Bachelor Thesis

Textual Analysis of German Online Media

Bachelor of Sience (B.Sc.)

International Media and Computing

written by

Jonathan Derin

September 4, 2018

First supervisor: Prof. Dr. Gafei Zhang

Second supervisor: Prof. Dr. Debora Weber-Wulff

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Contents

List of Figures

1. Introduction

1.1. Motivation

The field of natural language processing (NLP) has been gaining attention with its latest advancements and the increase of related technology. While speech recognition is a commonly known NLP application, textual analysis seems far less talked about. The intention of this thesis is to explore the possibilities of natural language processing in the field of textual analysis on the basis of a German news corpus. To illustrate the insights NLP can give into a topic, the thesis focuses on the German medias' various portrayals of migration to Germany between 2015 and 2017. While the focus of the analysis is to obtain information relating to the topic *migration to Europe*, the thesis will also illustrate the problem of searching for topics in a collection of text and how NLP approaches this issue. To solve this problem and to dive deeper into NLP, in addition to statistical text analysis, I will also use a machine learning technique called probabilistic topic modeling. The thesis is not setup to deliver a specific result, but rather to examine various natural language processing tasks.

1.2. Methodology

As the first step, the sources of the news articles, i.e. the newspapers, have to be selected. The desired data will then be collected through web scraping. To analyze the texts, the raw data will be prepared using various preprocessing techniques. Furthermore a topic model, more specifically an LDA (Latent Dirichlet Allocation) model, will be trained and improved to ultimately create a coherent topic model.

1.3. Related Work

It is important to mention David Kriesel's popular analysis of Spiegel Online presented at the Chaoc Communications Congress 2016. The presentation, titled "SpiegelMining", gives a detailed insight into Spiegel Online using extensive data analysis methods [16].

2. Basic Theory

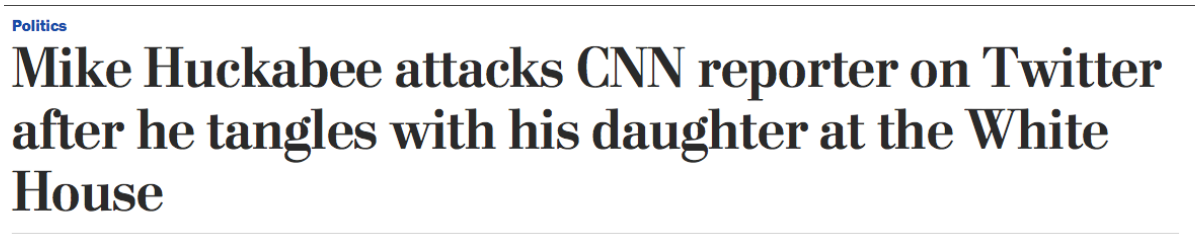
2.1. Natural Language Processing

Natural language processing is a field in computer science focused on processing data that consists of natural human language, either in written or spoken form. It is classified as a subfield of Artificial Intelligence, aiming at making computers understand language used by humans to communicate, called natural language. Natural language processing has become more prominent in the public eye since advancements in speech recognition led to its implementation in cars and home gadgets produced by Google, Amazon and others. In addition to speech recognition, natural language processing has a variety of applications. It has been implemented in conjuncture with the internet to overcome the language barriers inherent in globalization through Machine Translation, a field concerned with automatically translating text into different languages. Another major field of application is Information Retrieval, in which text is analyzed for specific information, such as topic and content of the text. The data can then be used for summarization, spam filtering and more.

Since natural language processing is concerned with natural language of any sort, there are huge amounts of data available, for example through the world wide web.

2.1.1. Challenges

Natural language processing researchers are struggling with some major challenges. Most prominent is probably the ambiguity intrinsic in every languages, where a word or a sentence can have more than one meaning. Linguists call words that sound the same and are pronounced the same homonyms and they are only distinguishable by context. An example of a homonym is the word *kind*. *Kind* can be used in context of classification, meaning type, as in "What kind of apple is that?", or as a description, meaning friendly or nice, as in "The boy was very kind". Sentences can be ambiguous as well. An example would be the recent headline of a Washington Post article, as can be seen in figure 1. The pronoun "he" could refer to the subject, Mike Huckabee, or the object, the CNN


Politics

Mike Huckabee attacks CNN reporter on Twitter after he tangles with his daughter at the White House

Figure 1: Ambiguous Washington Post headline [22]

reporter, leaving us with two possible meanings:

1. Mike Huckabee attacks CNN reporter after Mike Huckabee tangled with his daughter

2. Mike Huckabee attacks CNN reporter after the CNN reporter had previously tangled with his daughter

The meaning of this sentence is only clear through its context and our understanding of it. This ambiguity is a problem for machines, especially in unsupervised learning.

Another challenge in natural language processing is that the data is made up of language and not values. While numeric values are independent and can be compared with each other across the world, language is bound to certain users and contexts. Every language has its unique grammar, constraints, and speech patterns. This means a model that has been trained to parse or understand English cannot understand German, and the other way around. Building machine learning libraries for different languages is therefore complicated, especially for languages that are morphologically rich such as German.

Especially relevant for speech recognition and analysis is the ability to handle human flaws, such as incorrect grammar and mistakes and the constant evolution of natural language, such as slang and dialect. While this is crucial to gaining quality results in the field of analyzing spoken language or social media content, this is not so much of an issue for scientific or professional texts such as books or news articles.

2.1.2. Concepts

To clearly understand the following descriptions, there are a few terminologies that need to be explained.

- Corpus

  A corpus is a collection of texts, also called documents, of any sort. A corpus can consist of various documents with different topics, or even languages. While corpora are not defined to be coherent, they commonly exhibit the same documents from the same domain such as social media messages, newspaper articles or scientific papers.

- Token

  Tokens are the smallest unit of a document that is being examined and analyzed. Depending on the domain this might mean a word, a group of words or a sentence.

- Stop Words

  Natural language contains many words that are grammatically necessary and frequently used without containing relevant semantic information. Examples are words such as "the", "and", "or", that are necessary to compose a correct sentence, but do not contain further information about the topic of the sentence. These words are called stop words and are commonly filtered out of the text and removed from further analysis.

- Bag-of-words

  The bag-of-words model, also known as vector space model, is a representation model aimed at simplifying a text[1]. It does this by counting the occurrence of a word in a given text, ignoring the word order. A text such as *John went to a movie. It was a good movie. John liked it.* would therefore turn into a bag-of-words such as `John: 2, went: 1, to: 1, a: 2, movie: 2, it: 2, was: 1, good: 1, liked: 1`. By doing this the corpus is reduced in size without losing any words and additionally includes word frequencies, which is a useful feature for common natural language processing tasks.

- n-grams

  A sequence of words that co-occur unusually often in comparison to other word pairs are called collocations. An example of such collocation would be *red wine*, while *the wine* has no higher co-occurrence than any other pair of article and noun [7, p.20]. In computational linguistics such sequences are captured as n-grams, in which the n stands for the number of words in a sequence. Bigrams therefore capture a collocation of two words and trigrams a collocation of three words.

[1]`https://en.wikipedia.org/wiki/Bag-of-words_model`

2.1.3. Tasks

While natural language processing has a variety of applications and implementations, there are common tasks important to all application domains that are needed in the process of the analysis. Obtaining meaningful results in textual analysis greatly depends on the preparation of the raw data. The following selection of tasks are a selection of relevant tasks for my work.

- Tokenization

  It is common when analyzing text to break it down into smaller parts, most commonly sentences or words, which are then called tokens. To obtain tokens, a text is often stripped of unwanted punctuation characters and split, either by white space or sentence dividers, depending on the type of token needed. The difficulty of obtaining tokens can vary depending on language and task. While splitting sentences by sentence dividers can be difficult for German and English (e.g. U.K. or 60. Geburtstag), it may be even more difficult for languages such as Chinese or Arabic, which do not use whitespaces to split words.

- Text Normalization

  To remove unwanted distinctions between words, text is often normalized prior to the analysis. Normalization includes a variety of processes. A simple step towards removing unwanted distinctions is transforming every letter to lowercase. To remove distinctions between various morphological forms of a word, **stemming** can be applied. “Stemming is not a well-defined process” [7, p.107], but usually refers to removing the affixes of a word, only leaving its stem. Using the Lancaster Stemming Algorithm, the sentence *I have received many emails from other employees* would be processed as `i hav receiv many email from oth employ`. However, research could not conclude a positive effect of stemming on the outcome of the analysis. Most recently Schofield et al. have even correlated a deterioration of quality when applied to an LDA model. Another, more sophisticated process, is **lemmatizing**. Lemmatizing is concerned with finding the lemma, meaning the dictionary form of the word. The above sentence would result in the following output, using spaCy's lemmatizer algorithm[2]: `I have receive many email from other employee`. Different text normalization approaches also handle misspellings, slang and abbreviations. This step's importance differs depending on the application domain. While this step is common for social media corpora, using a newspaper corpus, the text will only have small amounts of misspellings and most likely no slang, making this step's influence on the quality of the analysis insignificant.

- Part-of-speech (POS) tagging

  Words and sentences are often ambiguous in meaning, only made clear by their context. A way of understanding context during analysis is by determining the grammatical structure of the sentence and the positions of the words. This can be done by using part-of-speech tagging or POS tagging. In POS tagging, every word of a sentence is categorized using grammatical tags. Common tags are noun, verb or adjective, but POS taggers can also describe words in more depth. Words can be tagged with a detailed description such as *adjective, superlative*, *verb, past tense* or describe its relation to adjacent words, such as *pronoun, possessive.* POS tagging helps reduce the ambiguity of a word. While the word *park* in "I am going to park the car." is a verb, in the sentence "I am going to the park.", it is a noun. The machine is then able to understand the difference.

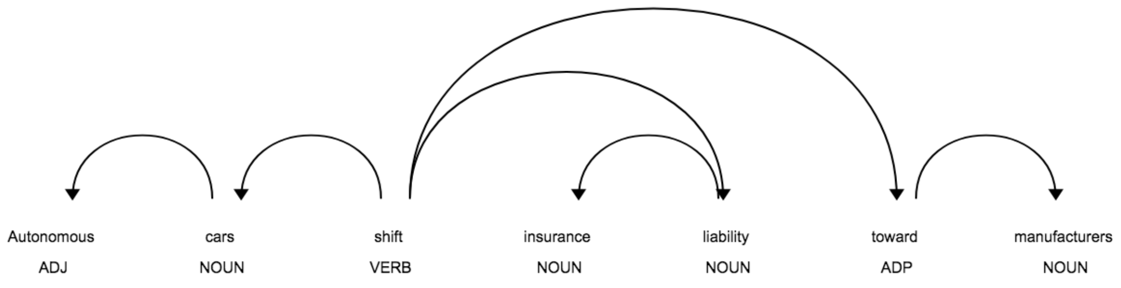


Figure 2: Part-of-speech tags[3]

[2]https://spacy.io/api/lemmatizer

2.2. Web Scraping

Collecting data has become one of the most important tasks in computer science. While a lot of data is readily available through public datasets or APIs, the world-wide-web offers billions of websites, containing a great amount of uncollected data that are not accessible through APIs or existing datasets. This kind of data can be gathered through web scraping. Web scraping can be defined as the process of "extracting data from websites". While web scraping has no standardized methodology, it usually refers to the automated process of downloading the content of a website to a database. To scrape the page of a website, the page is first downloaded. The downloaded content can then be processed as required. A typical step after downloading the website is parsing. Since websites are made up of HTML, libraries such as Beautiful Soup[4] create a parse tree which make it possible to parse the website and extract the desired content. To download the entire content of a website, a web scraping program can process a defined list of urls to download or use a so called web crawler. A web crawler, often used for web indexing, is code that follows hyperlinks on a website to quickly browse the entire website. This is especially useful for more complex website structures that can not easily be iterated through because of unknown or hidden structures.

2.2.1. Applications

Web scraping can be used for various purposes. A typical use case is data mining, in which the content of a website is scraped for further analysis or usage in another context. An example of this in the private sector is review scraping, in which product reviews or comments are scraped and then semantically analyzed. Web scraping is also the tool of political activists who believe in the freedom of data, and therefore scraping data that is not accessible to the public.

[3]`https://spacy.io/usage/linguistic-features`

[4]Available at `https://www.crummy.com/software/BeautifulSoup`

2.2.2. Challenges

While the internet's billions of website are a treasure for Natural Language Processing, the not standardized data poses a challenge for web scrapers. As the popular scraping library Beautiful Soup's website states, "You didn't write that awful page. You're just trying to get some data out of it" [2]. Websites can be written by anyone and often do not even follow the W3C standard for HTML and CSS[5]. But there are even more problems, making it hard to come up with one concise methodology or procedure:

- Inconsistency
  The more pages of a website are parsed, the more likely it is for there to be inconsistencies in format. The website might have changed the structure at some point but has not converted the old structure or pages might have moved.

- Changing Structure
  While a web scraper might work perfectly on a website at a given time, the website can change. Websites might add a paywall, change their structure or just rename a css class. Even the smallest changes can break a scraper.

- Authentication
  Some websites require a user account or a paid subscription to access certain areas of the website. Without being logged in it is not possible to scrape the entire website.

- Dynamic Rendering
  Javascript is becoming more popular with the rise of technologies such as Node.js and React. Parsing websites work best for static HTML. Parsing dynamic websites with changing css tags or content can be difficult for web scrapers and will often require additional software such as Puppeteer or Selenium, which act as a headless browser.

[5]https://www.w3.org/standards/

- Defensive Measures

  Some websites might have measures implemented to prevent web scrapers from collecting information. A simple method against web scraping and web indexing is a robots.txt, stating that the website does not wish crawlers to go through the website. More complex methods are blocking IP addresses or using CAPTCHA codes[6]. These methods can make it hard or even impossible for web scrapers to scrape data.

[6] https://en.wikipedia.org/wiki/CAPTCHA

2.3. Topic Modeling

With increasing access to internet and a growing amount of texts available online, the amount of information is also growing. Newspaper articles, scientific papers, blogs and comments are a powerful source of knowledge. To find this knowledge we use archives with labeled content. But how have these topics been defined? Defining topics can be difficult, considering that texts often have more than one topic. Manually defining these topics can therefore result in loss of information. Furthermore, labeling the amount of all available content is becoming humanly impossible. To better access this information, we need computers to identify the content or topic of the text.

In "Probabilistic topic models" [8] Blei envisions a New York Times archive. In it, it is possible to zoom into any topic, for example foreign politics, revealing more specific topics such as the Middle East or the American-Russian relationship. It would be possible to track any topic over time and, upon request, go to the original articles relevant for the selected topic in time. This is where we need topic modeling.

