Robert Thomann, BSc.

Exploring topical clustering of political statements in German language

Master's Thesis

to achieve the university degree of

Diplom-Ingeneur (Dipl.-Ing.)

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn. Elisabeth Lex

Advisor
Dipl.-Ing. Dr.techn. Simone Kopeinik

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Voitsberg, December 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Acknowledgments

First and foremost I would like to thank my supervisor Assoc.Prof. Dipl.-Ing. Dr.techn. Elisabeth Lex who gave me the opportunity to write my thesis at the Institute for interactive Systems and Data Science. I would also like to thank my advisor Dipl.-Ing. Dr.techn. Simone Kopeinik for her valuable insights, advice and patience. Additionally, I would like to thank the Know-Center for funding the research and interFact for providing the dataset. Another thank you goes to my colleagues at the Know-Center for countless inspiring discussions. I am also grateful to my family and friends who provided invaluable emotional support during this interesting time. Especially to my partner Nina for her love and understanding. Without them, none of this would have been possible.

Abstract

Political debate has no longer been restricted to the traditional media. With the advent of the World Wide Web, politicians have begun to disseminate their viewpoints online. Micro-blogs, online social networks and digital press releases have become a medium of political debate. The interested reader is now faced with the challenge of finding his way through this flood of information. Document clustering can help with this task, but the question arises as to which algorithm should be used. There is also the question of a suitable document representation that conveys the semantic information needed to subdivide along thematic boundaries. This work aims to provide an answer to these questions. We created a dataset from a partially labeled corpus of political statements provided by our industry partner. This dataset was used to evaluate several document representation approaches based on topic models and embeddings. Since our dataset does not contain similarities or distances between statements, we evaluated document representations based on their performance in a simple recommender system. Among the document representations evaluated, the averaged Word2vec proved to be the most useful approach for our dataset. This document representation was then used as the basis for evaluating several state-of-the-art clustering algorithms. The partitions produced by the clustering algorithms were compared to partitioning based on the expert labels. Here, Spectral Clustering provided the best results, although none of the clusterings found resembled well the clusters derived from human labeling. Finally, we implement a prototype for topic label recommendations based on the best performing document representation and similarity-based recommendations which was shown to outperform our industry partners' approach.

Contents

List of Figures

List of Tables

1 Introduction

With the rise of micro-blogging and online social networking services political debate is no longer limited to classical media channels like newspapers and television. Policymakers share their standings and ideas on social media and even engage in political debate online. A new and growing source of political information has emerged. In a representative survey from 2017, already 20% of Germans said they use social media as a source of political information [36]. This development does not come without challenges. One of these challenges is finding one's way through this flood of information. The vast amount of political statements leaves the user with an unstructured and confusing mix of statements. This is especially true when the statements are aggregated from various sources. Document clustering could aid the users in navigating the political debate by unearthing hidden structure in the data.

Since there are many state-of-the-art clustering algorithms, the question arises which of these algorithms produces a partitioning that makes sense to the user. This depends not only on the choice of algorithm, but also on the way the document is presented to the algorithm. The choice of an appropriate document representation therefore plays a central role in the quality of results that can be achieved. If the goal is partitioning along semantic properties, then the document representation should also be able to represent semantic relationships. How should we represent short political statements, so that the document representation may be useful in a topical clustering task? For this purpose, embeddings and topic models come to mind, since — depending on the approach — they assign a point in a semantic space to words, paragraphs, or even entire documents. But which one should we choose to represent short political statements, so that the transformed document may

be useful in a topical clustering task? And once the document representation approach is decided, which clustering algorithm will give us meaningful results?

In this work we strive to not only provide an answer to these questions, but we also aim to build on our findings. Our industry partner plans to replace their topical statement labeling algorithm with a superior one. To this end, we develop and present a prototype topic label recommender whose design is directly inspired by our findings from the experimental part of this work.

1.1 Problem statement

Our industry partner Interfact markets near real-time analyses of the political discourse to political parties, NGOs, lobbyists and large enterprises. Interfact collected a data set of more than one million statements from 5.200 Austrian and German politicians on which they run an in-house developed natural language processing algorithm. This highly optimized algorithm assigns at least one of nine topic labels to a given statement.

Interfact strives to improve their topic matching method to attract private businesses as customers. To satisfy the needs of this new customer group their topic matching algorithm needs to be replaced with an improved version.

In this work we aim to develop an alternative approach that outperforms the algorithm currently in use in terms of precision and recall. The model should be able to learn from the entire corpus, including the statements that have not been labeled. At the same time, the model should be able to generalize well enough so that it does not require re-training every time the debate shifts to a novel topic.

1.2 Research questions

In order to solve the stated problem, we need a better understanding of the problem's nature. To gain the required insights needed to master the task at hand, we address the

following two research questions.

RQ1: "Which method is most suitable to represent statements in similarity calculations?"
This research question deals with the representation of topical statement similarity. Different document representation algorithms capture different aspects of language. Preprocessing is expected to have strong impact on the outcome and is therefore addressed as well. The question aims to identify a method of document representation which capture the semantic similarity of statements referring to the same topic.

RQ2: "Which state-of-the-art clustering algorithm works best to categorize short political statements?"
Once a similarity measures is chosen, a clustering algorithm can be used to partition the data into clusters. If these clusters align with the partitions derived from human labeling interactions, then topic labels can be recommended per cluster instead of per statement. This research question aims to identify the usefulness of several state-of-the-art clustering algorithms to unearth the assumed underlying structure in the data.

1.3 Structure

This work is structured as follows:

Chapter 2 gives an overview on the relevant research related to this work. We will introduce the work who led to the document representations used, show the course of recent developments in topical text clustering and highlight some similar, contemporary research.

Chapter 3 explains how we address the stated research questions and also provides the fundamental knowledge to understand the experiments. We begin with an explanation how our dataset is assembled from the received database export in Section 3.1. In Section 3.2 we explain how the document representations are evaluated. For this we give a short introduction into the used metrics. Then we explain our experiment design in detail, where we also give a more detailed introduction into the evaluated document representations.

Subsequently, we describe the evaluation of clustering algorithms in Section 3.3. Again we describe the used metrics as well as the clustering algorithms under investigation. After that, we close with the description of the experiment design.

In chapter 4 the results of the experiments are given and discussed.

Based on the insights we gained from our experiments we created a prototype described in chapter 5. Besides the prototype we also describe the algorithm applied by our industry partner so far. We also conduct an experiment to compare the performance of both approaches.

In chapter 6, we summarize our findings, point out the limitations of this work and give ideas for future research.

2 Related Work

This chapter gives a short overview over the body of related work. Document representations used in this work are introduced in chronological order. This is followed by a very short introduction in topical text clustering using document embeddings. To give a context for this work, some contemporary research dealing with similar topics is highlighted as well.

2.1 Document Representation

Thanks to the work of Salton et. al [31, 32] TF-IDF weighted vector space model became the predominant document representation for similarity calculations of text documents. In this document representation the documents in a corpus are represented by a vector of term weights. Every term in the vocabulary corresponds with one element in this vector. This leads to large and typically sparse document representations. If the document vectors of a corpus are written as rows of a matrix, this creates the document-term matrix.

Deerwester et al. [11] introduced Latent Semantic Indexing (LSI) in 1990. They used singular-value-decomposition to decompose large term-document matrices into a predefined number of orthogonal factors. These factors are used to construct a semantic vector space and documents are approximated by points in this space. Typically only the k largest factors are used to represent a document. Therefore Latent Semantic Indexing (LSI) can be seen as a lossy compression of the documents resulting in a document vector of predefined size k.

Later Blei et al. [6] introduced a probabilistic model for document representation called Latent Dirichlet Allocation (LDA). Latent Dirichlet Allocation (LDA) is a three-level hierarchical probabilistic mixture model. The top level represents the document which is modeled as a mixture of topic probabilities. The second layer contains the topics which are modeled as mixtures of word probabilities from the third layer. Documents can be represented by vectors of their mixture coefficients. Like for LSI the size k of these vector is a model parameter.

The elements of a document vector form LSI or LDA can be interpreted as cluster labels membership scores but we do not follow this approach. Instead, we use these topic modes solely for the creation of document vectors, which are serve as input to the cluster algorithms we investigate.

It is also possible to concatenate documents to a longer document before using them to generate a model. Mehrotra et al. [25] showed that clustering performance on LDA representations of short documents can be significantly increased by using document pooling techniques. Their dataset was composed of Twitter messages pooled by their hash tags. This is particularly significant for this work, since hash tags can be seen to a degree as topic labels. While we do not user pooling techniques to compare the different models, the baseline approach in chapter 5 does.

Embeddings became popular in 2013 when Mikolov et al. introduced the Word2vec embedding [26, 27]. Word2vec uses artificial neural networks to learn vector representations of words. Neural networks have been used before to create word embeddings but Word2vec improved performance at lower computational cost. Document representations can be created from the word vectors of the terms they contain. In this work we chose to represent documents by the averaged word vectors of the terms they contain.

In 2014, Doc2vec was introduced under the name ”Paragraph Vector“ by Le and Mikolov [21]. The Doc2vec approach learns vector representations of sentences, paragraphs or even complete documents. This requires an additional step, which makes the model slower in comparison to Word2vec but in many cases Doc2vec outperforms Word2vec.

It achieved lower error rates in sentiment analysis and information retrieval [21] and performed better in duplicate question detection [20].

Another challenge is how to handle terms for which no vector representation was created because the word was not encountered during model training. This typically happens when a word is rare or misspelled. This issue is addressed by the FastText model introduced by Bojanowski et al. [7]. FastText learns vectors not only for complete words but also for word fragments. When the model encounters a word that does not appear in the training data, it can calculate a vector for that word by adding the vectors of its fragments. This use of sub-word information takes word morphology into account which is especially helpful when the model is trained on a small corpus. Less frequent words may not be present in a smaller corpus, but their fragments probably are. Therefore, FastText can assign vectors to words in situations where Word2vec cannot.

2.2 Similarity Measures

A wide variety of similarity measures have been used in text document clustering [18]. The notion of similarity and dissimilarity is in the core of every clustering problem. While some clustering algorithms are fixed to a particular similarity metric, for others it is interchangeable. In this work we will focus mostly on cosine similarity and Euclidean distance.

2.3 Topical Clustering

Clustering is a technique for data analysis which is used in many fields. Definitions for clustering developed over time from intuitive approaches to more formal ones. On page 7 of his book “Cluster Analysis” [15] Everitt notes the difficulty of a formal definition for “cluster”.

Clustering divides data into groups (or partitions) of similar objects. These groups are called clusters. Objects within one cluster should be similar to each other while dissimilar to objects outside of their cluster. If clusters are disjoint then this is called a hard (or crisp) clustering. If they overlap then we speak of soft clustering.

Using word or document embeddings for text clustering is a relatively new trend. Alnajran et al. [2] reviewed 13 papers on cluster analysis of Twitter data from 2011 to 2017. None of them used embeddings as features.

Similarly, the "Brief Survey of Text Mining" from Allahyari et al. [1] from 2017 did not even mention embeddings. It's worth mentioning, that they stated tree challenges for clustering text data:

- Text representations are sparse although the dimensionality, is high. This is especially true for short documents like tweets.
- The correlation of words with each other needs to be considered.
- The need for normalization of document representations during the clustering process.

All these challenges can and have been mitigated by the use of word or document embeddings since then.

In their survey on topic detection from Twitter streams, Ibrahim et al. [19] give a taxonomy of topic detection techniques. In their taxonomy topic detection techniques are divided into five categories: clustering, frequent pattern mining, Exemplar-based, matrix factorization, and probabilistic models. Except for "Frequent Pattern Mining", we use at least one technique from every main category in this work. Although we don't use LSI and LDA directly for clustering but instead as document representation before the clustering process.

Dai et al. [10] proposed a text clustering method based on word embeddings for Twitter data. Following a pre-processing step, words are represented as word vectors. Cluster analysis is performed on this vectors to create cluster of semantically similar words. Every cluster is represented by a cluster vector, which is calculated by averaging all

contained word vectors. Further they choose a key word for a topic they are interested in and represent this word by its embedding, called the topic vector. Clusters are considered related to topics based on the similarity between topic vectors and cluster vectors. If a tweet contains words from a cluster which is related to a topic, then the whole tweet is considered related to a topic.

Fraj et al. [16] also represent Word2vec embeddings and cluster words in topical clusters. Tweets are further represented as TF-IDF vectors, but here the word clusters serve as terms instead of the words themselves. This gives not only smaller vectors, but also captures semantic similarities between the words. As final step, k-means is applied to the tweet vectors to generate topical clusters.

Li et al. [23] tested Word2vec embeddings for sentiment analysis and topic classification on data sets comprised of Twitter statements and general data. They found that their best performing training data set for topic classification of Twitter statements was a combination of Twitter statements and general text data aggregated from news articles, Wikipedia and two other published corpora.

Dorani et al. [13] investigated the performance of different word embeddings in comment clustering. They compared Word2vec, Glove and FastText embeddings with a traditional TF-IDF representation on several data sets. Embeddings consistently outperformed TF-IDF and Word2vec showed the best performance in terms of silhouette score.

Curiskis et al. [9] evaluated techniques for document clustering and topic modeling on online social network data. They used several document representations including Doc2vec embeddings, the mean of Word2vec embeddings and a Latent Dirichlet Allocation (LDA) topic model. The clustering algorithms used include k-means, hierarchical agglomerative clustering and others. This paper is especially noteworthy in the context of our work, since Curiskis et al. have followed a very similar approach and there is also an overlap in the investigated algorithms.

