Czech
Technical University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Extracting Keywords from Textual Data Clusters

Bachelor's Thesis of
Diana Korladinova

Supervisor: **Ing. Jan Drchal, Ph.D.**
Study program: **Open Informatics**
Specialization: **Artificial Intelligence and Computer Science**
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Korladinova Diana**	Personal ID number:	**491899**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Extracting Keywords from Textual Data Clusters

Bachelor’s thesis title in Czech:

Extrakce klíčových slov z textových shluků

Guidelines:

The first part of the task is to experiment with different clustering methods aimed at topic detection in Twitter data. The second, more important part, deals with keyword/keyphrase extraction to provide a concise description of the clusters as well as texts in general.
1) Review methods of text clustering in NLP. Also explore methods of keyword (keyphrase) extraction including abstractive approaches (generating keywords not necessarily present in the source text).
2) Implement (or reuse existing implementations) of selected clustering and keyword extraction methods.
3) Perform experiments on clustering Twitter data supplied by the supervisor. Evaluate keyword extraction methods on both tweet clusters and the news dataset corpus (provided by the supervisor as well).

Bibliography / sources:

[1] Ibrahim, R., S. Zeebaree, and K. Jacksi. "Survey on semantic similarity based on document clustering." Adv. sci. technol. eng. syst. j 4.5 (2019): 115-122.
[2] Firoozeh, Nazanin, et al. "Keyword extraction: Issues and methods." Natural Language Engineering 26.3 (2020): 259-291.
[3] Papagiannopoulou, Eirini, and Grigorios Tsoumakas. "A review of keyphrase extraction." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10.2 (2020): e1339.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Drchal, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Drchal, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.
Date of assignment receipt Student’s signature

Acknowledgements

I'm incredibly grateful to my thesis supervisor Ing. Jan Drchal, Ph.D. for his valuable advice, timely help and weekly meetings through the course of writing this thesis.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” is also gratefully acknowledged.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 26. May 2023

Abstract

With the tremendous amount of data people have at their disposal, automatic topic recognition can make analyzing it exceedingly faster. Much effort has been put into developing topic detection tools in past years, but most of those solutions have been constructed with the English language in mind. Using two Czech corpora, one scraped from Twitter and the other consisting of annotated news articles, this thesis delves into topic detection with Latent Dirichlet Allocation and a combination of K-Means and keyword extraction and generation. Over 219K tweets are clustered, and while the Latent Dirichlet Allocation results in a list of topics directly, I experiment with various statistical and graph-based methods to extract keywords from the K-Means clusters and use the annotated data set to train a model in generating keyphrases not present in the text. Keyword extraction produced valuable results regardless of the approach, and generating keyphrases with a trained model combined with diverse beam search showed great potential as well. These findings have a broad spectrum of applications, from automatic text tagging to document summarization.

Keywords: Topic Detection, Textual Clustering, Keyword Extraction, Keyphrase Generation

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

Vzhledem k nepřebernému množství dat, které mají lidé k dispozici, může automatické rozpoznávání témat nesmírně zrychlit jejich analýzu. V uplynulých letech bylo vynaloženo velké úsilí na vývoj nástrojů pro detekci témat ale většina těchto řešení byla konstruována pouze pro angličtinu. V této práci byly použity dva české datasety, jeden extrahován z Twitteru a druhý sestávající se z anotovaných novinových článků. Tato práce se zabývá detekcí témat pomocí Latentní Dirichletovy alokace a kombinací K-Means s metodami extrakce a generování klíčových slov. Více než 219 tisíc tweetů bylo seskupeno podle témat a zatímco při použití Latentní Dirichletovy alokace byl vrácen rovnou seznam témat, tak při použití K-means bylo nutné klíčová slova získat jiným způsobem. Experimentovala jsem s různými statistickými a grafovými metodami pro extrakci klíčových slov a také jsem pomocí anotovaných dat natrénovala model pro generování klíčových frází, která nemusí být přítomná v textu. Extrakce klíčových slov přinesla cenné výsledky bez ohledu na zvolený přístup a generování klíčových frází pomocí natrénovaného modelu v kombinaci s paprskovým prohledáváním také ukázalo velký potenciál. Tato zjištění mají široké spektrum aplikací od automatického označování textu po sumarizace.

Klíčová slova: detekce témat, shlukování textů, extrakce klíčových slov, generování klíčových frází

Contents

Figures

Tables

Chapter 1
Introduction

Since one of the predominant humanly comprehensible forms of communication is textual, a large part of the available information is generated and stored in this manner. Domo's annual Data Never Sleeps[1] report notes that the colossal 97 zettabytes of data were circulating in 2022 alone. As stated in the article, every minute of the day 231.4 million emails are sent, 347 200 tweets (Twitter posts) are shared on Twitter, and 510 000 comments are posted on Facebook.

With that in mind, this bachelor thesis explores various techniques to make the process of sifting through massive collections of written documents as easy and efficient as possible. The main focus is handling compilations of tweets - Twitter's nature allows conveying information at a rapid speed via short text-based and semantically rich posts. Furthermore, the methods analyzed and tested on Twitter data can also apply to articles' titles, captions, or scientific abstracts.

One way of enabling easy text filtering is categorizing by topic - an often lengthy and complex process for humans can turn into a swift operation. The problem has two major components, first grouping the tweets that belong together and then detecting the shared topic because not all clustering approaches present the common theme. Since no annotated Twitter training set was available in Czech, the topic modeling, in this case, relies primarily on unsupervised methods such as Latent Dirichlet Allocation (LDA) and K-Means.

When determining the subject of the related tweets, identifying the keywords and keyphrases best representing the text proves to be an adequate and effective concept. However, the problem certainly has its challenges, which stem from the following facts - natural languages, especially the Czech language, are highly complex to analyze, and the input documents (clusters of tweets) are not homogenous either.

The task can be tackled in two ways: extracting keywords already contained in the cluster or generating keyphrases not necessarily present in the text. Whereas statistical and graph-based tools can handle extraction without supervision, generation requires supervised learning. For that part of the experiments, the Czech News Center (produces titles like Reflex, E15, iSport, Živě, and Blesk) supplied a proprietary dataset of already annotated articles.

[1]`https://www.domo.com/data-never-sleeps`

1.1 Motivation

The advance of machine learning algorithms and techniques in recent years has significantly boosted natural language processing (NLP) problem-solving. Nevertheless, the majority of the proposed and tested tools are suitable for English corpora, and topic modeling and keyword extraction are no exception. Therefore, the primary goals of this work are clear: ascertain whether methods applicable to English texts are fitting for Czech documents and identify the best approach when processing Czech tweets. Comparing and evaluating the most widespread methods can bring their advantages and limitations to light and help determine the most appropriate domain of application.

1.2 Thesis Outline

This work is structured in the following way:

- **Chapter 2** provides a theoretical basis and state of the art overview
- **Chapter 3** describes the datasets used and the preprocessing performed on them
- **Chapter 4** is dedicated to text and topic clustering experiments
- **Chapter 5** contains all keyword extraction and generation experiments
- **Chapter 6** concludes this thesis

Chapter 2
Theoretical Basis and State of The Art Overview

2.1 Feature Extraction

Since most machine learning (ML) and NLP models do not accept language units as input, numerical representation is instrumental. Mere scalars are insufficient conveyors of the linguistic and semantic information born by words and sentences, so their vectors, often with a specified number of dimensions, are used instead. The vectors are then mapped to a vector space in such a way that semantically similar words or sentences hold close positions. The process of transition from language to vector space is described below.

2.1.1 Tokenization

In the first step of preprocessing, the textual stream of data is broken into discrete elements - characters, words, or sentences. Many tools are available for this task, and its nature makes it mostly language-independent[1] in the Indo-European languages, where English and Czech belong.

An alternative to character, word, or sentence tokenization is subword tokenization. As Tunstall et al. (2022) state, subword tokenization splits rare words into smaller units to allow the model to deal with complex words and misspellings and to keep frequent words as unique entities while the length of the inputs is manageable.

Let's examine the differences between word and subword tokenization on an example text: *"Inteligence je schopnost přizpůsobit se změně."*. Whereas word tokenization described above produces a list containg all the words from the sentece, subword tokenization returns the tokens with some added information.

```
['Inteligence', 'je', 'schopnost', 'přizpůsobit', 'se', 'změně']
['cs_CZ', '▁Intel', 'i', 'gence', '▁je', '▁schopnost', '▁přizpůsob', 'it', '▁se', '▁změn', 'ě', '.', '</s>']
```

Figure 2.1: Word (upper list) and subword tokenization. Some special tokens are added to the subword token list to distinguish the language and the end of the sequence. The _ prefix in _schopnost means that the preceding string is whitespace. Tokenization tool is a part of the `gensim.utils` module.

For instance, Transformer (Vaswani et al., 2017) based models, the state-of-the-art deep learning architectures in NLP, process the raw text at the token level and use subword

[1]Language independence stems from the fact that all these languages separate their words using spaces, and punctuation marks define sentence boundaries. Many Asian languages (e.g., Hindi, Chinese, Korean, Urdu), however, have different text construction, and this method is not applicable.

tokenization to take advantage of the benefits of word tokenization while keeping the vocabulary[2] size reasonable.

2.1.2 Vectorization

The tokens derived from the input text are characters, words, subwords, or sentences depending on the chosen type of tokenization and need to be transformed into real-valued vectors. There are multiple approaches to this task with varying complexity and robustness. Shahmirzadi et al. (2019) suggest that vector representation techniques form two main categories:

- Count-based vector representations that rely on word frequency and treat words as atomic units (word order is ignored)
- Methods that embed the vectors into an $\mathbb{R}^n$ space and consider semantic relations and meaning (word order is taken into account)

Determining the correct approach depends on the purpose and the available dataset. It has been observed that simple models trained on vast amounts of data tend to outperform complex systems trained on fewer data (Mikolov; Chen, et al., 2013).

Count Vectorization

As its name suggests, count vectorization belongs to the group of methods which disregard word order and sentence structure and concentrate solely on the number of appearances of each word. It produces a matrix where each column represents a word from the vocabulary, and the rows contain the words' numbers of appearance in the input documents. The following example demonstrates the result after vectorization of the documents *["Není z Prahy, ale chodí do školy v Praze.", "Do konce měsíce obdržíme fotky Měsíce"].*

```
   ale  chodí  do  fotky  konce  měsíce  není  obdržíme  prahy  praze  školy
0    1      1   1      0      0       0     1         0      1      1      1
1    0      0   1      1      1       2     0         1      0      0      0
```

Figure 2.2: Count vectorization, each row of the count matrix represents one input sentence. Library used: `CountVectorizer` by `scikit-learn`.

The vectorizer uses word-level tokenization by default, and some of its shortcomings are apparent from Figure 2.2. While lemmatizing[3] the input before vectorizing it will eliminate the problem of diverse word forms ("Prahy", "Praze"), the fact remains that different meanings of the same word are overlooked ("měsíc", "Měsíc"), and the final matrix is of high dimension but sparsely populated.

Tf-Idf Vectorization

Term Frequency - Inverse Document Frequency (TF-IDF) is one of the most commonly used algorithms for vectorizing textual data. The relevance of a word to a text from the corpus increases proportionally with the increasing word frequency in the text. However,

[2]The set of unique tokens found within the corpus.

[3]Conversion of words into their headword (it is done in order to avoid treating inflected forms of the same word as different words). The headword of "Praze", "Prahy", and "Praha" is "Praha".

its number of appearances throughout the dataset balances it out, which helps to counteract the fact that some words are generally used more frequently than others. According to Shahmirzadi et al. (2019), the correlation between a term's weight and its occurrence in a document is positive, while its correlation with its overall occurrence in the corpus is negative. The weight of a term depends on two quantities

- **Term Frequency** ($TF(t,d)$): represents the proportion of the number of instances of a term $|t|$ in a document d to the document's length $|d|$.

$$TF(t,d) = \frac{|t|}{|d|}$$

- **Inverse Document Frequency** ($IDF(t)$): document frequency can be obtained by calculating the number of documents containing the term t ($D(t)$). Inverse document frequency is the number of all documents in the corpus (D) divided by the document frequency of t. For scaling purposes, a logarithm of the value is used.

$$IDF(t) = \log \frac{D}{D(t)}$$

and is defined as their product:

$$TF\text{-}IDF(t,d) = TF(t,d) \cdot IDF(t) = \frac{|t|}{|d|} \cdot \log \frac{D}{D(t)}$$

The difference between count Vectorization and Tf-Idf Vectorization is thus evident - CountVectorizer calculates frequency based on vocabulary alone, whereas Tf-Idf Vectorization regards words' significance across the whole corpus.

Continuous Vector Representations

Perceiving words as discrete entities strips away a fundamental layer of semantics - context. Let's consider the following Czech sentence: *"Na závodním kole objel jedno kolo a upadlo mu kolo."*. Even though "kolo" has three distinct meanings in that particular sentence, the vectorization methods described above will treat it as three occurrences of the same word and hence lose a portion of the information. To rectify that, some vectorization techniques examine surrounding words to reflect word relations, and Shahmirzadi et al. (2019) explain the premise behind this approach: words with akin meanings tend to appear in similar contexts.