A topic model is a statistical model, that, given a selection of unlabeled text, splits up the vocabulary into word clusters, which then can be interpreted as themes, or topics. A topic model's output is a cluster of similar words, which can be interpreted as a topic. There are various types of clustering, typically split up in hard clustering and soft clustering [29].

- Hard clustering: clusters are binary, either an object belongs to a cluster or not.
- Soft clustering: Each item belongs to every cluster to a certain degree.

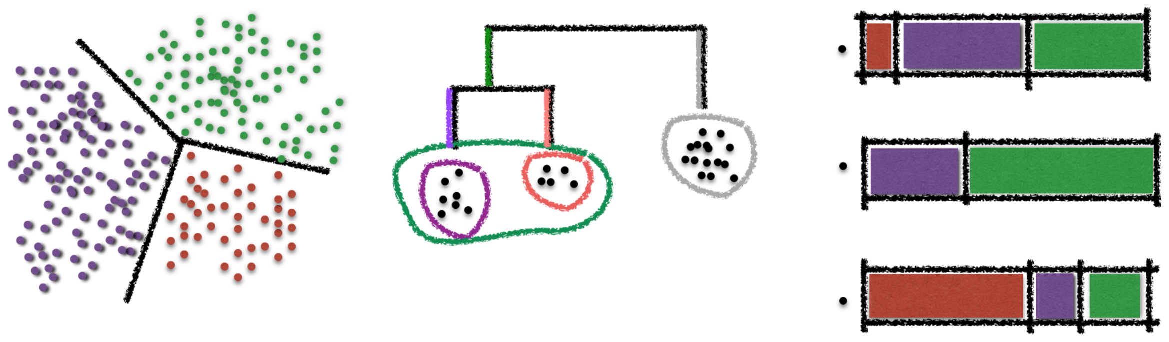


Figure 3: Cluster types. From the left: hard cluster, hierarchical cluster, soft cluster[7]

It is possible to define cluster types with even more granularity, resulting in types such as hierarchical clustering and overlapping clustering. In topic modeling, topics can be seen as soft clusters. For my analysis I used the Latent Dirichlet Allocation (LDA) algorithm, but to illustrate the state of topic modeling I will briefly introduce other relevant methods that ultimately led to LDA.

[7]Image from `http://chdoig.github.io/pytexas2015-topic-modeling`

2.3.1. Tf-idf

Term frequency-inverse document frequency, or tf-idf, is a statistical measure to evaluate the relevance of a word to a document. A simple way to determine a word's relevance would be to count the occurrence of the word in the document. The higher the frequency, the higher would be its relevance. However, the problem with this approach is that a term that occurs more often is not necessarily more relevant. To determine words with a high relevance, the occurrence among all documents in the corpus is calculated. Words that occur in many documents can therefore be used in various topics. This means they are less specific, or less "semantically focused words" [19, p.543]. The more semantically focused a word is, the more relevant it is to a document's topic, if it occurs.

There are various ways to calculate the tf-idf weight of a word [30]. The most simple form is to calculate the terms frequency in the document, $tf(t, d)$, and and then multiplying it with the inverse document frequency $idf(t, D)$. The inverse document frequency can be calculated by counting the amount of documents the word occurs in, n_t, and dividing it by the total number of documents D. The fraction is then logarithmically scaled, resulting in following formula:

$$weight = idf(t, D) * \log \frac{D}{n_t}$$

The terms can then be sorted by weight, showing the most relevant terms for a document. While this reduction offers some insight to a document, Blei, Ng, and Jordan note that tf-idf only "reveals little in the way of inter- or intra- document statistical structure" [10, p.994]

2.3.2. Latent Semantic Analysis

Latent Semantic Analysis, also referred to as Latent Semantic Indexing, was first introduced in 1990 by Deerwester et al. The problem of tf-idf is that while it allows someone to look up documents by their most relevant words, a predefined list of words was necessary. If the goal was to look up a document by a topic, tf-idfs problem is that "individual words provide unreliable evidence about the conceptual topic or meaning of a document" [17, p.1]. Building on tf-idf, LSA assumes that conceptually similar words will occur together in a document more frequently. Using Singular Value Decomposition of the tf-idf term-document matrix and reducing dimensionality, LSA is able to display the words in a k dimensional space. Terms used frequently together will therefore appear close to each other, allowing one to analyze similarity between terms as well as between documents.

2.3.3. Probabilistic Latent Semantic Analysis

While LSA solved some of the fundamental issues Information Retrieval was having, it was also criticized for its expensive computation and its use of SVD, which assumes a gaussian distribution, while there are more appropriate distributions for a term-document matrix [19]. To overcome these issues, a probabilistic version of LSA was introduced in 1999 by Hofmann [18], namely Probabilistic Latent Semantic Analysis (PLSA), that did not use SVD.

2.3.4. Latent Dirichlet Allocation

A further improved model was introduced in 2003 by Blei, Ng, and Jordan called Latent Dirichlet Allocation, or LDA. LDA is a generative probabilistic model and is currently the state of the art of topic modeling algorithms. LDA assumes that every document is made up of multiple topics. In addition, every document in a collection contains every topic, but with different distributions. A topic can be defined as a group of words, connected by a theme, or more formally, as "distribution over a fixed vocabulary" [8, p.78]. LDA assumes a document is created by picking words out of a set of topics.

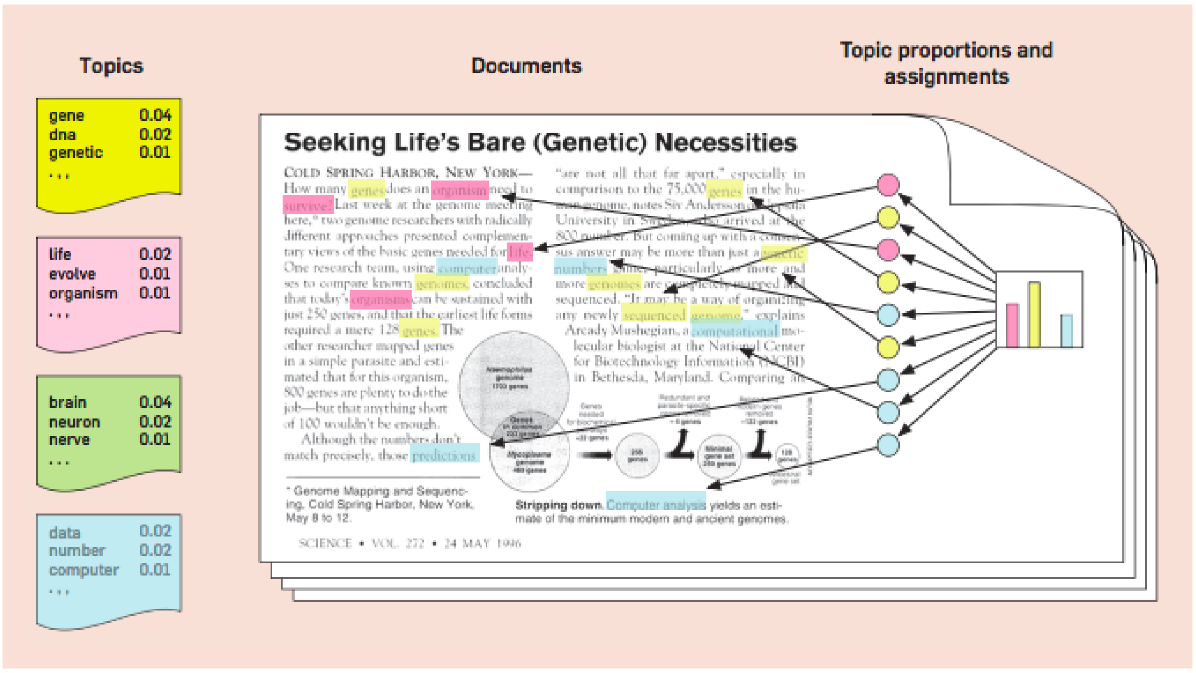
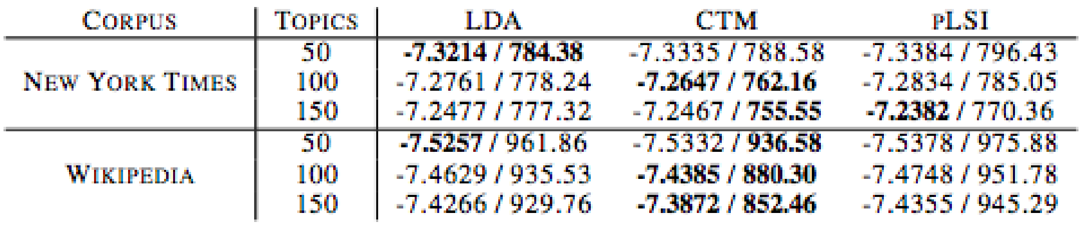


Figure 4: The intuitions behind latent Dirichlet allocation [8]

Figure 4 shows this process, in which each topic is represented by a color. Blue, for example, is the color of a topic that could be interpreted as data science and green of a topic that could be biology or neuro-science. On the left the topics are displayed by showing the words they are made up of with their appropriate probability of occurring in a given text. The histogram on the right shows the document's distribution over topics. To create a document, first a distribution over topics is defined. Then, words are picked from each topic, accordingly to their distribution. This process illustrates the model's assumption, or as Blei calls it, intuition, that a document is made up of different topics to a different degree, namely their probability. LDA uses this intuition, that a document is created by randomly picking words from topics, and calculates what the distribution over topics and words is to create such document. This is the model's *generative* process. This uncovers the *latent* structure, that is the "amount" a topic is present in a document and a word is present in a topic, giving us detailed insights into inter- and intra-document relations.

Human Judgement

Evaluating a model is key to determining its accuracy and therefore success. Evaluating the result of a topic model, i.e. a set of words grouped by an assumed theme, can be difficult. Since there is no data to verify the results of a topic model, various mathematical approaches have been discussed. For the longest time estimating the held-out probability has been the most common way to evaluate a model. Held-out probability is the probability a document can be created using the model's generative process, based on held-out documents, that the model has not yet seen. This is supposed to illustrate how well a model can generalize. Chang et al. were critical of this approach and published a paper in 2009 called “Reading Tea Leaves: How Humans Interpret Topic Models”, researching the correlation between held-out likelihood and topics' semantic qualities.



Corpus	Topics	LDA	CTM	pLSI
New York Times	50	**-7.3214 / 784.38**	-7.3335 / 788.58	-7.3384 / 796.43
	100	-7.2761 / 778.24	**-7.2647 / 762.16**	-7.2834 / 785.05
	150	-7.2477 / 777.32	-7.2467 / **755.55**	**-7.2382** / 770.36
Wikipedia	50	**-7.5257** / 961.86	-7.5332 / **936.58**	-7.5378 / 975.88
	100	-7.4629 / 935.53	**-7.4385 / 880.30**	-7.4748 / 951.78
	150	-7.4266 / 929.76	**-7.3872 / 852.46**	-7.4355 / 945.29

Figure 5: CTM ranks best, while LDA only ranks second, followed by pLSI [11].

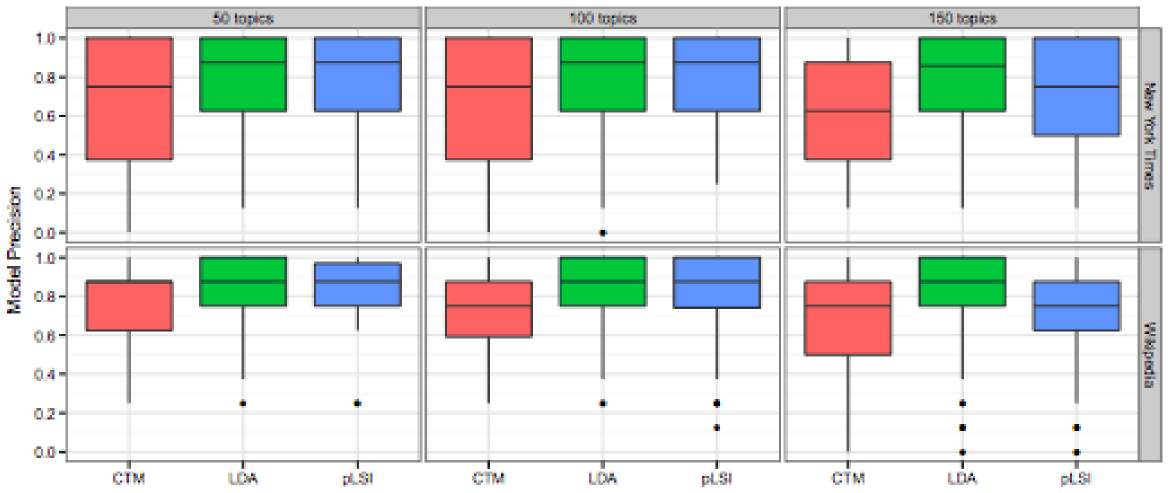


Figure 6: Compared to human judgement, LDA ranks highest [11].

In figure 6 Chang et al. compared three different topic models: pLSI, LDA and CTM [9] using word likelihood and average rank. Using predictive likelihood and predictive rank LDA only ranks second. But compared to human interpretation of the topics output, LDA ranks first while CTM performs worse, as can be seen in figure 6. They pointed out that "topic models which perform better on held-out likelihood may infer less semantically meaningful topics" [11, p.1] and instead describe a different approach, using human judgement.

- Word Intrusion

  Word intrusion is an evaluation method aiming at measuring the coherence of topics. To do so, a topic is picked at random and its five most probable words are selected to illustrate the topics theme. To test the coherence of this theme, one word is picked as an intruder, which should not belong to the topic. To get optimal results, it is important to ensure the intruder is different enough. This is done by choosing a word with low probability in the selected topic, ensuring it is not a word native to the topic, and with a high probability in a different topic, to ensure it is not irrelevant to every topic. This procedure creates a list of six words, with five supposedly coherent words and one incoherent word. The subject then selects a word from this set that does not follow the interpreted theme.

  An example would be

  – dog, cat, horse, apple, pig, cow

  It is easy to conclude apple does not belong to the thematic group of animals, which could be the theme of the topic. With a given set of

  – car, teacher, platypus, agile, blue, Zaire

  selecting the intruder is more difficult since there is no obvious theme connecting the words. Repeating this with a subject over multiple topics can be a good measure to estimate the models accuracy.

- Topic Intrusion

  With the help of topic intrusion a documents distribution over topics can be evaluated. The test set contains the documents three highest ranking topics and one intruder topic, which is randomly chosen among the lowest ranking topics. The topics are illustrated by displaying their eight highest ranking words, respectively. The subject is then shown the four topics and the documents title and and excerpt of the text. By choosing what the subject thinks is the intruder, the process can evaluate the quality of the chosen topic distribution in regards to the humanly interpreted content of the document.

