISSN 15700844. Available from: doi:10.3233/SW-140134

[30] WANG, Alex, et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* [online]. Brussels, Belgium: Association for Computational Linguistics, November 2018, pp. 353—355 [cit. 2023-02-26]. ISSN 15700844. Available from: doi:10.3233/SW-140134

[31] WANG, Alex, et al. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* [online]. Red Hook, New York, USA: Curran Associates Inc., December 2019, (294), pp. 3266–3280 [cit. 2023-02-26]. Available from: `https://dl.acm.org/doi/pdf/10.5555/3454287.3454581`

[32] MIKOLOV Tomáš, et al. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations (ICLR)* [online]. Scottsdale, Arizona, USA: ICLR, May 2013 [cit. 2023-03-10]. Available from: `https://arxiv.org/pdf/1301.3781.pdf`

[33] VASWANI, Ashish, et al. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* [online]. Red Hook, New York, USA: Curran Associates Inc., 2017, pp. 6000–6010 [cit. 2023-03-11]. ISBN 9781510860964. Available from: `https://dl.acm.org/doi/pdf/10.5555/3295222.3295349`

[34] YANG, Zhilin, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* [online]. June 2019 [cit. 2023-03-11]. Available from: `https://arxiv.org/pdf/1906.08237.pdf`

[35] BROWN, Tom B., et al. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* [online]. Red Hook, New York, USA: Curran Associates Inc., 2020, (159) [cit. 2023-03-11]. ISBN 9781713829546. Available from: `https://dl.acm.org/doi/pdf/10.5555/3495724.3495883`

[36] LEPIKHIN, Dmitry, et al. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *arXiv preprint arXiv:2006.16668* [online]. July 2020 [cit 2023-03-11]. Available from: `https://arxiv.org/pdf/2006.16668.pdf`

[37] FEDUS, William, Barret ZOPH and Noam SHAZEER. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint arXiv:2101.03961* [online]. January 2021 [cit. 2023-03-11]. Available from: `https://arxiv.org/pdf/2101.03961.pdf`

[38] WAHBA, Yasmen, Nazim MADHAVJI and John STEINBACHER. A Comparison of SVM Against Pre–trained Language Models (PLMs) for Text Classification Tasks. In: NICOSIA, Giuseppe, et al., ed. *Machine Learning, Optimization, and Data Science* [online]. Cham: Springer Nature Switzerland, 2023, pp. 304–313 [cit. 2023-02-30]. Lecture Notes in Computer Science. ISBN 978-3-031-25890-9. Available from: doi:10.1007/978-3-031-25891-6_23

[39] XUL, Haoyin, et. al. When are Deep Networks really better than Decision Forests at small sample sizes, and how? *arXiv preprint arXiv:2108.13637* [online]. September 2021 [cit. 2023-03-11]. Available from: `https://arxiv.org/pdf/2108.13637.pdf`

[40] STRAKOVÁ, Jana, Milan STRAKA and Jan HAJIČ. Open–Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* [online]. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 13–18 [cit. 2023-04-30]. Available from: doi:10.3115/v1/P14-5003

[41] BENGIO, Yoshua, et al. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research* [online]. March 2003, **3**, pp. 1137–1155 [cit. 2023-03-15]. ISSN 1532-4435. Available from: `https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf`

[42] PENNINGTON, Jeffrey, Richard SOCHER and Christopher MANNING. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* [online]. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543 [cit. 2023-03-15]. Available from: doi:10.3115/v1/D14-1162

[43] BOJANOWSKI, Piotr, et al. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* [online]. 2017, **5**, pp. 135–146 [cit. 2023-03-15]. ISSN 2307-387X. Available from: doi:10.1162/tacl_a_00051

[44] MARON, M. E. Automatic Indexing: An Experimental Inquiry. *Journal of the ACM* [online]. 1961, **8**(3), pp. 404–417 [cit. 2023-03-15]. ISSN 0004-5411. Available from: doi:10.1145/321075.321084

[45] BOSER, Bernhard E., Isabelle M. GUYON and Vladimir N. VAPNIK. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* [online]. New York, New York, USA: ACM, 1992, pp. 144–152 [cit. 2023-03-15]. ISBN 089791497X. Available from: doi:10.1145/130385.130401

[46] GOWDA, Harsha S., et al. Semi-supervised Text Categorization Using Recursive K-means Clustering. *arXiv preprint arXiv:1706.07913* [online]. August 2017 [cit. 2023-04-02]. Available from: `https://arxiv.org/pdf/1706.07913.pdf`

[47] GROOTENDORST, Maarten. Creating a class-based TF-IDF with Scikit-Learn: Extracting informative words per class. *Medium.com* [online]. October 2020 [cit. 2023-04-04]. Available from: `https://medium.com/towards-data-science/creating-a-class-based-tf-idf-with-scikit-learn-caea7b15b858`

[48] PLECHÁČ, Petr and Robert KOLLÁR. The Corpus of Czech Verse. *Studia Metrica et Poetica* [online]. June 2015, **2**(1), pp. 107–118. [cit. 2022-03-26]. Available from: doi:10.12697/smp.2015.2.1.05