The embeddings produced that way are considered a part of an inner product vector space where the vectors of synonymous or related phrases are positioned close to each other. This mathematical outlook not only drastically simplifies measuring word similarity but also allows arithmetic operations to be performed on words. Let $v(word)$ represent the vectorization of the word *word*:

$$v(kralovna) \approx v(kral) - v(muz) + v(zena)$$

In their paper on linguistic regularities, Mikolov; Yih, et al. (2013) discover that vector representations capture semantic and grammatical likeness by adding constant vector offsets between pairs of words sharing a particular relationship. The authors assume that all pairs of words with the same connection (e.g. singular and plural forms) also share the

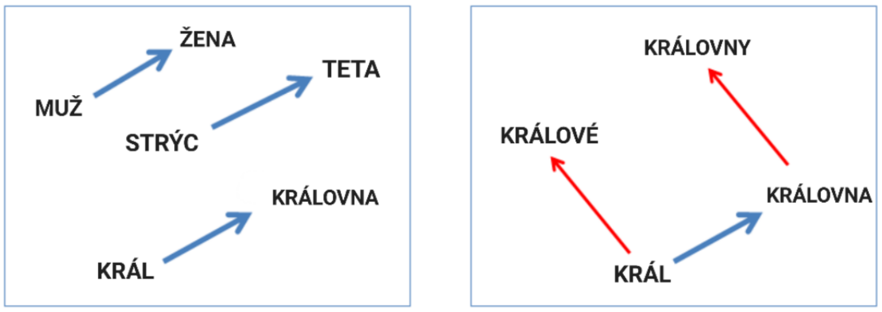


Figure 2.3: Inspired by Mikolov; Yih, et al. (2013). Left scheme shows the gender relation, right includes singular/plural relation. A high-dimensional space means multiple relations embedded for a single word.

same constant offset (Figure 2.3). These findings have a significant impact on NLP tasks that rely on the discovery of analogous words.

Both tasks of this thesis, clustering tweets by their topic and finding keywords, depend on determining how similar two words, more precisely their vectors, are. One of the most common measures of similarity is *cosine similarity* which calculates the cosine of the angle θ between the vectors. For vectors u and v cosine similarity is defined as the their dot product divided by the product of their lengths:

$$cos(\theta) = \frac{u \cdot v}{||u|| \cdot ||v||}$$

It holds that smaller angles mean a higher degree of similarity; thus, the cosine value has to be as close to 1 as possible. Table 2.1 shows the top five nearest neighbours of the word "modrá" calculated via cosine similarity. The fact that the closest words are other colours confirms that semantic meaning and context have been considered when training the model.

Word	Cosine Value
žlutá	0.8671872019767761
červená	0.8648878931999207
zelená	0.8549166321754456
oranžová	0.833976686000824
fialová	0.8107661604881287

Table 2.1: Nearest neighbours of the word "modrá" generated with the `fastText` library introduced by Bojanowski et al. (2017). The top five words are sorted by the cosine value.

It is important to note that not only words are embedded in vector spaces: Reimers et al. (2019) have introduced Sentence-BERT (SBERT), which generates sentence embeddings whose distance in the vector space is determined by their semantic similarity. SBERT is a modification of BERT, a language representation model, presented by Devlin et al. (2019).

2.2 Clustering Methods

In essence, one of the main tasks of this thesis is to divide a large Twitter dataset into subgroups based on a shared topic between the tweets to derive a maximal amount of information about the data. Since the input is represented as vectors belonging to the $\mathbb{R}^n$ vector space, the resulting clusters should be generated in such a way that only sufficiently similar vectors belong to the same category. The available Twitter dataset is not labeled, and for that reason, only unsupervised methods are applicable.

2.2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised ML technique which creates a probabilistic model of the corpus. It detects topics within the dataset based on the assumption that documents are *"represented as random mixtures over latent topics, where a distribution over words characterizes each topic"* (Blei et al., 2003). In other words, when given a dataset D containing M documents, the posterior probability is calculated, reflecting the conditional distribution of topics over the corpus documents. Figure 2.4 illustrates LDA schematically with its parameters:

- M denotes the documents in the corpus D
- N denotes the words in a specific document
- w is a particular word in a document
- z is the topic assigned to a particular word in a document
- θ denotes the topic distribution per document
- α is an initialization parameter; handles the initialization of θ with Dirichlet distribution $\theta \sim Dir(\alpha)$
- β denotes the word distribution per topic

Chipidza et al. (2022) summarize that the conditional distribution is a quotient of the joint probability distribution of z, β, and θ across all w and the probability of observing D across all topics.

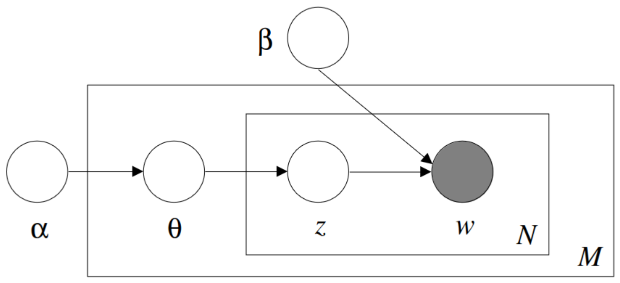


Figure 2.4: LDA scheme adapted from Blei et al. (2003).

The parameters θ and β are randomly initialized, and each word is randomly assigned a topic. Topic reassignment of a word happens on one of two conditions: i) appearances of the word in the topic are infrequent, ii) the topic is rare for the examined document.

The algorithm converges if no new reassignments were performed in the last step or if a predetermined number of iterations has been reached. The result is a collection of topics, and each topic is represented as a cluster of top N words that best describe it.

A somewhat challenging aspect of the algorithm is that the number of topics, k, has to be specified beforehand. Although no precise method exists and the results are data-reliant, the number of topics can be chosen so the resulting sets of words representing the detected topics are as *coherent* as possible. Röder et al. (2015) state that *“a set of statements or facts is said to be coherent, if they support each other. Thus, a coherent fact set can be interpreted in a context that covers all or most of the facts.”*. However, it is unclear how exactly coherence is to be quantified. Many metrics work with word co-occurrence, and the **UMass coherence score** is an example of that. Stevens et al. (2012) define it as

$$C_{UMass}(w_i, w_j) = \log \frac{D(w_i, w_j) + \epsilon}{D(w_i)}$$

where $D(w_i)$ denotes the number of occurrences of the word w_i in the input documents, $D(w_i, w_j)$ indicates how many times the words w_i and w_j appeared together, and ϵ[4] ensures that the score is a real number. The overall coherence of a specific topic is then computed by averaging the pairwise coherence scores of its top N words. The optimal k produces the lowest UMass score. Thus, a range of candidate values can be tested, and the one minimizing the score should be set as the number of expected topics.

2.2.2 K-Means

K-Means is a classic unsupervised learning method operating on an unlabeled dataset. The dataset containing $n \in \mathbb{N}$ input documents is divided into $k \in \mathbb{N}$ clusters based on high similarity within the cluster and low similarity between clusters. Each cluster has a centroid $c_i, i \in \{1, \ldots, k\}$, and all cluster points x are closer to that centroid than other centroids. If viewed as an optimization problem, the objective is to assign each datapoint a cluster so that the (squared) distance between the point and the cluster's centroid is minimal. Let $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ denote the set of clusters:

$$\min_{\mathcal{T}} \sum_{i=1}^{k} \sum_{x \in \mathcal{T}_i} ||x - c_i||^2$$

Algorithmically, the process consists of the following steps (Ibrahim et al., 2018):

1. Initialize the centroids randomly
2. Repeat until convergence (all clusters have remained unchanged):
    a. Assign each datapoint to the closest centroid
    b. Update the centroids

Although the algorithm's structure is set, there is much room for alterations to improve the result. As described above, K-Means is not a guaranteed global minimizer and can reach a local minimum. However, a clever initialization[5] of the centroids in the first step can at least aid it in finding better local minima. Other modifications of K-Means substitute the Euclidean distance used to determine the closest cluster center for other

[4] Usually set to 1.
[5] Known as K-Means++.

distance metrics or use diverse methods to update the centroids. Ibrahim et al. (2018) provide detailed descriptions of many of the algorithm's variations.

Just like LDA, K-Means relies on the predefined argument k - the number of expected topics / clusters. Finding the optimal value of the parameter poses a considerable hardship, especially without any foreknowledge about the dataset.

The **Elbow method** is an empirical technique for graphically determining the most appropriate number of clusters. The concept is relatively straightforward – K-Means clustering is run for a range of candidate values, and the sum[6] of squared distances from each point to its assigned centroid is calculated for each one (Schubert, 2022). When these distortions are plotted, the curve's inflection point, the "elbow", is the optimal k.

An alternative approach is available for higher-dimensional data: the **Silhouette method**. For a candidate value of k, so-called silhouette coefficients are estimated for all datapoints to indicate how similar each point is to its cluster members[7] compared to the rest of the clusters[8]. Wang et al. (2017) formulate the silhouette coefficient s_x of a point x as:

$$s_x = \frac{b_x - a_x}{\max(a_x, b_x)}$$

where a_x denotes the average distance between the point x and all other points from the same cluster, and b_x is the average distance of x to the members of the closest cluster. The averaged silhouette coefficients of all input points form the *silhouette score* $\in [-1; 1]$, and the goal is for it to be as close to 1 as possible because that implies that the samples are far from neighbouring clusters. Apart from determining the optimal value of k, the Silhouette method can also detect whether the cluster has any outliers.

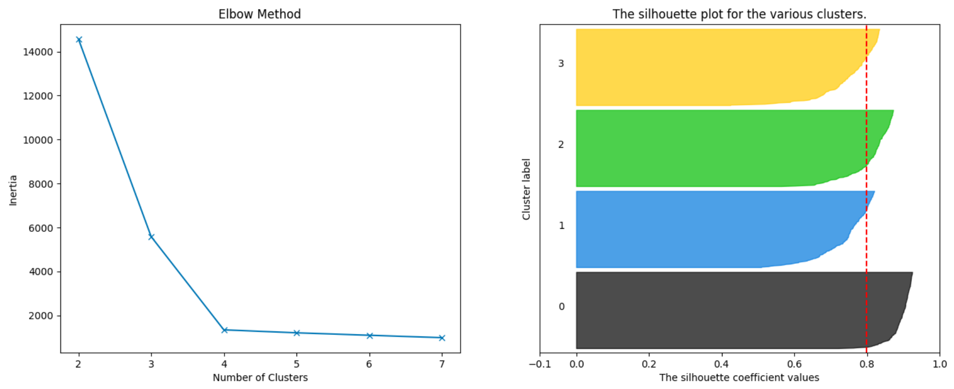


Figure 2.5: According to the Elbow method (left), the optimal number of clusters for a sample of 700 points is 4. The silhouette coefficient is also maximal for that number of clusters.

2.3 Keyword Extraction and Generation

Despite being a very practical unsupervised classification technique, K-Means undoubtedly has its flaws. One of the main issues, selecting the right number of clusters, and its possible

[6]This sum is also known as inertia.
[7]Cluster cohesion.
[8]Separation between clusters.

solutions are discussed in section 2.2.2. In the context of text clustering, however, another problem arises: although the algorithm places relevant tweets in the same cluster, the shared topic of the cluster remains concealed. Fortunately, there is a way around that obstacle - treating a cluster of similarly themed tweets as text can help reveal the subject by identifying its keywords.

When seeking keywords present in the text, two main viewpoints come into consideration – keyword extraction can be treated as a classification problem, where each word is labeled either “keyword” or “non-keyword”, or it can fall into the ranking problem category, where all words receive scores, and subsequently the highest ranking are chosen. The need for a comprehensive, already labeled training set is a significant drawback of the classification approach since its preparation requires manual annotation. On the other hand, unsupervised approaches like ranking evade the use of training data and are directly applied to the test set.

State-of-the-art keyword extraction approaches are either supervised or unsupervised, and Beliga (2014) proposes further dividing unsupervised methods into four categories: **Statistical**, **Graph-based**, **Linguistic**, and **Other** (Figure 2.7). Linguistic approaches focus predominantly on lexical and discourse analysis, and “other” methods rely on some heuristic knowledge about the input data, e.g. the length and positions of the words (Zhang et al., 2008). Out of the four, statistical and graph-based algorithms require no insight into the dataset, and both are language-independent.

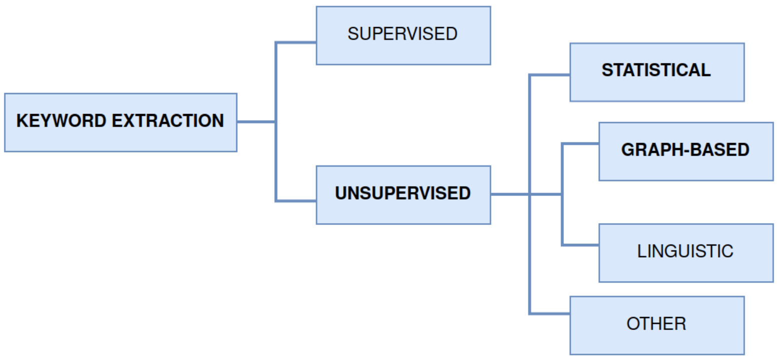


Figure 2.6: Keyword extraction methods classification inspired by Beliga et al. (2015).

2.3.1 Statistical Methods

Language and domain independence are notable reasons why many unsupervised methods rely on statistical features when extracting the most important words from a text. One of the most prominent statistical features widely used is TF-IDF (described in detail in section 2.1.2) because it reflects the relevance of a word to a document in the corpus. One of its many beneficial qualities is that it depends both on word occurrences in a document and the number of documents containing the word. When extracting keywords this way, the TF-IDF of each term in the document is calculated. The words and phrases containing the term (the score of a phrase is the sum of the TF-IDF of its components) are then ranked based on their TF-IDF value, and the most relevant are chosen as keywords.