While Chang et al. have shown how to evaluate quality measurements for topic models, the methods proposed are unlikely to replace quantitive approaches, due to the need of human participants. Assembling a panel of judges is expensive, in time as well as financially.

Coherence Measures

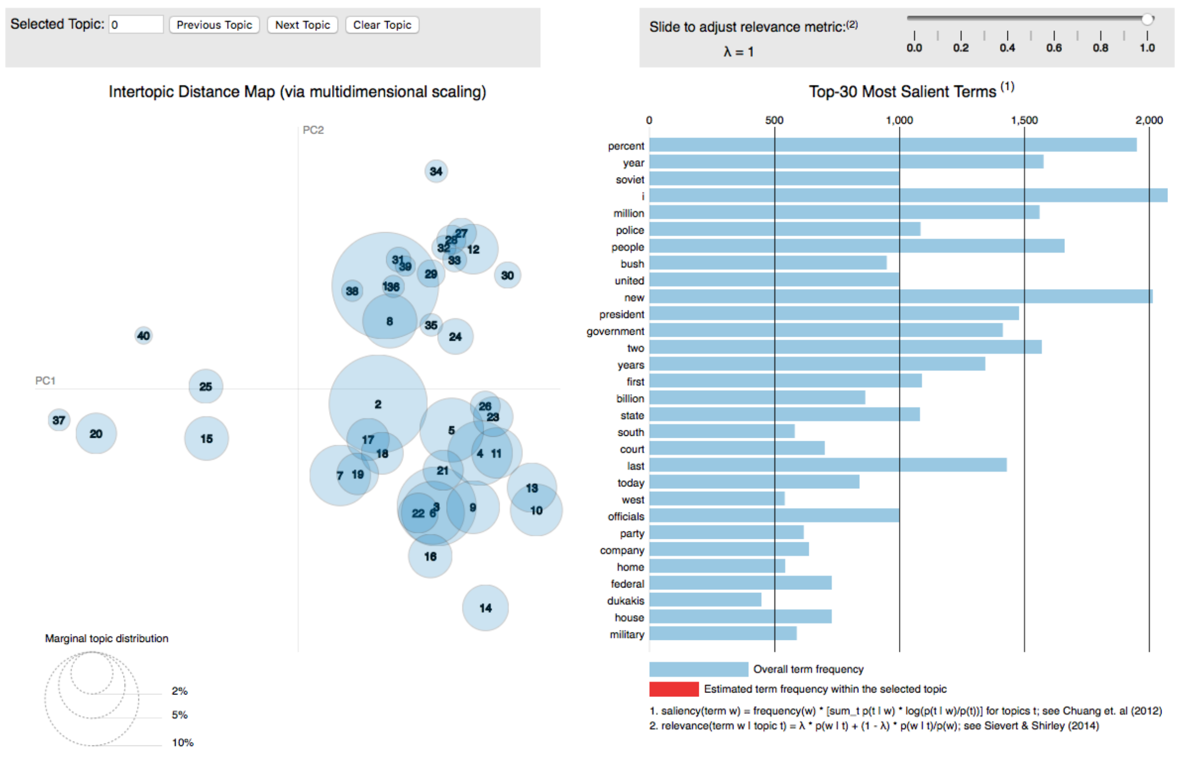
Introducing the measure of human judgement and disproving the common belief of a correlation between semantic quality of inferred topics and held-out likelihood (or perplexity), sparked a new discussion about topic model evaluation. While human judgement would not find mainstream adoption, due to its expensive process, it led to new measures which better correlate with human interpretation. One of those measures is Cv. Introduced in 2015 by Röder, Both, and Hinneburg [24], it measures a topic's coherence by measuring the cooccurrence of a topic's words in a corpus and is commonly considered the best coherence measure[8]. The disadvantage, and the reason I did not choose Cv for this thesis, is that it is computationally expensive. A more time efficient measure was proposed by Mimno et al. in 2011 called Umass. The idea behind the calculation of Umass is that every high ranking word is more likely to occur together with another high ranking word from its topic.

Visualization

As the quality of LDA depends on human judgement, interpreting the output of the model has become an increasing issue of research. Furthermore, creating humanly understandable topics is the ultimate goal of topic modeling and LDA. For a human to understand a topic's theme, a term's probability in a topic might not be the best ranking. Terms that occur often across a corpus tend to have a higher probability in multiple topics, making them less distinguishable. Consequently these words lose their descriptiveness and ultimately decrease the understandability of a topic. To address this issue, Sievert and Shirley [26] introduced LDAvis, a tool to visualize LDA's topics. The idea behind LDAvis is to facilitate topic model interpretation by interactively visualizing these topics. Additionally, Sievert and Shirley introduced the *relevance* of a term, a new ranking method for words of a topic. The visualization is split up in two main areas, as can be seen in Figure 7. On the left the topics are displayed on a two-dimensional graph, showing the distance between topics and between the words of a topic. On the right, the

[8]`http://palmetto.aksw.org/palmetto-webapp/?coherence=cv`

[9]`http://www.kennyshirley.com/LDAvis/`



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 7: LDAvis demo using the AP data[9]

words of the selected topic are displayed as a bar chart, sorted by their relevance. The bar chart shows each word's overall corpus frequency and topic specific frequency. By allowing the user to interact with the presentation of the topics, the meaning of a topic can be comprehended more easily. Furthermore, LDAvis allows for an overview of topics and puts them in context, resulting in a better overall understanding of the model.

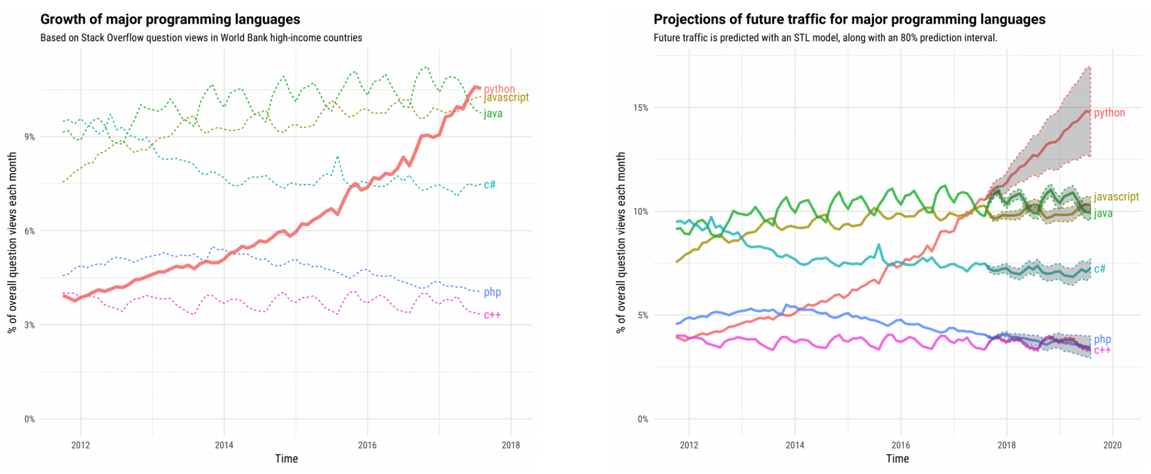
Criticism

As explained above, the assumption of LDA is that a document is created by randomly picking words out of topics to represent their distribution. This also means that the order of words is irrelevant in the process of LDA. Critics have argued that word order is relevant to reduce ambiguity and to understand the semantics of a sentence [14]. The bag-of-words model does not allow for the word order to be considered and therefore lacks semantic intelligence.

3. Technical Tools

3.1. Python

Python, released in 1991, is one of the most popular computer languages for programmers world wide, according to the PYPL index[23] and a study conducted by hackerrank [28]. A recent Stackoverflow study also showed the extraordinary growth of Python in developed countries, indicating that Python is the fastest-growing language [3].



(a) Growth of major programming languages (b) Prediction of growth of major programming languages

Figure 8: Stackoverflow study on the growth of Python

Furthermore, Python has a great reputation in the machine learning community resulting in the biggest advantage Python has to offer and the reason I chose Python for my thesis: packages. Python has numerous packages for machine learning, natural language processing, statistics and data visualization. The packages cover complex areas and are actively being developed with a big community, helping developers with well written code and extensive documentation.

3.2. Scrapy

Scrapy, first released in 2008, is a Python framework web scraping. Scrapy comes with numerous features built in that can increase the speed and robustness of a scraper [1].

- Asynchrony

  By using asynchronous requests, Scrapy can download and process multiple pages at once, making it many times faster than a successive, blocking approach, where each request has to wait for the other to finish.

- Error Handling

  Scrapy has various error handlings built in, for example following redirects, retrying failed requests and handling delays and timeouts.

- Sessions

  Scrapy automatically handles cookies and sessions

- Community

  Being an open source product, Scrapy has a big following that actively develops tools and helps to make the framework more robust.

Architecture

An overview of Scrapy's architecture can be seen in figure 9. Scrapy's architecture is split up in five main components: engine, spider, downloader, scheduler and the item pipeline. To scrape an article, the spider sends a request to the engine (1). Since Scrapy works asynchronously, the engine forwards the request to a scheduler (2). The scheduler then returns the next request (3), which subsequently gets downloaded (4). Once downloaded, the downloader returns a response with the page (5), which the engine forwards to the spider for further processing (6). The spider returns the processed page and the new request to the engine (7), which sends the processed page into the item pipeline and the next request to the scheduler (8). The item pipeline can validate the content, check for duplicates and save the item to a database or file.

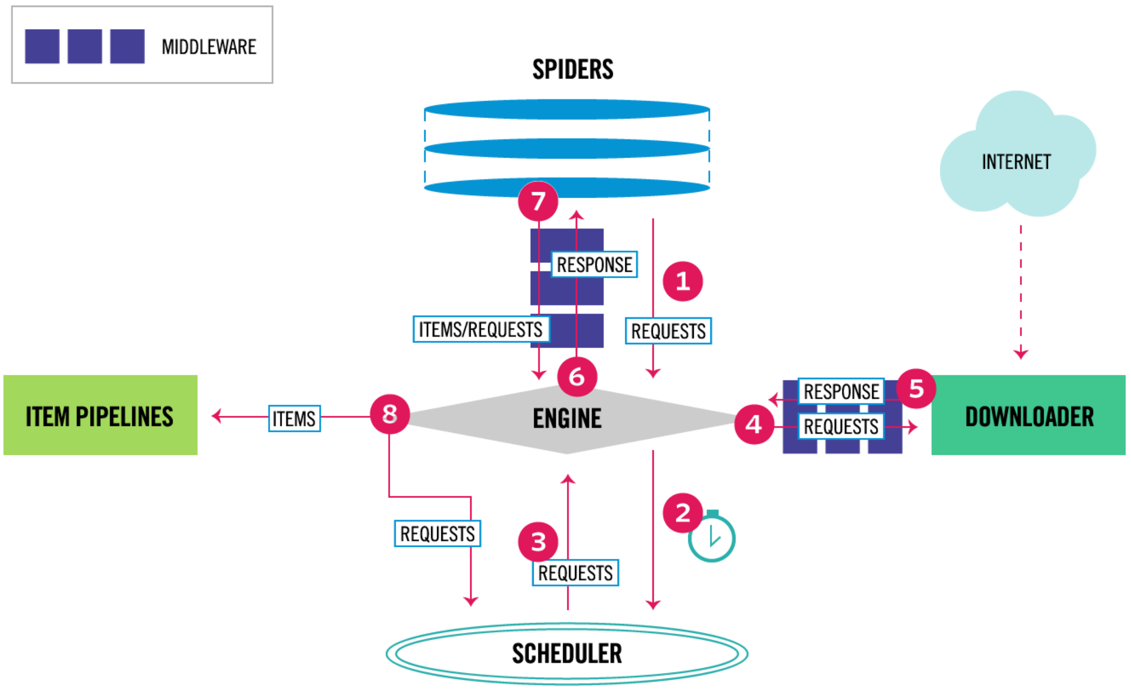


Figure 9: Scrapy's architecture split up in components[10]

3.3. MongoDB

Because scraped data can come in many forms and not every document will have the same fields, I decided against a relational database scheme and to go with a NoSQL database. First released in 2009, MongoDB is one of the most popular NoSQL databases. A NoSQL database is a database that is not relational, meaning the database structure is non-tabular, like such of a MySQL database. While there are multiple NoSQL database types such as the simple key-value store, MongoDB uses the document store. Document databases exist of documents which are made up of key-value pairs [12]. Values in a document can be of different types, such as string, integer, array or object. Furthermore, a group of documents, called collection, has dynamic schemas, meaning they do not have to be cohesive and can differ in keys and value types. This can be a great advantage over relational databases, allowing for freedom to store the data in any form desired.

3.4. Natural Language Processing Software

Python offers a vast amount of libraries and programs to use, each with their own advantages. While most tools share a variety of features, my most specific requirements were German language support, text preprocessing and topic modeling. The following tools provide those features and have been used for my textual analysis. The tools are some of the most popular open-source NLP projects, exceeding the popular Natural Language Toolkit (NLTK) project on Github, as can be seen in figure 10.