[49] CVRČEK, Václav and Olga RICHTEROVÁ. Původní morfologická značka (tag před SYN2020). *Wiki Českého národního korpusu* [online]. Příručka ČNK, [2022] [cit. 2023-04-04]. Available from: `https://wiki.korpus.cz/doku.php/seznamy:tagy_archiv`

[50] X-SAMPA. *Wikipedia: The free encyclopedia* [online]. [n.d.] [cit. 2023-03-21]. Available from: `https://en.wikipedia.org/wiki/X-SAMPA`

[51] REIMERS, Nils and Iryna GUREVYCH. Sentence-BERT: Sentence Embeddings using Siamese BERT–Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP–IJCNLP)* [online]. Hong Kong, China: Association for Computational Linguistics, November 2019, pp. 3982—3992. [cit. 2022-03-26]. Available from: doi:10.18653/v1/D19-1410

[52] ALBERT, Eduard. *Na zemi a na nebi: Básně*. Praha: František Šimáček, 1900.

Appendix **A**

Acronyms

BERT Bidirectional Encoder Representations from Transformers

BoW Bag of Words

c–TF–IDF Class–based Term Frequency – Inverse Document Frequency

DL Deep Learning

HDBSCAN Hierarchical Density–Based Spatial Clustering of Applications with Noise

IoT Internet of Things

JSON JavaScript Object Notation

ML Machine learning

NB Naive Bayes

NLP Natural language processing

PLM Pre–trained Language Model

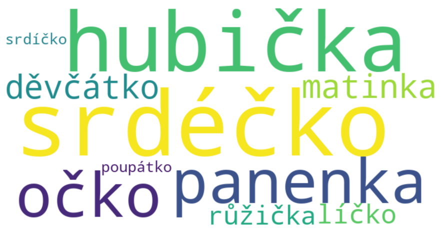
SVM Support Vector Machine

TF–IDF Term Frequency – Inverse Document Frequency

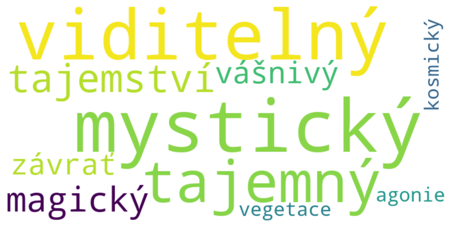
UMAP Uniform Manifold Approximation and Projection

Appendix B

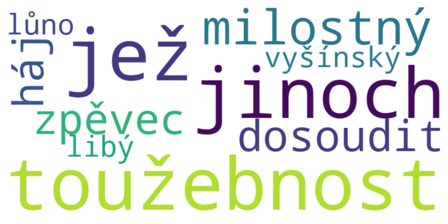
Top2Vec (reduced): topic representations



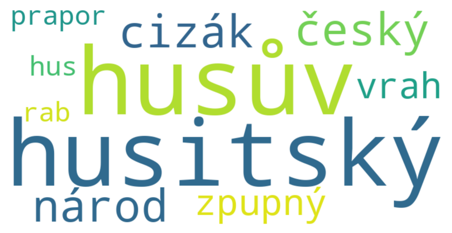
(a) Topic 1.



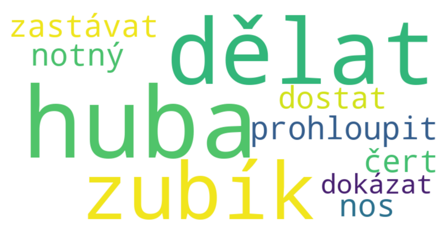
(b) Topic 2.



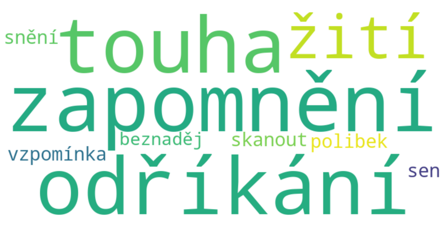
(a) Topic 3.



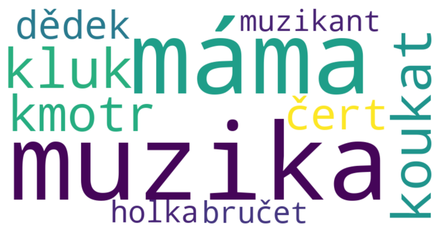
(b) Topic 4.



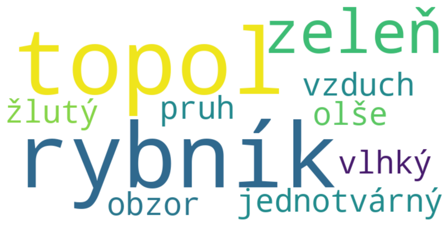
(a) Topic 5.



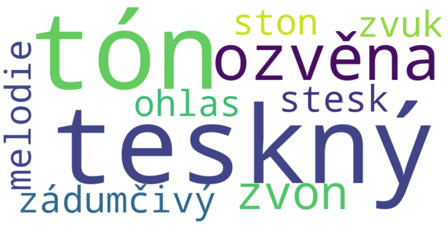
(b) Topic 6.