2.4 Summary

Our work draws from a long tradition of document representations and clustering approaches. Following the vector space model [31, 32], we represent documents as vectors in a semantic vector space. This allows us to use measures such as cosine similarity and Euclidean distance. The vectors are generated either from one of the aforementioned topic models [11, 6] or one of the mentioned embeddings [26, 27, 21, 7]. By using the vectors of text documents as input to state-of-the-art clustering algorithms, we implicitly address the three challenges described by Allahyari et al. [1] for clustering text documents: We avoid sparse document representations, consider word correlation, and create normalized document representations.

We highlight several recent publications in the area of text clustering using word or document embeddings. Some, such as Dai et al. [10] and Fraj et al. [16] address the challenge by presenting new clustering methods. We take a different approach by evaluating established standard clustering approaches with different document representation approaches and distance measures. Similar work was published by Dorani et al. [13] and Curiskis et al. [9]. We add to the area by introducing additional document representation and clustering approaches not considered in these works. By focusing on short policy statements in German, we extend the focus to an additional language and domain.

3 Method

This chapter explains how we answer the research questions.

The first section describes the creation of the dataset. We begin with a description of the database dumps we received from our industry partner, from which we assemble our dataset. First we describe the database structure and give a short overview about its contents. Then we explain how we select and clean the data to remove duplicate statements. Our text preprocessing routine is explained as well. Finally, we explain our approach to calculate topic scores and cluster labels from the expert labels in the database.

We approach our research questions in sequential order, treating the second research question as dependent on the answer to the first one. Otherwise, the parameters space to explore for answering the second research question would be too large.

To answer research question RQ1 (*"Which method is most suitable to represent statements in similarity calculations?"*) we apply the document representation algorithms to the collected statements. We calculate pairwise similarity between these vectors. To judge the usefulness of the document representations for similarity calculations, we facilitate the calculated statement similarities in a simple topic label recommender.

For research question RQ2 (*"Which state of the art clustering algorithm works best to categorize short political statements?"*) we apply state of the art clustering algorithms to the vectors created by the most promising document representation. The resulting clusters are then compared to the partitions derived from expert labels.

3.1 Dataset

The data we use in the course of this work is provided to us by our industry partner in form of database exports from their deployed database. Their database is constantly updated in real time and therefore our dataset is based on a snapshot of their live system.

In this section we first describe the contents and structure of the database export. Then we explain how we derive our dataset from the database.

3.1.1 Description of the database

Our partner provides us with a database export in MySQL format. The database contains statements of politicians made online as well as labeling interactions where experts assigned topical labels to the statements. It also contains meta information about the politicians who made the statements.

To prepare the data for analysis the SQL file was imported into an instance of the open source MariaDB database management system. As an initial step we analyzed and documented the structure of the relational database.

Database structure

The structure of the database export is shown in Figure 3.1. The data base consists of 14 tables, which hold various information on politicians, political positions, and of course the political statements collected from online sources.

Over the course of the project, we received two database exports. The second export extends the first export with additional labels and statements. We requested the second export, because we suspected that additional labels would increase the performance of label recommendation and the original dataset was too small to verify this hypothesis.

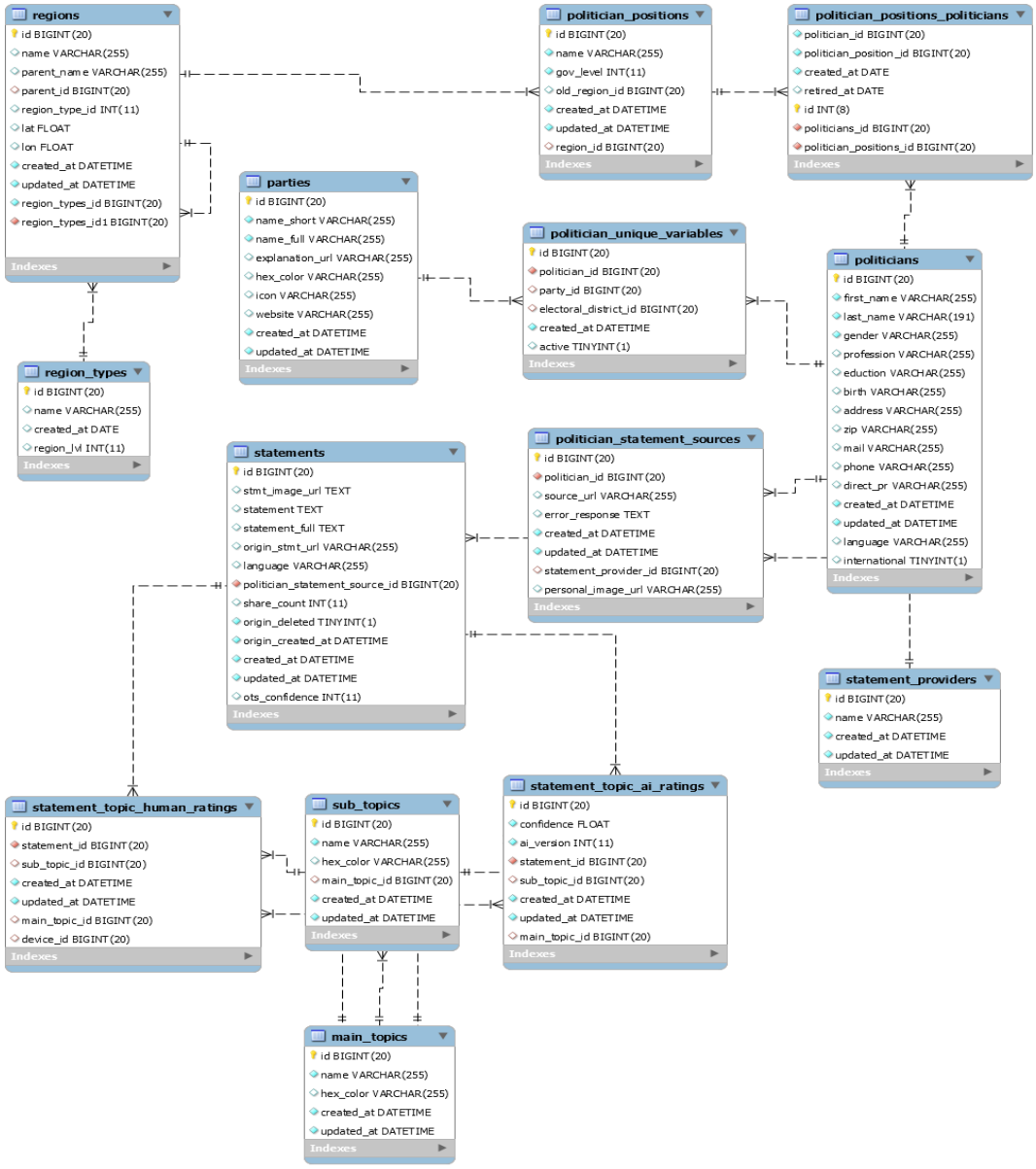

Figure 3.1: Database structure

The first export contained a total of 1175680 statements, where 5724 statements were labeled by human experts. With the second export the number of statements increased to 1489267 and the number of labeled statements is now 7523.

	Number of entries	
	1st export	**2nd export**
topics	9	9
political parties	123	124
politicians	5351	5646
labeling interactions	8505	10539
labeled statements	5724	7523
statements	1175680	1496790

Table 3.1: Entries in the 1st and 2nd database export.

A side by side comparison of the two exports is shown in Table 3.1.

Table 3.1 shows the number of entries for some key features in the first and second export. The number of collected statements increased by 321110 and the number of recorded labeling interaction increased by 2034.

Most of the parties in the dataset are parties from Austria and Germany, followed by pan-European political parties. Also governing parties from other European countries and even some parties from non-European countries are listed.

Statement distribution

Figure 3.2 pictures the distribution of the topic labels in the dataset, which shows non-uniform characteristics. Note that one statement can have several labels. Therefore the sum of the labels is larger than the number of labeled statements.

As Figure 3.3 shows, all labeled statements originate from members of Austrian or German parties. The prevalence of statements from Austria originates in the company's market disposition. The company is rooted in Austria and expands from there to other countries.

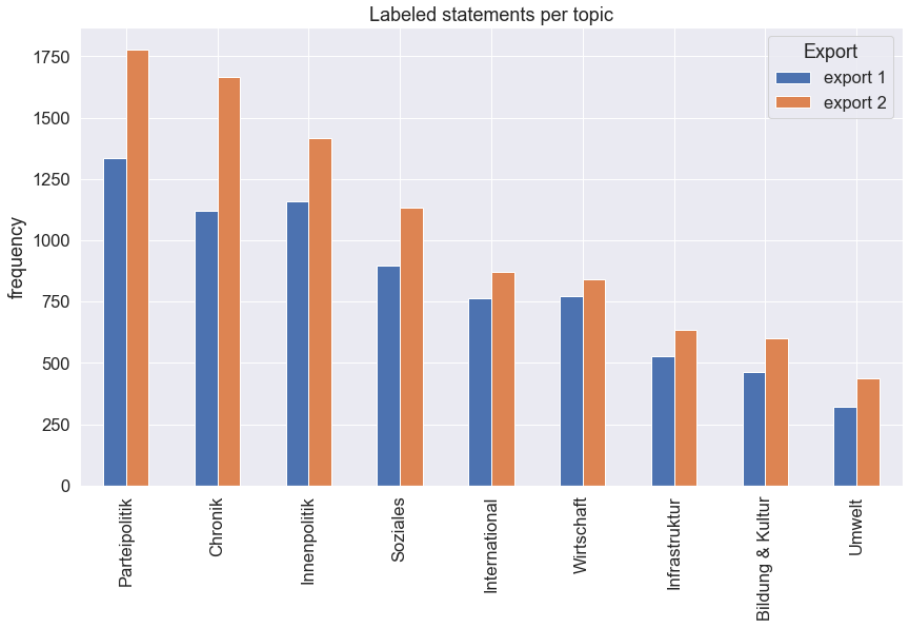

Figure 3.2: Distribution of topic labels

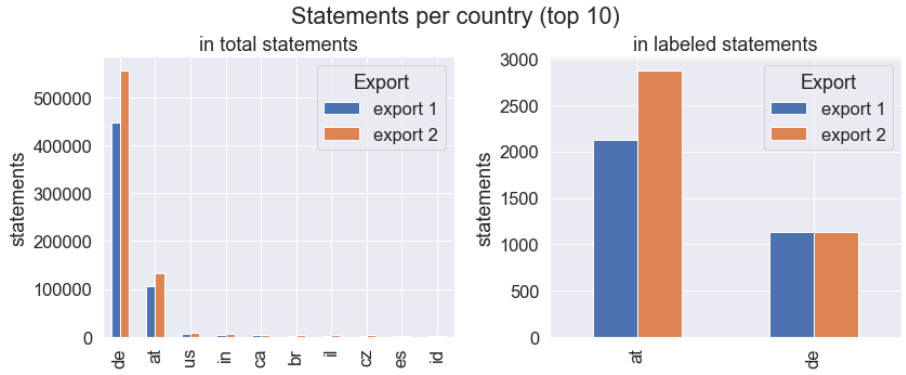

Figure 3.3: Origin countries of statements

The distribution of labeled statements per party shows non-uniform characteristics as well. This is caused by differences in size and online activity of the parties.

Most of the labeled statements were collected from Twitter and Facebook as shown in Figure 3.4. A relatively small amount of labeled statements was collected from press releases at the “Originaltext-Service” (OTS) of the Austrian press agency.

Figure 3.4: Statements per provider name

Figures 3.5 and 3.6 show the number of labeled statements per topic of the five most represented parties from Austria and Germany. We find that all major parties make statements on every topic category. However, the emphasis on each issue differs, sometimes significantly, across parties.

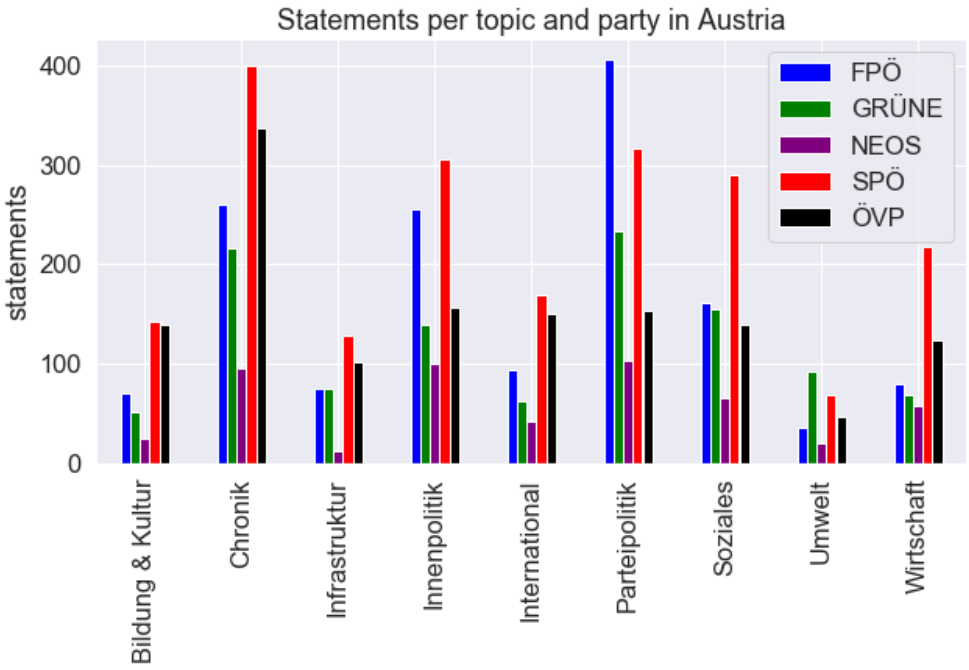

Figure 3.5: Statements per topic and party in Austria.