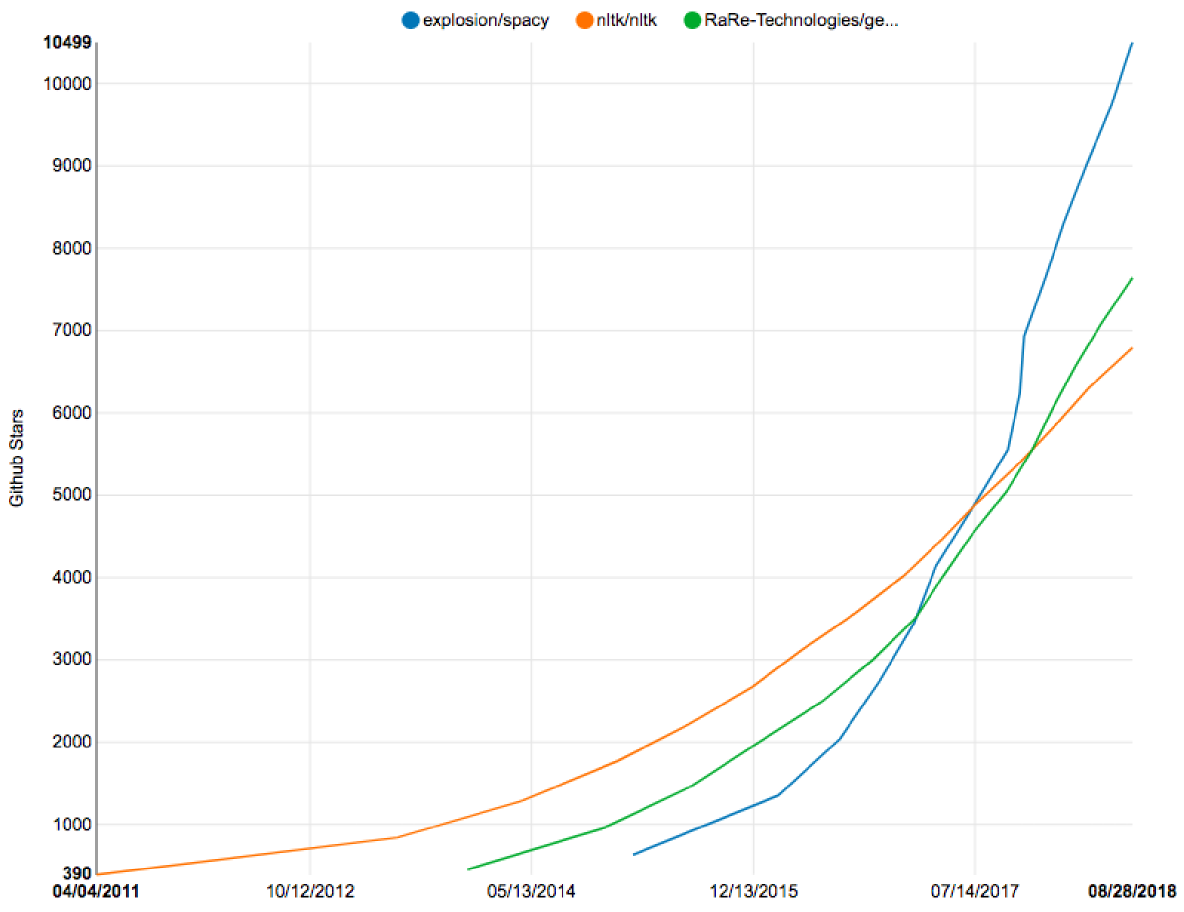


Figure 10: NLP projects and their Github stars over time[11]

[11]http://www.timqian.com/star-history/

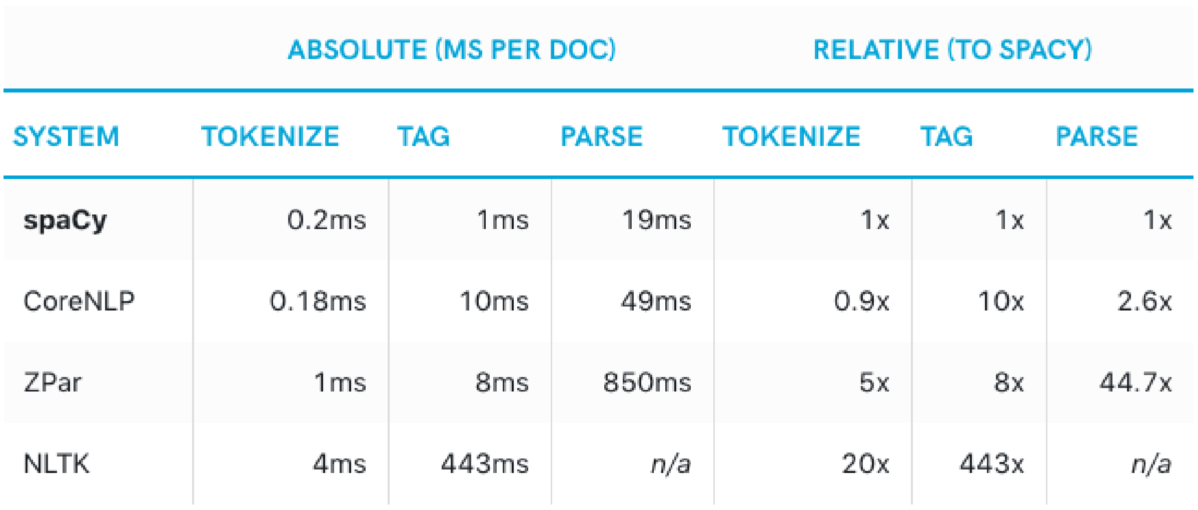
3.4.1. Natural Language Toolkit (NLTK)

The Natural Language Toolkit, or NLTK, is a Python based toolkit offering libraries, programs, corpora and other resources useful for natural language processing. Released in 2001, NLTK has a broad set of natural language processing tools such as a Part-of-speech tagger, tokenizer and n-gram model. While NLTK is certainly one of the most popular tools, its catch-all approach seemed too cluttered and did not convince me.

3.4.2. SpaCy

SpaCy, released in 2015, is a software library for natural language processing. Written in Python and Cython, a superset of Python acting as a Python to C compiler, spaCy aims at production usage, focusing on speed and accuracy. According to research done by Choi, Tetreault, and Stent, spaCy in 2015, spaCy's syntactic parser ranked first in speed and was among the top 1% in regarding accuracy [13]. SpaCy offers a wide variety of linguistic features such as tokenization, part-of-speech (POS) tagging and sentence segmentation. Next to deep learning integration and convolutional networks, spaCy additionally offers extensive support for more than 30 languages, including a 645 MB large German news corpus.

Figure 11 shows an evaluation conducted by spaCy using 100,000 English documents and four different natural language processing libraries: spaCy, CoreNLP, ZPar and NLTK. The time displayed is the mean time for the appropriate task in ms. SpaCy is therefore multiple times faster than NLTK.

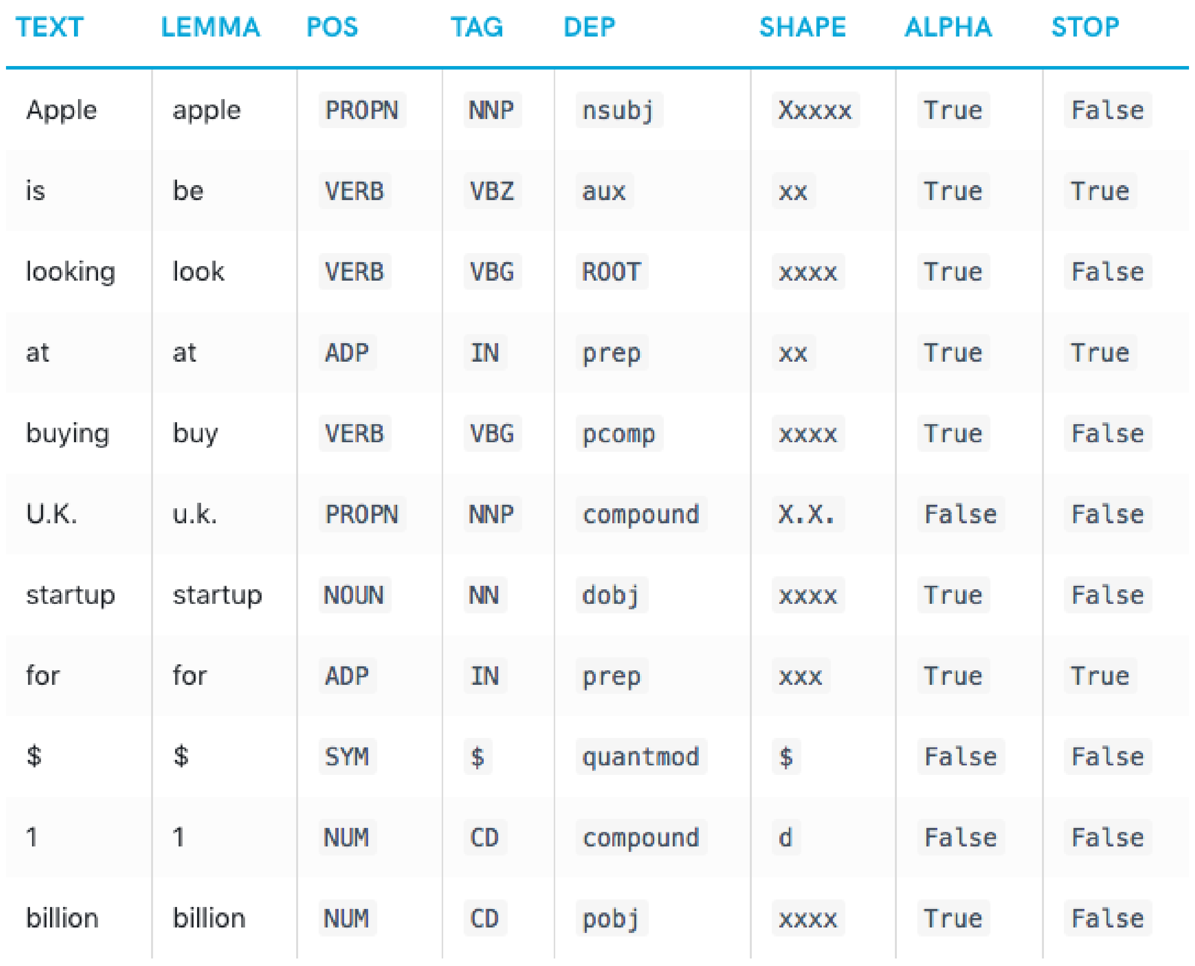


	ABSOLUTE (MS PER DOC)			RELATIVE (TO SPACY)		
SYSTEM	TOKENIZE	TAG	PARSE	TOKENIZE	TAG	PARSE
spaCy	0.2ms	1ms	19ms	1x	1x	1x
CoreNLP	0.18ms	10ms	49ms	0.9x	10x	2.6x
ZPar	1ms	8ms	850ms	5x	8x	44.7x
NLTK	4ms	443ms	*n/a*	20x	443x	*n/a*

Figure 11: Per-document processing time of different natural language processing tasks compared across various libraries[12]

SpaCy also includes linguistic information about each token. Figure 12 displays a selection of the linguistic information offered by spaCy. The lemma attribute is the token's base form, or lexical root. The pos and tag attributes are the part-of-speech tags. The tag attribute follows the Universal Dependencies scheme for English[13] and the TIGER treebank for German[14] and the pos attribute represents the Google Universal POS tag set. The dep tag displays the token's syntactic dependency, showing information such as subject, dative or negation. The shape represents the token's shape, i.e. capitalization, punctuation and digits. Alpha and stop are boolean attributes, indicating if the token is an alphanumeric character and if the token is a stop word.

[12]`https://spacy.io/usage/facts-figures#speed-comparison`
[13]`http://universaldependencies.org/u/pos`
[14]`http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/TIGERCorpus/annotation/index.html`



TEXT	LEMMA	POS	TAG	DEP	SHAPE	ALPHA	STOP
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	VERB	VBZ	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
$	$	SYM	$	quantmod	$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

Figure 12: Selection of Spacy's linguistic annotations of the sentence *Apple is looking at buying U.K. startup for $1 billion*[15]

[15] `https://spacy.io/usage/linguistic-features`

3.4.3. Gensim

Gensim is a python based natural language processing toolkit with focus on topic modeling and vector space modeling. Radim Řehůřek developed Gensim in 2009 partly with the aim of handling large amounts of data[16]. To overcome the issue of scaling, Gensim can process a stream of data, resulting in the data not having to be stored in RAM. This is still one of Gensim's main features. Furthermore, Gensim's use of NumPy, SciPy and Cython results in tasks partly outperforming optimized C code[17], such as the word to vector tool, which is also used by spaCy[18]. Next to scalability and speed, Gensim's main advantages are customizable Python implementations of popular natural language processing concepts such as latent Dirichlet allocation, hierarchical Dirichlet processes and latent semantic analysis or indexing. Gensim also offers informative material such as extensive documentation and tutorials, as well as community support, e.g. Google Groups and Github.

[16]`http://williambert.online/2012/04/interview-with-radim-rehurek-creator-of-gensim/`

[17]`https://decisionstats.com/2015/12/07/decisionstats-interview-radim-rehurek-gensim-python/`

[18]`https://spacy.io/usage/facts-figures#other-libraries`

4. Implementation

4.1. Data Collection

It is possible to either use an existing dataset or to create your own dataset. Regarding time-crucial projects, creating a new dataset is often not a feasible option. Other than saving time, the benefit of an existing dataset is also the community effect, assuming that the dataset is open for public access. The advantages of publicly accessible data, often referred to as open data, are various. Openly accessible data often means the dataset has already been used in projects and either independently verified or rejected. If questions arise, it will be easier to find knowledge on the appropriate platforms. Furthermore there might be projects that have worked with the dataset that can be built on. The disadvantage of working with open data is the limitation. While all the data is readily available, its data and its structure is fixed. If a dataset of articles for example does not include the date, it is not possible to analyze articles per period or any sort of change over time. Often limitations are not clear from the beginning since it is not completely clear which kind of data will be needed, so the limitations can occur at any time in the course of the project. While it is common to find an interesting dataset that is publicly available, it is difficult to find a dataset that fulfills specific criteria. Searching for open data that fulfills all the criteria needed for the research will often lead to compromises. The more specific the research goals get, the harder it will be to find a dataset that can fulfill all the requirements. For natural language processing, finding the appropriate dataset is more difficult than in other fields, since the data usually has to be in one specific language. For languages other than English, the amount of open data available is dramatically reduced. Finding an appropriate German language dataset that is both extensive and current is difficult, so that the benefits of creating a custom dataset become more clear, despite the challenges. The biggest selling point of a custom dataset in natural language processing is that there is a vast amount of raw data readily available on the world-wide-web. Using web scraping it is possible to access these texts. Compared to data that does not consist of natural language, the data is easier to gather. There are no measurements, experiments or surveys necessary, all the raw data is available for free. However, scraping data has the disadvantage of not being verified. The data can be corrupted due to errors in the scraper itself, for example by scraping the same article more than once or by not downloading articles behind a paywall.

Selecting Media to Compare

To analyze German online media I wanted to focus my work on a primary dataset representing the political middle as well as gather two secondary datasets representing the left and the right. For the primary dataset I would calculate a broad set of statistics, while the secondary datasets should act as comparates. The main newspaper should be a bigger newspaper that is more or less situated in the political middle, the other two should be oriented more on the extreme ends of the political spectrum, to highlight the difference in writing style.

- Neues Deutschland

  For the far left newspaper I decided to use Neues Deutschland. During the DDR, between 1946 and 1989, the newspaper was the official news platform of the socialist party in east Germany. After being owned by the PDS, the successor party of the SED, it is now partly owned by the left-wing political party Die Linke. It has a circulation of 24.477 [4] and is sold nationwide, with its offices currently in Berlin.

- Junge Freiheit

  For the far right newspaper my research came up with Junge Freiheit. The newspaper was founded in 1986 as a party newspaper of the FVP, a right wing faction. It is categorized as a right-wing newspaper, and has a nationwide circulation of 29.287 [5], with its offices in Berlin.