Oftentimes statistical methods are paired with more complex tools to enhance the results. One such example would be KeyBERT, introduced by Grootendorst (2020). Key-

BERT combines simple count vectorization (section 2.1.2) with BERT embeddings which acknowledge word meaning. Its basic functionality consists of three steps:

1. A list of candidate keywords is generated from the input document (via count vectorization)
2. Both the document and the list of candidate keywords are embedded using BERT
3. The candidates most similar to the document are found by calculating the cosine similarity between vectors

To obtain a more diverse set of keywords the *Max Sum Similarity* algorithm is applied: it maximizes the candidate similarity to the document whilst minimizing the similarity between candidates. In practice, the top n keywords are selected, and from them the m that are the least similar to each other are picked to represent the document.

2.3.2 Graph-based Methods

Even though vector embeddings successfully capture semantic information, the structure of the text as a whole is often neglected. Graph-based representation of the input documents, however, would facilitate efficient analysis of the relationships between terms. Let a directed graph be defined as $G = (V, E)$, where words constitute the set of vertices V and their relations form the set of edges E. The edges can symbolize various relations: co-occurrence[9], syntax dependence, semantic relations and others (Beliga et al., 2015).

Most ranking algorithms are based on Google's PageRank by Page et al. (1999) and aim to determine a node's significance within the graph. One of the current state-of-the-art models, TextRank, uses co-occurrence windows[10] to create edges and when *"one vertex links to another, it is basically casting a vote for that other vertex"* (Mihalcea et al., 2004, p. 1). In other words, a node's importance is directly proportional to the number of "recommendations" it has received from other nodes. Furthermore, the ranking model takes into consideration the score of the vertex casting the vote: this way, votes from more "prominent" vertices have more weight. Let $In(V_i)$ denote the set of vertices pointing to a given vertex V_i, and let $Out(V_i)$ be the set of vertices that V_i points to. As established by Brin et al. (1998), the ranking score of V_i would be:

$$S(V_i) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} \cdot S(V_j)$$

where $d \in [0, 1]$ is a factor incorporating the probability of jumping from a given vertex to another random vertex in the graph usually set to 0.85 (Mihalcea et al., 2004).

This algorithm is run until convergence and once each node has received its score, the top N nodes are selected for post-processing. During the last phase, the N candidate keywords are highlighted in the text, and sequences of adjacent keywords are collapsed into a multi-word keyphrase. For example, suppose the input sentence is *"Podle ministerstva školství je matematická olympiáda velmi populární soutěž pro děti ve školním věku."* and TextRank chooses the words *"matematická"* and *"olympiáda"* as potential keywords. In that case, the words will be combined in the phrase *"matematická olympiáda"*, since they are adjacent.

[9]In this context it means that neighbouring terms co-occurring within the window of a fixed size are connected.

[10]Typically of size 2-10 words.

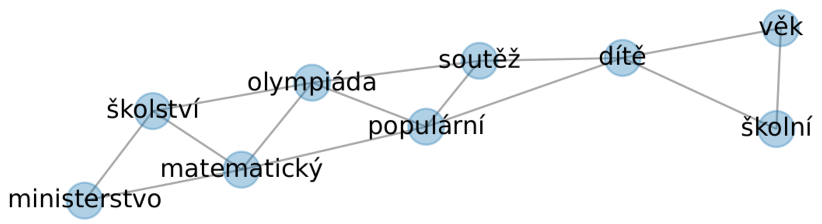


Figure 2.7: Graph-based representation of the sentence *"Podle ministerstva školství je matematická olympiáda velmi populární soutěž pro děti ve školním věku."* generated with the libraries `NLTK` and `NetworkX`.

2.3.3 Supervised Methods

As mentioned before, the most substantial disadvantage of supervised learning in general is the required training dataset. In the case of keyword extraction, manual annotation is not only tedious, but also inconsistent since determining what words best describe the text is decidedly subjective. However, if a training dataset is at disposal, a classifier can be trained to decide whether a word should be labeled as a keyword.

Witten et al. (1999) propose the *Keyphrase Extraction Algorithm* (KEA), which disregards stop words[11] and trains a Naive Bayes classifier utilizing the position of the first occurrence of the word within the document and the word's TF-IDF feature. Nevertheless, standard machine learning techniques such as support vector machine (SVM) and multi-layer perceptron neural networks outperform KEA in word classification (Firoozeh et al., 2020).

2.3.4 Keyphrase Generation

All of the approaches listed above tackle keyword *extraction*, yet it is hardly an inconceivable notion that the terms best reflecting the topic of the document may not appear in it at all. For example, the word "sport" would be a suitable keyword for a text reporting the results of a basketball playoff but is not a necessary part of its content. Finding such keywords is referred to as *keyphrase generation*, and as of now, no tools directly designed for this task are available. Instead, Koloski et al. (2022) suggest viewing keyphrase generation as a sequence-to-sequence[12] (seq2seq) problem and employing transformer architectures and supervised learning. Regarding seq2seq models, Meta AI's (Lewis et al., 2019) denoising autoencoder[13] BART (combines Bidirectional and Auto-Regressive Transformers) is a leading technology in the field. Pretraining consists of two steps: first, the text is corrupted with an arbitrary noising function, and then a sequence-to-sequence model learns to reconstruct the original text. Since the Twitter dataset is in Czech, MBART (Liu et al., 2020), the multilingual version of BART pretrained on corpora in various languages, including Czech, is the most fitting option.

[11] A set of commonly used words in a language. For example, the words "od", "oni", and "a" are considered stop words in the Czech language.

[12] Both the input and the output of the model are sequences of items (characters, words, etc.).

[13] A modified autoencoder that prevents the network simply learning the data.

2.3.5 Diverse Beam Search

A technique often employed in the context of sequence generation, beam search presented by Graves (2012) predicts the most likely sequence of words. An initial word is supplied to the algorithm, and based on it, the language model generates a set of candidate words for the next position in the sequence. Beam search does not focus solely on the most likely word; it keeps track of a fixed number of top candidates, known as the *beam width*[14]. The following word(s) are predicted based on each candidate word, resulting in multiple potential sequence continuations. A probability score is kept for each sequence continuation, and the candidates with the highest probability scores are retained while the rest are pruned. The process is iterative (the retained candidates become the input for the next iteration) and continues until a predefined end condition is met - e.g., reaching a maximum sentence length or a specific end token.

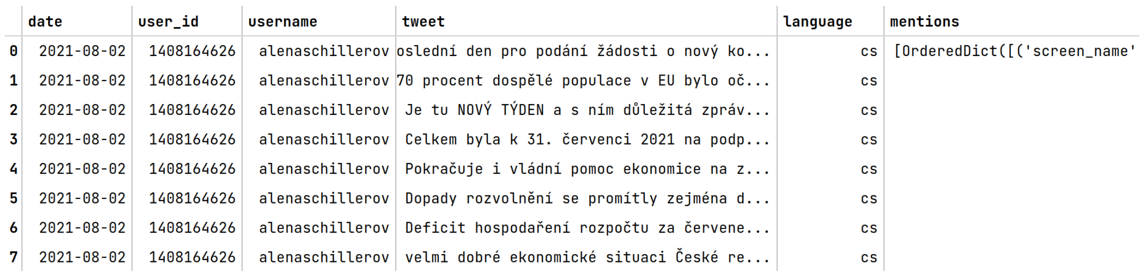
Vijayakumar et al. (2016) propose *diverse* beam search, which aims to produce a more varied set of outputs by discouraging similar or redundant sequences. Exploring different potential continuations is encouraged by penalization and diversity metrics such as n-gram overlap or Hamming distance. This technique promotes diversity and expands the range of possible solutions.

[14]Determines how many candidates are considered at each step; a higher value generally leads to better results but comes at a higher computational cost.

Chapter 3
Datasets and Preprocessing

3.1 Twitter Dataset

In order to assess whether the selection of methods described in Chapter 2 performs adequately and manages to form textual clusters and extract their keywords so the shared topic is evident, a Twitter dataset consisting of 219 524 tweets was used. Scraped from Twitter and provided by Jan Drchal, the dataset is stored as a pickle pandas file and apart from tweets, various metadata such as `date`, `time`, `user_id`, `language` and others[1] is included (Figure 3.1). According to it, 93 %[2] of the tweets were posted in Czech, with the oldest dating from 12/02/2009, while the newest was authored on 13/07/2022.



	date	user_id	username	tweet	language	mentions
0	2021-08-02	1408164626	alenaschillerov	oslední den pro podání žádosti o nový ko...	cs	[OrderedDict([('screen_name'
1	2021-08-02	1408164626	alenaschillerov	70 procent dospělé populace v EU bylo oč...	cs	
2	2021-08-02	1408164626	alenaschillerov	Je tu NOVÝ TÝDEN a s ním důležitá zpráv...	cs	
3	2021-08-02	1408164626	alenaschillerov	Celkem byla k 31. červenci 2021 na podp...	cs	
4	2021-08-02	1408164626	alenaschillerov	Pokračuje i vládní pomoc ekonomice na z...	cs	
5	2021-08-02	1408164626	alenaschillerov	Dopady rozvolnění se promítly zejména d...	cs	
6	2021-08-02	1408164626	alenaschillerov	Deficit hospodaření rozpočtu za červene...	cs	
7	2021-08-02	1408164626	alenaschillerov	velmi dobré ekonomické situaci České re...	cs	

Figure 3.1: Twitter dataset structure, not all metadata fields are displayed.

3.1.1 Preprocessing

Preprocessing is a crucial stage of almost all NLP tasks, and before any technique can be applied to the dataset, some adjustments are necessary. In this particular case, the tweets needed text and format modifications, so I wrote a custom function dealing with the following:

- **Stop word removal.** The idea behind this step is to remove words that occur frequently but bear little semantic information. This way, such "meaningless" words will be excluded from the list of keywords and will not create additional noise. Though prepositions, conjunctions, pronouns, and modal verbs are generally considered stop words, each language has its own specific set. For Czech, I used a list available online[3] since no official compilation was to be found. The Czech language uses the

[1]The total number of metadata fields is 37, and they carry information about attached photos and videos, the amount of likes and retweets, and the conversation thread.

[2]204 288 out of the 219 524 tweets are in Czech.

[3]`https://github.com/stopwords-cs`

Latin script with three diacritic marks: acute accent “ ´ ”, caron “ ˇ ”, and overring “ ° ” (*Czech and Slovak*, 2017). However, the accessible stop word list did not include diacritic marks, because such characters are standardly encoded in Unicode, a system supporting languages with special characters, and not ASCII. Therefore, the tweets were temporarily converted to ASCII representation with the `Unidecode` library[4], and upon stop word removal, the original representation was restored.

```
Není z Prahy, ale chodil do školy v Praze.
['Není', 'Prahy', 'chodil', 'školy', 'Praze']
```

Figure 3.2: Example of stopword removal on a Czech sentence. The words “z”, “ale”, “do” are considered stop words.

- **Lemmatization** is a process which reduces words to their root form called *a lemma*. The Czech language has seven grammatical cases, and nouns, pronouns, adjectives, numerals, and determiners all inflect to indicate their case. Verbs are conjugated as well, which means that the majority of Czech words can appear in numerous forms. Regarding each form as a separate word is undesirable for the goal of this project to provide a *semantically diverse* set of keywords, hence the need for lemmatization. For the tweets' lemmatization, I used `spaCy`'s UDPipe NLP pipeline[5] mainly beacuse of its pre-trained Czech language model.

```
Není z Prahy, ale chodil do školy v Praze.
['být', 'z', 'Praha', ',', 'ale', 'chodit', 'do', 'škola', 'v', 'Praha', '.']
```

Figure 3.3: Example of a Czech sentence lemmatization.

- **Link, emoji, and special symbols removal.** Due to their nature as short social media statements and comments, tweets often contain links, emojis, and other symbols[6]. None of those special characters and strings can be treated as potential keywords; thus, their removal reduces the noise of the dataset. To filter them out, I created specific regular expressions and used Python's `re` module.

Figure 3.4 shows an example of the final result of the preprocessing stage.

```
Není z Prahy, ale chodil do školy v Praze. #studiumvpraze 👍

'Praha chodit škola Praha studiumvpraha '
```

Figure 3.4: Preprocessing result example of a sentence with a typical tweet structure.

3.2 Czech News Center Dataset

While LDA, K-Means clustering, and all of the outlined keyword extraction methods, do not require an annotated dataset, the keyphrase generation experiments can only be performed with one. The dataset supplied by Jan Drchal for this part of the task is proprietary and courtesy of the Czech News Center (CNC)[7], a large media house in Central

[4]`https://github.com/takluyver/Unidecode`
[5]`https://spacy.io/universe/project/spacy-udpipe`
[6]For example, the symbols “@” and “#” often appear across social media, especially on Twitter.
[7]`https://www.cncenter.cz/en`

Europe. Their newsrooms produce popular titles such as Blesk, Sport, Reflex, and E15, and a generous collection of articles with manually added tags was kindly provided to experiment with.

All of the articles are written in Czech, and each title is stored in a separate XML (Extensible Markup Language) file. Each of the three files contains the essential data fields depicted below, the most important being `<ChapterBody>`, which holds the article itself, and all of the `<house>` elements, since those represent the tags/keywords. There is no set number of keywords per article; it typically ranges between four and twelve words. Another potentially useful element is the `<perex>` - it carries the articles' lead paragraphs, which summarize their main ideas and can be utilized in NLP summarization tasks.

```
<article>
    <date></date>
    <title></title>
    <perex></perex>
    <ChapterBody></ChapterBody>
    <url></url>
    <ArtID></ArtID>
    <house></house>
    ...
    <house></house>
</article>
```

3.2.1 Preprocessing

The preprocessing of the CNC dataset had two primary objectives: extract the relevant data from the XML file (articles and keywords) and remove invalid entries (wrong format, encoding issues). A tailored function dealt with it and split the processed data into train and test sets. All preprocessing functions can be found in the repository.