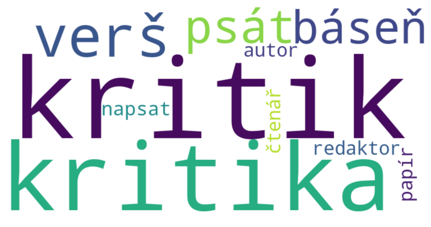
(a) Topic 7.



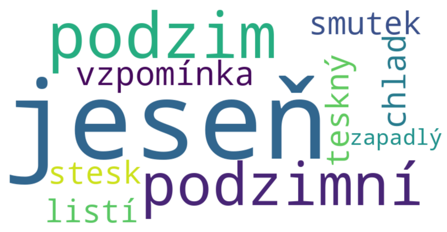
(b) Topic 8.



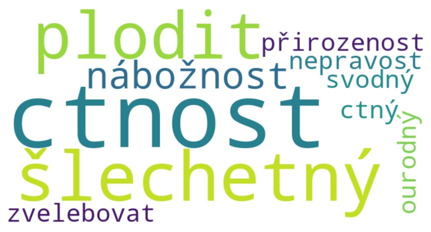
(a) Topic 9.



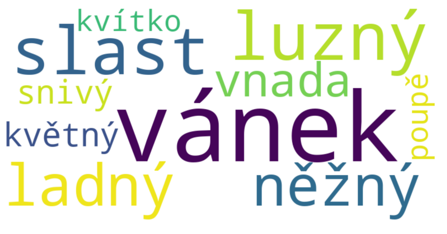
(b) Topic 10.



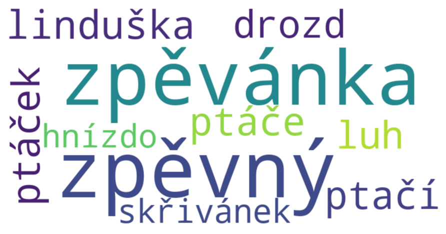
(a) Topic 11.



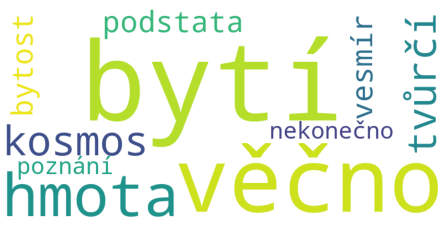
(b) Topic 12.



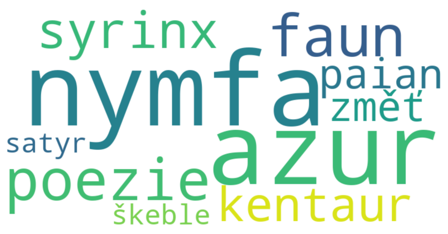
(a) Topic 13.



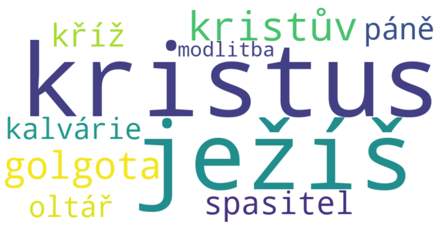
(b) Topic 14.



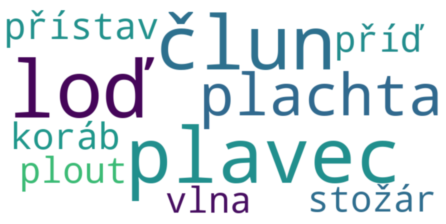
(a) Topic 15.



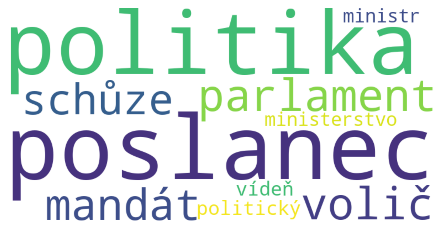
(b) Topic 16.



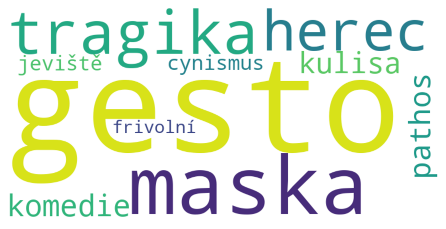
(a) Topic 17.



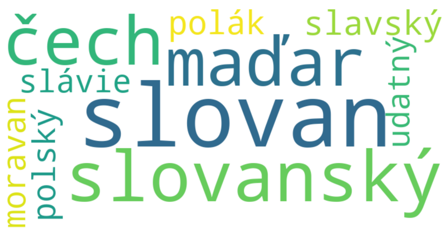
(b) Topic 18.



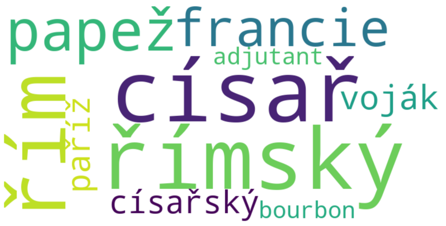
(a) Topic 19.



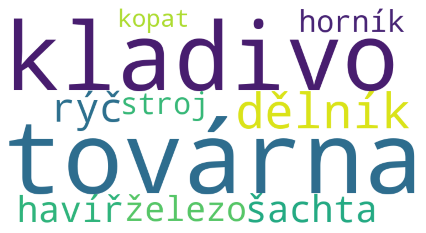
(b) Topic 20.