- Spiegel Online

The mainstream newspaper was difficult to select. An important aspect was the political orientation, which is hard to objectively quantify. Moreover, there is no political middle, each newspaper will lean somewhere more left or more right. Another aspect was to find a newspaper that has a high circulation, meaning it is a relevant newspaper for the German political society and can be seen a representation of the German political point of view. Additionally it was important to find a newspaper with a high amount of articles, accessible through the internet. Spiegel Online best met these criteria. Spiegel Online, founded 1994, is a branch of the weekly magazine Der Spiegel, founded 1947, which has been a relevant part of the journalistic history in Germany. Spiegel Online is one of the most widely read online newspaper platforms, after Upday[19], the news feed for Samsung phones and Bild.de[20] [6].

[19]Can be found under `https://www.upday.com`

[20]Can be found under `https://www.bild.de`

4.2. Scraping

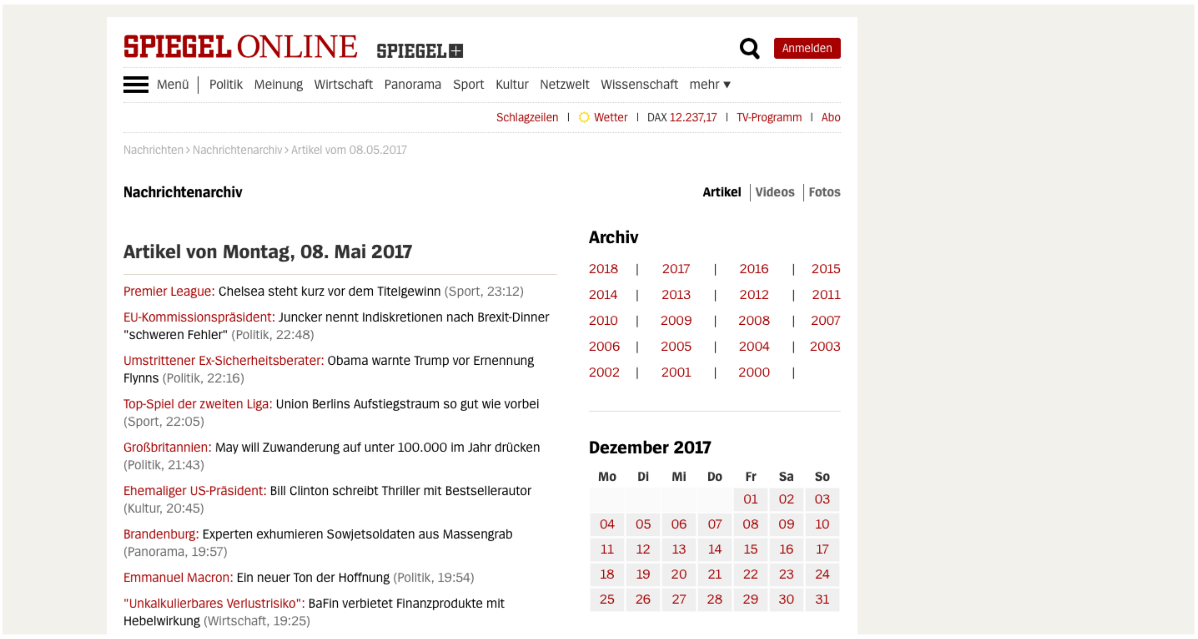
Once the newspapers to analyze were chosen, the data had to be collected. As explained above, I decided to create my own dataset, meaning I had to scrape the websites and filter through the information I wanted to have. Scraping the websites is split up into two tasks: gathering all article links and scraping the content of the articles. To gather all article links it is necessary to have a chronological collection of articles. A preexisting archive would have been useful here, but often times an archive only saves articles that appeared in the print edition, which is not the objective of this thesis. I used the request library[21] for downloading the webpages and Beautiful Soup[22] to parse the article's content during most of my thesis. While my scraper downloaded articles sequentially, upon furhter reseach I found Scrapy[23], which, among other featues, implements an asynchronous web scraper. This changed my code entirely, reduced the file size and decreased processing time dramatically.

4.2.1. Spiegel Online

Prior to my decision on Spiegel Online as my main research objective, I inspected various newspapers on their difficulty to be scraped. Websites like Bild.de do not have an openly accessible chronological list of articles and require access to their archive. Spiegel Online has an chronological list of articles publicly available, called Nachrichtenarchiv[24].

Articles are sorted by date and go back to the year 2000, making it possible to crawl the website's articles day by day, by iterating over the date. Rather than automatically clicking through the links on the page, which would have been difficult, I found it was possible to manipulate the URL parameters to display the appropriate date. Accordingly `http://www.spiegel.de/nachrichtenarchiv/artikel-08.05.2017.html` would display all articles from 8 May 2018. To crawl all archive pages I used Scrapy's Rule object, which allows specifying which links Scrapy should follow using regex. I set up a

[21]Available at `http://docs.python-requests.org/en/master/`
[22]Available at `https://www.crummy.com/software/BeautifulSoup/`
[23]Available at `https://scrapy.org`
[24]Available at `http://www.spiegel.de/nachrichtenarchiv/`


SPIEGEL ONLINE SPIEGEL+

Anmelden

Menü | Politik Meinung Wirtschaft Panorama Sport Kultur Netzwelt Wissenschaft mehr ▾

Schlagzeilen | Wetter | DAX 12.237,17 | TV-Programm | Abo

Nachrichten › Nachrichtenarchiv › Artikel vom 08.05.2017

Nachrichtenarchiv

Artikel | Videos | Fotos

Artikel von Montag, 08. Mai 2017

Premier League: Chelsea steht kurz vor dem Titelgewinn (Sport, 23:12)

EU-Kommissionspräsident: Juncker nennt Indiskretionen nach Brexit-Dinner "schweren Fehler" (Politik, 22:48)

Umstrittener Ex-Sicherheitsberater: Obama warnte Trump vor Ernennung Flynns (Politik, 22:16)

Top-Spiel der zweiten Liga: Union Berlins Aufstiegstraum so gut wie vorbei (Sport, 22:05)

Großbritannien: May will Zuwanderung auf unter 100.000 im Jahr drücken (Politik, 21:43)

Ehemaliger US-Präsident: Bill Clinton schreibt Thriller mit Bestsellerautor (Kultur, 20:45)

Brandenburg: Experten exhumieren Sowjetsoldaten aus Massengrab (Panorama, 19:57)

Emmanuel Macron: Ein neuer Ton der Hoffnung (Politik, 19:54)

"Unkalkulierbares Verlustrisiko": BaFin verbietet Finanzprodukte mit Hebelwirkung (Wirtschaft, 19:25)

Archiv

2018 | 2017 | 2016 | 2015
2014 | 2013 | 2012 | 2011
2010 | 2009 | 2008 | 2007
2006 | 2005 | 2004 | 2003
2002 | 2001 | 2000 |

Dezember 2017

Mo	Di	Mi	Do	Fr	Sa	So
				01	02	03
04	05	06	07	08	09	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 13: Spiegel Online Nachrichtenarchiv showing all articles for 8 May 2017[25]

rule for Scrapy to follow each link resembling an archive page. Another Rule object is set up for the article links, which then get parsed.

To process the content of the articles, it was necessary to understand the structure of the page and which information I wanted to extract. The most important information to scrape was the text. Secondly, the date was important in order to put the articles in context and compare events across newspapers. Because articles differ so much in content, speech and relevance, I also decided to extract the categories of the article to later filter articles, if necessary, before further analyzing them. Furthermore, for an easy identification and to look up the actual website again, I selected the url as a quasi ID. In general it is better to gather more information than needed, since going back and selecting additional information will be more time consuming. Consequently, I chose every text included in the article, such as *headline intro*, a short sentence or keyword regarding the topic of the article, *headline, article intro*, a short introduction of the following article. Another headline could be found in the breadcrumbs, which represents the title in the url. Why this differs from the original headline is unclear, but most likely it is a technique for Search Engine Optimization.

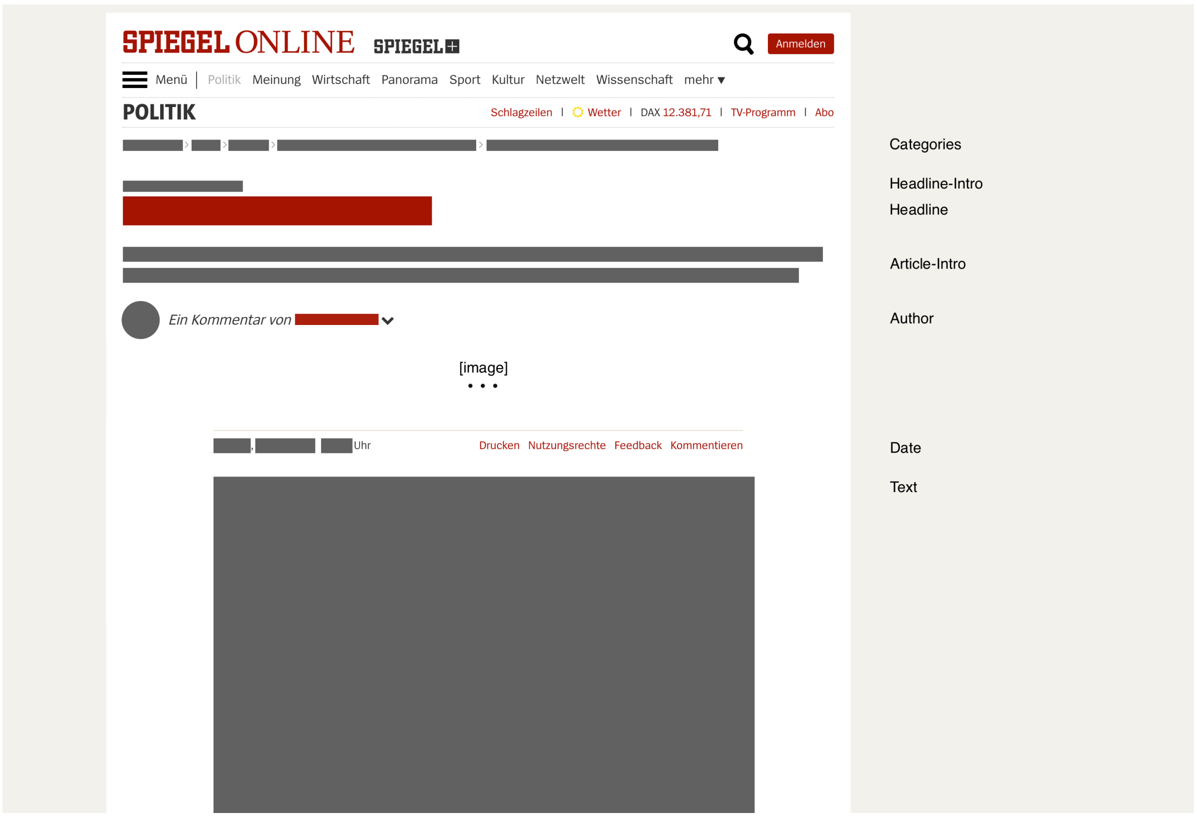


Figure 14: Typical structure of a Spiegel Online article (own figure)

To measure the amount of premium content, meaning content that is only accessible after purchasing a subscription, I wanted to record if the page has a paywall. Spiegel Online introduced its premium content Spiegel Plus in 2016 using the payment service LaterPay[26]. The paywall worked by displaying the first 1000 characters free to read. The content after that was obfuscated using Javascript. Therefore, the presence of the class `.obfuscated-content` meant the page had a paywall. As reported by Andreas Zeller[27], Spiegel Online used the Caesar cipher[28]. This encryption technique uses the letters order of the alphabet, say A has the order 1 and Z has the order 25, and shifts the letter by a certain amount. For example, if a given word "Hello" is shifted by 3,

[26]`http://www.spiegel.de/dienste/klaus-brinkbaeumer-und-florian-harms-start-von-spiegel-plus-a-1099762.html`

[27]`http://andreas-zeller.blogspot.com/2016/06/spiegel-online-nutzt-unsichere-casar.html`

[28]`https://en.wikipedia.org/wiki/Caesar_cipher`

its cipher would be "khoor". To decrypt the obfuscated text I wrote following short function:

```python
def decrypt(string):
    def decipher(char):
        return chr(ord(char)-1) if char != ' ' else char
    return ''.join([decipher(char) for char in string])
```

After losing my database to an SSD failure, I had to scrape Spiegel Online again. By then, Spiegel Online had reorganized their premium content structure[29]. All Spiegel Plus articles up to 28 May 2018 were now free to read, rendering my statistical analysis of paywall content on Spiegel Online obsolete.

4.2.2. Neues Deutschland

Neues Deutschland, while not having an interactive archive, has a paginated list of articles grouped by publishing date, e.g. `https://www.neues-deutschland.de/ausgabe/2017-05-08`. Using Scrapy's follow function, I used the pagination button to crawl through all editions available online. In addition to the data structures I used for the Spiegel Online, I also extracted the website tags, which describe the articles content in keywords. Additionally I recorded if the article had a paywall. Neues Deutschland introduced their paid subscription category, called ndPlus, in 2017[30]. NdPlus content can be accessed only as a logged in user with a valid subscription. Once logged in as such, the article does not include information about being an ndPlus article. This led to some difficulty in recording paywalls. To avoid scraping each page twice, once as a logged in user and once without authentication, I added the boolean, ndPlus or not, to the response as meta information. This allowed me to access the responses meta information later on in the pipeline, during processing.

[29] `http://www.spiegel.de/plus/willkommen-beim-neuen-spiegel-a-b615ca16-2e30-43eb-87d2-43d3cf0965df`

[30] `https://www.neues-deutschland.de/artikel/1064206.das-neue-neue-deutschland.html`

4.2.3. Junge Freiheit

Junge Freiheit does not have any form of archive or collection of articles grouped by date. Nevertheless I found that `https://jungefreiheit.de/page/2` displays articles in chronological order with a pagination menu on the bottom. Since Junge Freiheit does not have a paywall, I only extracted the main data points also extracted from the Spiegel Online articles.

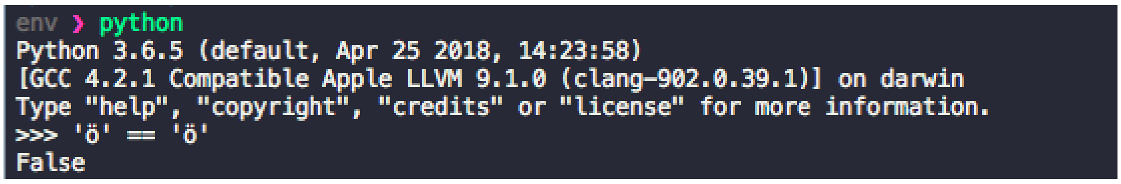
4.2.4. Cleaning Data

To ensure the data's logical integrity and uniformity, it is important to normalize the data before saving it to the database. While I did not want to normalize it too much, since reducing a layer of information before saving it to the database can not be undone, there are a few important steps.