Since the keyphrase generation is to be handled as a summarization task, the dataset structure had to be adapted to the training process. Initially, each article had an array of keywords (`<house>` elements), but they could not be regarded as one summary. Therefore, each keyphrase was considered a separate label. Adopting this perspective on the matter meant that if an article had six keywords, the text would be added to the dataset six times, and each entry would have one of the six keywords as its label (see Table 3.1).

Text Field	Summary Field
"Síť O2 měla poslední den v roce rekord..."	"T-Mobile"
"Síť O2 měla poslední den v roce rekord..."	"mobilní operátor"
"Síť O2 měla poslední den v roce rekord..."	"Vodafone"
"Síť O2 měla poslední den v roce rekord..."	"Nový rok"
"Síť O2 měla poslední den v roce rekord..."	"O2 Czech Republic"

Table 3.1: Each element of the training set had to consist of a "text" field containing the article and a "summary" field containing a single keyphrase corresponding to the article. The example article had five tags originally, so five copies of it were added to the training set, one for each keyphrase.

Table 3.2 shows the initial content of the dataset and its expanded structure after the preprocessing. Ninety-two percent of it was used for training (48 848 out of 52 955 entries), and the remaining data went towards a validation and a test set (evenly distributed).

Title	Number of articles	Expanded version
E15	3 063	24 535
Reflex	1 983	11 991
Živě	1 802	16 429
Total	6 848	52 955

Table 3.2: An overview of the provided dataset and its contents before and after expanding it to accomodate the training requirements.

Chapter 4
Text Clustering

4.1 Clustering with Latent Dirichlet Allocation

As discussed in Section 2.2.1, LDA is a topic modeling algorithm which portrays corpora as mixtures of latent topics. A cluster of its most relevant words defines each topic, and detecting those clusters requires no annotated training sets. An undeniably valuable feature of the LDA technique is that no prior knowledge of the topics is needed, and that makes it exceptionally suitable for a dataset of 219 524 tweets spanning over thirteen years.

4.1.1 Implementation

The first phase of the implementation, preprocessing, was done as proposed in Section 3.1.1 with my custom functions and the libraries `Unidecode`, `spaCy`'s UDPipe, and Python's `re` module.[1] Each tweet was split into separate sentences before preprocessing, yet it had no impact on the algorithm because LDA considers the vocabulary of the entire corpus and disregards word order. Figure 4.1 shows an example of the input structure passed to the subsequent stage.

```
['Ekonomická situace v eurozóně se i přes opětovné šíření koronaviru postupně zlepšuje 📈
Nejrychleji se podle výhledu @ CNB_cz zotavuje sektor služeb 👍']
[['ekonomický', 'situace', 'eurozóna', 'opětovný', 'šíření', 'koronavir', 'postupně',
'zlepšovat', 'rychle', 'výhled'], ['CNB', 'zotavovat', 'sektor', 'služba']]
```

Figure 4.1: An example of a tweet before and after preprocessing. The tweet belongs to Alena Schillerová and was authored on 23/07/2021.

The LDA topic model expects a dictionary and a document-term matrix as its primary arguments, so I had to generate those next (Figure 4.2). The dictionary holds information about all unique tokens present in the input documents. To create it, I used Gensim's `corpora.dictionary`[2] module, which performs a mapping between words and their integer ids. Gensim's `doc2bow` function helped produce the document matrix from the dictionary: it is essentially the so-called bag of words (BOW) - a data structure reporting the number of occurrences of every word in the documents.

Before running the LDA model, one more step had to be carried out: determining the parameter k (the number of expected topics). For this procedure, I employed the *UMass*

[1] References to the libraries are enclosed in Chapter 3.

[2] `https://radimrehurek.com/gensim/corpora/dictionary.html`

```
Dictionary<14 unique tokens: ['ekonomický', 'eurozóna', 'koronavir', 'opětovný', 'postupně']...>
Document-term matrix:  [[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1),
(9, 1)], [(10, 1), (11, 1), (12, 1), (13, 1)]]
```

Figure 4.2: The dictionary and document matrix generated for Alena Schillerová's example tweet from Fig 4.1.

coherence score method (Section 2.2.1). Since the total number of tweets to be processed is around 219 thousand, I examined ten k candidates: 10 000, 11 000, ..., 20 000. The experiments consisted of running the LDA algorithm for each particular k and calculating the UMass coherence score of the result. The optimal number of topics was the one minimizing the score; for the Twitter dataset, that turned out to be 13 000 (Figure 4.3).

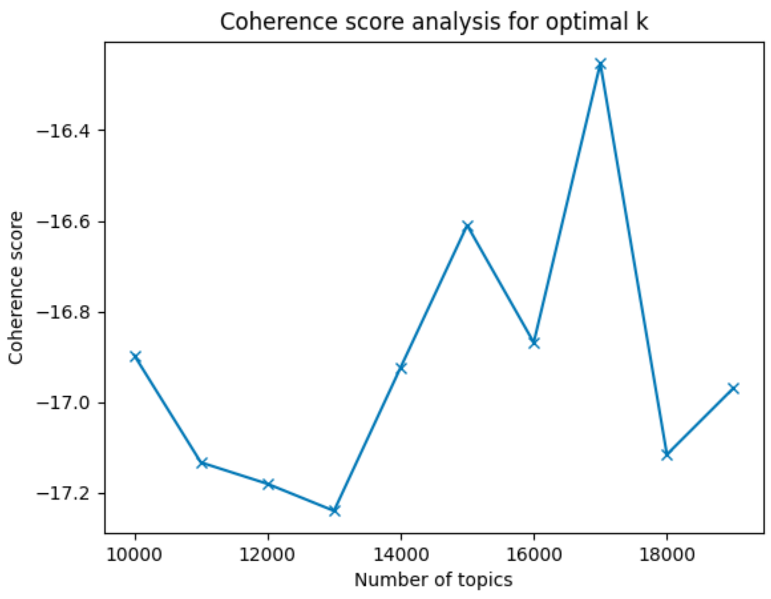


Figure 4.3: UMass coherence score analysis fot the Twitter dataset. According to it, the optimal number of topics for this particular dataset is 13 000.

Next in order was running the LDA model itself, and for that purpose, I utilized Gensim's `ldamodel` library[3] optimized in Python. Apart from the dictionary of unique tokens, the document-term matrix containing word occurrences, and the number of expected topics, the number of documents to be used in each training chunk, and the number of passes through the corpus during training had to be set as well (to 200 and 20 respectively - the chunk size did not really influence the outcome, but for the number of passes it turned out that 20 passes yield much better results than 10 passes while 30 passes only slowed down the process without notably improving it).

The LDA model assumed k topics, iterated over all tweets and randomly assigned each word to a topic. Then, for every word of every tweet, it computed:

1. the proportion of words in document d_i assigned to topic t_k
2. the proportion of all documents assigned to topic t_k for a given word w_j

[3]`https://radimrehurek.com/gensim/models/ldamodel.html`

Then the product of those two probabilities was calculated and, based on its value, the word w_j from document d_i was reassigned to a new topic t_k. After the number of loops was exhausted, the model returned the identified topics which were defined by their top words and probabilities as Figure 4.4 shows. More examples of topics detected in the Twitter dataset can be found in Appendix A.

```
(1,
 '0.044*"obec" + 0.034*"stav" + 0.031*"pomáhat" + 0.031*"nouzový" + '
 '0.031*"dostat" + 0.028*"pravidlo" + 0.027*"změna" + 0.027*"výroba" + '
 '0.024*"příspěvek" + 0.024*"kraj"'),
```

Figure 4.4: An example of a Twitter topic identified by LDA.

4.1.2 Results

To gain a better insight into the recognized topics, I pulled the top thirty words for each of them. Since one of the most effective ways to comprehend data is through visualization, I took a batch of 4000 tweets (for the sake of better readability) and visualized their detected topics[4] with the aid of the `pyLDAvis` library[5] which specializes in interactive topic model visualization. At first glance, some topics seemed reasonably defined, especially the most prominent ones (Figure 4.5).

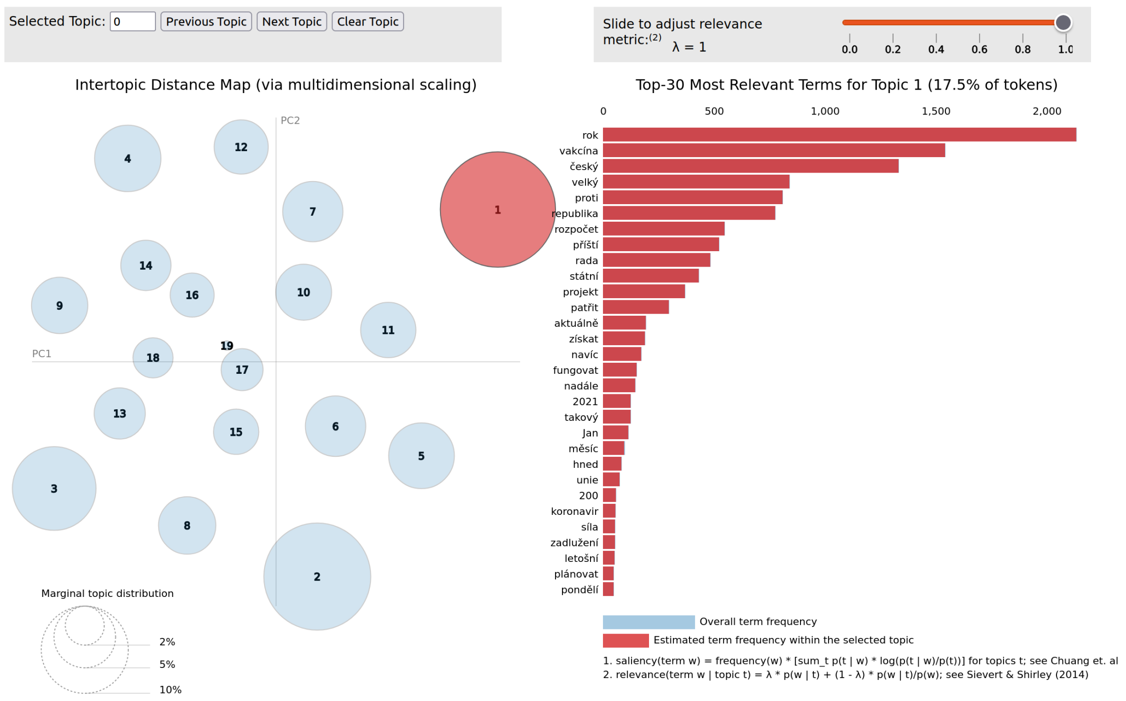


Figure 4.5: LDA topic visualization over 4000 tweets with the `pyLDAvis` library. The most prominent topic is marked and its top words are listed on the right.

The rest, however, had many semantically non-significant words appear in the top spots as the most important terms (Figure 4.6). Those "insignificant" words are not considered stop words, yet they provide no understanding of the topic. The topics of the entire dataset suffered from the same problem: the most relevant terms were exceptionally noisy even

[4] Based on the UMass coherence score analysis, 19 topics were to be expected.
[5] `https://github.com/bmabey/pyLDAvis`

though the data had been preprocessed. Thus, it was impossible to determine coherent themes. This demonstrates that devising an objective metric to numerically assess the quality of the detected topics and their definitions is not simple by any means. One way to evaluate the results would be to measure the global UMass coherence score C_{UMass}. Generally, the closer C_{UMass} is to 0, the better the coherence is, because it is calculated over the logarithm of probabilities (Section 2.2.1). In this case, C_{UMass} equalled -17.24 for the optimal k. The issues with that are two: first, there is no way to judge whether that is an acceptable score - no frame of reference can be built because the results largely depend on the dataset. Secondly, although C_{UMass} is well below zero, manual interpretation showed that most detected topics are not coherent enough.

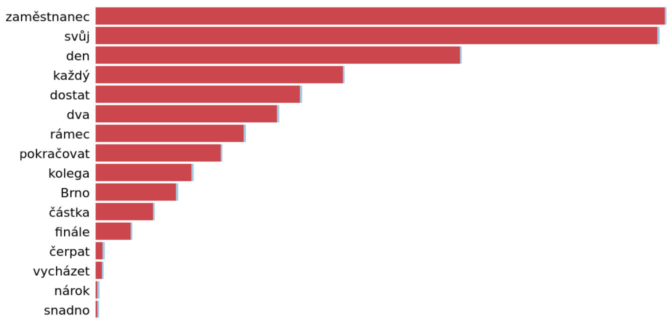


Figure 4.6: The top words defining topic number 5. Its theme cannot be reasonably concluded.

4.2 Clustering with K-Means

As it became evident in the previous section, experiments with LDA resulted in an overview of the latent topics present in the Twitter dataset, but did not classify the tweets themselves. Therefore, I decided to apply K-Means (Section 2.2.2) on the dataset to acquire clusters of tweets grouped together because of a shared topic. It is an unsupervised ML technique that guarantees convergence and scales to sizable datasets like the Twitter one.