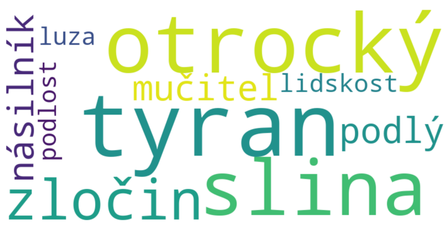
(a) Topic 21.



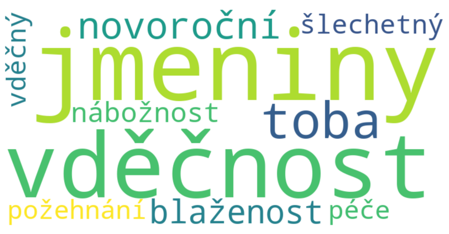
(b) Topic 22.



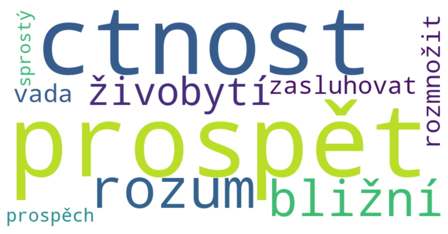
(a) Topic 23.



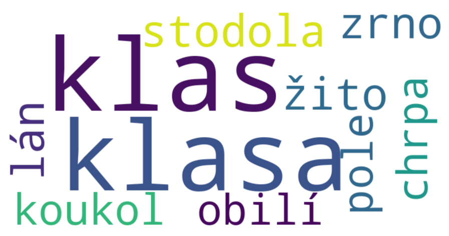
(b) Topic 24.



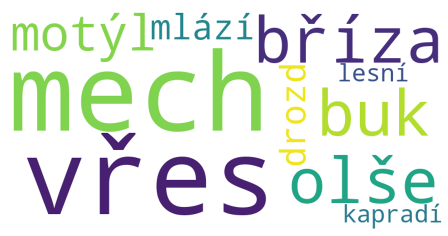
(a) Topic 25.



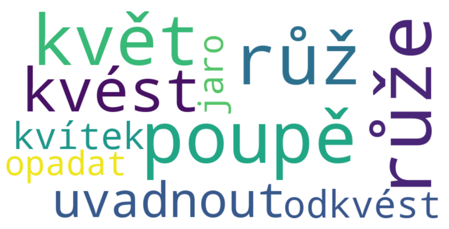
(b) Topic 26.



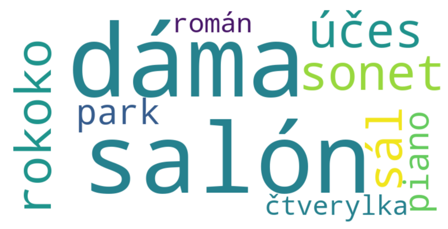
(a) Topic 27.



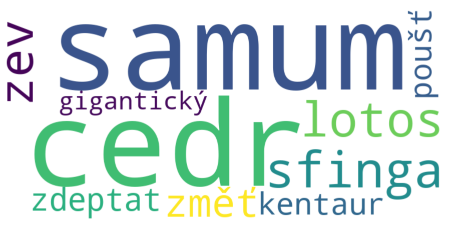
(b) Topic 28.



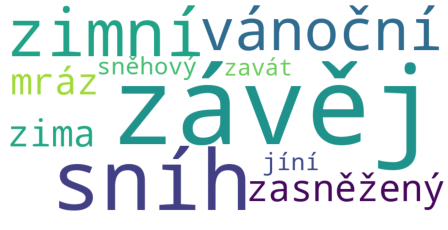
(a) Topic 29.



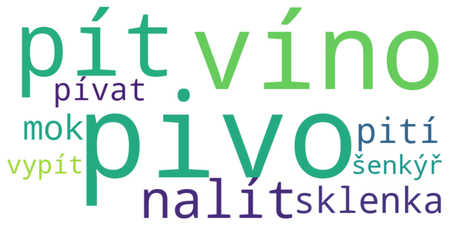
(b) Topic 30.



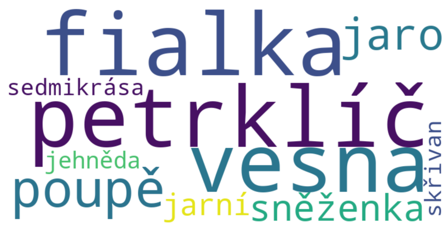
(a) Topic 31.



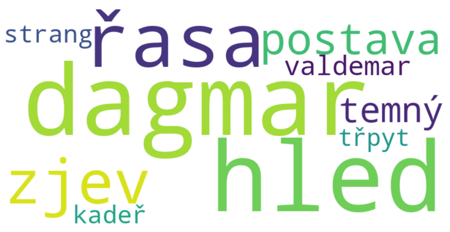
(b) Topic 32.



(a) Topic 33.