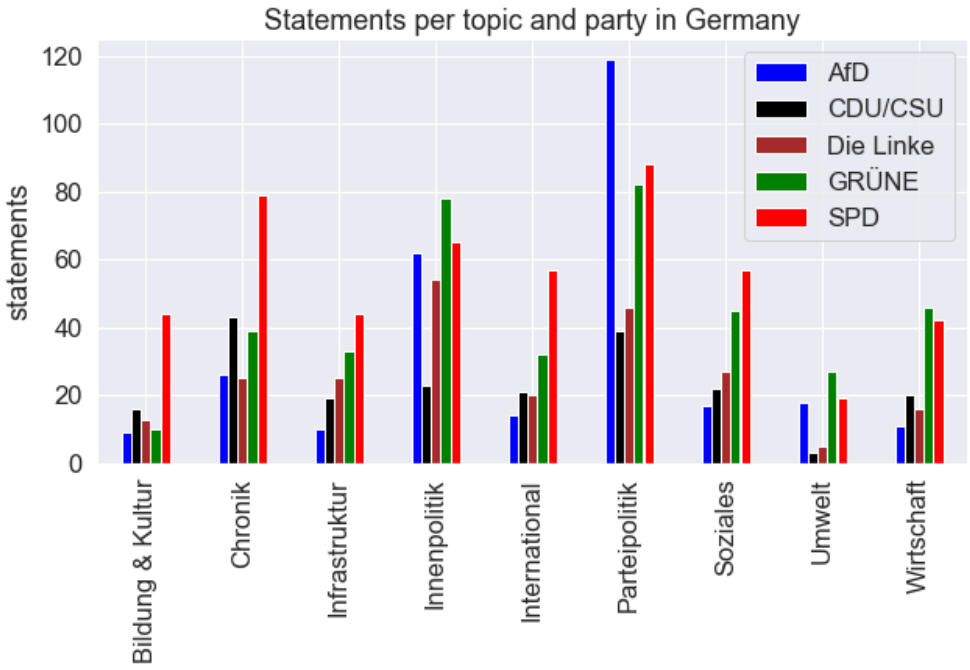

Figure 3.6: Statements per topic and party in Germany.

3.1.2 Creation of the dataset

This subsection describes how we generate our dataset from the database. We explain which data is selected and how it is pre-processed.

Since the goal of this work is to explore the topical clustering of the statements, we focus on the statement text and the topic labels. The unique statement-ID is kept and the recorded labeling interactions are used to create topic-label-vectors and topic-score-vectors later on.

Data cleaning and text preprocessing

We observed that some of the statements are highly similar or duplicates. This usually happens if the same statement is posted to two different channels in parallel. Another common source of duplicate statements is when a politician posted the same statement a second time because a hash-tag or link was missing. When we perform a random split into a training-set and a test-set, the same statement may appear in both sets. If this happens, then the measured performance will be better than the actual performance of the algorithm. Therefore, we consider statements where the first 100 characters are exactly the same as duplicate statements and keep only the first one we encounter.

The remaining statements are preprocessed as follows:

1. First we use regular expressions to remove URLs.
2. Words starting with an '@' are removed as well.
3. All non-alphanumeric characters are replaced with a space character (Unicode U+0020).
4. Remove OTS headers: Texts from Austrian Press Agency's Originaltext-Service (OTS) start with an OTS specific header. If such a header is present then this header is removed.

5. Split the text into tokens: At this point the text is split into tokens. Tokens shorter than 2 characters or longer than 15 characters are dropped. Tokens containing only numbers are dropped as well.

Topic labels and scores

The database contains labeling interactions where human experts have assigned a topic label to a statement.

We calculate a topic-score vector $\boldsymbol{s}$ for every statement where at least one label was assigned by an expert.

In Equation 3.1, we define $\boldsymbol{s}$ as a vector consisting of a topic scores for every of the T topics.

$$\boldsymbol{s} = [s_1, \ldots, s_T] \tag{3.1}$$

Equation 3.2 shows how our topic score is calculated. The topic score is a value between zero and one. To calculate it, we count the number of times the statement has received a particular topic label and divide this by the number of all topic labels assigned to this statement.

$$s_t = \frac{\text{number of times the topic label } t \text{ was assigned}}{\text{total number of labels assigned}} \tag{3.2}$$

The statements topic-label vector $\boldsymbol{l}$ indicates, if a topic label has been assigned to the statement at least once or not. If the topic-label was assigned to the statement, then the corresponding entry is one — otherwise the entry is zero. This means, that the topic-label vector can be easily derived from the topic-score vector as shown in Equation 3.3.

$$\begin{aligned} \boldsymbol{l} &= [l_1, \ldots, l_T] \\ &= [\lceil s_1 \rceil, \ldots, \lceil s_T \rceil] \end{aligned} \tag{3.3}$$

3.2 Evaluation of document representations

In this section we evaluate state-of-the-art document representation algorithms for their usefulness in feature engineering. Our goal is to answer research question RQ1: ("*Which method is most suitable to represent statements in similarity calculations?*").

The dataset does not contain statement similarity. Therefore, we can not use the straight forward approach of calculating statement similarities and compare them to the ground truth. Instead, we measure the achieved performance when the document representations are used in a topic-label recommender. We argue that if the same documents are used to train a simple recommender system and only the feature representation changes, the observed performance changes are due to the document representation approach chosen.

During this section we use the document representation algorithms to create document vectors. This document vectors serve as training data for a similarity based recommender. We optimize the preprocessor settings and the recommender parameters to get the best performance with every document representation algorithm. Then we use information retrieval metrics to measure the performance of the optimized system.

We consider a document-representation useful if it leads to good results, when used in a similarity based recommender.

3.2.1 Similarity metric

To create a similarity based recommender, a similarity metric is required. We use the cosine similarity [4, chap. 3.2.6]. Equation 3.4 shows the cosine similarity between two vectors $\mathbf{d}$ and $\mathbf{q}$.

$$sim_{cos}\left(\boldsymbol{d}, \boldsymbol{q}\right) = \frac{\boldsymbol{d} \cdot \boldsymbol{q}}{\|\boldsymbol{d}\|_2 \cdot \|\boldsymbol{q}\|_2} \tag{3.4}$$

Here $\|\cdot\|_2$ denotes the Euclidean vector norm.

The cosine similarity measure can take values between -1 and 1. A cosine similarity of 1 means that two vectors point exactly in the same direction, while a cosine similarity of -1 means that the vectors are pointing in exactly opposite directions. An angle of 90° gives a cosine similarity of 0.

3.2.2 Information retrieval metrics

We use Precision, Recall and F1-Measure to evaluate the performance of the label recommendation and for also to determine the optimal parameter settings.

We denote the set of labels predicted for statement d as $\widehat{L}_d$ by the system under investigation. L_d is the set of Labels for statement d in the test-set. These labels were assigned to the statement d by human experts beforehand. These labels are considered to be "relevant" to the statement.

Precision $P@k$ gives us the portion of the k recommended labels which are relevant to the statement d:

$$P@k = \frac{|L_d \cap \widehat{L}_d|}{|\widehat{L}_d|} = \frac{\text{Nr. correctly predicted labels}}{\text{Nr. predicted labels}} \tag{3.5}$$

Recall $R@k$ gives us the portion of relevant labels which were assigned to statement d by the system under investigation.

$$R@k = \frac{|L_d \cap \widehat{L}_d|}{|L_d|} = \frac{\text{Nr. correctly predicted labels}}{\text{Nr. relevant labels}} \tag{3.6}$$

The $F1$ score is the harmonic mean of precision and recall.

$$F1@k = \frac{2 \cdot P@k \cdot R@k}{P@k + R@k} \tag{3.7}$$

When we report $P@k$, $R@k$ and $F1@k$ for the test-set, we always report the average over all statements in the set.

3.2.3 Training and test set

We select all German language statements from Austria which are longer than 51 characters from the data set. We decided for these conditions because the system currently in use uses these settings as well. This way, our experiment takes place under conditions that resemble a production environment.

We randomly split our data set into a training and a test set, where the training set contains 80% of the statements and the test set 20%. The training set is used for training and validation while the test set is used for performance measurements.

3.2.4 Preprocessing

Preprocessing has a strong impact on the performance of the different algorithms. We expected that some of the preprocessing steps would impact the different document representation approaches in a different way. This leads to an unfair comparison between the document representations. To tackle this issue, we also optimize some of the preprocessing together with the recommender parameters. To do so, we add optional preprocessing steps. This allows us to optimize preprocessing together with the document representation algorithms. This way we aim to create stronger baselines and a fairer comparison between the algorithms.

The optimizable preprocessor parameters are:

- Stemming: yes / no
- Lemmatization: yes / no
- Stop word list:
  - None
  - common words
  - politician names
  - common words + politician names
- Filter for nouns: yes / no

3.2.5 Creation of document vectors

After preprocessing, the statements should be represented as vectors. To do so, we train different language representation model on our data. Then we use these models to represent the statements in a vector space.

We evaluate several state of the art document representation algorithms: LSI, LDA, Word2vec, FastText, and Doc2vec.

Here we give an overview on the algorithms under investigation, explain how we use them to create document vectors. We also define for which parameters we optimize before the algorithms performance is compared.

Vector size

For ease of comparison, we decided to choose the same vector size for all document representation algorithms, taking into account the expected model quality and the computational resources required. Vector size affects quality in that larger vectors tend to produce better results, but the relationship is not linear. For a similar dataset, Li et al. [23] observed that performance did not increase significantly for word embeddings above a size of 300. Smaller vector sizes lead to smaller models and faster similarity computations. Our partner uses a LSI model with a vector space of size 200 in the existing system. For our investigations, we choose a vector space dimension of size $k = 200$. We consider this an acceptable trade-off between computational efficiency and predictive ability. This also gives us another advantage: since we use the same size as in the existing system, it is also easier to make comparisons with the system later.

Latent Semantic Indexing

LSI [11] was introduced as a method for automatic indexing and retrieval in 1990.

LSI is based on the singular value decomposition of a term-document matrix created from a collection of documents. This results in a set of orthogonal factors spanning a semantic vector space. By only using the k largest singular values the resulting vector space is reduced to dimension k. All documents which are part of the training set are therefore approximated by a vector of dimension k within this space.

For our investigations we created a vector space of dimension $k = 200$. The entries in our term-document matrix are weighted with the "term frequency – inverse document frequency" (TF-IDF) weighting scheme [31]. We used the cosine-similarity between the vectors in the semantic space as document similarity metric. Since the documents we want to represent are not part of the training set, we approximated their document vector with the fold-in method described by Deerwester et al. [11].

Latent Dirichlet Allocation

LDA is a generative, probabilistic model of a corpus. Documents are modeled as mixture distribution of latent topics. The latent topics are also mixture distributions of word probabilities. These explicit representations of documents can be viewed as vectors in a semantic space spanned by the LDA topics. [6]

Important parameters are:

- the number of topics in the LDA model, which is also the size of the resulting vector space.
- a parameter alpha, that controls the shape of the topic distribution
- and optional the a-priory word probabilities.

We do not make assumptions on the distribution of topics or the word probabilities. Therefore, we do not set a value for the two associated parameters and let the algorithm learn them from the supplied data. We choose the number of topics to be 200, which gives a vector space with 200 dimensions.

We train this model on our training set and then we use it to represent new statements in the found vector space.

Word2vec

Word2vec uses neural networks to learn vector representations of words. The words in a defined proximity of a word are denoted as the word's context. The neural network is trained using the words and their context. If the neural network is trained to predict a word from its context then this is called a Continuous Bag-of-Words model (CBOW). If the context is predicted from the word then this is called a Continuous Skip-gram model. [26]

We train Word2vec CBOW model on the complete corpus of collected political statements. The dimensions of the resulting word vectors is determined by a model parameter which we choose to be 200.
We derived the word vectors for all words in a statement and then used the average of these vectors as document vector.

FastText

FastText aims to enrich Word2vec with sub-word information. It is based on the Word2vec model but the words are represented as bag of character n-grams. [7]

We choose 200 for the vector size of the word vectors. The document vectors are created the same way as for the Word2vec model: by averaging the word vectors. The FastText model is trained on the complete corpus of collected statements.

Doc2Vec

Doc2vec or Paragraph Vector is an algorithm that builds on Word2vec. Doc2vec extends the Word2vec model by adding a document vector which represents the particular para-

graph or document. Document vectors for unseen documents are inferred at prediction time [21].

We train a Doc2vec model with 200 dimensions on the complete corpus of collected statements and use this model to create document vectors of new statements.

3.2.6 Label recommendation

We use a simple k-nearest-neighbor recommender. The recommender holds instances of labeled document vectors. The k most similar statements are considered to be neighbors of the statement under investigation. The label which is most common for the neighbors is recommended to the statement under investigation. Statements are considered similar if the cosine similarity given in Equation 3.4 is large. The neighborhood size k is the only hyper-parameter for the recommender.

3.2.7 Optimization

For the optional preprocessing steps, we define different settings for our preprocessor and then conduct grid search on them. We do this two times for our preprocessor: At first, we search for the optimal strategy for word filtering where we investigate three different stop word sets and the removal of words which are not nouns. Then we investigate the impact of two normalization strategies: Stemming and lemmatization. For each of these investigations we keep the other parameters fixed.

We combine the best preprocessing settings into one parameter set and use it to optimize the label recommender parameter. We conduct grid search over the neighborhood size, which is the only hyperparameter of our simple recommender. For this optimization step, we optimize for the F1 score given in Equation 3.7.

Then we use the test set to measure and report the performance of the optimized system.

3.3 Evaluation of clustering algorithms

In this section we tackle research question RQ2 ("*Which state of the art clustering algorithm works best to categorize short political statements?*").