4.2.1 Implementation

The first step of the process was to transform the tweets into real-valued vectors to enable K-Means classification. Tweets tend to be really short, often consisting of a single sentence (if that), and for that reason, I created their embeddings with SBERT's SentenceTransformers framework, more specifically with the `paraphrase-multilingual-mpnet-base-v2` model.[6] It is pretrained on parallel data for over fifty languages, including Czech, and maps sentences and paragraphs to a 768 dimensional dense vector space.

I would like to emphasize that the tweets were *not* preprocessed prior to creating the embeddings, and there is a reason for that. As mentioned above, tweets have limited length, and people often express themselves with more than words - emojis and hashtags, in particular, carry much information. Flag emojis unite many political tweets, while medal ones regularly appear in the context of sports events. Taking these social media phenomena into account, I opted to vectorize the dataset as it is.

Next, the sought number of clusters k had to be specified. This aspect of K-Means, having to set the number of clusters in advance, is one of its major drawbacks. I wanted

[6]`https://www.sbert.net/docs/pretrained_models.html`

to examine all methods from Section 2.2.2 in practice, so I ran both the Elbow method and the Silhouette method and tried to use them to discover the optimal k. The candidates were the same as the ones tested for the LDA topic detection: 10 000, ..., 20 000. The graph in Figure 4.7 shows that the Elbow method failed for an extensive dataset and large numbers of clusters. No “elbow” formed, so the optimal value of k is unclear. The Silhouette method, on the other hand, revealed that 16 000 is the most suitable number of clusters (maximizes the Silhouette score) and that is the parameter value I went with.

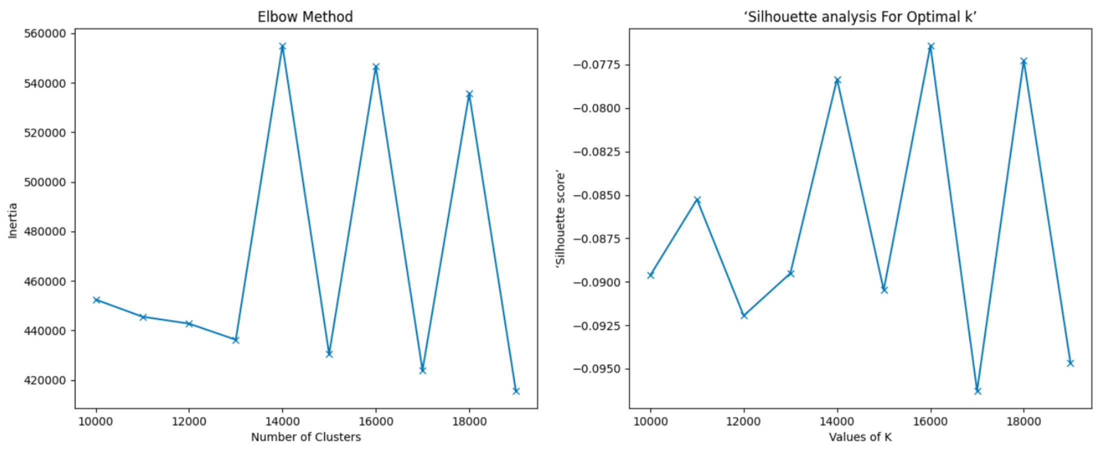


Figure 4.7: The left panel shows the result of the Elbow method, the right one contains the graph of the Silhouette method. Both were applied on the Twitter dataset.

For the clustering[7] I decided to put `scikit-learn`'s `MiniBatchKMeans` module[8] to use. As the library's user guide[9] states, *“the MiniBatchKMeans is a variant of the K-Means algorithm which uses mini-batches to reduce the computation time while still attempting to optimize the same objective function. Mini-batches are subsets of the input data, randomly sampled in each training iteration. These mini-batches drastically reduce the amount of computation required to converge to a local solution. In contrast to other algorithms that reduce the convergence time of K-Means, mini-batch K-Means produces results that are generally only slightly worse than the standard algorithm.”* (Pedregosa et al., 2011). The batch size was set to 2048 to enable parallelism on eight cores.

4.2.2 Results

When determining the quality of the obtained clusters, the Silhouette score can be used, for it is a metric revealing how similar an element is to its assigned cluster compared to the rest of the clusters. The score s ranges between 1 and -1:

- $s \approx 1$ means that the samples are far from neighbouring clusters
- $s \approx 0$ means that the samples are on a decision boundary between two or more clusters
- $s \approx -1$ means that the samples are assigned to the wrong clusters

For the optimal number of clusters, 16 000, the Silhouette score equalled -0.076. It is closest to zero, meaning that most tweets gravitated close to the decision boundaries

[7]Not only the final clustering of the tweets, but the clusterings needed for the Elbow and the Silhouette methods.

[8]`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`

[9]`https://scikit-learn.org/stable/modules/clustering.html`

between clusters. Nevertheless, it is unsurprising that tweets pertain to more than one topic - posts about politics and covid, finances and education, or culture and Prague are hardly rare. Upon manual inspection[10], most of the clusters were coherent, consistent, and logically structured.

[10]2000 clusters were inspected manually.

Chapter 5

Keyword Extraction and Generation

After acquiring reasonably shaped clusters of related tweets, I wanted to gather their topics automatically rather than reviewing them manually. Obtaining the keywords of each cluster was a viable method since its most important words would indicate the common topic of the group of tweets. There were two options on how to proceed: extract keywords present in the cluster or generate new ones that are not explicitly mentioned. For the extraction, I decided to compare the statistical (including the KeyBERT hybrid) and graph-based methods because of their completely distinct approaches described in Chapter 2.3. Every cluster was regarded as a separate input text and underwent the same preprocessing from Section 3.1.1. Generating new keyphrases was a trickier task, and I tried two different concepts: finding the nearest neighbours of already extracted keywords and training a transformers model, which was the only supervised method.

It is imperative to preface the unsupervised experiments by pointing out that no scientific metric can measure the quality of the results; no exact number can reflect the "keyness" of the extracted keywords. Even when done manually, the set of keywords best describing a text chosen by one annotator almost always differs from the set picked by another. Despite the subjective nature of the task, 2 000 clusters[1] were reviewed and annotated manually. The selected keywords then served as a reference when determining if the experimentally extracted ones were accurate enough.

5.1 Statistical Keyword Extraction

As stated in Section 2.3.1, when it comes to statistical keyword extraction methods, TF-IDF is one of the most suitable features to be explored because it reveals the relative importance of words with regard to the whole corpus (see Section 2.1.2).

5.1.1 Implementation

The implementation of the TF-IDF keyword extraction method consists of three main steps:

1. Calculate the **term frequency** of each document: divide the number of occurrences of a word by the total number of words in the document.
2. Calculate the **inverse document frequency** as the logarithm of the total number of documents divided by the number of documents containing the word w_i. Each

[1]2 000 out of the 16 000 clusters produced with K-Means in the previous chapter.

word's TF-IDF weight is a product of the term frequency and the inverse document frequency.

3. Sort keyphrases by their TF-IDF weights and choose the top N as keywords

For steps 1 and 2, I decided to use scikit-learn's `feature_selection` module and its optimized `TfidfVectorizer` class[2] to speed up the process. The documents are passed as input, and based on the vocabulary, the document-term matrix of the corpus is returned. In this context, the document-term matrix is a table whose rows represent each document from the corpus, and every word from the corpus' vocabulary has its designated column. For example, position (i, j) holds the TF-IDF score of the j-th vocabulary word for the i-th document (see Figure 5.1).

```
     chodit    nárůst     praha  studiumvpraha   uchazeč  zaznamenat     škola
0  0.497675  0.000000  0.708199       0.354100  0.000000    0.000000  0.354100
1  0.000000  0.470426  0.334712       0.334712  0.470426    0.470426  0.334712
```

Figure 5.1: TF-IDF document-term matrix example for the cluster ["Není z Prahy, ale chodil do školy v Praze. #studiumvpraze", "Školy v Praze zaznamenaly nárůst uchazečů #studiumvpraze."]. The illustrative cluster only contains two documents for the matrix to be readable.

Defining what is considered the corpus and what is referred to as a document played a significant role in this experiment. Since TF-IDF is a feature that takes into account the entire corpus, there were two possible approaches:

- The corpus could consist of all 16 000 clusters produced in the K-Means experiment. In that case, the whole cluster would be considered one document from the corpus, resulting in 16 000 documents overall. I will refer to this experiment as experiment A.
- Each cluster could be considered a separate corpus where every tweet assigned to it would be a document. The number of documents in the corpus would equal the number of tweets in it. I will refer to this experiment as experiment B.

In order to compare the two techniques, I implemented them both. For experiment A, I merged all tweets within every cluster into one text so that the clusters could be manipulated as documents. The corpus consisted of 16 000 documents of various lengths. For experiment B, I left the clusters as they were (lists of tweets) and ran the algorithm for each cluster separately because they formed distinct corpora. Both experiments, A and B, were run in three different modes. Firstly, I only considered single words (unigrams) when calculating the TF-IDF weights and choosing the keywords. Next, I allowed only two-word phrases (bigrams) to be considered. Lastly, a mix of two and three-word units (bigrams and trigrams) was examined to analyze whether the keyphrases' length improved their quality.

5.1.2 Results

Regardless of the considerable differences between their corpora, both in size and vocabulary, experiments A and B yielded decidedly similar results in all three tested modes.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

Table 5.1 shows the keywords of a random cluster[3] extracted in experiment A for the three analyzed keyphrase lengths (unigrams, bigrams, bigrams and trigrams) and Table 5.2 contains the keywords for the same cluster obtained in experiment B. Generally, the majority of clusters exhibited the illustrated properties: in most cases, the main discrepancy came from the ranking, not the chosen words. In fact, for 93%[4] of the clusters, at least 10 of the 15 extracted keywords/phrases were completely identical[5] for experiments A and B, save for their position. The difference between the two experiments was most noticeable in their sets of extracted unigrams. The unigrams experiment B derived from the clusters frequently contained more common words like “rok”, “díky”, “státní”, “kč”, and “miliarda” (Table 5.2). Experiment A rarely came up with such words, and I believe that this is a manifestation of the difference between their corpora: experiment B only considers a singular cluster as its corpus, so words like “díky”, “státní”, and “kč” seem relevant. On the other hand, experiment A has all 16 000 clusters at its disposal, and since such words appear in many of them, their TF-IDF weight is negligible.

Rank	Unigrams	Bigrams	Bigrams and Trigrams
1	iii	program covid	program covid
2	úvěr	covid iii	program covid iii
3	program	záruční program	covid iii
4	covid	malý střední	záruční program covid
5	záruka	investiční úvěr	záruční program
6	firma	firma podnikatel	malý střední
7	podnikatel	státní záruka	investiční úvěr
8	investiční	čerpat půjčka	firma podnikatel
9	záruční	čerpání úvěr	státní záruka
10	střední	investice prodlužovat	covid iii konec
11	investice	iii konec	podpořit investice firma
12	podpořit	investice firma	firma podnikatel investice
13	cmzrb	likvidita firma	likvidita firma podnikatel
14	prodloužení	investiční účel	podpořit likvidita firma
15	banka	záruka investiční	zčerpat půjčka

Table 5.1: The top 15 keywords extracted from a cluster with experiment A in all three modes.

When compared against the manually extracted keywords, the TF-IDF ones[6] matched at least a third of them precisely and had around two to four very similar phrases. All in all, the topic of the cluster could be successfully deduced given the keywords.

The biggest flaw of the method, however, could not be avoided no matter the corpus (experiment A versus B) or the length of the keyphrases (unigrams, bigrams, bigrams and trigrams): some of the top keywords tended to be semantically analogous and brought no

[3]The contents of the cluster are attached in Appendix B, Figure B.1.
[4]14 947 out of 16 000 clusters.
[5]In the respective mode.
[6]All three groups of extracted keywords in each experiment were compared against the manually extracted ones.

Rank	Unigrams	Bigrams	Bigrams and Trigrams
1	program	program covid	program covid
2	úvěr	covid iii	covid iii
3	covid	miliarda kč	program covid iii
4	firma	státní záruka	miliarda kč
5	iii	záruční program	státní záruka
6	záruka	investiční úvěr	záruční program
7	podnikatel	firma podnikatel	záruční program covid
8	investice	malý střední	investiční úvěr
9	podpořit	záruka investiční	firma podnikatel
10	investiční	výše miliarda	malý střední
11	rok	investice firma	záruka investiční
12	díky	podpořit investice	výše miliarda
13	státní	investiční účel	výše miliarda kč
14	kč	čerpat půjčka	čerpat půjčka
15	miliarda	čerpání úvěr	investice firma

Table 5.2: The top 15 keywords extracted from a cluster with experiment B in all three modes.

new information about the topic of the cluster. That was especially the case for the sets of bigrams and trigrams. As displayed in Table 5.1 and Table 5.2, the cluster's top fifteen most relevant words could be shrunk by half. One could argue that such issues would only arise when working with smaller clusters, but even the most enormous clusters with more than 900 tweets had the same problem. For most clusters, the sets of bigrams balanced semantic diversity and usefulness of the keywords the best.

5.2 Keyword Extraction with KeyBERT

KeyBERT presented in Section 2.3.1 is a partially statistical method for keyword extraction which picks candidate keywords based on their number of occurrences, creates BERT embeddings of both the potential keywords and the original document, and selects the words closest[7] to the input text as its most important ones. Grootendorst's idea to combine a basic statistical feature and transformers architecture certainly has merit: BERT embeddings incorporate word meaning in the produced vectors, so the distance between the text and the candidate words mirrors their semantic closeness.