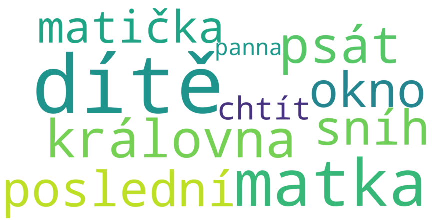
(b) Topic 34.


strang
řasa
postava
valdemar
dagmar
temný
třpyt
zjev
hled
kadeř

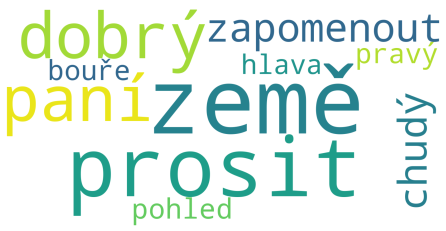
(a) Topic 35.

Appendix C

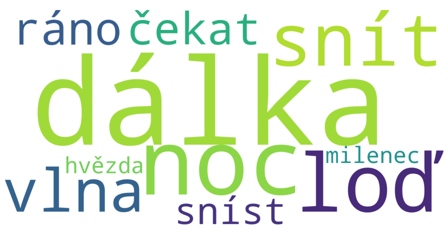
Annotated data: topic representations



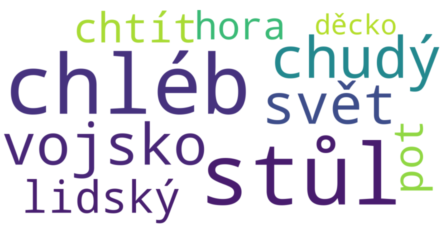
(a) Topic "Childhood/Motherhood/-Parenthood".



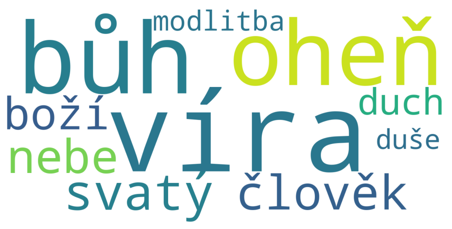
(b) Topic "Home".



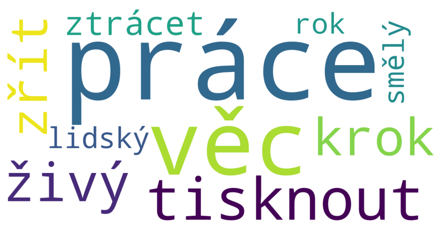
(a) Topic "Exotics/Travel".



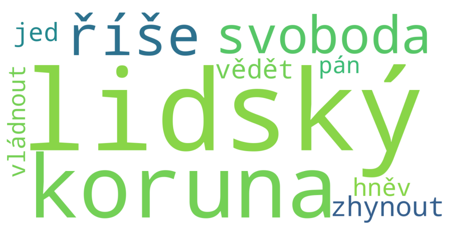
(b) Topic "Poverty/Opression".



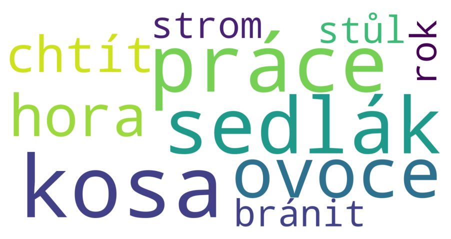
(a) Topic “Religion/Faith”.



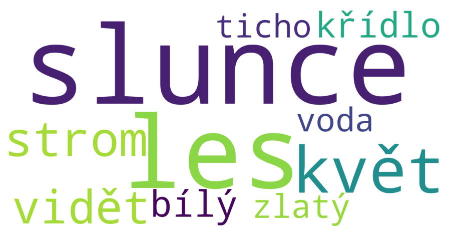
(b) Topic “Progress/Technology”.



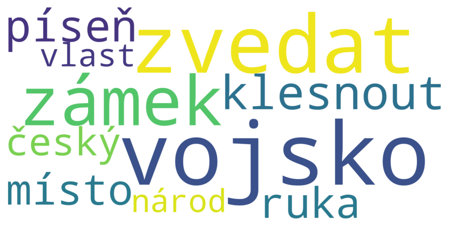
(a) Topic “Politics/Society”.



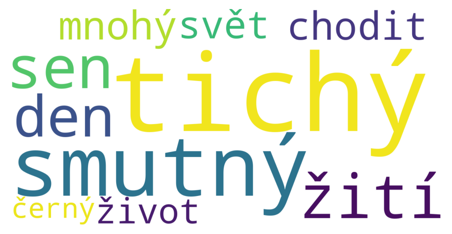
(b) Topic “Work”.



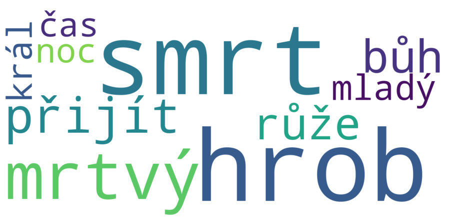
(a) Topic “Nature”.



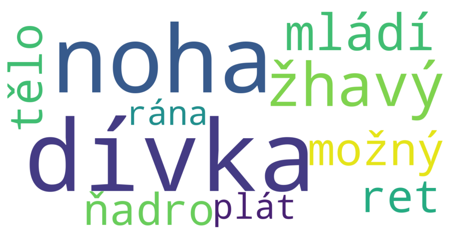
(b) Topic “Revolution/Liberation”.