- Date Normalization
  To persist the data in MongoDB, I transformed the string data into a Python datetime object. While Python offers the `strftime` method, I used the powerful parser module, which allows an easy way to automatically parse a date string into a datetime object.

- Unicode Normalization
  While representations of a character in a text might look the same, that does not mean they are the same combination of bits. This stems from the different encodings that can implement unicode. Different encodings can have distinctly identifiable code points, while rendering an identically looking glyph. As an example I had an issue not being able to filter out words that were on my stop word list. On further inspection, I noticed every problematic word contained an umlaut. Reducing it to the minimum, I compared the umlaut from my stop word list with the assumed same umlaut from my database. As can be seen in Figure 15, the two characters were not the same. Even though the two characters look the same, one might be an umlaut while the other is defined as a diaeresis. The

Latin, the Greek as well as the Cyrillic alphabet have identically looking glyph[31]. Therefore, unicode normalization is essential to ensure the uniformity of encoding. Using one of the four unicode normalization algorithms can resolve this issue. The best solution specific to the umlaut - diaeresis issue was the Normalization Form Canonical Composition (NFC), which decomposes the characters before it recomposes them by canonical equivalence, meaning the assumed similarity of the characters graphical display[32].



```
env ❯ python
Python 3.6.5 (default, Apr 25 2018, 14:23:58)
[GCC 4.2.1 Compatible Apple LLVM 9.1.0 (clang-902.0.39.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 'ö' == 'ö'
False
```

Figure 15: Unicode glyph ambiguity (own figure)

[31]https://en.wikipedia.org/wiki/Diaeresis_(diacritic)
[32]https://en.wikipedia.org/wiki/Unicode_equivalence

4.3. Data preprocessing

Over the course of my thesis the applied techniques for preprocessing varied. This list will give a short summary of all the techniques used over time.

4.3.1. Tokenization

After using Python's `strip` method to split words by whitespace for most of the time, I later used spaCy's tokenization included in their processing pipeline. An improved approach would have been to additionally use an n-gram model to keep words such as *New York* together.

4.3.2. Removing Characters

To clean the text of unwanted characters such as punctuation, I used a `translation table` object to remove all `string.punctuation` characters from a text, which uses string operations in C for high performance. It is possible to just filter out all non-letter characters, but this would remove too much information, such as numbers and hyphens. The report of the disappeared Malaysian airline flight *MH370* would result in a token `[MH]`, deleting crucial information. There can also be unicode escape characters, such as \n, that need to be removed. To do so I developed a regex to get rid of numbers that are not part of a word, unicode escape characters and specific Latin unicode characters that are not included in `string.punctuation`.

4.3.3. Normalization

To ensure the consistency and uniformity of the text I applied various normalization techniques. To begin with I lowercased the entire document, since capitalization only carries little semantic value, if at all. Furthermore I used lemmatization to remove morphological redundancy. I first used the IWNLP lemmatizer[33], but later replaced it with spaCy's lemmatizer for higher accuracy and better performance.

[33]`https://github.com/Liebeck/IWNLP.Lemmatizer`

4.3.4. Word filtering

Removing unwanted words also means reducing noise and dimensionality. The problem with removing unwanted words is figuring out which words are unwanted. A simple form is stop word removal, filtering out all words that occur in the stop word list. Since most tools do not have a extensive German stop word list, NLTK for example has a German stop word list with 231 words, I used the German stop word collection from `https://github.com/stopwords-iso`, a repository dedicated to collecting stop words for various languages. The German stop word list contains as many as 623 stop words. Because of spaCy's German language support I experimented with their stop word detection functionality. Because spaCy was using a lowercase stop word list but did not lowercase the text, the stop word detection was faulty. After a conversation on a Github thread[34], I later replaced the stop word list with spaCy's fixed stop word tagging for ease-of-use and better performance.

While this helps remove known words with low semantic value, it most likely only includes a small percentage of unwanted words. A more radical approach I took on later in the thesis was filtering out a relative or absolute amount of the highest frequency words and lowest frequency words.

[34]`https://github.com/explosion/spaCy/issues/1922`

5. Data Analysis

5.1. Insights

5.1.1. Dataset size

As can be seen in figure 16, Spiegel Online and Neues Deutschland both have comparable amounts of data. Junge Freiheit, with 23,391 articles, is notably smaller. While there is

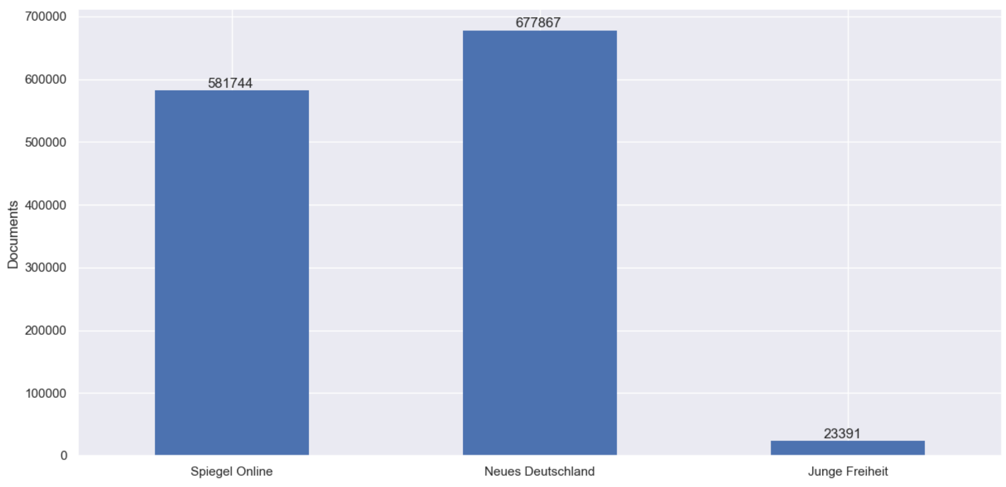


Figure 16: Overview of the dataset and its size

a minimum size for a corpus to contain enough linguistic elements to analyze, Deerwester et al. stated that a reasonable size for a corpus is 1000 to 2000 documents [17]. Crossley, Dascalu, and McNamara have researched the correlation of corpus size and quality of the model, in regards to LDA and LSA, and have not come to conclusive results [15]. While a too small corpus might not have enough linguistic elements, a too large corpus might have too much noise and will therefore be harder to analyze. For reference, 20 Newsgroups, a popular and widely used dataset containing social media posts, is a collection of 20,000 documents[35].

[35]`http://qwone.com/~jason/20Newsgroups/`

5.1.2. Articles Published a Day

The articles collected are articles posted online by the newspaper. This means the data is not equivalent to the print edition of the newspaper, in regards to quantity and quality of articles. Counting published articles by date gives an overview of the current publishing behavior and the newspaper's publishing trend. Because the publishing behavior changed from year to year (as can be seen in figure 20), I decided to display articles from 2010 until 2017 to smooth out yearly variation. The selection creates a more cohesive and better comparable dataset.

In figure 17 we can see Spiegel Online's publishing behavior over the last eight years. After observing the visible spike around mid 2012 I inspected my data to check what

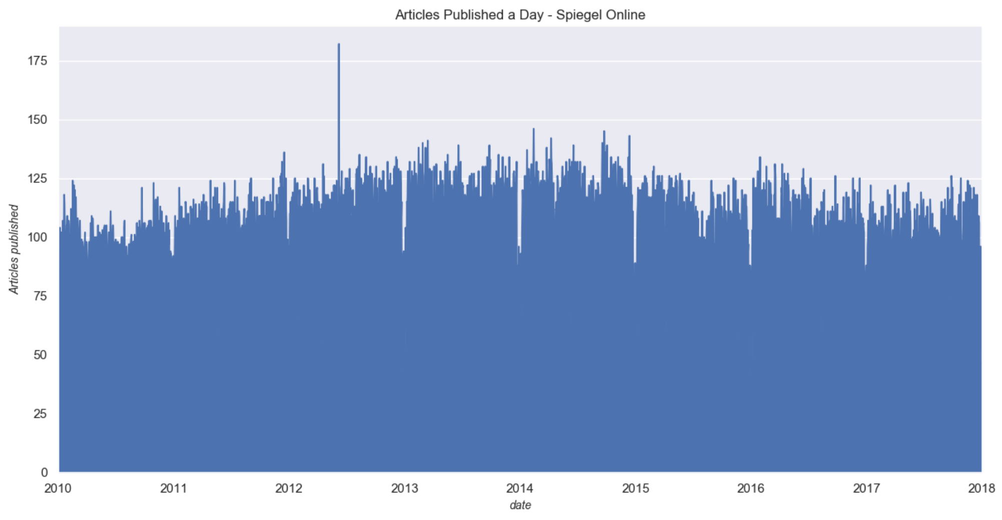


Figure 17: Articles published a day - Spiegel Online

happened that day. With 182 published articles, it is the busiest day in the history of Spiegel Online. Upon further research there was no dramatic event that was reported on. But I found out that over 50 articles had been published in one minute, between 8:10 am and 8:11 am, all of them in the category *health*. It is unclear what happened that day, but it would seem plausible to assume that there was some sort of technical malfunction or human error - a mistake. The variation of data points can therefore have purely technical reasons. The visible gaps that occur regularly are the publishing lows in December, Christmas time. The published articles per day are relatively consistent over the span of eight years. In 2010, Spiegel Online published 85 articles on average. While there was a slight increase, with a high of 103 articles a day in 2014, in 2017 Spiegel Online decreased their articles again to an average of 90 per day.

Analyzing the publishing behavior of Neues Deutschland, as seen in figure 18, shows a strong increase in daily articles around mid 2010 and again in 2012. With a strongly varying article count in 2010 and 52 daily articles on average, the publishing rhythm became more consistent over the years. In 2012 Neues Deutschland increased the articles per day to 91 on average, increasing their daily published articles by 77% compared to 2010. While not necessarily correlated, in 2012 Jürgen Reents stepped down as chief editor[36]. Consequently, variation can be the results of unrelated internal decision making or editorial policy. Similar to Spiegel Online, but not as clearly, we can see a decrease in articles published around the end of the year.

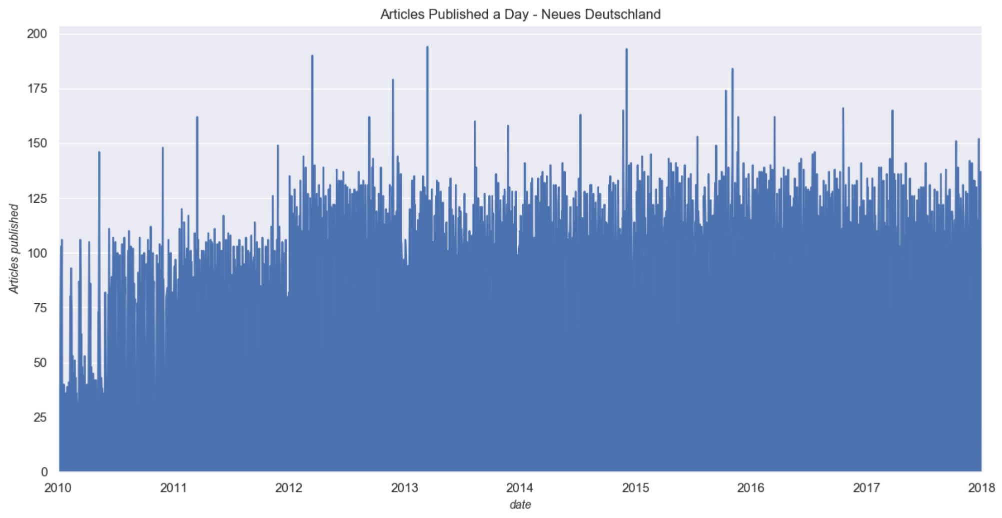


Figure 18: Articles published a day - Neues Deutschland

As already evident by the amount of articles available from Junge Freiheit, figure 19 shows that the daily published articles are significantly lower than the average published

[36]https://de.wikipedia.org/wiki/Neues_Deutschland#Geschichte

by Spiegel Online or Neues Deutschland. Due to the low number of articles, especially on weekends, when Junge Freiheit regularly only publishes one or two articles, the mean articles per day go from four in 2010 to six in 2017. If we zoom out we can see that it has not always been the case. Before 2010 Junge Freiheit's publishing behavior looked significantly different, as figure 20 shows. Research showed Junge Freiheit relaunched their newspaper in 2010, including a new layout and more pages[37].

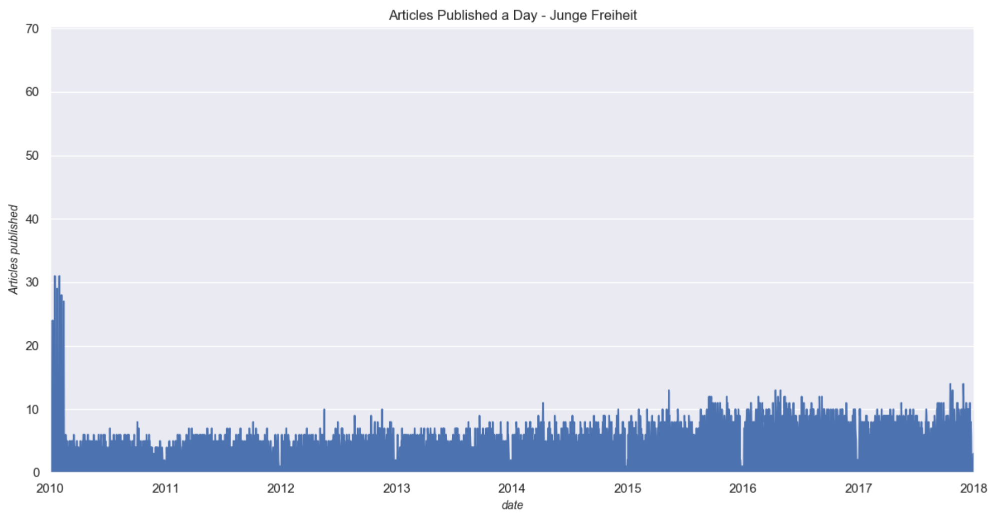


Figure 19: Articles published a day - Junge Freiheit

[37]`https://jungefreiheit.de/informationen/die-geschichte-der-jf/`

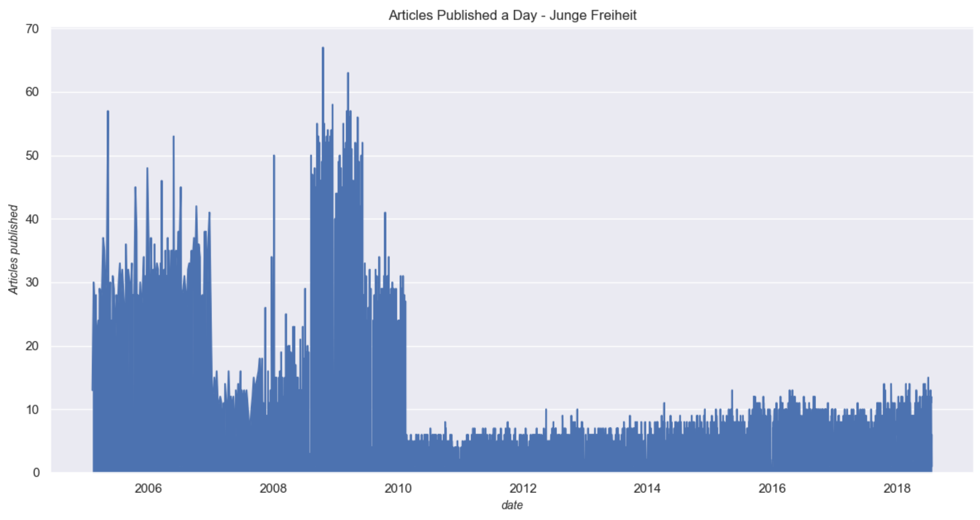


Figure 20: Articles published a day - Junge Freiheit pre 2010

5.1.3. Category distribution

Breaking up a newspaper by its categories can be helpful to show what a newspaper's focus is. The following graphs show the top ten categories, sorted by the amount of articles they contain.