We chose the most promising document representation algorithm and use it to create vector representations of the labeled statements from our dataset. Then we apply several state-of-the-art clustering algorithms to these vectors. We vary the algorithm parameters and measure the clustering performance. Where possible, we use the cosine similarity as similarity metric. If this is not the possible we use Euclidean distance instead.

3.3.1 Evaluation metrics for clustering

We decide to use two metrics for cluster evaluation: Silhouette score and Adjusted Mutual Information (AMI). Silhouette score serves as an indicator for the cluster structure. We use AMI to compare the clustering results with the clustered extracted from the expert labels.

Silhouette score

The silhouette score is calculated for every clustered item and can take a value from -1 to 1. The silhouettes sore measures how well the item matches clustering at hand.

Cluster silhouettes were mainly introduced as a graphical representation for cluster evaluation: The items of a cluster are sorted by their silhouette scores and then the scores are plotted as a horizontal bar chart or histogram. Besides the graphical inspection of the silhouette plot, the average silhouette scores are useful as well. Clusters with a large average silhouette score are more pronounced. If the output of a clustering algorithm leads to a very large overall silhouette score this can be taken as indication that the algorithm discovered a very strong clustering structure. [30]

We denote the average dissimilarity of an item i to all other items in the same cluster as $a\,(i)$ and the average dissimilarity to all items of the closest other cluster as $b(i)$.

The silhouette score $s(i)$ for an item i is given by Equation 3.8.

$$s(i) = \frac{b\,(i) - a(i)}{max(a\,(i)\,, b\,(i))} \tag{3.8}$$

Equation 3.8 requires a measure of dissimilarity on which the Silhouette score is based. The obvious choices are Euclidean distance and cosine distance.

Equation 3.9 shows how cosine distance between two vectors $\boldsymbol{d}$ and $\boldsymbol{q}$ is calculated. Cosine distance is based on the cosine similarity we already covered in Equation 3.4.

$$\begin{aligned} dist_{cos}(\boldsymbol{d}, \boldsymbol{q}) &= 1 - sim_{cos}(\boldsymbol{d}, \boldsymbol{q}) \\ &= 1 - \frac{\boldsymbol{d} \cdot \boldsymbol{q}}{\|\boldsymbol{d}\|_2 \cdot \|\boldsymbol{q}\|_2} \end{aligned} \tag{3.9}$$

We calculate and report silhouette scores based on both dissimilarity metrics.

Adjusted Mutual Information (AMI)

AMI is an information theoretic measure introduced in 2009 by Vinh et al. [34]. AMI is a measure for the similarity of two clusterings and is based on the mutual Information measure (MI). AMI adjusts MI for chance to reduce MI's bias towards a higher number of clusters.

Given two clusterings U and V, we denote the entropy of the clustering U as $H(U)$ and the entropy of clustering V as $H(V)$. The expectation for the mutual information between U and V is denoted as $E(MI(U, V))$

The adjusted mutual information of U and V is given by Equation 3.10.

$$\mathrm{AMI}\,(U,V) = \frac{\mathrm{MI}\,(U,V) - E(\mathrm{MI}\,(U,V))}{\frac{H(U)+H(V)}{2} - E(\mathrm{MI}\,(U,V)} \tag{3.10}$$

Note that there are alternative definitions of AMI which are also given in the original paper [34].

3.3.2 Data selection and preprocessing

The corpus contains over 1.2 million statements written in German language. For the evaluation of the clustering algorithms, the 2656 labeled, unique, German language statements from Austria are used as data points. The restriction to one country is done to account for the fact that country context contributes to the experts' labeling decision. For example, a statement about an event in Berlin will be considered internal politics from a German perspective, but international news from an Austrian one. The restriction to labeled statements is due to technical reasons, because some clustering algorithms cannot cluster such large amounts of data in an acceptable amount of time with the available resources. This is a relevant limitation, as clustering all statements and not only the labeled sample may lead to more meaningful results, since the clustering result would not be affected by the sampling error.

Another issue arises with the AMI metric. The expert labels do not lead to hard clusters where every statement belongs only to one topic. AMI, on the other hand, only works for data where every data point has exactly one label. Therefore, we preprocess the statement labels and leave only the label with the highest score. This new set of labels can be used with AMI.

The settings for statement preprocessing and document representation are given in chapter Results and Discussion for consistency reasons.

3.3.3 Clustering algorithms

In this section we describe the clustering algorithms under investigation. We selected several state-of-the-art standard algorithms.

For our investigation we used the implementation of the open source machine learning toolkit Scikit-learn [28].

K-Means

The k-means algorithm was introduced as a quantization algorithm for pulse-code modulation [24]. K-means partitions a data set into k non overlapping sets. The sets are defined by their centroids and form convex clusters. The number of the clusters must be defined as a parameter. K-means is a fast and relatively simple clustering algorithm that scales well. It uses Euclidean distance as a distance measure. Since k-means effectively assigns data points to the clusters with the closest cluster center. It works best on data of spherical or hyper-spherical structure.

We vary the number of cluster and optimize for the highest AMI.

Spectral Clustering

Spectral clustering treats the data segmentation problem as a graph partitioning problem. Data points are viewed as nodes in a graph. The edges between the nodes are weighted by their similarity. Then the eigenvalues (spectrum) of the graph's Laplacian matrix are used to partition the graph. Usually this is done by facilitating a standard clustering algorithm like k-means but other approaches like the normalized cut criterion exist as well [33].

Spectral clustering takes the number of clusters as parameter. The nodes in the graph can be weighted by different similarity measures. The implementation that comes with Scikit-learn [28] knows several similarity-measures out of the box but not cosine-similarity. It

is however possible to calculate the pairwise cosine-similarities beforehand and use them as an input parameter.

Due to its graph based approach, Spectral Clustering is able to find highly non-convex clusters.

We optimize for AMI by varying the number of clusters. This is done twice: During the first run the graph is constructed based on the Euclidean nearest neighbors. In the second run, the graph is based on the precomputed cosine similarity between the statements instead.

In the further course of this work we distinguish these two runs by denominating the latter as “Spectral Clustering (cos)”.

Agglomerative Clustering

The agglomerative clustering approach recursively merges the pair of clusters that minimally increases a given linkage distance. We chose Ward’s minimum variance method [35] as linkage distance. For comparison, we also cluster with the single-linkage method, where always the two clusters are merged that contain the two closest points which are not from the same cluster.

For the single-linkage clustering we use cosine-similarity as affinity measure. Ward’s method only works on Euclidean distances. We set the desired number of clusters as stopping criterion for cluster merging. To find the optimal parameters we vary this number and optimize for the largest AMI.

Gaussian Mixtures

We use the expectation-maximization algorithm [12] to fit a Gaussian-mixture-model as described in chapters 2 and 9 of Bishop’s book on pattern recognition and machine learning [5].

We set the number of components to the desired number of clusters and optimize for the largest AMI.

Birch

"Balanced Iterative Reducing and Clustering using Hierarchies" (BIRCH) is an algorithm designed for large data-sets. First BIRCH scans the data and creates a tree structure. The leaf nodes of this tree structure represent local sub-clusters. This tree structure can be viewed as a compressed clustering problem. A global clustering algorithm is then applied to this tree for clustering the sub-clusters [37].

BIRCH takes two parameters that control the creation of the tree structure: "threshold" and "branching factor". We set the threshold to 0.5, and the branching factor to 50.

The global clustering algorithm accepts the number of clusters as parameter. Subsequently, this number is varied to optimize for the largest AMI.

Mean-Shift

Mean shift is a kernel-based algorithm which iteratively shifts the kernel towards high density regions. After the algorithm converges, near-duplicates kernels are removed and the remaining kernels are presented as cluster centroids [8].

The implementation at hand uses a flat kernel on Euclidean space and takes the kernel bandwidth as parameter.

DBSCAN

DBSCAN is a density-based clustering algorithm that can discover clusters of arbitrary shape. It takes two input parameters: a distance "eps" and a number "MinPts". DBSCAN starts from an arbitrary point and determines if the point is a "core point" or a "border point". It is a core point if at least MinPts are within the distance eps, otherwise it's a

border point. If the point is core point all points within eps are added to the cluster of the core point. If an added point is already part of a cluster, then the clusters are merged. If the point is a border point no points are added to the cluster and DBSCAN picks the next arbitrary point. [14]

In the Scikit-learn implementation [28] the MinPts parameter is named “min_samples”.

We vary both parameters to optimize for the largest AMI. This is done with euclidean distance and cosine distance metric.

OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) does not produce a clustering explicitly but an ordering of instances. This ordering contains all necessary information to extract density-based clusters from the data. Like DBSCAN, OPTICS uses a distance “eps” and a number of “MinPts”, but eps is not a parameter here. Instead, OPTICS determines the minimum eps for every point to reach MinPts neighbors. The eps value for each instance together with the distances between instances is used to create an ordered set of instances. Clusters are then extracted from this ordered set. [3]

Affinity Propagation

Affinity propagation views every data point as a node in a network. Messages are passed between nodes to determine which nodes are suited to represent a cluster (“exemplars”) and which nodes are represented by which exemplar.

Two types of messages are sent between data points: responsibility messages and availability messages. Responsibility messages indicate how well a data point is suited to represent a cluster center in the eye of the sender. Availability messages indicate how appropriate it would be for the receiver to choose the sender as an exemplar. After receiving these messages, the nodes update their beliefs. Affinity Propagation then iterates

until the beliefs converge. Messages are sent between pairs of nodes and only simple, local computation are required. [17]

This design makes it a promising choice for self-organizing systems and parallel computing, but in a classical setting the quadratic time and memory complexity is a serious drawback, especially when de data sets are very large.

The algorithm takes two parameters: a damping factor used to prohibit numerical oscillation and an optional preference list. The preference list represents a-priory knowledge on which data point are likely to be exemplars.

3.3.4 Clustering

Once we created the document vectors we use the clustering algorithms to cluster the statements. We vary the parameters of the clustering algorithm and calculate the AMI score with respect to the processed expert labels. Where possible, we also report the silhouette score as given by Equation 3.8. The silhouette score is calculated two times: First we calculate the silhouette score based on cosine similarity and then a second time with Euclidean distance.

3.3.5 Summary

We create our dataset from a database dump provided by our industry partner. The database dump contains over 1.4 million statements and 10539 labeling interactions, with statements labeled from a set of 9 different topic labels. The majority of statements are from Germany, followed by Austria, and are written in German. The statements were collected via Twitter, Facebook and the Austrian press release platform OTS.

The first step is to remove duplicate statements based on the first 100 characters. The labeling interactions associated with the statements are then used to compute a topic score vector for each labeled statement. A binary version of the topic score vector, the topic label vector, is also computed.

In order to find the most suitable method for document representation, we investigate different approaches to document representation. Two topic models (LSI and LDA) and three embeddings (Word2vec, FastText, and Doc2vec) are considered. Cosine similarity is used to calculate the similarity of the resulting document vectors. Since there is no statement similarity in the database for comparison, the suitability of the document representations is assessed based on the recommendation performance when used in a simple k-nearest-neighbor recommender. To ensure a fair comparison between the different approaches, each recommender is optimized for the best F1 score. During this step, the following things are optimized: the size of the neighborhood for the recommender, the parameters of the document representation approach, and a parameterizable preprocessor. All document vectors are created in the same size, which we choose based on efficiency, quality, and practical considerations. The most promising document representation is then used to assess the clustering algorithms.

The performance of nine state-of-the-art clustering algorithms is investigated. These are K-Means, Spectral Clustering, Agglomerative Clustering, Gaussian Mixtures, Birch, Mean-Shift, DBSCAN, OPTICS and Affinity Propagation. All labeled statements from Austria serve as input to the clustering process. The quality of the clustering is based on the Adjusted Mutual Information (AMI) with the expert labels. Since AMI requires two crisp sets of cluster labels as input, the expert labels are hardened first. The clustering parameters are optimized for the best AMI for all clustering algorithms before comparison. In addition, the silhouette score is calculated to gain a better understanding of the cluster structure.

4 Results and Discussion

This chapter aims to present the result of our experiments outlined in chapter 3 and highlight the most significant findings. We give the optimal parameters found for the document representations based on the measured performance. Then we compare the top performances for every document representation approach with each other. For the clustering experiment we give silhouette scores and AMI scores for the calculated clusters. For comparison, we also give the silhouette scores of the clusters derived from the expert labels.

4.1 Evaluation of document representations

Following the approach presented in Section 3.2, the statements were preprocessed and represented by different types of document vectors. Subsequently, these vectors are used in a simple similarity based label recommender. The parameters for every document representation approach are optimized together with the recommender parameters using grid search.

4.1.1 Optimal parameters for document representation

We followed the approach outlined in Section 3.2 to find optimal parameters for the document representation algorithm in respect to the topic label recommendation task. A simple grid search was performed for the parameters of the algorithms and the neighborhood size of the similarity-based recommender system. The parameter set that gave the

best F1 score was considered the optimal setting. Table 4.1 shows the optimal parameters obtained for every document representation algorithm and also the optimal neighborhood size for the similarity based recommender.

	normalization	**stop words**	**parts of speech used**	**number of neighbors k**
LSI	lemmatization	common words + names	nouns	9
LDA	lemmatization	common words + names	nouns	2
Word2vec	none	common words	all	22
FastText	none	common words + names	all	30
Doc2vec	none	common words + names	all	13

Table 4.1: Optimized parameters for different document representations.

Unlike the two topic models LSI and LDA, none of the embeddings worked best with one of the tested normalization strategies. This is not surprising since the embeddings were trained on the complete corpus without normalization. To train the embeddings on a normalized corpus was not feasible since normalization takes too much computing time. While this seems to be a disadvantage at first glance it is not: All three investigated embeddings possess the capability to be trained on a vast amount of text compared to the topic models. This enables them to capture the semantic similarity between non-canonical variants of a word. Beyond that, FastText even takes sub-word information into account, which makes word normalization redundant altogether.