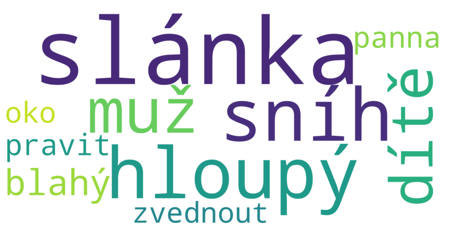
(a) Topic “Loneliness/Alienation”.



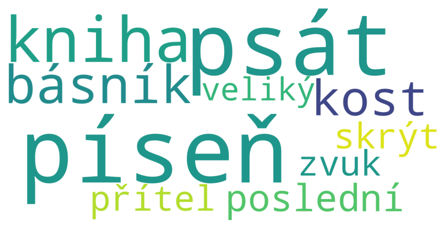
(b) Topic “Death/Dying/Old age”.



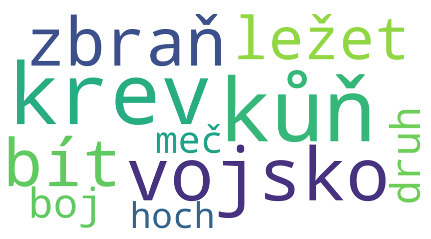
(a) Topic “Body/Sport”.



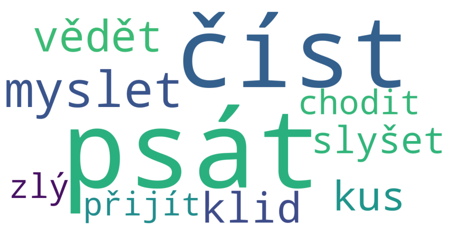
(b) Topic “Tradition/Folklore”.



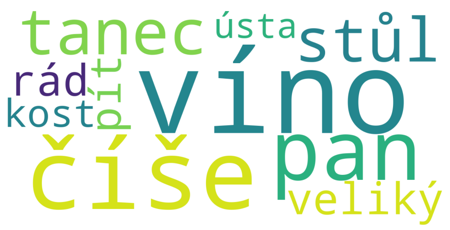
(a) Topic "Art/Literature, Poetry/Creation".



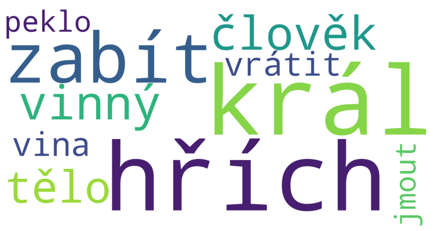
(b) Topic "War".



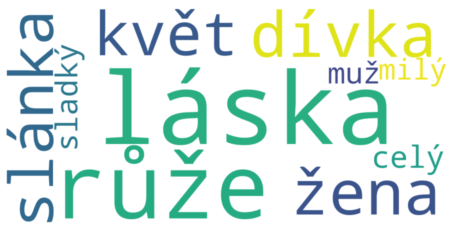
(a) Topic "Education/School".



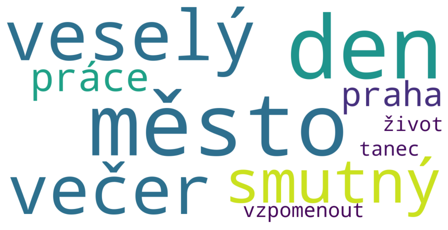
(b) Topic "Entertainment/Leisure".



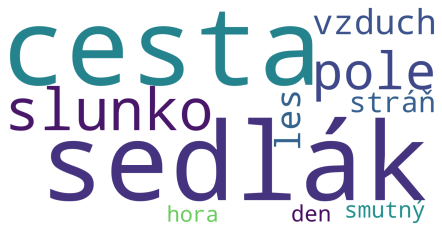
(a) Topic "Crime/Punishment".



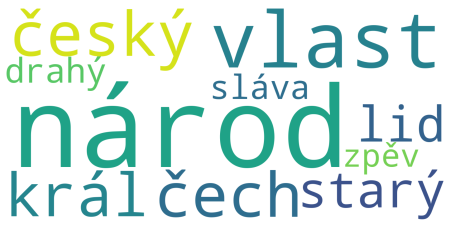
(b) Topic "Love/Romantic relationship".



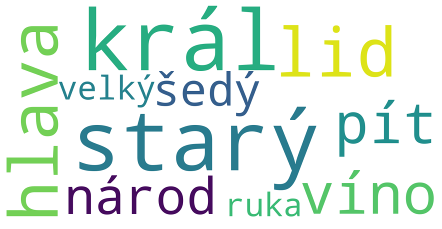
(a) Topic "City".



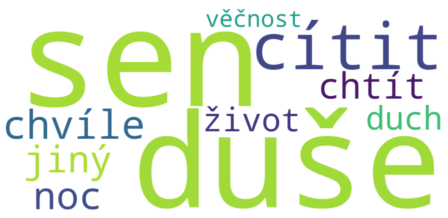
(b) Topic "Countryside".



(a) Topic "Nation/Homeland".