In figure 21 we can see Spiegel Online's distribution of topics. The strongest category bar by far, *news*, with 581,671 articles, shows that Spiegel Online mainly sees itself as a news outlet. To make the graph clearer I filtered this category out. Next to *politics* (145,517 articles), *panorama* - a category including articles about society and social issues and *business*, Spiegel Online also has a strong focus on sports, with *sports* and *soccer-news* both being in the top ten strongest categories. It can also be seen that Spiegel Online is a nationwide newspaper, covering national and international topics, with a slightly larger focus on international topics (83,311 articles) than national topics (62747 articles).

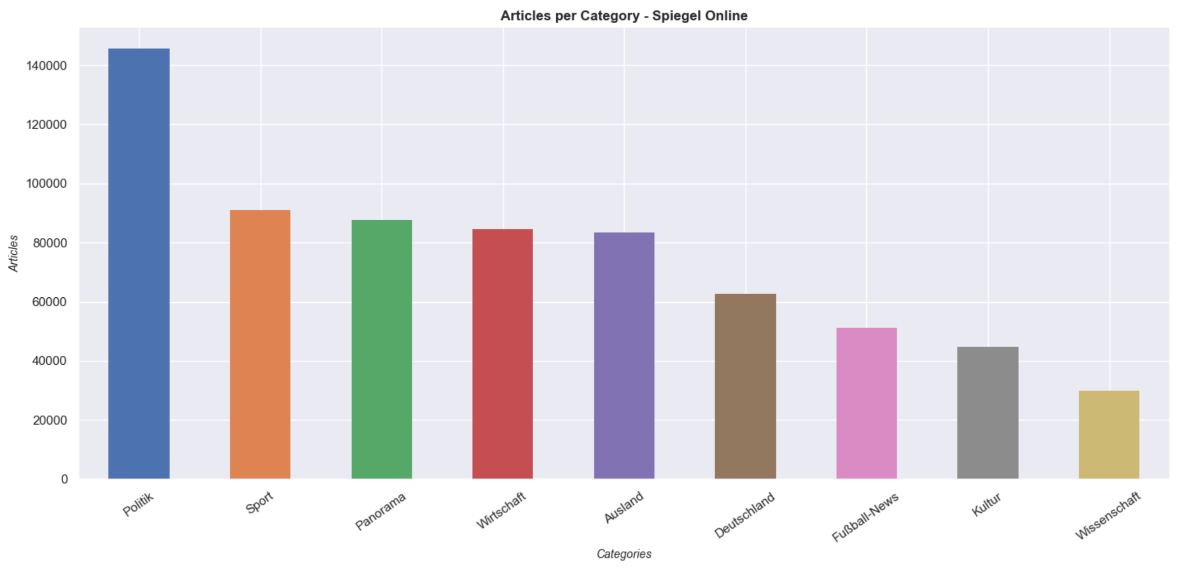


Figure 21: Articles per Category - Spiegel Online

Neues Deutschland's category distribution, as seen in figure 22 shows a strong focus on *politics*. Next to *politics* we can see *Brandenburg* and further down also *Berlin*, while there is no category *world*, indicating the regional focus.

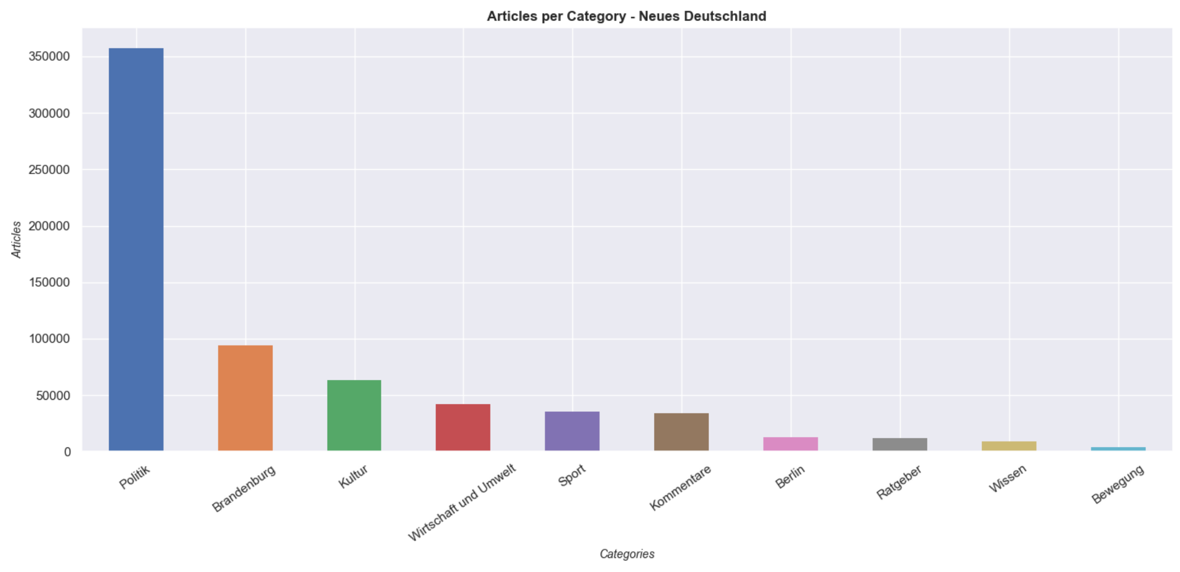


Figure 22: Articles per Category - Neues Deutschland

Figure 23 shows Junge Freiheit's category distribution. We can clearly see a focus on *politics* and *Germany*. Furthermore, *Discussion* is one of the strongest categories, indicating a stronger focus on opinionated articles.

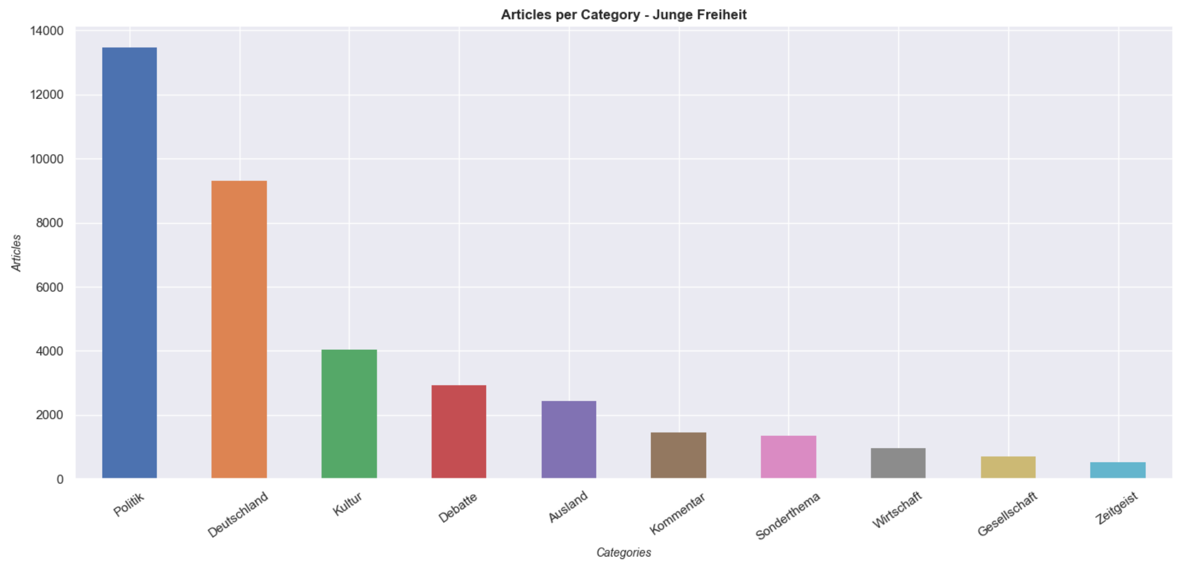


Figure 23: Articles per Category - Junge Freiheit

5.2. Authors and Text Lengths

It is common practice for newspapers to reuse and merge texts published by agencies. Articles that contain agency texts are marked by the agency's name and, depending on the extent of an author's contribution, the authors abbreviation on the bottom of the page. While this is an understandable decision for articles with little contribution, even articles written entirely by the author sometimes only include the author's name as an abbreviation on the bottom of the page. When and why this is done is unclear and differs between newspapers and editors. For my dataset I only extracted the author's name if it was written out in full on the top of the page. To find out why and when this is done, I analyzed how these articles differ in content. To do this, I first compared the amount of articles that contain the author's name with articles that do not show the authors name.

Figure 24 shows the percentage of articles with and without author name. For Spiegel Online, 18% of all articles include the author's name. With 29% almost a third of Neues Deutschland's articles contain the author's name. Junge Freiheit almost exclusively add author's names on opinion pieces and therefore has the lowest count with only 9% of all articles containing the author's name.

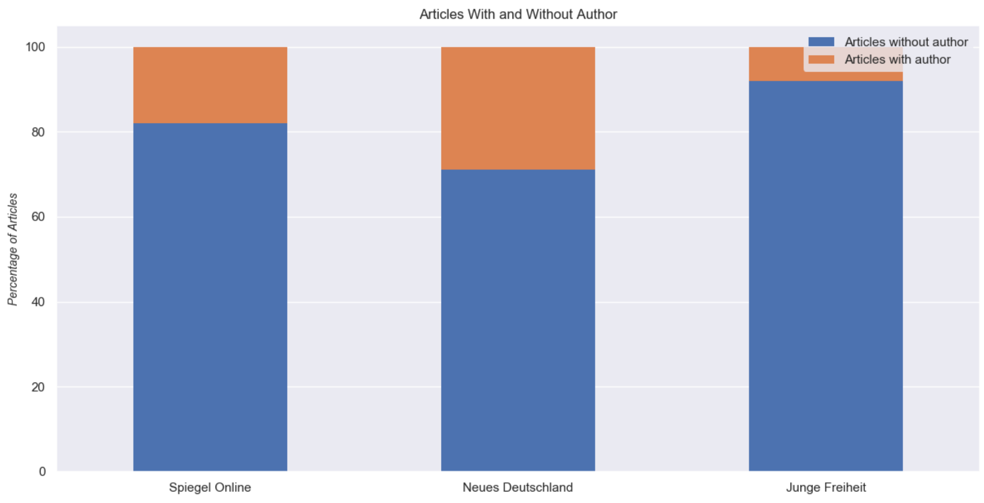


Figure 24: Overview of articles with and without authors

To further analyze the difference between articles with and without the author's name, figure 25 shows the average text length of the two categories. It shows clearly that articles that include the author's name tend to be longer. The most significant difference can be seen with articles from Neues Deutschland. Here articles including the author's name are 43% longer.

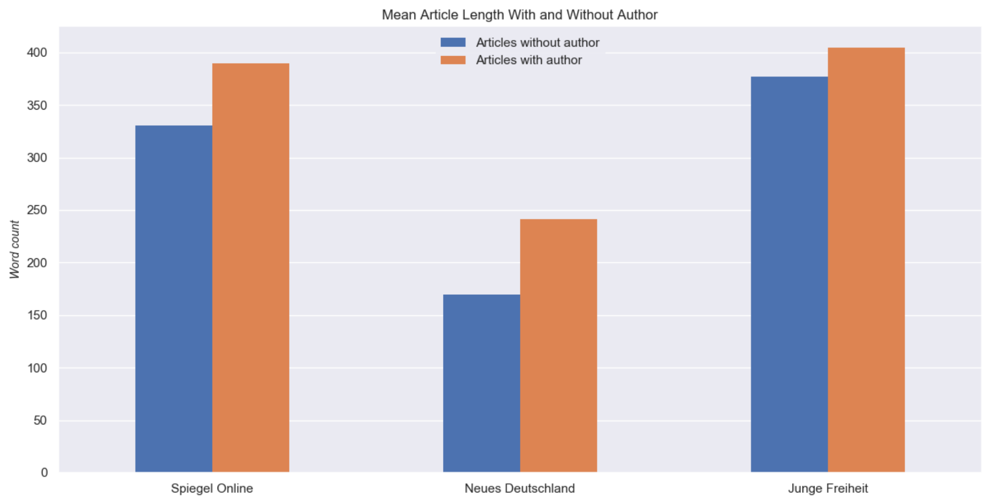


Figure 25: Text length of articles with and without authors

A shorter text length could be interpreted as a lower quality article, indicating that author's names are written out in full if the article is of higher quality.

5.3. Topic Development

In 2014 203,000 refugees applied for asylum in Germany [27]. After asylum applications in Germany kept on rising, with 477,000 in 2015 and 746,000 in 2016, the media declared the European refugee crisis. I wanted to research how the rise of migration manifested itself in German news. To quantify the media attention given to this topic, I wanted to count the amount of articles a newspaper published about refugees and migration.

One approach was to evaluate the keywords a newspaper itself added to their articles. The limitation with this approach is that the quality of the results relies on the quality and consistency of the keywords and is therefore completely dependent on the editorial decisions of the newspaper. Additionally, Spiegel Online recently removed their keywords, which is a good example for the single point of failure of this approach.