5.2.1 Implementation

Grootendorst's implementation of KeyBERT is available for installation[8] and direct application. Unlike the TF-IDF extraction method, KeyBERT operates on one text at a time, disregarding the rest of the corpus. Because of that, I chose to view each cluster as a separate input, merged its tweets into one body of text and passed its preprocessed

[7]Distance is measured via the cosine similarity.
[8]`https://github.com/MaartenGr/KeyBERT`

version as the main argument. Once again, I explored what influence the length of the sought keyphrases has on their quality by running the experiment for unigrams, bigrams and bigrams and trigrams. Another intriguing option was to use *Max Sum Distance* (Section 2.3.1) to attempt to diversify the results.

5.2.2 Results

The keywords KeyBERT extracted from the same cluster as in the previous experiment[9] are shown in Table 5.3. All sets of keywords had the same defect as the TF-IDF ones: semantically similar words and phrases were chosen as the most significant. Even the unigrams often contained nouns and adjectives constructed from the same lemma, e.g. “investice”, “investiční” (Table 5.3). The supposedly diverse *Max Sum Unigrams* indeed tended to consist of more various words, yet some of them were hardly key for the cluster and were regularly ranked higher.

Rank	Unigrams	Max Sum Unigrams	Bigrams	Bigrams and Trigrams
1	covidplus	portfoliový	covid investa	covid investa umožnit
2	covid	návrh	covidplus možnost	program covid investa
3	úvěr	záruka	program covidplus	covid investa
4	financování	schválený	investiční	covidplus možnost čerpat
5	investiční	investa	záruka úvěr	firma potřebný úvěr
6	firma	podnikatel	úvěr podpořit	program covidplus možnost
7	investice	byznys	komerční úvěr	záruka investiční úvěr
8	půjčka	český	úvěr poskytovat	přispět program covidplus
9	podnik	banka	záruka investiční	covidplus možnost
10	banka	půjčka	využít úvěr	úvěr program covid
11	český	investiční	zaručený úvěr	investiční úvěr podpořit
12	byznys	financování	potřebný úvěr	program covidplus
13	podnikatel	úvěr	dosáhnout úvěr	investiční úvěr program
14	investa	covid	úvěr program	český firma potřebný
15	záruční	covidplus	úvěr záruční	žádat investiční úvěr

Table 5.3: KeyBERT: Extracted keywords of different lengths (unigrams, bigrams and bigrams and trigrams). Max Sum Unigrams are supposed to be semantically diverse.

Comparison with the manually extracted keywords showed, however, that for 91% of the clusters[10], the words extracted by KeyBERT also matched at least a third of the manual ones, and similar phrases were present as well. Due to that, the topic of each cluster was easily recognizable in most cases. Another fascinating feature of KeyBERT was its ability to recognize the importance of hashtags. While TF-IDF only recognized Twitter hashtags when they appeared often, KeyBERT managed to pull them almost always. One such example would be the keyword “covidplus”: originally, it was a hashtag (#CovidPlus) included in a single tweet. KeyBERT considered it rare and significant enough to give

[9]The contents of the cluster are attached in Appendix B, Figure B.1.
[10]14 569 out of 16 000 clusters.

it the highest rank. Similar things happened for many tweets, and it is essential to note that while KeyBERT successfully detected rare words, the lemmatization done during preprocessing the clusters saved it from deeming misspelled words unique.

5.3 Graph-based Keyword Extraction

The most substantial dissimilarity between statistical and graph-based methods lies in the fact that when creating a graph-based representation of a document, the structure of the text is taken into account as well, something statistical methods omit entirely. To assess how graph-based approaches handle keyword extraction, I experimented with the TextRank algorithm outlined in Section 2.3.2.

5.3.1 Implementation

Although TextRank has an available implementation in the form of a library called `PyTextRank`, I had to make a multitude of adjustments to the open-source code[11] for it to accommodate the Czech language. While the statistical methods listed above operate with language-independent statistical features such as number of occurrences or TF-IDF, graph-based techniques work with linguistic properties like lexical classes and grammar. PyTextRank's original implementation uses `spaCy`'s English language model to prepare the required linguistic features, but no such model is supported for the Czech language. That is why I opted to substitute it with `spaCy-udpipe`, whose Czech model I had already used for preprocessing (Section 3.1.1).

The first step of the process was to analyze the input document and perform part-of-speech (POS) tagging where every word corresponds to a particular part of speech[12]. SpaCy's udpipe offers such a feature for its Czech language model, and an example of a POS analysis can be seen in Figure 5.2.

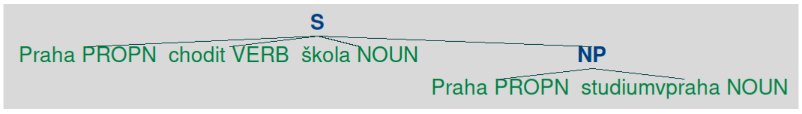


Figure 5.2: Part-of-speech tagging performed for the preprocessed version of the example "Není z Prahy, ale chodil do školy v Praze. #studiumvpraze".

The issue, however, arose from the fact that the descriptive POS tags differ for English and Czech. In graph-based keyword extraction, these tags are indispensable because not only do they reflect text structure and word relations, but they are also needed to create the so-called "noun chunks". *Noun chunks* are phrases that always contain a noun and some other words describing the noun (e.g. "nejvyšší budova na světě"[13] is a noun chunk revolving around the noun "budova"). Apart from indicating word co-occurrence, they are later used to combine the extracted keywords into longer phrases. I had to implement the noun chunk creation myself, and I did so by coming up with a grammar on what to consider a noun chunk based on the Czech POS tags. I needed to take into account the structure of the language and what typical phrases look like. Three types of phrases made the most sense: adjectives and a noun with the possibility of a determiner or a possessive

[11]`https://derwen.ai/docs/ptr/`
[12]Example POS categories would be a noun, verb, adjective, and other.
[13]Czech for "the world's highest building".

pronoun to precede them, a sequence of at least two nouns and a sequence of proper nouns and nouns.

```
NP: {<DT|PP\$>?<ADJ>+<NOUN>} # determiner/possessive, adjects, noun
    {<NOUN><NOUN>+} # a sequence of nouns
    {<PROPN>+<NOUN>+} # a sequence of proper nouns and nouns
```

After modifying the rest of the code so that the Czech POS tags and noun chunks could be applied, I passed each cluster as a separate document[14] and its graph was created according to the algorithm described in Section 2.3.2. During the last phase, the top N candidate keywords (the nodes with the highest score) were marked in the text, and sequences of adjacent keywords were collapsed into multi-word keyphrases.

5.3.2 Results

Albeit challenging to read at times, the graph representations of the clusters produced by the modified PyTextRank accentuated nearly the same words marked as key during the manual annotation of clusters. Not only that, but the "most popular" nodes often contained terms also extracted by the statistical methods. Figure 5.3 shows the epicenter of the graph representing the same cluster[15] as in the previous experiments and confirms the statement: the words with the most connections match a big part of the TF-IDF set of keyword unigrams.

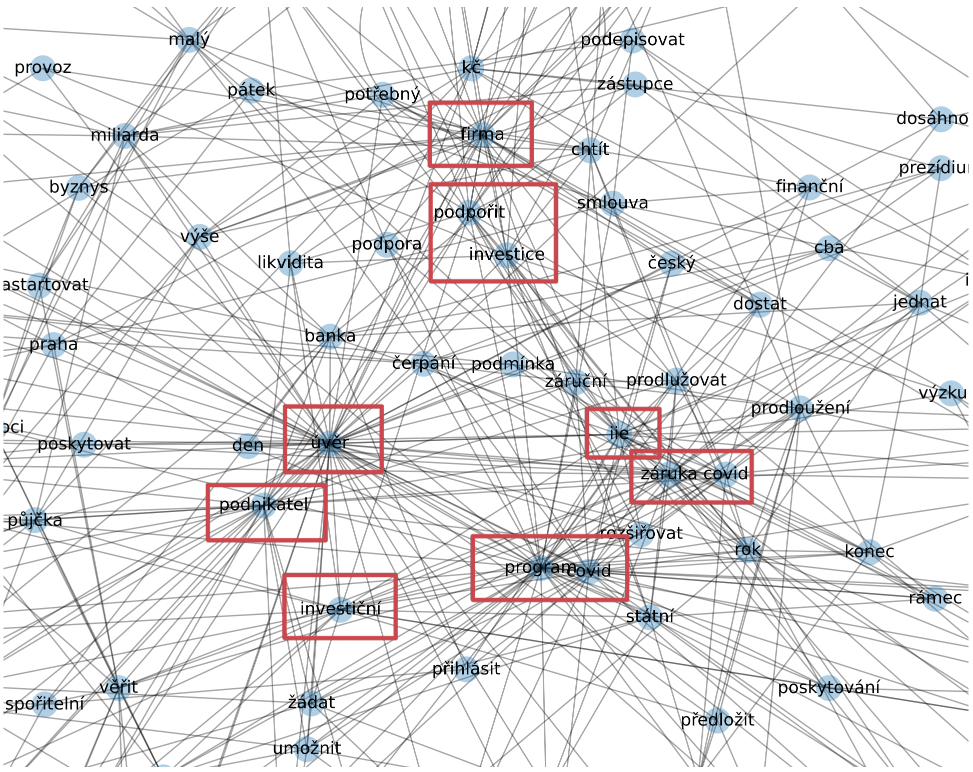


Figure 5.3: The graph generated for the same Twitter cluster as in the previous experiments. The nodes marked with the red boxes have the highest score (highest number of edges) and are considered most relevant. Visualised with the `NetworkX` library.

[14]The cluster's tweets were merged into one text.
[15]The contents of the cluster are attached in Appendix B, Figure B.1.

In my opinion, combining the highest-scoring words into phrases produced mixed results for most clusters. In some cases, the phrases truly provided more insight into the cluster topic and facilitated comprehension of its key aspects. An example of that would be "oživení ekonomika program covid investa" from Table 5.4, which conveys an essential theme of the examined cluster. On the other hand, not all phrases necessarily made sense, one such being "covid iii konec rok". Probably the most significant advantage of the method, however, was the fact that semantic repetition appeared to a much lesser extent than with the statistical methods.

Rank	Keyword/phrase
1	program covid iii záruka úvěr
2	záruka program covid iii
3	rámec program covid iii konec
4	investiční úvěr
5	záruční program
6	investice firma podnikatel výzkum inovace zvýšení efektivita výroba
7	vláda návrh prodloužení program
8	zelený program
9	čerpání úvěr
10	likvidita firma podnikatel investice
11	žádost úvěr
12	covid iii konec rok
13	oživení ekonomika program covid investa
14	prodloužení konec rok podmínka čerpání úvěr
15	potřebný úvěr

Table 5.4: Keywords and phrases extracted with the graph-based extraction method from the same cluster as in the previous experiments.

5.4 Nearest Neighbours Keyword Generation

All the techniques tested so far extract keywords; they pick words from the text and declare them most relevant. This next experiment is an attempt to generate keywords not present in the input document. The idea is to take a set of extracted keywords, vectorize them and seek their nearest neighbours in the vector space. As observed in Section 2.1.2, a short distance between embeddings indicates close meaning of the encoded words.

5.4.1 Implementation

To realize this idea, I utilized the `fastText`[16] library for handling text representations. It distributes pre-trained word vectors for 157 languages, including Czech, trained on Common Crawl[17] and Wikipedia. It automatically creates word embeddings with dimension

[16]`https://fasttext.cc`
[17]`https://commoncrawl.org`

300, and the built-in function `get_nearest_neighbors` returns the ten words nearest[18] to the passed argument. I took the clusters' sets of unigrams generated by the TF-IDF method and used them as input.

5.4.2 Results

From a theoretical viewpoint, the concept seemed logically sound, practically though, several hindrances sabotaged the experiment. The first obstacle was evident even in the example with the nearest neighbours of the word "modrá" (Table 2.1) in chapter 2. The words closest to the colour blue, "modrá", were, at first glance, unsurprisingly, other colours like yellow, red and green. Oddly, "barva", the Czech word for colour, did not appear on the list in any shape or form. This was problematic because the nearest neighbours tended to be words on the same semantic level as the extracted keyword and failed to bring diversity and generalisation into the set. After all, if for a cluster of World Cup tweets the word "football" is extracted, I would like the word "sport" to be generated instead of "basketball", "volleyball" or "baseball".

The second issue was of technical nature. The aforementioned `get_nearest_neighbors` function worked well for common words ("modrá", "fotbal") but failed to produce reasonable neighbours for more complex ones. In some cases, it simply returned the passed keyword in different forms (e.g. "záruka", Table 5.5), while in others, the returned strings were straightforwardly unrelated to the passed argument. For those two reasons, the Nearest Neighbours experiment was ultimately unsuccessful.

Rank	Keyword záruka	Keyword covid
1	záruka.	00024341
2	záruka-	8080000246
3	zárukaNa	Geovid
4	.záruka	ostrovid
5	Záruka	Dovid
6	M-záruka	00024
7	záruka3	Peirsol
8	záruky	Morazzone
9	záruka38	Mulenga
10	záruka52	EverNew

Table 5.5: The top ten nearest neighbours of the keywords "záruka" and "covid" produced with the `fastText` library.

5.5 Supervised Keyword Generation

Coming up with words best describing the primary notions of a text essentially means summarizing it, and viewing keyphrase generation as a sequence-to-sequence[19] problem is

[18]Distance is measured via the cosine similarity.

[19]The input is a sequence of items (characters, numbers, etc.), and the output is also a sequence of items.

an alternative take on the task. The goal of this experiment was to train a seq2seq model, which, when presented with a document, produces its keywords regardless of their presence or absence from the text. Since the job required supervised learning, the annotated CNC dataset presented in Section 3.2 served as a training set.