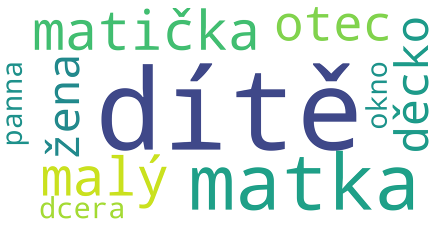
(b) Topic "History".



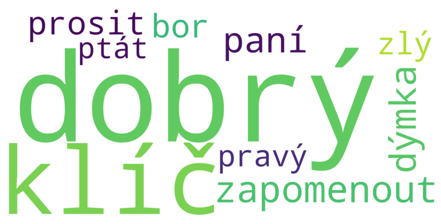
(a) Topic "Self–reflection/Inner life".

Appendix D

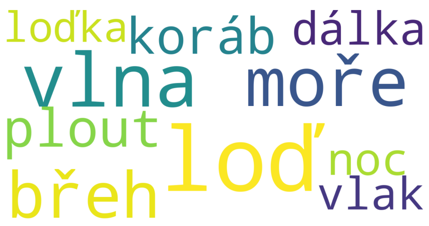
c–TF–IDF: topic representations



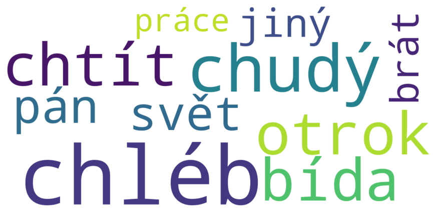
(a) Topic "Childhood/Motherhood/-Parenthood".



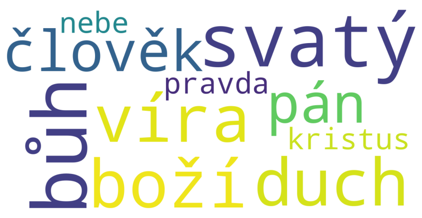
(b) Topic "Home".



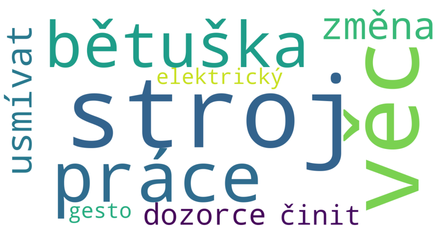
(a) Topic "Exotics/Travel".



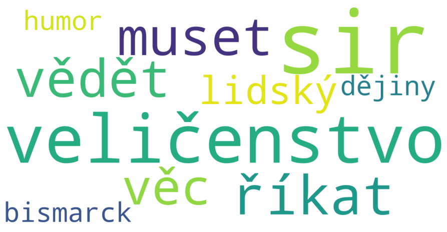
(b) Topic "Poverty/Opression".



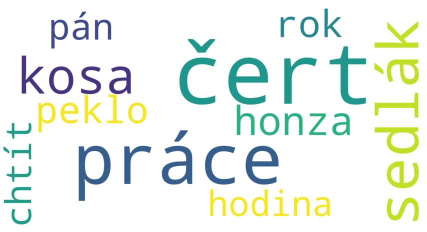
(a) Topic "Religion/Faith".



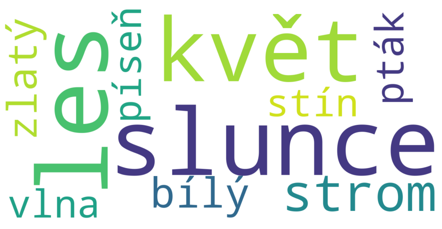
(b) Topic "Progress/Technology".



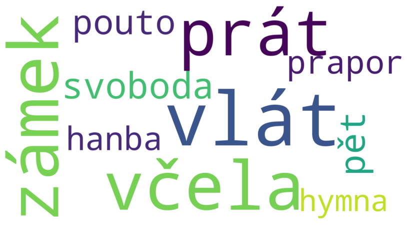
(a) Topic "Politics/Society".



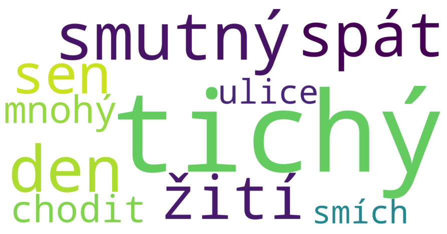
(b) Topic "Work".



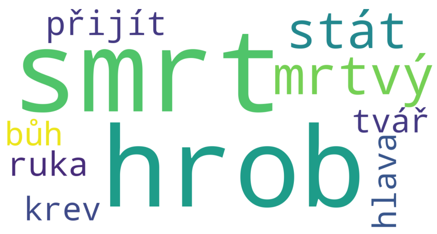
(a) Topic "Nature".



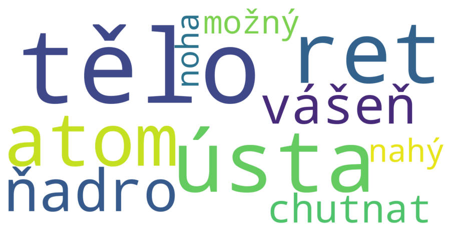
(b) Topic "Revolution/Liberation".