The topic models worked best when only nouns were used as input data. Part of speech (POS) tagging is required to filter out words which are not nouns, which can be a resource intensive task. The embeddings did not require this step, which is another advantage over the two topic models.

The use of a stop word list on the other hand was beneficial to every single of the investigated algorithms.

4.1.2 Performance with different document representations

For performance comparison of the label recommendation with the different document representations, we chose a graphical representation. Figure 4.1 shows the resulting precision-recall curve for top-N recommendation with the different document representations used. The number or recommended labels N were varied from one label to all nine possible labels. We also add a random label recommendation for comparison.

The topic label recommender based on Word2vec consistently outperformed the other configurations. Word2vec is followed by FastText and Doc2vec, while the two topic models gave the lowest performance. Within the two topic models LSI outperformed the newer LDA.

A study by Curiskis et al. on document clustering [9] found that Doc2vec performed better than Word2vec in many cases. The authors pointed out that the most likely explanation for this effect is a dataset with insufficient size to accurately train the Doc2vec model. While the same might be true in our case, more experiments would be needed to test this hypothesis.

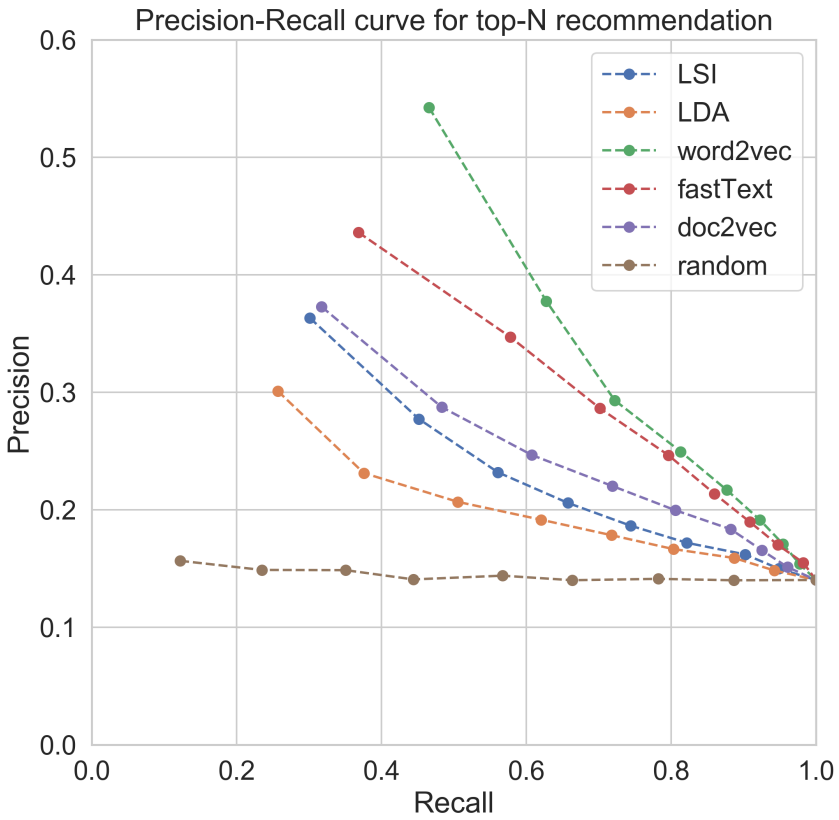

Figure 4.1: Performance with different document representations.

4.2 Evaluation of clustering algorithms

To give a better idea of the nature of the data, we first give the silhouette scores for the clusters derived from the expert labels. We optimized the different parameters of the clustering algorithm for the best AMI score and give the best parameter sets found. These results are divided into two subsections: One for the clustering algorithms that use the cluster number as a parameter, and one for those that do not. We compare the performance of all tested algorithms at the end of this section.

4.2.1 Clusters from expert labels

Before we cluster the data, we take a short look at our ground truth and calculate the overall silhouette score for the clusters derived from expert labels. These silhouette scores are given in Table 4.2. Both scores are small and negative. This suggests that the clusters are either not convex, they overlap, or both.

distance used	silhouette score
cosine	-0.017
Euclidean	-0.046

Table 4.2: Silhouette scores of expert labels.

4.2.2 Optimal clustering settings

Algorithms with the cluster number as parameter

Some clustering algorithms under investigation take the cluster number as parameter. These clustering algorithms are KMeans, Spectral Clustering, Agglomerative Clustering, clustering with Gaussian Mixtures, and Birch.

For these algorithms we varied the number of clusters to find the optimal settings with respect to the largest resulting AMI. The progression of AMI over the number of clusters is shown in figure 4.2. For the most clustering algorithms the AMI curve shows a steep increase at the beginning but saturates soon. The exception is Agglomerative Clustering with single linkage, which does not provide usable results.

The optimal settings found for clustering algorithms that take the cluster number as parameter are given in Table 4.3. In this group Spectral Clustering performs best, but not by a large margin. This is true for Spectral Clustering based on Euclidean distance calculation and for the one based on cosine similarity.

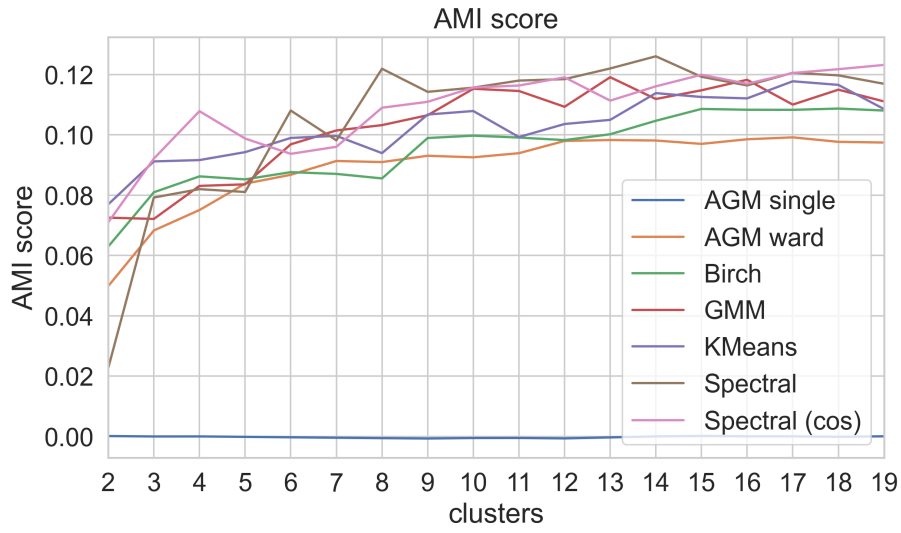

Figure 4.2: AMI score vs cluster number.

Algorithm	AMI	parameters
AGM single	0.000	n_clusters=15, affinity=cosine, linkage=single
AGM ward	0.099	n_clusters=17, affinity=euclidean, linkage=ward
Birch	0.109	n_clusters=18
GMM	0.119	n_components=13
KMeans	0.118	n_clusters=17, init=k-means++
Spectral	0.126	n_clusters=14, affinity=nearest_neighbors
Spectral (cos)	0.123	n_clusters=19, affinity=precomputed

Table 4.3: Clustering with cluster number as parameter: optimal settings

Algorithms without the cluster number as parameter

In this section we present the results for the clustering algorithms which do not take the cluster number as a parameter. These are Mean Shift, DBSCAN, OPTICS, and Affinity Propagation.

For Mean Shift we varied the bandwidth parameter to optimize for the largest AMI. Figure 4.3 shows the AMI with varying bandwidth. Note that Mean Shift is only defined on

Euclidean space.

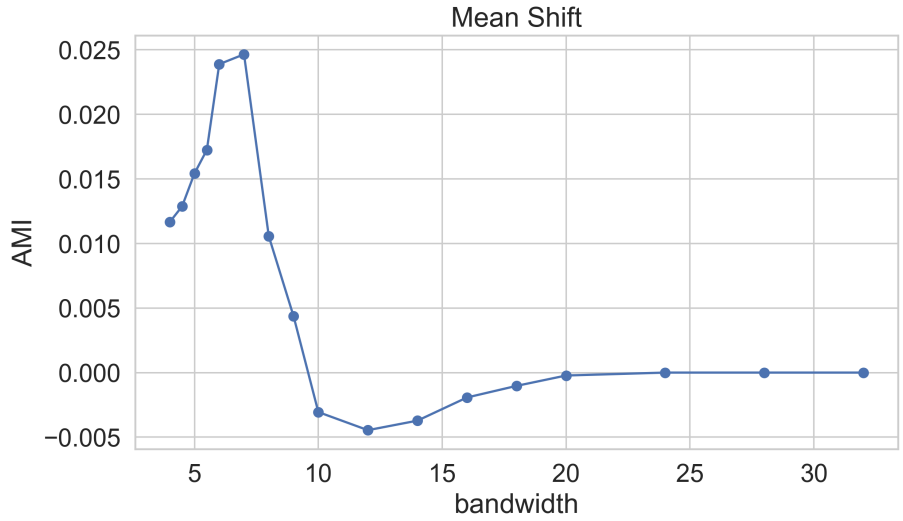

Figure 4.3: Mean Shift: AMI vs bandwidth

DBSCAN works best with MinPts set to 2. This could be an indication that the data set is too small for DBSCAN. Since DBSCAN did not provide meaningful results with the Euclidean distance metric, figure 4.4 only shows the performance with cosine distance. The best achieved AMI score is below 0.08.

OPTICS, which is similar to DBSCAN, produces meaningful results for both distance metrics, but the performance is well below DBSCAN´s. OPTICS also gives best results for MinPts set to 2.

For Affinity Propagation, we could only find one working parameter set, which we report as optimal parameter set. We also introduced an additional processing steps for this clustering algorithm, because Affinity Propagation did not produce meaningful results without them: We removed the mean and scaled our vectors to unit variance before we applied Affinity Propagation.

The found parameter set is listed in Table 4.4, together with the optimal parameters of the other clustering algorithms from this subsection.

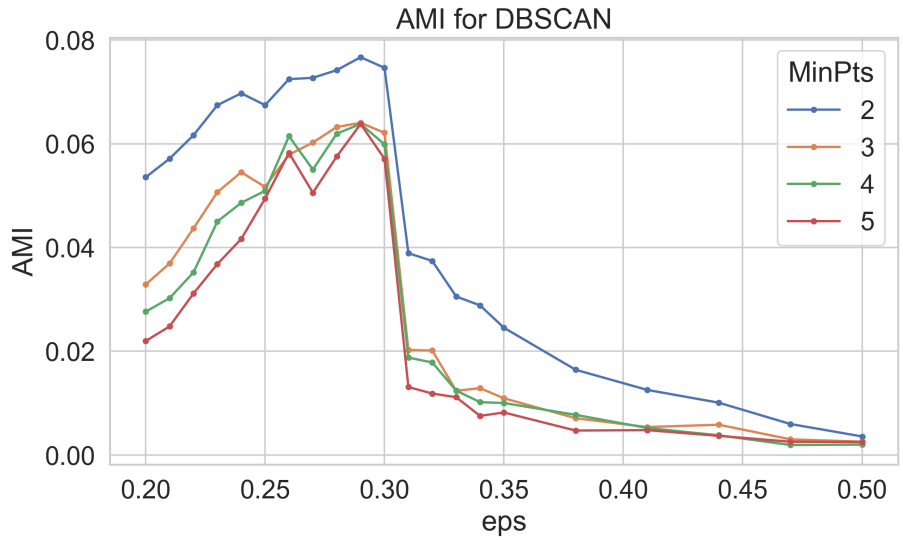

Figure 4.4: DBSCAN: AMI vs eps (cosine distance used)

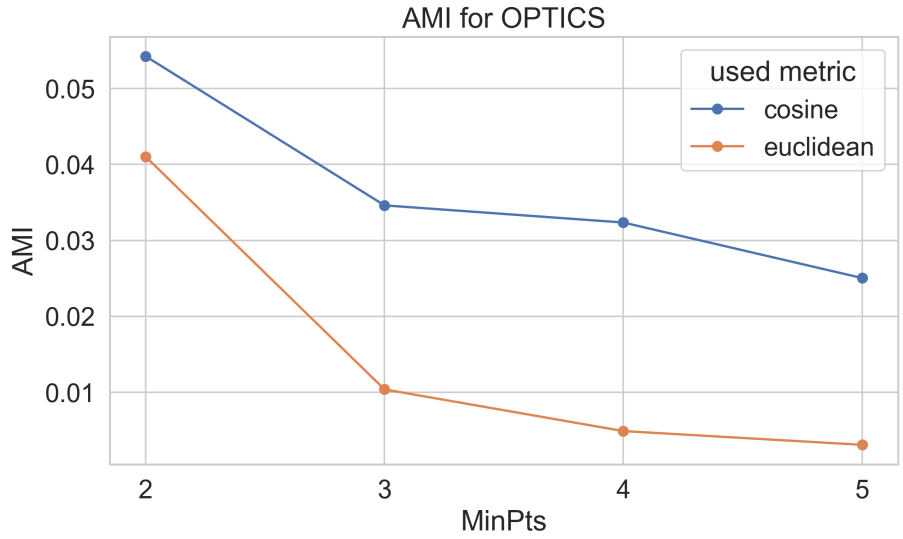

Figure 4.5: OPTICS: AMI vs min samples

Algorithm	AMI	parameters
Affinity Propagation	0.075	scaled=True, metric=cosine
DBSCAN	0.077	min_samples=2, metric=cosine, eps=0.29
Mean Shift	0.025	bandwidth=7.0, bin_seeding=False, cluster_all=True
OPTICS	0.054	min_samples=2, metric=cosine, max_eps=inf

Table 4.4: Clustering without cluster number as parameter: optimal settings

4.2.3 Clustering Performance

Table 4.5 shows the clustering performance of all investigated algorithms. Spectral Clustering outperforms all other approaches in terms of AMI. This is true for clustering with Euclidean distance as well as for clustering with cosine distance. The highest silhouette scores are also produced by the both Spectral Clustering results. This is noteworthy since Spectral Clustering does not necessary give convex clusters. Agglomerative Clustering with single linkage did not work at all. All density based clustering algorithms (DBSCAN, OPTICS, and Mean Shift) fall at the lower end in terms of AMI. This is also true for Affinity Propagation.