5.5.1 Implementation

The model I decided to employ is Meta AI's state-of-the-art denoising autoencoder BART (Section 2.3.4), more specifically, its multilingual version mBART[20]. Of the available mBART modules, I combined the pretrained `MBartForConditionalGeneration`, the MBART Model with a language modeling head, and `MBart50Tokenizer`. Apart from tokenizing the data, the tokenizer includes language metadata and special tokens and can also take care of padding, so the resulting tensors all have the same size.

The training process was set in the following way:

- The training set consisted 48 848 of articles and their corresponding keywords. As explained in Section 3.2, each article was assigned precisely one keyword; the articles that were associated with several tags had multiple copies in the dataset - one for each tag. The articles represented the input documents; their respective keywords were the expected outputs/labels.
- Input (articles) and output (keywords) are tokenized separately, and since some of the articles were of considerable length, the maximal token length set accepted by the model (1024) was exceeded. The graph in Figure 5.4 shows the token length distribution of the E15 articles[21], and it is clear that a sizable part of the tokens surpassed the 1024 mark by a lot. I had two feasible courses of action: truncate sequences longer than 1024 or try to cut off specific parts of the articles which may be redundant. Since the articles had different structures and content, I went with truncation, even if that meant a rather hefty loss for some of the longest ones. Auspiciously, news articles tend to concentrate their paramount concepts in the opening paragraphs, so the semantic loss should not be detrimental.

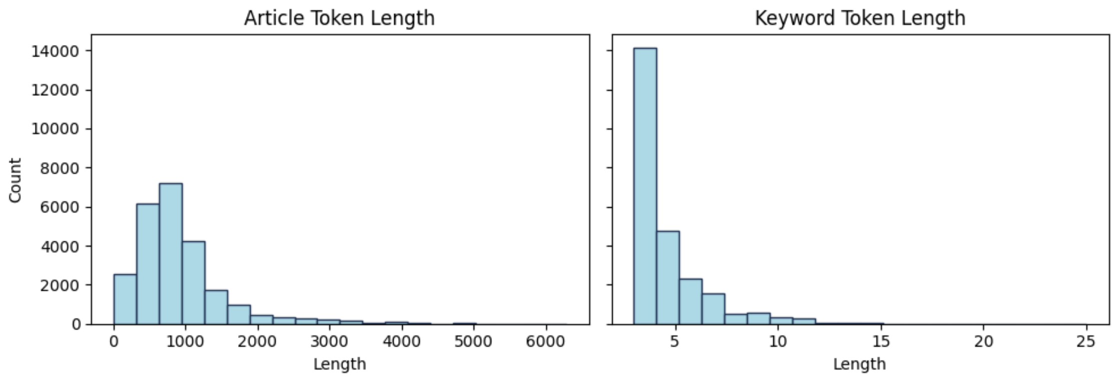


Figure 5.4: Token length distribution of the E15 articles.

- The training itself was mediated by the `Seq2SeqTrainer` - a feature-complete training and evaluation loop for PyTorch, and ran on the RCI cluster.

[20]`https://huggingface.co/docs/transformers/model_doc/mbart`
[21]The token length distribution of the rest of the dataset can be found in Appendix C.

5.5.2 Results

It turned out during training that after the third epoch overfitting ensues.[22] The validation dataset comprised 6 105 labeled articles, and the model's performance was evaluated using the *ROUGE* (Recall-Oriented Understudy for Gisting Evaluation) metric. Defined by Lin (2004), it is a set of metrics suitable for automatic evaluation of summarization and translation tasks in NLP. The model's summaries or translations are compared against reference summaries and translations produced by humans. MBART's results after three epochs of training were the following:

Metric	Value[%]
`eval_rouge1`	8.5375
`eval_rouge2`	2.7711
`eval_rougeL`	8.5507
`eval_rougeLsum`	8.5338

Table 5.6: Evaluation results of the MBART model after three epochs of training.

Fundamentally, ROUGE assesses how many n-grams[23] in the generated summary match the n-grams in the reference summaries. Thus, the reason behind the low percentages shown in Table 5.6 is apparent: the model predicts a single keyphrase (a unigram, bigram or a longer phrase) for every input text, but the reference keywords available per article range between four and twelve, so even when the generated phrase matches exactly, it cannot cover all references. Since the ROUGE scores are averaged for texts with multiple references, the resulting values make sense for the CNC dataset.

Although the model had several reference labels per article at its disposal during training, its predictions always consist of a single keyphrase. While the chosen keyword undeniably reflects a noteworthy theme of the document, even when passed a K-Means cluster, one phrase is largely insufficient. For example, the chosen keyword for the cluster examined in the rest of the experiments is "úvěr". The word belongs to the manually picked keywords for the cluster and is also present in the sets extracted with the statistical and graph-based methods. However, it is tough to deduce the full topic of the tweets based on it.

MBART generation	Manual Annotation
Ukrajina	Rusko, ropa, plyn, uhlí, suroviny, Komodity, válka na Ukrajině, Evropa, Evropská unie, Čína, indie, plynovod
elektřina	České dráhy, Brusel, Poplatek, Evropská komise, Martin Krupka, ODS, elektřina, peníze
Twitter	burza, jedovatá pilulka, Elon Musk, peníze, Twitter, Dolar, akcie, Tesla Motors
New York	ForMen, móda, hudba, rap, New York, Harlem, Bronx

Table 5.7: The keywords generated by MBART (left column) and the sets of human-produced keywords/tags (right column).

[22]Conclusion based on the loss.

[23]rouge1 is a unigram (1-gram) based scoring, rouge2 is a bigram (2-gram) based scoring, rougeL is a Longest common subsequence based scoring, and rougeLSum splits text using "\n"

To remedy that, I tried generating predictions using diversity beam search explained in Section 2.3.5. The beam width was set to five in the hopes of producing five keywords, and the diversity metric used was Hamming distance. The resulting sequences were longer than expected and shared a similar trait: the first couple of words were usually related to the input documents, but the rest seemed random, if not nonsensical. In my opinion, it was a sign that the MBART model was underfitted. Still, some keyphrases could be derived from the sequences - due to beam search's nature, the beginning of each sequence contains the words with the highest probability of appearing. Relying on that fact, I removed the stopwords from each sequence and picked the first three words of the preprocessed sequences as keywords. A few example results of this foray can be seen in Table 5.8.

Diverse Beam Search	Manual Annotation
válka, Ukrajina, Rusko	Rusko, ropa, plyn, uhlí, suroviny, Komodity, válka na Ukrajině, Evropa, Evropská unie, Čína, indie, plynovod
dohoda, ochrana, hospodářský	České dráhy, Brusel, Poplatek, Evropská komise, Martin Krupka, ODS, elektřina, peníze
Tesla, Motors, Dolar	burza, jedovatá pilulka, Elon Musk, peníze, Twitter, Dolar, akcie, Tesla Motors
New York, Industrial, Brooklyn	ForMen, móda, hudba, rap, New York, Harlem, Bronx

Table 5.8: The keywords generated with diverse beam search (left column) and the sets of human-produced keywords/tags (right column).

Chapter 6

Conclusion

The ultimate ambition of this thesis was to transform topic detection across large Czech corpora into an agile and smooth operation. The first phase included extensive research of state-of-the-art text clustering methods and keyword extraction techniques as a way to provide succinct descriptions of texts. For me to practically test those theoretical approaches, two distinct datasets were needed. The first one contained Czech tweets scraped from Twitter and the second one provided by the Czech News Center enclosed tagged articles of the titles “E15”, “Živě” and “Reflex”.

First, I employed Latent Dirichlet Allocation to discover the underlying topics of the Twitter dataset. A good coherence score was measured, and the most prominent topics seemed well-defined. Upon closer inspection, however, the rest of the detected topics were not as coherent and straightforward to interpret. Next, I decided to experiment with an alternative approach: instead of uncovering the hidden topics, I vectorized the tweets and performed K-Means clustering on their embeddings. As a result, the Twitter posts were grouped based on a shared theme, and the final clusters were consistent, logically structured and of decent size. It has to be noted though that due to the nature of the K-Means algorithm, each tweet is assigned to exactly one cluster, even though it may pertain to more than one topic. This shortcoming aside, K-Means yielded way more useful results than LDA.

After acquiring the clusters of related tweets, I wanted to reveal their topics automatically and obtaining the keywords of each cluster was a feasible method to do so. The first experiments I undertook relied on statistical features to extract keywords from the clusters. The two variations, TF-IDF word ranking and KeyBERT extraction, proved to be adequate strategies of comparable quality: the sets of keywords were semantically somewhat homogenous but informative enough for the topic to be inferred. Next, I tried my hand at graph-based algorithms. The nodes with the highest degree definitely contained the words most relevant to the text. Those words were then merged into longer phrases, and while some questionable combinations were formed, many phrases did indeed enhance the topic comprehension. Generating keywords not necessarily present in the documents was an even more arduous task. Staying in the realm of unsupervised methods, I tried to find the nearest neighbours of already extracted keywords. The results, however, consisted either of different grammatical forms of the same word or synonyms, which failed to bring diversity. Switching to supervised learning, I used the annotated CNC dataset to train a transformers model, MBART, to generate keyphrases for Czech texts. From a semantic point of view, the words selected by the model did, in fact, represent the clusters’ themes; still, it has certain limitations - the model only predicts one keyword per document. To overcome this issue, I integrated diversity beam search into the generation process and managed to produce longer sequences the beginnings of which turned out to be very useful.

Testing the limits further is what I plan to pursue in my future work. Assembling an even more extensive training set of manually annotated documents can vastly improve and develop the undeniable potential shown by the trained MBART model combined with diversity beam search. Not experimenting with other models would be a massive oversight on my part as well - Maarten Grootendorst's BERTopic, for example, is certainly intriguing and should be evaluated on Czech data.

Bibliography

BELIGA, Slobodan, 2014. Keyword extraction: a review of methods and approaches. *University of Rijeka, Department of Informatics, Rijeka*. Vol. 1, no. 9.

BELIGA, Slobodan; MEŠTROVIĆ, Ana; MARTINČIĆ-IPŠIĆ, Sanda, 2015. An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*. Vol. 39, no. 1, pp. 1–20.

BLEI, David M; NG, Andrew Y; JORDAN, Michael I, 2003. Latent dirichlet allocation. *Journal of machine Learning research*. Vol. 3, no. Jan, pp. 993–1022.

BOJANOWSKI, Piotr; GRAVE, Edouard; JOULIN, Armand; MIKOLOV, Tomas, 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*. Vol. 5, pp. 135–146.

BRIN, Sergey; PAGE, Lawrence, 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*. Vol. 30, no. 1, pp. 107–117. ISSN 0169-7552. Proceedings of the Seventh International World Wide Web Conference.

CHIPIDZA, Wallace; AKBARIPOURDIBAZAR, Elmira; GWANZURA, Tendai; GATTO, Nicole M., 2022. Topic Analysis of Traditional and Social Media News Coverage of the Early COVID-19 Pandemic and Implications for Public Health Communication. *Disaster Medicine and Public Health Preparedness*. Vol. 16, no. 5, pp. 1881–1888.

Czech and Slovak, 2017 [online]. [visited on 2023-05-03]. Available from: `https://sites.psu.edu/symbolcodes/languages/europe/czechslovak/#about`.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina, 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of naacL-HLT*. Vol. 1, p. 2.

FIROOZEH, Nazanin; NAZARENKO, Adeline; ALIZON, Fabrice; DAILLE, Béatrice, 2020. Keyword extraction: Issues and methods. *Natural Language Engineering*. Vol. 26, no. 3, pp. 259–291. Available from DOI: `10.1017/S1351324919000457`.

GRAVES, Alex, 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

GROOTENDORST, Maarten, 2020. *Keyword Extraction with BERT* [online]. [visited on 2023-04-21]. Available from: `https://www.maartengrootendorst.com/blog/keybert/`.

IBRAHIM, Rania; ELBAGOURY, Ahmed; KAMEL, Mohamed S.; KARRAY, Fakhri, 2018. Tools and approaches for topic detection from Twitter streams: survey. *Knowledge and Information Systems*. Vol. 54. Available from DOI: `10.1007/s10115-017-1081-x`.

KOLOSKI, Boshko; POLLAK, Senja; ŠKRLJ, Blaž; MARTINC, Matej, 2022. Out of Thin Air: Is Zero-Shot Cross-Lingual Keyword Detection Better Than Unsupervised? In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 400–409. Available also from: `https://aclanthology.org/2022.lrec-1.42`.

LEWIS, Mike; LIU, Yinhan; GOYAL, Naman; GHAZVININEJAD, Marjan; MOHAMED, Abdelrahman; LEVY, Omer; STOYANOV, Ves; ZETTLEMOYER, Luke, 2019. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. Available from arXiv: `1910.13461 [cs.CL]`.

LIN, Chin-Yew, 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. Available also from: `https://aclanthology.org/W04-1013`.

LIU, Yinhan; GU, Jiatao; GOYAL, Naman; LI, Xian; EDUNOV, Sergey; GHAZVININEJAD, Marjan; LEWIS, Mike; ZETTLEMOYER, Luke, 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*. Vol. 8, pp. 726–742.

MIHALCEA, Rada; TARAU, Paul, 2004. TextRank: Bringing Order into Text. In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, pp. 404–411. Available also from: `https://aclanthology.org/W04-3252`.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey, 2013. *Efficient Estimation of Word Representations in Vector Space*. Available from arXiv: `1301.3781 [cs.CL]`.