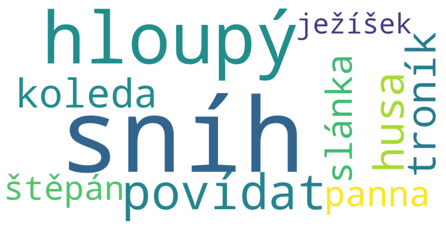
(a) Topic "Loneliness/Alienation".



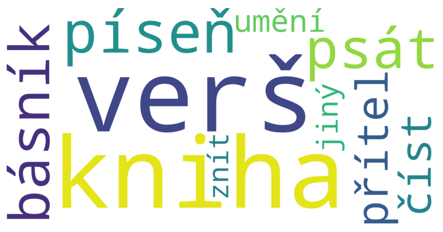
(b) Topic "Death/Dying/Old age".



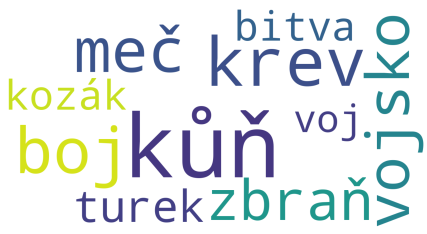
(a) Topic "Body/Sport".



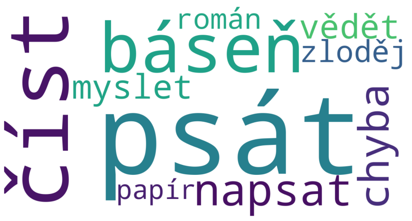
(b) Topic "Tradition/Folklore".



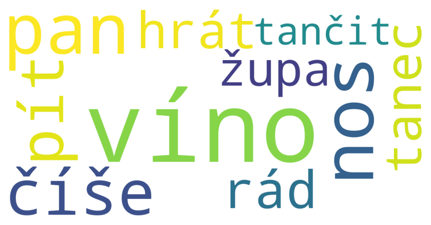
(a) Topic "Art/Literature, Poetry/Creation".



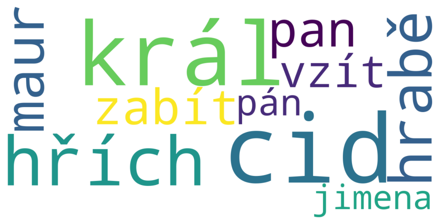
(b) Topic "War".



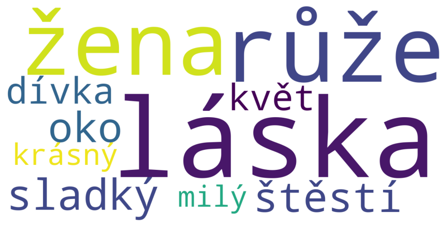
(a) Topic "Education/School".



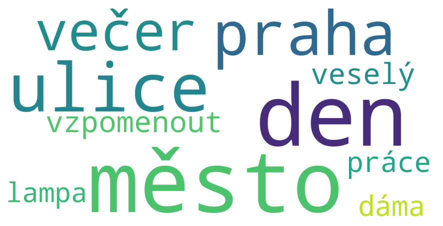
(b) Topic "Entertainment/Leisure".



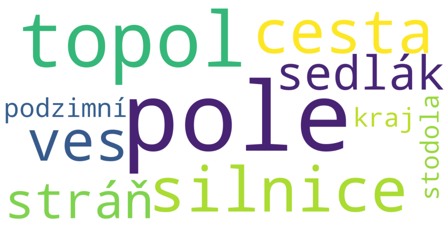
(a) Topic "Crime/Punishment".



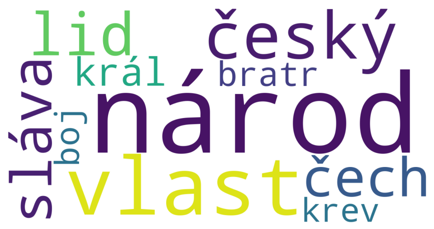
(b) Topic "Love/Romantic relationship".



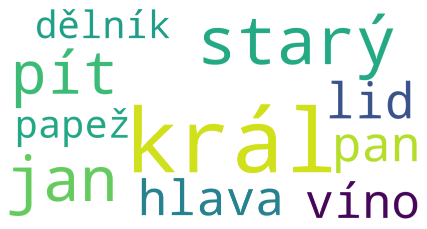
(a) Topic "City".



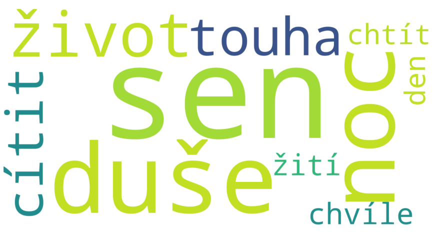
(b) Topic "Countryside".



(a) Topic "Nation/Homeland".



(b) Topic "History".



(a) Topic “Self–reflection/Inner life”.

Appendix E

Contents of the attached repository

```
README.md ........................ description of the repository contents
├── data .............. the directory with annotations exported from the DB
├── notebooks ......... the directory with source codes in Jupyter notebooks
├── results .......................... the directory with the selected results
├── scripts ........... the directory with source codes of the training scripts
└── text ....................................... the thesis text directory
    ├── thesis.pdf .......................... the thesis text in PDF format
    └── src ................................ the thesis text in LaTeX format
```