Algorithm	AMI	silhouette (cos)	silhouette (euclidean)
Spectral	**0.126**	-0.043	**0.028**
Spectral (cos)	**0.123**	**0.055**	0.008
GMM	0.119	0.038	0.018
KMeans	0.118	0.029	0.007
Birch	0.109	-0.016	-0.004
AGM ward	0.099	-0.005	0.001
DBSCAN	0.077	-0.282	-0.205
Affinity Propagation	0.075	-0.049	-0.070
OPTICS	0.054	-0.230	-0.163
Mean Shift	0.025	-0.286	-0.055
AGM single	0.000	-0.069	-0.273

Table 4.5: Clustering performances.

Figure 4.6 shows a visualization of the different cluster algorithm outcomes as well as the human labeling. The different clusters (or labels) are shown in different colors. Although this is just a non-linear two-dimensional projection, we can see that the clustering algorithms produce very different results compared to the human experts. This conclusion is supported by the AMI scores that we calculated from the outcomes of the cluster algorithms.

The silhouette scores are often low and sometimes negative, which indicates overlapping and/or non-convex clusters.

We were not able to produce a clustering, which resembles the labeling done by human experts. None of the above clustering algorithms seem to unearth the same structure as a human expert would. However, this conclusion is premature for two reasons. On the one hand, the data set is very small and experts often disagree on which keywords apply. On the other hand, the labels used by the experts were not derived from the data set, but were given by the company's knowledge engineer. The latter one might even be the reason for the first one.

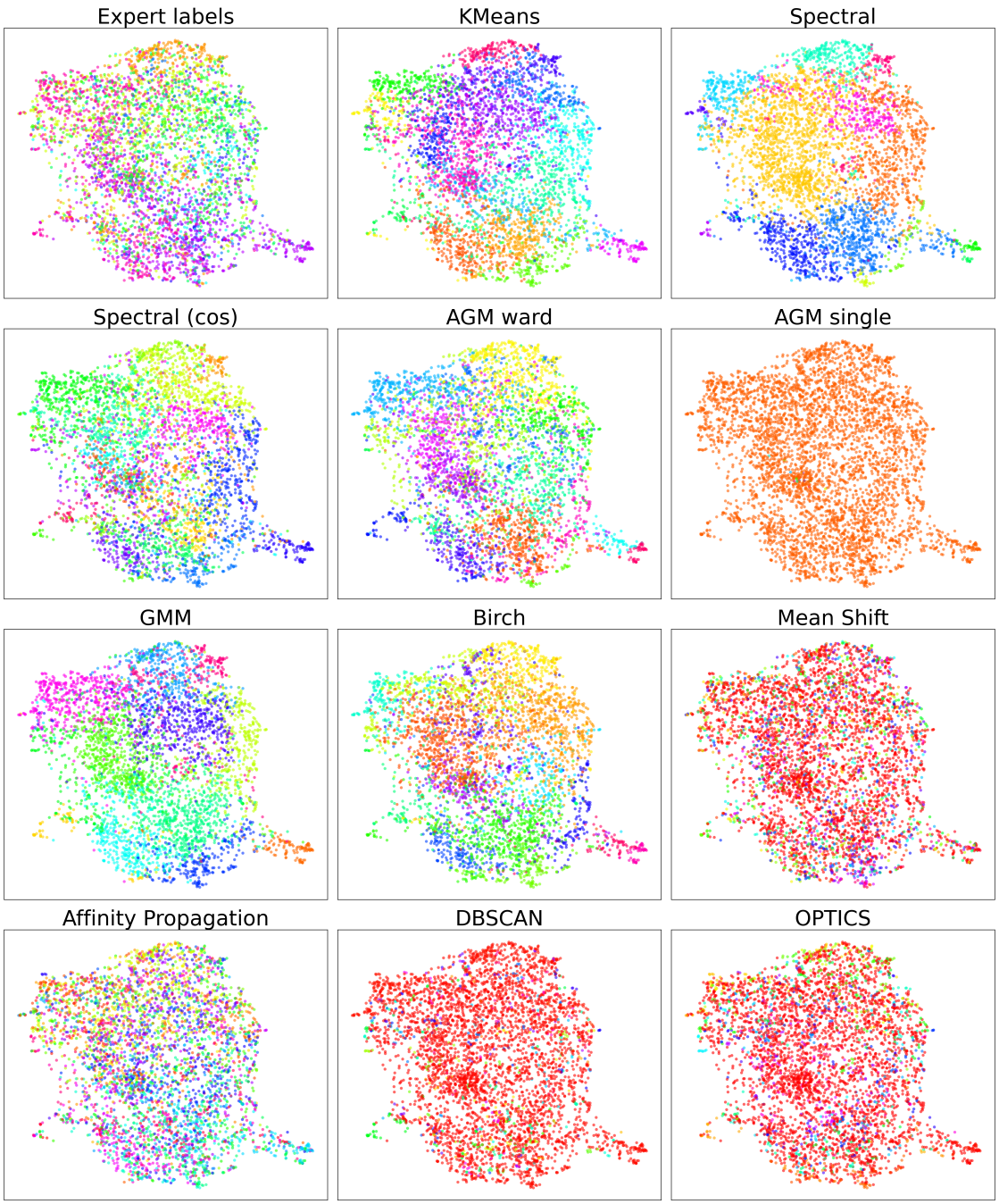

Figure 4.6: Visualization of different clusterings.

5 Prototype

In this chapter we present our prototype for label recommendation, which we developed based on the insights gained from our experiments. In Section 5.1 we also analyze and describe the approach currently in use. The current algorithm serves as the baseline for our prototype's performance. Section 5.2 describes our prototype in detail and highlights how our findings impacted the prototype design. We compare both algorithms and present the results in Section 5.3.

5.1 Baseline approach

The currently implemented baseline algorithm is in house developed by our partner and is tightly coupled with the rest of the system for better performance. So far no documents describing the baseline approach exists. Therefore, we analyzed the implemented algorithm. This enabled us to develop a standalone version, which we further used for performance comparison. To the best of our knowledge, this section is the only documentation of the currently implemented approach.

5.1.1 Design

In its core, the baseline algorithm uses an LSI model and extensive preprocessing. The LSI model is trained on the labeled statements only and not on the whole corpus. The algorithm uses the topic labels for document pooling. Tag based pooling schemes have

been shown to significantly increase the clustering performance of the LDA topic model [25]. It is therefore likely, that this pooling strategy benefits the LSI model as well. The unlabeled part of the corpus does not contribute to the generation of the LSI model. Since the creation of an LSI model is computationally expensive, it would not be feasible to include these statements anyhow.

The data flow of the algorithm is depicted in Figure 5.1. The diagram is divided into two areas separated by a dashed line. The area on the left side shows the data flow during model training, while the right side depicts the flow during label prediction. Intersections with the doted line mark the information transfer from training to prediction. This way it is easy to see which data artifacts contain the learned information. For the baseline algorithm these artifacts are the labeled document pool vectors and the LSI model.

5.1.2 Preprocessing

The preprocessing steps are exactly the same for training and prediction:

1. Remove non alpha-numeric characters: First non-alphanumeric characters are removed.
2. Remove OTS headers: Texts sourced from Austrian Original-Text-Service (OTS) start with an OTS specific header. If such a header is present then this header is removed.
3. Split into tokens: The document is split into tokens along whitespace characters.
4. Capitalize hashtags: If any hashtags are present they are set to upper case.
5. Part-of-speech-tagging: At this point we have a list of string tokens, without special characters. The list of tokens is given to a part-of-speech tagging algorithm to identify nouns in the text.
6. Filter for nouns: All tokens who are not classified as nouns are dropped.
7. Filter for Names: Our partner maintains a list of known politician names. If such a name is found, then the token is dropped.
8. Lemmatize: All remaining tokens are lemmatized with a lemmatizing algorithm for German language.

5.1.3 Training

At the beginning of the training all labeled statements in the dataset are preprocessed as described in Section 5.1.2.

This is followed by the topical pooling: All documents belonging to a certain topic which is given by their label are concatenated to form one large document we denote as document pool. This results in several pools, where each pool represents one topic.

Then an LSI model with 200 dimensions is trained on this corpus of document pools. The pools are transformed to the semantic space of the resulting LSI model and the document vectors of the pools are stored.

5.1.4 Prediction

The new statement is preprocessed as described in Section 5.1.2 which results in a list of string tokens. Then the LSI model is used to transform the preprocessed statement in the semantic LSI vector space. In the LSI vector space the cosine similarity between the vector of the new statement and the vectors of all pools are calculated. In the live system the similarity to the document pools is used in a threshold classifier: If the similarity exceeds 0.7, then the according label is assigned to the new statement. In our experiment we use the similarity score for ranking: The topic with the highest score is recommended first, the others follow ordered by descending similarity.

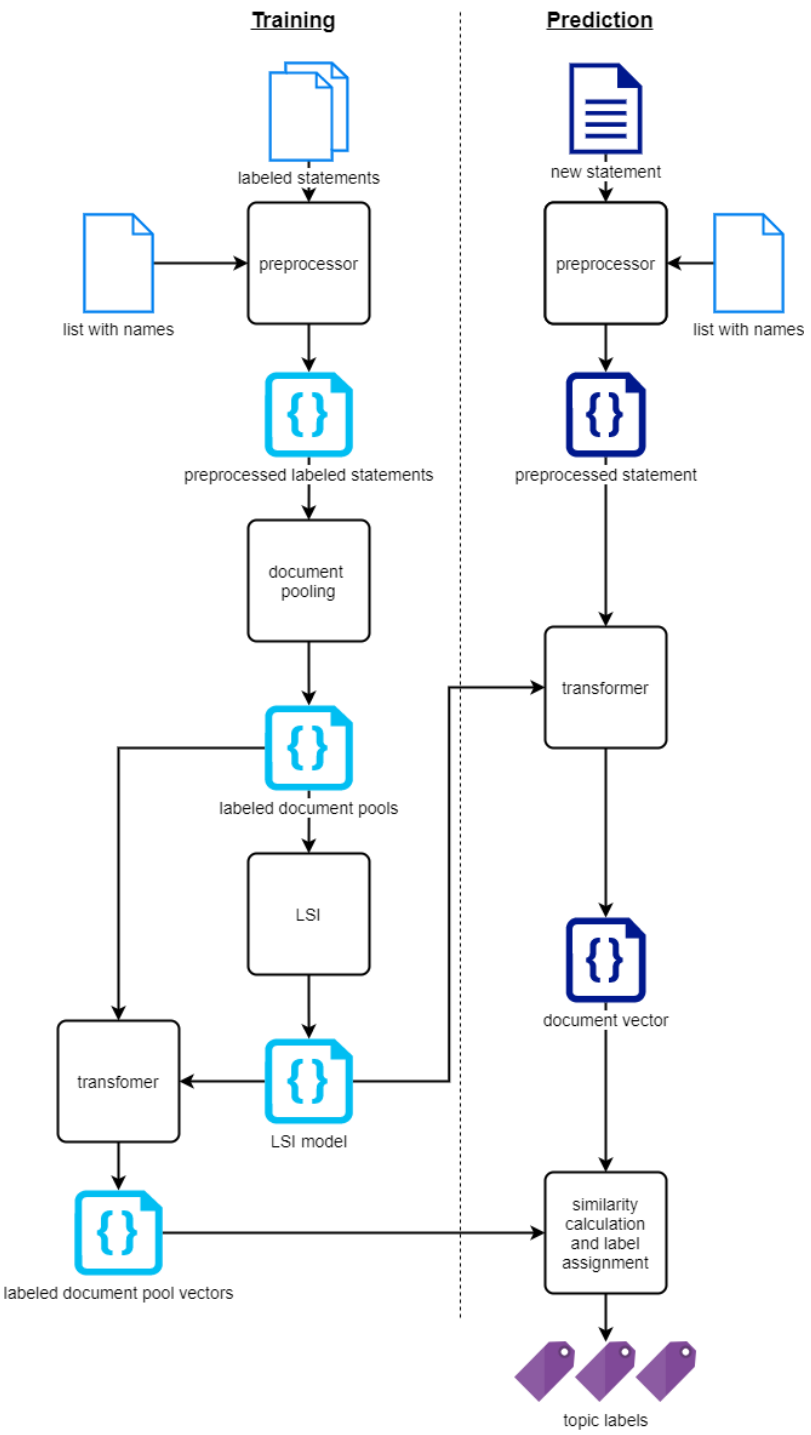

Figure 5.1: Data flow of the baseline algorithm.

5.2 Our prototype

We implemented a label recommendation prototype that outperforms the algorithm in use, based on our findings presented in chapter 4. In this section we describe the prototype design and explain the underlying design decisions.

5.2.1 Design

Our prototype is based on Word2vec document embeddings and a similarity-based label recommender.

Figure 5.2 shows the data flow during prediction and training of the recommender system. The left side of the figure shows the data flow for recommender training and the right side shows the data flow for the topic label prediction.