MIKOLOV, Tomas; YIH, Wen-tau; ZWEIG, Geoffrey, 2013. Linguistic Regularities in Continuous Space Word Representations. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751. Available also from: `https://aclanthology.org/N13-1090`.

PAGE, Lawrence; BRIN, Sergey; MOTWANI, Rajeev; WINOGRAD, Terry, 1999. The PageRank Citation Ranking : Bringing Order to the Web. In: *The Web Conference*.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. Vol. 12, pp. 2825–2830.

REIMERS, Nils; GUREVYCH, Iryna, 2019. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Available from arXiv: `1908.10084 [cs.CL]`.

RÖDER, Michael; BOTH, Andreas; HINNEBURG, Alexander, 2015. Exploring the Space of Topic Coherence Measures. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. Shanghai, China: Association for Computing Machinery, pp. 399–408. WSDM '15. ISBN 9781450333177. Available from DOI: `10.1145/2684822.2685324`.

SCHUBERT, Erich, 2022. Stop using the elbow criterion for k-means and how to choose the number of clusters instead. *arXiv preprint arXiv:2212.12189*.

SHAHMIRZADI, Omid; LUGOWSKI, Adam; YOUNGE, Kenneth, 2019. Text similarity in vector space models: a comparative study. In: *2019 18th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, pp. 659–666.

STEVENS, Keith; KEGELMEYER, Philip; ANDRZEJEWSKI, David; BUTTLER, David, 2012. Exploring Topic Coherence over many models and many topics. In: TSUJII, Jun'ichi; HENDERSON, James; PAŞCA, Marius (eds.). *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 952–961. Available also from: `https://aclanthology.org/D12-1000`.

TUNSTALL, Lewis; VON WERRA, Leandro; WOLF, Thomas, 2022. *Natural language processing with transformers*. "O'Reilly Media, Inc."

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia, 2017. Attention is all you need. *Advances in neural information processing systems*. Vol. 30.

VIJAYAKUMAR, Ashwin K; COGSWELL, Michael; SELVARAJU, Ramprasath R; SUN, Qing; LEE, Stefan; CRANDALL, David; BATRA, Dhruv, 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

WANG, Fei; FRANCO-PENYA, Hector-Hugo; KELLEHER, John D.; PUGH, John; ROSS, Robert, 2017. An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity. In: PERNER, Petra (ed.). *Machine Learning and Data Mining in Pattern Recognition*. Cham: Springer International Publishing, pp. 291–305.

WITTEN, Ian H; PAYNTER, Gordon W; FRANK, Eibe; GUTWIN, Carl; NEVILL-MANNING, Craig G, 1999. KEA: Practical automatic keyphrase extraction. In: *Proceedings of the fourth ACM conference on Digital libraries*, pp. 254–255.

ZHANG, Chengzhi; WANG, H.; LIU, Y.; WU, Dan; LIAO, Y.; WANG, B., 2008. Automatic keyword extraction from documents using conditional random fields. Vol. 4, pp. 1169–1180.

Appendix A
LDA Results

LDA Extracted Topics Example

Cluster ID	Top Words
0	’návrh’, ’schválit’, ’člověk’, ’místo’, ’hlásit’, ’očkovací’, ’volný’, ’zákon’
8	’podpora’, ’změna’, ’platit’, ’trh’, ’obec’, ’stále’, ’pracovat’, ’kontrola’, ’žádný’
17	’podnikatel’, ’město’, ’konec’, ’tunel’, ’iDNEScz’, ’policista’, ’solární’, ’Jan’, ’prostřednictví’
89	’500’, ’navíc’, ’příští’, ’zpráva’, ’týden’, ’nízký’, ’platba’, ’přinést’, ’nadále’
154	’svůj’, ’tisíc’, ’hodně’, ’velký’, ’kdy’, ’vycházet’, ’srovnání’, ’pan’, ’přilepšit’
672	’stát’, ’začít’, ’evropský’, ’jednání’, ’spojený’, ’politický’, ’echo24’, ’pomsta’, ’dohromady’
1064	’moci’, ’procento’, ’jednat’, ’hodina’, ’smlouva’, ’výsledek’, ’tisíc’, ’schůzka’, ’září’
1996	’český’, ’dva’, ’dělat’, ’dráha’, ’poprvé’, ’ministerstvo’, ’mio’, ’investovat’, ’kdy’
2903	’miliarda’, ’koruna’, ’program’, ’letos’, ’celkem’, ’přitom’, ’rada’, ’čerpat’, ’projednat’
4365	’tenhle’, ’proti’, ’růst’, ’daňový’, ’chodit’, ’držet’, ’francouzský’, ’ukazovat’, ’příjem’
5528	’díky’, ’2020’, ’dobrý’, ’situace’, ’velmi’, ’celkový’, ’dojít’, ’zdravotnictví’, ’navýšení’
7011	’chtít’, ’všechen’, ’zástupce’, ’myslit’, ’2013’, ’potkat’, ’sestra’, ’Belgie’, ’dluh’
12087	’psát’, ’říci’, ’rodina’, ’tvůj’, ’hned’, ’těšit’, ’sedět’, ’pozvat’
14863	’teď’, ’nikdy’, ’úřad’, ’operátor’, ’výkup’, ’vyhrát’, ’vědět’, ’ktorý’, ’2002’

Table A.1: Examples of topics defined by the Latent Dirichlet Allocation.

Appendix B
K-Means Cluster Example

Example of a Cluster Produced with K-Means

The following cluster was used to generate the exapmple sets of extracted keywords in the experiments described in Chapeter 5.

```
 1 Podnikatelé mohou od dneška žádat o investiční úvěry z programu COVID III. Záruku na tyto
   úvěry poskytuje stát prostřednictvím @CMZRB_Praha. Věřím, že i díky zaručeným úvěrům
   zrychlíme oživení naší ekonomiky 👍
 2 Program COVID Invest umožní žádat o komerční úvěry se státní zárukou i na investiční
   účely. Věříme, že i tento program pomůže nastartovat byznys malých a středních firem👍
 3 Podpoříme tím likviditu našich firem a podnikatelů a jejich investice. Díky zárukám
   programu COVID III dostaneme k českých firmám potřebné úvěry až do výše 500 mld. Kč.
   Chceme udržet zaměstnanost, podpořit investice firem a nakopnout hospodářský růst.
   https://t.co/Ptu1mIC4BW
 4 Hospodářský výbor @SenatCZ dnes podpořil prodloužení záručního programu COVID III do
   konce roku 2021. Díky návrhu z dílny @MinFinCZ budou moci podnikatelé využít úvěry nejen
   na financování provozu svých firem, ale také investic.
 5 Prodlužujeme státní záruky v rámci programu COVID III až do konce příštího roku. Poslanci
   právě dali zelenou návrhu novely zákona o státní záruce. Současně program COVID III
   rozšiřujeme o investiční úvěry. Podpoříme tím likviditu našich firem a podnikatelů a
   jejich investice.
 6 Prodlužujeme záruční program COVID III do roku 2021. Chceme podpořit investice našich
   firem a podnikatelů do výzkumu, inovací a zvýšení efektivity výroby. Proto jsme program
   rozšířili o poskytování záruk za investiční úvěry.  https://t.co/rJhs33RBjo
 7 Zítra předložím na vládu návrh na prodloužení programu Covid III. S Evropskou komisí
   budeme jednat o prodloužení až do konce roku 2021. Podmínky čerpání úvěrů zůstanou
   stejné, pouze vypustíme slovo „provozní“. Podnikatelé tak budou moci čerpat půjčky i na
   investiční účely.
 8 Podle @asociace_export dojde v následujících 5 měsících k postupnému oživení vývozu.
   Věřím, že k tomu přispěje i program #CovidPlus a možnost čerpat půjčky až do výše 2 mld.
   Kč.
 9 Se zástupci bank jsme dnes jednali na prezidiu @cba_cz o čerpání úvěrů ze záručního
   programu Covid III. Ke dni 13.7. banky přijaly 2536 žádostí o úvěr v celkovém objemu 16,4
   mld. Kč. Schvalovány jsou průměrně 2 ze 3 žádostí.
10 Dobrá zpráva pro malé a střední podniky! @CMZRB_Praha od pátku podepisuje první smlouvy s
   finančními institucemi do programu COVID III. Dosud se přihlásilo celkem 20 bank a
   spořitelních družstev, které podnikatelům do 500 zaměstnanců umožní získat úvěry za téměř
   půl bilionu korun
11 Vláda dala zelenou programu COVID III na podporu malým a středním firmám. Ty potřebují
   překlenout stávající období, než se jim zase zakázky a tržby rozjedou. Díky schválené
   portfoliové záruce státu dosáhnou na úvěry až za 495 mld. Kč. Pomůžeme tím udržet
   ekonomický výkon země.
```

Figure B.1: Cluster produced by K-means containing eleven tweets.

Appendix C
MBART Token Length

Token Length Distribution of the CNC Dataset

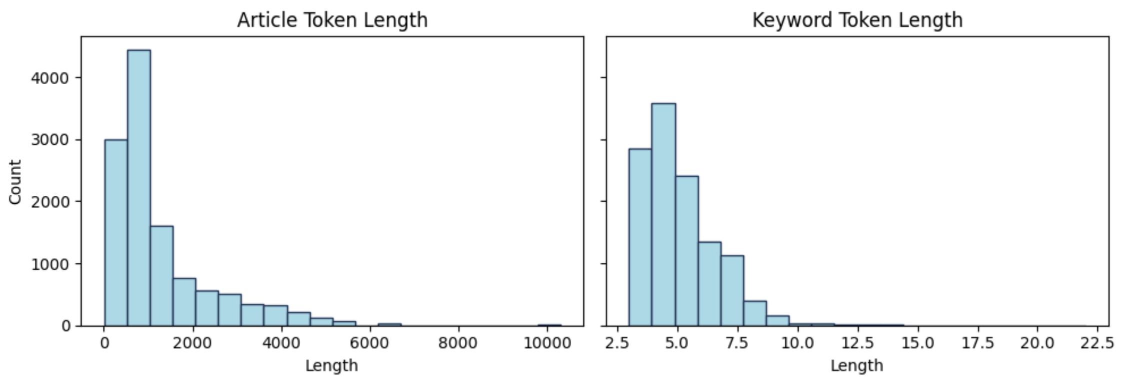


Figure C.1: Token length distribution of the Reflex articles.

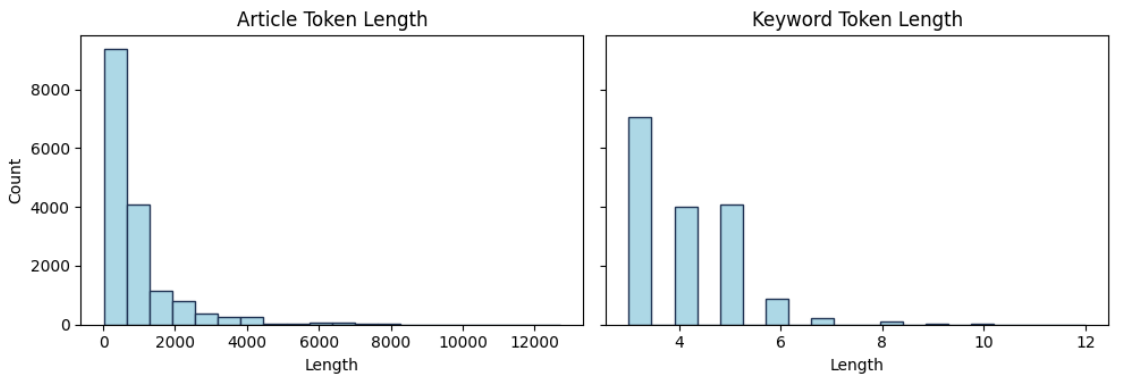


Figure C.2: Token length distribution of the Živě articles.

Appendix D
Acronyms

BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag Of Words
CNC	Czech News Center
KEA	Keyphrase Extraction Algorithm
LDA	Latent Dirichlet Allocation
ML	Machine Learning
NLP	Natural Language Processing
POS	Part Of Speech
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SBERT	Sentence-BERT
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency

Appendix E

Repository Structure

Description of Enclosed Repository

```
korladia_NLP_BP.zip ............................................. repository file
 ├── preprocessing.py ..................... preprocessing script and utility functions
 ├── LDA .................................. LDA implementation folder (Section 4.1)
 │    ├── lda.py ......................................... LDA implementation script
 │    └── lda.ipynb .................................... LDA notebook with examples
 ├── K-Means ........................... K-Means implementation folder(Section 4.2)
 │    └── kmeans.py ................................. K-Means implementation script
 ├── Keyword_Extraction.................. keyword extraction implementation folder
 │    ├── keyword_extraction_BERT.py .......... KeyBERT keyword extraction script
 │    │   (Section 5.2)
 │    ├── keyword_extraction_graph.py ........ graph-based keyword extraction script
 │    │   (Section 5.3)
 │    └── keyword_extraction_TFIDF.py ........... TF-IDF keyword extraction script
 │        (Section 5.1)
 ├── Keyword_Generation ................. keyword generation implementation folder
 │    ├── summarization_train.py .............. MBART training script (Section 5.5)
 │    └── summary_run_args....command line arguments for running the training script
 ├── kmeans_keywords.ipynb ........ K-Means and keyword extraction notebook with
 │   examples
 ├── czech_stopwords.txt ......................... contains a list of Czech stopwords
 └── tweets_new.csv ......................................... Czech tweets dataset
```