On the left side of Figure 5.2 we can see that the topic label recommender is created from three data sources. These are a dump of the English Wikipedia, the corpus of collected political statements, and topical labels elicited from domain experts. The corpus of collected statements and the Wikipedia dump serve together as the training corpus for the creation of a Word2vec model, which is later used to calculate document vectors. A sample of the collected statements was given to domain experts for labeling. Together with document vectors calculated from the Word2vec embeddings these labels serve as instances in the similarity based recommender.

For prediction a statement is preprocessed and transformed in the same way as the statements labeled by the experts were. The resulting document vector is processed by the recommender to generate topic label recommendations.

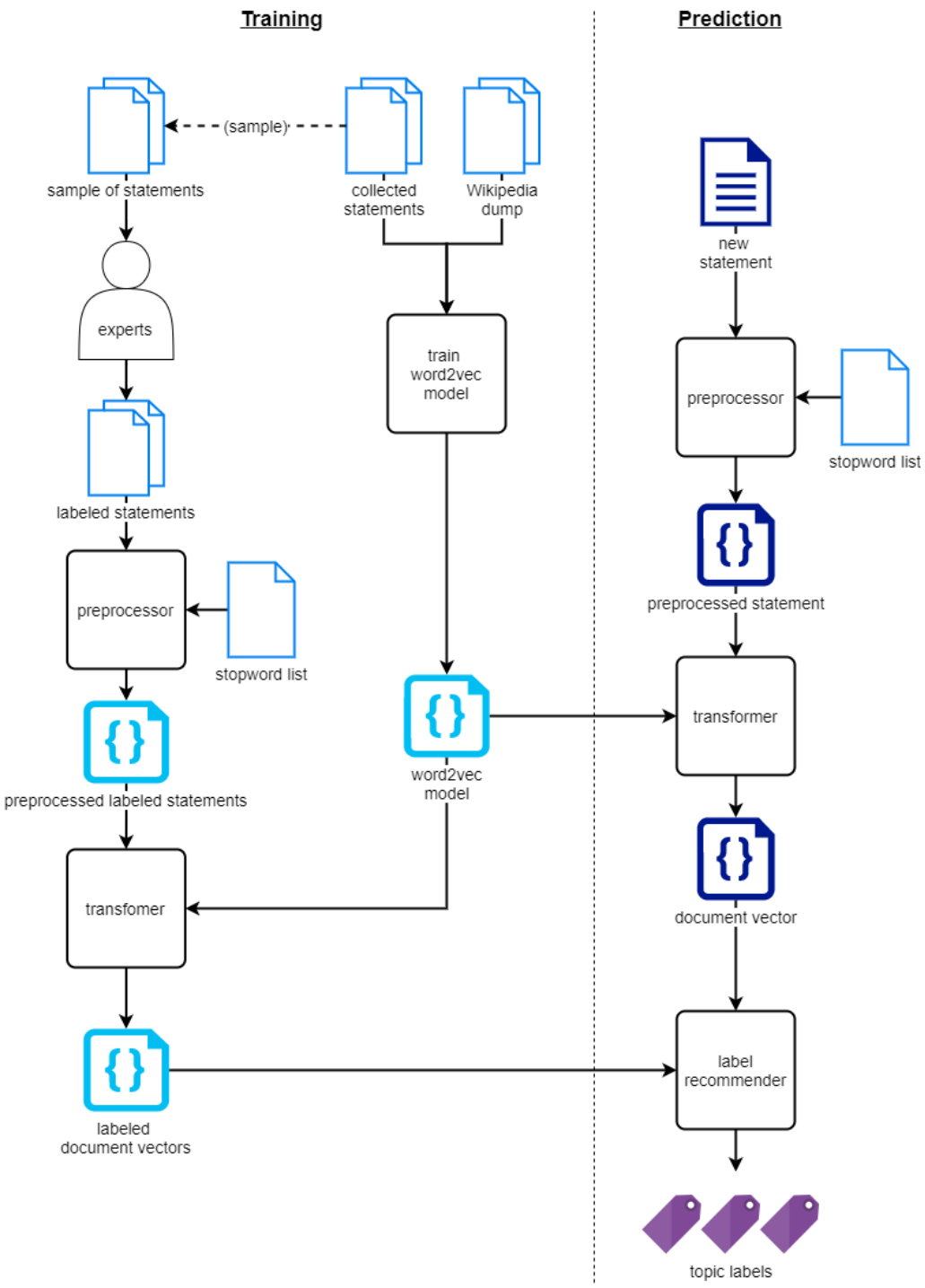

Figure 5.2: Data flow of our prototype.

5.2.2 Statement labels

Our partner provided us with a set of statements labeled by human experts. These were sampled from their corpus of collected statements and the resulting expert labels were added to their database. This means that the same labels were also used to create the dataset for answering the research questions from Section 1.2.

The labeling procedure was conducted as follows: Around 200 statements are sampled from the corpus and compiled into a list. This list takes the form of a CSV-Document which contains statement IDs and text of these statements. The first line of the document shows the possible topic IDs and spaces where the experts may enter their age and gender. The document also contains a column for the topic IDs. This document is handed to a human expert who is asked to enter the topic IDs separated by a column. Optionally the experts may give their age and gender. The first lines of such a document are shown in Figure 5.3.

INFO LINE	Topic ids: INTERNATIONAL: 1 || INNENPOLITIK: 2 || WIRTSCHAFT: 3 || SOZIALES: 4 || INFRASTRUKTUR: 5 || UMWELT: 6 || BILDUNG & KULTUR: 7 || CHRONIK: 8 || PARTEIPOLITIK: 9	AGE:	GENDER:
stmt_id	stmt_text	topic_ids	
96023923	Wir fordern eine #Mautfreistellung auf der Autobahn, um das drohende #Verkehrschaos im Zuge des Baus der Koralmbahn abzuwenden. Die WOCHE Graz-Umgebung Süd berichtet .	5, 8	
96010941	jede einzelne forderung ist richtig und wichtig. und weil mich die eine oder andere noch an den unistreik '87 erinnert, wünsch ich euch umso mehr erfolg #danke #unsreichts #unibrennt https://t.co/Krhk1JiZAv	7	
96020825	!!! Save Uighurs !!! "Die Unvereinbarkeit der Religion mit den Menschenrechten liegt so wenig im Begriff der Menschenrechte, daß das Recht, religiös zu sein, auf beliebige Weise religiös zu sein, den Kultus seiner besonderen Religion auszuüben, vielmehr ausdrücklich unter die Menschenrechte ...	1,7	

Figure 5.3: Document handed to the experts for statement labeling.

After the experts have finished their task the labels are added to the dataset. It is not required to use this exact label set for recommender training. The user may choose to re-train the recommender with a completely different set of labels at any time.

5.2.3 Model generation

Section 4.1.2 shows that the best performing document representation for similarity calculations is the document representation based on a Word2vec model.

Embeddings like Word2vec can be learned from a much larger corpus than topic models, because they are computationally inexpensive in comparison. When the corpus is large and diverse enough, re-training periods can be extended further. Learning Word2vec embeddings from big data sets takes a significant amount of time, however, this is only a problem on the first glance. A large, non domain specific corpus can be used to create a well generalizing word2model, which does not require re-training. In a second step this model can be refined with a smaller but domain specific corpus on a regular basis within just a short amount of time.

We decided against using a published Word2vec model for several reasons: To save disk space, public available Word2vec models often omit data which is required for refining the model. Some models are re-trainable, but they are published under licenses which do not permit commercial use. Another disadvantage is that we can not control the model's parameter if we use a published model. Therefore, we decided to create a model from scratch on a corpus published under a permissive license.

We trained our Word2vec model on a dump of the German Wikipedia from 1st January of 2020. Then we refined the model with our corpus of political statements.

By training the model on the Wikipedia corpus, we get a model which generalizes well but is not well suited to our domain. In the second training step, we refine the model with our own corpus which we compiled from collected political statements. This fits the model to our domain, without loss of the model's capacity to generalize. Training on the Wikipedia corpus takes hours to days, dependent on the used hardware and the exact settings.

For our prototype, we created a CBOW Word2vec model with a vector size of 200 dimensions. We set the training parameters to 15 passes with a window size of 5. This

was done on an Intel i7-4790K with 16 GiB of RAM overnight. The resulting model is persisted to disk and archived for later use.

Finally, we refine the model learned from Wikipedia with the newest version of our corpus. For the prototype this took less than an hour on a laptop.

5.2.4 Preprocessing

Since we use the document representation based on the Word2vec model, we also use the optimal preprocessing steps found in Section 4.1.1.

Thus, the preprocessor steps are the following:

1. Remove URLs: A regular expression is used to remove all http, https, ftp, and ftps URLs.
2. Remove words starting with an '@' character.
3. Replace non-alphanumeric characters with a space character (Unicode U+0020).
4. Remove headers specific to texts from Austrian Press Agency's OTS, if such a header is present.
5. Tokenize: The text is split into tokens along whitespace characters. All tokens shorter than 2 or longer than 15 characters are dropped. Tokens containing only numbers are dropped as well.
6. Tokens are converted to lower case.
7. Remove stop words: All tokens which appear in the given stopword list are dropped.

The output of the preprocessor is the preprocessed statement: a list of string tokens which represent the statement.

5.2.5 Document vectors

Once a statement is preprocessed it can be transformed into a document vector.

As mentioned before, the document vectors are calculated from Word2vec embeddings of the statement's words. The prototype follows the approach introduced in Section 3.2.5: First we use the Word2vec model to gather the document vector for all tokens of the preprocessed statement. Then we calculate the average of these vectors and use the resulting vector as document vector.

The prototype uses the Word2vec model which was created before.

5.2.6 Label recommender

The partitions found by the clustering algorithms under investigation did not resemble the clusters derived from human labels as shown in Section 4.2. Therefore, an approach based on statement clusters did not look promising. However, the similarity based topic label recommender introduced in Section 3.2.6 produced reasonable results. So we decided to improve on its design and implement a similarity based topic label recommender.

Again we use an instance based system, where new labels are recommended based on the similarity between the new statement and the learned instances. For similarity, we calculate the cosine similarity given in Equation 3.4 between document vectors. During prediction the k most similar neighbors are considered for label recommendation.

This approach leads to problems with unbalanced training data. As clearly shown in Section 3.1.1, the dataset is indeed unbalanced. To mitigate this issue, the prototype calculates a topic weight for every topic during training. Topics with fewer labels in the training data, are weighted higher than these with more labels.

Training

The recommender is trained with labeled document vectors. These document vectors are created from the statements as described in Sections 5.2.4 and 5.2.5. The Labels are elicited from experts beforehand as described in Section 5.2.2. These labeled document vectors are stored within the recommender and represent the gathered knowledge.

The topics are weighted to mitigate the negative effects caused by unbalanced data. High frequent topic labels are weighted lower than those which appear in fewer examples. For this a topic weight w_t is calculated for every topic t.

We write the topic labels for every document d in form of a topic label vector $\boldsymbol{l}_d$ as given by Equation 5.1. The dimension of the topic label vector is determined by the number of topics T.

$$\boldsymbol{l}_d = [l_{d,1}, \cdots, l_{d,T}] \tag{5.1}$$

If the topic label for topic t is assigned to statement d, then the corresponding vector element is one. Otherwise, it is set to zero as shown in Equation 5.2.

$$l_{d,t} = \begin{cases} 1, & \text{if d has label t} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

We use the topic label vectors to form a topic label matrix and sum up the columns. In Equation 5.3 this is shown as a left multiplication with a vector of ones. This gives us the number of labels for every topic.

$$[1 \ \cdots \ 1] \cdot \begin{bmatrix} \boldsymbol{l}_1 \\ \vdots \\ \boldsymbol{l}_D \end{bmatrix} = [n_1 \ \cdots \ n_T] = \boldsymbol{n} \tag{5.3}$$

The topic weight is then directly calculated from the number of labels for each topic as given in Equation 5.4. The multiplication with $min(\boldsymbol{n})$ guarantees that the largest topic weight equals one. This is done for numerical stability.

$$\begin{aligned} \boldsymbol{w} &= min(\boldsymbol{n}) \cdot \begin{bmatrix} \frac{1}{n_1} & \cdots & \frac{1}{n_T} \end{bmatrix} \\ &= [w_1 \ \cdots \ w_T] \end{aligned} \tag{5.4}$$

Prediction

For prediction the label recommender takes an unlabeled document vector and recommends topic labels for it. The document vector is created from a statement as described in sections 5.2.4 and 5.2.5. The recommender module calculates the cosine similarity (as defined in Equation 3.4) between the input vector $\boldsymbol{q}$ and the labeled instances in its memory.

The labeled document vectors are then sorted by similarity to the input vector in descending order from 1 to n, so that d_1 is the most similar document vector and d_n is the least similar one. The k most similar instances are taken into account for label recommendation, where k is set to 15. This neighborhood size was based on numerical stability and execution speed considerations.

A topic score is calculated with the scoring function given in Equation 5.5. Note that w_t is the topic weight for topic t as introduced in Equation 5.4 and l_i, t is the corresponding element of the topic label vector as given in Equation 5.2.

$$\text{score}_t(\mathbf{q}) = w_t \cdot \sum_{i=1}^{k} \text{sim}\left(\mathbf{q}, \mathbf{d}_i\right) \cdot l_{i,t} \tag{5.5}$$

Topic labels are recommended based on the calculated topic score. The topic label are recommended according to the topic scores in descending order, so that the topic label with the highest score is recommended first and the label with the lowest score is recommended last.

5.3 Comparison

To compare our prototype to the baseline algorithm, we measure their performance on our dataset from Section 3.1. Statements written in German language which originated in Austria were selected. An additional requirement was a statement length of 51 characters. These particular settings were chosen, because the baseline algorithm was optimized

for them. The argument for this approach is the following: The baseline algorithm was optimized for use in the live system. Hence, these are the conditions under which the prototype must outperform the baseline to provide real live performance benefits.

We perform a training/test set split with a ratio of 80/20. Both algorithms are trained on the test set and then the performance is measured on the test set. Precision, Recall and F1-Measure as presented in Section 3.2.2 are used as performance metrics. Precision and Recall for top-N recommendation are calculated as given in Equation 3.5 and 3.6. The calculation of the $F1$ score follows Equation 3.7. The number of recommended labels were varied from only one to all nine.

The measured performance for both algorithms is given in Table 5.1, which clearly shows that our prototype outperforms the legacy algorithm under all tested conditions by a significant margin. A graphical representation of the performance comparison is shown in figure 5.4.

	baseline algorithm			**prototype**		
k	**P@k**	**R@k**	**F1@k**	**P@k**	**R@k**	**F1@k**
1	0.453	0.374	0.399	**0.539**	**0.458**	**0.483**
2	0.372	0.599	0.444	**0.398**	**0.649**	**0.478**
3	0.301	0.714	0.410	**0.318**	**0.769**	**0.436**
4	0.251	0.792	0.370	**0.267**	**0.847**	**0.393**
5	0.219	0.854	0.338	**0.231**	**0.904**	**0.357**
6	0.193	0.900	0.309	**0.203**	**0.950**	**0.325**
7	0.174	0.945	0.286	**0.179**	**0.973**	**0.294**
8	0.157	0.973	0.264	**0.159**	**0.987**	**0.267**
9	0.144	1.000	0.246	**0.144**	**1.000**	**0.246**

Table 5.1: Top-N recommendation performance of the prototype and the baseline algorithm.

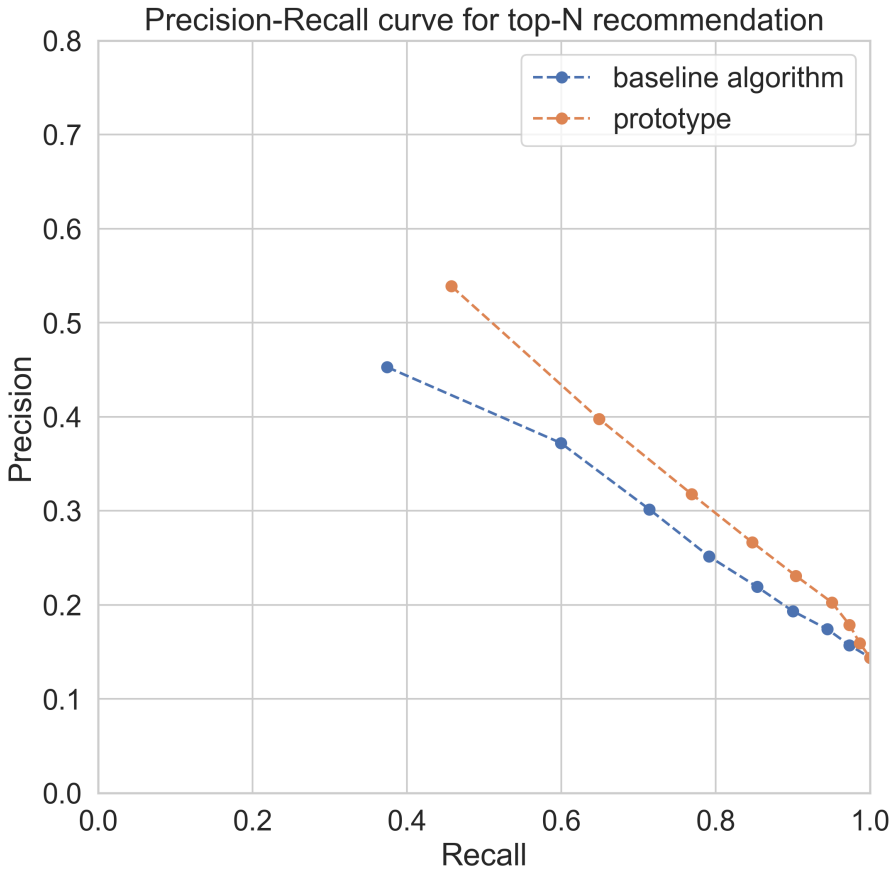

Figure 5.4: Prediction performance of the baseline algorithm and the prototype for $N = [1 \ldots 9]$.

5.4 Summary

We have developed a prototype for label recommendations based on the Word2vec model and a similarity-based recommendation system. Our Word2vec model is created in a two-step process: First, a general model is created from a dump of the German Wikipedia, which is then refined using the corpus of collected statements. To create the document vector, the document is preprocessed and the Word2vec vectors of the words are averaged. The similarity based recommender service is trained using labeled document vectors. The topic labels are weighted based on the label frequency to compensate for label imbalance.

We analyzed the currently used algorithm and created a standalone version to serve as a baseline for comparison. The baseline algorithm is based on an LSI model, extensive preprocessing, and a topic-pooling strategy. Labels are recommended based on their similarity to topic pools in semantic LSI space.

Our prototype consistently outperformed the baseline algorithm in terms of precision, recall, and F1-measure.

6 Conclusion

In this thesis we investigated topical clustering approaches for short text documents of political statements.

We evaluated various document representation approaches for their usefulness in similarity calculations of these political statements. In the absence of a similarity baseline, we developed a simple similarity-based recommender to infer the usefulness of document representations from recommendation performance. Document representations based on averaged Word2vec embeddings showed the best performance on our dataset. It was somewhat surprising that the averaged Word2vec representation turned out to be the most suitable document representation for our data set. Intuitively, one would expect that the additional use of sub-word information (FastText) or a paragraph vector (Doc2vec) would perform better. Obviously, the added complexity does not always pay off, and one should not dismiss simpler approaches out of hand. The conditions under which Word2vec has an advantage over the other two embeddings are not yet entirely clear. Curiskis et al. noted in their work that Word2vec may have an advantage over Doc2vec when the number of documents in the training corpus is small [9]. Another influence that can be easily overlooked is the number of words that make up a document. If the document is very short, it often happens that it is connected to a particular topic by only one or two words. In such cases, a paragraph vector may not contain any additional information compared to the word vector. Similarly, if the author of a document uses no or only a few words outside the vocabulary, then there is nothing to be gained by using sub-word information, since this information is already encoded in the word vectors. If the training corpus is sufficiently large and the authors use common language, this is likely to be the case.

We further evaluated various clustering algorithms for their consistency with topic labels assigned by human experts. It turned out that the clustering algorithms behaved quite differently from the human experts and produced clusterings which did not resemble the one derived from the expert labels. On closer inspection, however, this is not surprising. The clustering algorithms only partitioned based on the available documents, while the experts worked with a set of predefined topic labels. One could say that the experts partitioned the data based on predefined cluster centers. Moreover, these labels were not derived from the data itself, but were predefined. Thus, the two processes do not compare well which reflects in the overall low AMI scores.

Based on the findings of the above research, we have developed a prototype for topic label recommendation. Our prototype is based on a similarity-based recommender system that uses the averaged Word2vec vectors as the document representation. The Word2vec model used by the prototype is first trained on a dump of the German Wikipedia and then re-trained on our dataset. The topic label recommendation algorithm developed by our project partner was used as a baseline, which was consistently outperformed by the prototype. Given the previously performed evaluation of document representations (see Figure 4.1), one would expect the Word2vec based prototype to outperform the LSI based baseline by a huge margin. Surprisingly, the gains of the prototype compared to the baseline were lower than anticipated. We hypothesize that the good performance of the baseline is due to the document pooling strategy used. Mehrotra ad al. showed that document pooling strategies improve the performance of LDA in a labeling task on micro-blog posts [25]. It is likely that LSI also benefits from document pooling. Considering this, it is possible that a pooling strategy would also improve our prototype when applied to the documents in the second training cycle, where the Word2vec model is adapted to the target domain.

6.1 Limitations

As with all such studies there are limitations that offer opportunities for further research. The work is based solely on one dataset, which also contains only a relatively small set

of labeled statements. Therefore, it remains to be seen how the document representations perform when trained on a larger corpus, and how a larger set of labeled statements would affect the quality of the prototype's recommendations. Due to the process the human experts followed, the expert labels form a fuzzy set of labels. In the creation of our dataset, this fuzzy set was converted into a crisp set of labels. However, this conversion is lossy, which means that the information encoded in the expert labels could not be used to its full extend. When statements are published on online social networks, images and links often play a significant role to express the conveyed message. This information can help in clustering statements, but was not considered since this work focuses on text data. For the evaluation of clustering algorithms, only the labeled statements were used. This was done because some of the clustering algorithms would not be able to cluster the entire corpus with the available computational resources and in an acceptable time. Also, no user study was conducted to evaluate the topic recommendations due to time and resource constraints.

6.2 Future work

For future work we propose to broaden the scope to include next-generation embeddings, such as Glove [29], which could be a good alternative for feature representation.

Specialized document clustering algorithms have been developed in the past for specific tasks, such as event detection from news sources [22] or for public health monitoring from posts on online social networks [10]. These could be adapted for political statements or serve as inspiration for a specialized statement clustering approach. Techniques such as document pooling strategies [25] and topic vectors [10] may be useful to increase clustering performance. We hypothesize that for certain events, some topics occur more frequently than others. Therefore, the time at which a statement is made could be an important feature for clustering statements and should be considered in future evaluations.

Another area that needs investigation includes the clusters found by the evaluated algorithms. They may not resemble the expert labels from our dataset, but that does not mean they are useless. As mentioned earlier in this section, the clustering procedure performed by the experts does not compare well with the algorithmic approach used. We would like to propose two approaches to address this problem in future work. The first approach is to use or develop algorithms that accept predefined topics in a manner similar to how they are presented to experts. The "topic vectors" used by Dai et al. [10] are a good example of this approach. For the second approach, we propose a setting without predefined topics. The experts would not be asked to label from a predefined set, but to divide the dataset into partitions that would be labeled later. In this case, the workflow of the experts is similar to the algorithmic approach of clustering documents followed by a keyword extraction task.

Finally, a users study should be conducted to ultimately judge the quality of label recommendations in a real life setting.

Bibliography

[1] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[2] N. Alnajran, K. Crockett, D. McLean, and A. Latham. Cluster analysis of twitter data: A review of algorithms. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pages 239–249. INSTICC, SCITEPRESS - Science and Technology Publications, 2017. ISBN 978-989-758-220-2. doi: 10.5220/0006202802390249.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, volume 28, pages 49–60, New York, NY, USA, jun 1999. Association for Computing Machinery. doi: 10.1145/304182.304187.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval–the concepts and technology behind search 2nd edn.* Addison-Wesley, Pearson, Harlow, England, 2011. ISBN 9780321416919.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2006. ISBN 0387310738.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, Mar. 2003. ISSN 1532-4435.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606v2*, 2016.

[8] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 603–619, may 2002. doi: 10.1109/34.1000236.

[9] S. A. Curiskis, B. Drake, T. R. Osborn, and P. J. Kennedy. An evaluation of document clustering and topic modelling in two online social networks: Twitter and reddit. *Information Processing & Management*, 57(2):102034, mar 2020. doi: 10.1016/j.ipm.2019.04.002.

[10] X. Dai, M. Bikdash, and B. Meyer. From social media to public health surveillance: Word embedding based clustering method for twitter classification. In *SoutheastCon 2017*, pages 1–7. IEEE, mar 2017. doi: 10.1109/secon.2017.7925400.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[12] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[13] E. Dorani, N. Duru, and T. Yildiz. An empirical investigation of performances of different word embedding algorithms in comment clustering. In *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2019. doi: 10.1109/asyu48272.2019.8946379.

[14] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, 1996. AAAI Press.

[15] B. S. Everitt. *Cluster analysis*. Wiley series in probability and statistics. John Wiley & Sons, 5th edition, 2011. ISBN 978-0-470-97780-4.

[16] M. Fraj, M. A. B. Hajkacem, and N. Essoussi. A novel tweets clustering method using word embeddings. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, oct 2018. doi: 10.1109/aiccsa.2018.8612816.

[17] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, feb 2007. doi: 10.1126/science.1136800.

[18] A. Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.

[19] R. Ibrahim, A. Elbagoury, M. S. Kamel, and F. Karray. Tools and approaches for topic detection from twitter streams: survey. *Knowledge and Information Systems*, 54(3): 511–539, 2017. doi: 10.1007/s10115-017-1081-x.

[20] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/w16-1609.

[21] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 2014.

[22] G. Leban, B. Fortuna, and M. Grobelnik. Using news articles for real-time cross-lingual event detection and filtering. In *NewsIR@ECIR*, 2016.

[23] Q. Li, S. Shah, X. Liu, and A. Nourbakhsh. Data sets: Word embeddings learned from tweets and general data. In *Proceedings of the eleventh international conference on web and social media*, pages 428–436, Montréal, Québec, Canada, 2017.

[24] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on InformationTheory*, 28(2):129–136, mar 1982. doi: 10.1109/tit.1982.1056489.

[25] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information*

retrieval, pages 889–892, New York, NY, USA, jul 2013. ACM. doi: 10.1145/2484028.2484166.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[27] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546v1*, 2013.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. doi: 10.3115/v1/d14-1162.

[30] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, nov 1987. doi: 10.1016/0377-0427(87)90125-7.

[31] G. Salton and C. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29(4):351–372, 1973. doi: 10.1108/eb026562.

[32] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, nov 1975. doi: 10.1145/361219.361220.

[33] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[34] J. N. X. Vinh, J. Epps, and null Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the International Conference on Machine Learning*, pages 1073–1080. ACM, 2009. doi: 10.1145/1553374.1553511.

[35] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[36] YouGov. Politisches informationsverhalten der deutschen, 2017.

[37] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114, 1996.