A better way was to parse the articles and then determine if the topic was migration. To determine the topic my first approach was using tf-idf. This meant calculating every word's rank by its frequency in the text compared to its overall frequency. The highest tf-idf scores would give an overview of the text's most significant words, resulting in a summary of the text. While this returned a set of words for each document, I needed a boolean to indicate if the article is about migration or not. To check if a topic was related to the topic I needed a list of words that describe the topic. I could then see if any of these words appear among the highest ranking tf-idf values. Upon implementing this approach, I realized its redundancy. Tf-idf is a simple statistical technique to retrieve information about the text with no previous knowledge. I already knew the topic the text should have and due to the boolean approach did not need all the information tf-idf was giving me. The words defining this topic were not ambiguous, meaning a text that includes the word *Flüchtling* will contain the migration topic to some degree.

```
['flüchtling', 'asyl', 'migra',
'geflüchtet', 'einwander', 'zuwander', 'refugee']
```

Therefore I created a list of words that defined the topic and searched the text for any occurrence. If the word occurred, it could be considered an article about migration. Due to German's rich morphology and the large amount of compound words, matching exact words would be an issue. However, only checking for the word stem, e.g. *migra*, would find positive matches with words such as *Migration, Migrant, Migrationspolitik*.

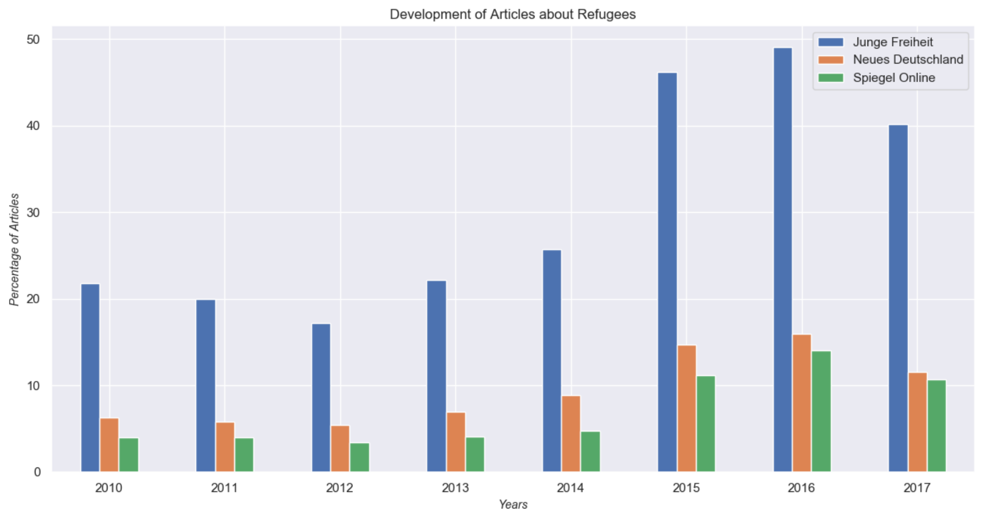


Figure 26: Articles about refugees over time

To show the development, figure 26 shows the article counts by year and as a percentage of all articles published in that year. In 2010 Spiegel Online wrote in 4% of all articles about migration. While 6% of Neues Deutschland's articles contained the topic migration, Junge Freiheit already focused more strongly on the topic, with 22% of all articles being about migration. The next two years the coverage of this topic went down for all newspapers, reaching each newspaper's low for the selected time frame. In 2013 all newspapers increased their coverage again, slightly exceeding their values from 2010. The next year we can see a trend, with coverage increasing between 38% (Spiegel Online) and 64% (Neues Deutschland) compared to the low of 2012. By 2015, the so called refugee crisis was declared. The graph clearly represents this by showing an overall increase, with Spiegel Online increasing their articles about migration by 135% to 11%. In 2016 the topic reached its maximum attention level. Spiegel Online covered migration in 14% of all articles, Neues Deutschland in 16% and Junge Freiheit in 49% - close to every second article. In 2017 all newspapers decreased their coverage about migration by an average of 31%. This still leaves the overall percentage of articles about migration on a significantly higher level than 2015, even though migration to Europe is back to the levels of 2014[38]. This could be interpreted as the lasting effect of the media coverage during 2015 and 2016.

This approach has various issues. While this approach is certainly time efficient, it lacks accuracy and gives little insight to the document. A topic containing *Migrationspolitik* might contain the topic migration to some degree, but the article could also describe the talking points of a presidential visit, with no further information about migration. It is difficult to deal with this, because the article does contain the topic migration to some extent, but at the same time is not an article *specifically* about migration. Additionally, precisely defining a topic is an issue. While migration has certainly been an issue in Germany pre 2014, the topic might have addressed Turkish migrants living in Germany or migration happening in other parts of the world, such as Africa or South East Asia. An article about a North Korean refugee seeking asylum in South Korea would also be included in the topic migration. While this topic still contains the topic migration, it is vastly different than migrants coming to Europe, seeking asylum. The difficulty of this task, taking inter- as well as intra-document context into account, is what led to the research around topic modeling and the approaches we have today such as LDA.

[38]`https://nyti.ms/2N2V6Ad`

5.4. Word Development

The word we use to describe a person migrating to a country varies. It can be *asylum seeker, refugee, migrant, immigrant* or *expat*. In German there are even more words that can be used. While the question why and when a word is used falls in the domain of social science, I wanted to analyze the development of the words over the years. I selected the following nine words that are used to describe a migrant to track their usage: *Asylant, Asylbewerber, Einwanderer, Flüchtling, Geflüchteter, Immigrant, Migrant, Refugee and Zuwanderer.* While these words in fact have different definitions, they are often used interchangeably by journalists. Assuming those are all possible words for *migrant*, I calculated the percentage of the word, relative to the count of times any word was used to describe a migrant.

Figure 27 shows the development of the words used by Spiegel Online. In 2010 the most popular words were Migrant(30%), *Einwanderer* (27%), *Flüchtling*(20%) and *Zuwanderer*(14%). Over time words such as *Zuwanderer* and *Einwanderer* slowly disappear while *Flüchtling* takes up all the lost percentages with a high of 72%. We can also see that the word *Asylbewerber* became more frequently used after 2012. The word *Geflüchteter* was up until 2015 below 1% and is since gaining frequency.

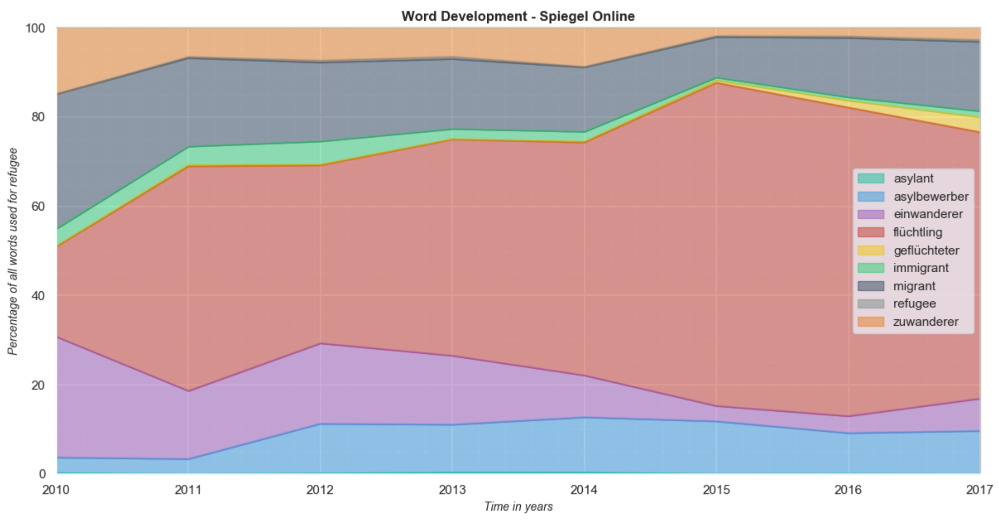


Figure 27: Word development - Spiegel Online

In figure 28 the word development as used by Neues Deutschland is displayed. Neues Deutschland predominantly used the word *Flüchtling* with 44% and *Migrant* with 26%. Over the course of the years *Migrant* became less frequent, with a low of 7% in 2015, the same year *Flüchtling*, which has gotten constantly more frequent since 2010, reached its high with 75%. From 2015 onwards *Geflüchteter* has been getting more frequent, instead of *Flüchtling*, with a usage of 16% in 2017 and an increase of 245% since 2015.

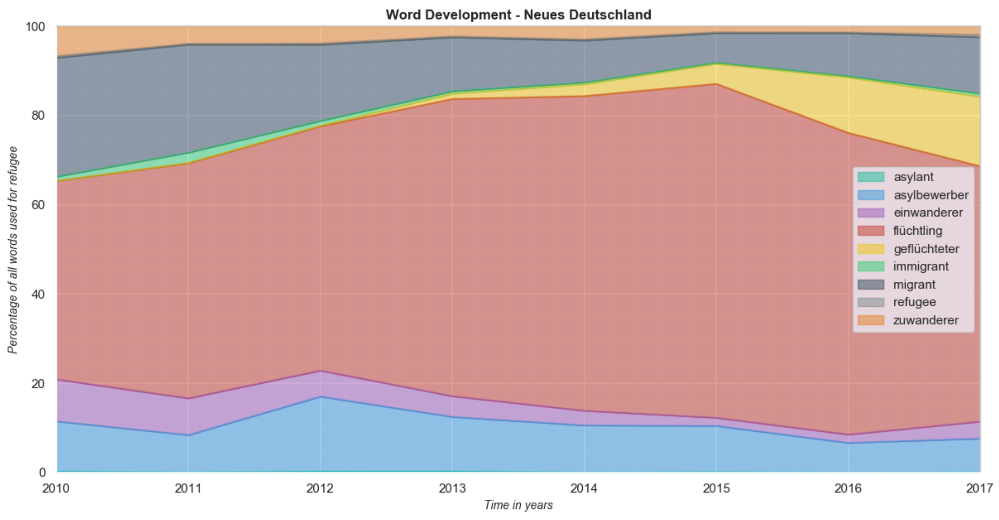


Figure 28: Word development - Neues Deutschland

The development of words used by Junge Freiheit to describe a migrant can be seen in figure 29. In 2010 Junge Freiheit's choices of words were diverse, with *Einwanderer* being used 32%, *Zuwanderer* 30%, *Asylbewerber* 16%, *Flüchtling* 10% and *Migrant* 9% of the time. Over the course of the time, *Zuwanderer* almost completely disappeared, being replaced by *Asylbewerber*. By 2015, *Asylbewerber* reached its high with a usage of 50%. The recent development shows that *Flüchtling* is now becoming the most used word. With an increase of 124% since 2014, it became the most used word in 2017, transcending *Asylbewerber*. An interesting observation was that, while *Refugee* seems not to be a popular word in German, it has been used to some degree by Spiegel Online and Neues Deutschland since 2011. The first time Junge Freiheit used the word was 2016 and has so far used it a total of six times.

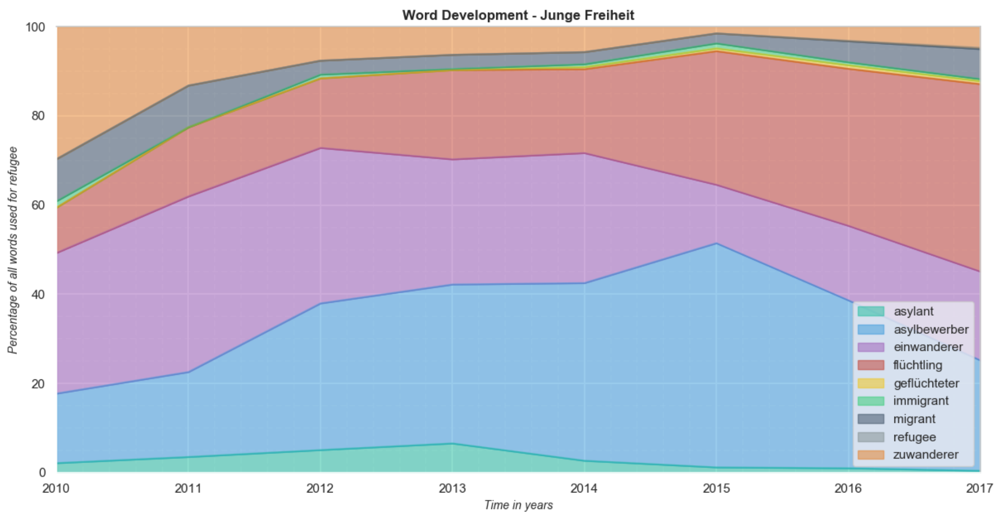


Figure 29: Word development - Junge Freiheit

The interpretation of this analysis is difficult. While there surely is a political motivation in using different words, there is also a contextual motivation. The graph can therefore also be seen as measure for context change. Words such as *Migrant, Einwanderer* und *Zuwanderer* might have been used in a different context, describing the migration by Turkish immigrant workers. The increase of *Flüchtling* could come from a change of migration that was discussed. We are able to see overall trends, such as the popularity of the word *Flüchtling*, but we can also see the Neues Deutschland and Spiegel Online using a more similar vocabulary than the right-wing newspaper Junge Freiheit, especially in regards to the words *Asylbewerber* and *Geflüchteter*. Interestingly it seems that, while for a long time avoided by the right, *Flüchtling* is now becoming the most popular word, while losing popularity on the left.

5.5. Topic Modeling

To gain a deeper insight into the corpus I created a topic model. Topic models can group semantically similar words to a topic. The model can give information about document relations in a corpus but also about word relations in a topic. To obtain quality results, an iterative process is necessary to evaluate and compare results of different parameter settings. To illustrate this process, I will show five iterations of topic models and how they were implemented.
The process of creating a topic model is broken down into the following steps:

1. Preprocess text
2. Create corpus
3. Create LDA model
4. Interpret and evaluate model

In Gensim a corpus is created by converting the tokenized text into a bag-of-words model. To have a more efficient bag-of-words model, the words are mapped to ids, using a so called *dictionary*. Assuming we have a sentence such as: *"The generation of random binary unordered trees"* - after stop word removal a dictionary might look like this:
`{'random': 1, 'unordered': 3, 'generation': 0, 'trees': 4, 'binary': 2}`
And the appropriate bag-of-wods representation would look like this:
`[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)]`
the first value being the id and the second value its count, stating that every word occurs once.

To create an LDA model, Gensim offers the `LdaModel` class. It accepts various parameters to customize the model. The most important parameters during my process were `num_topics`, to define the number of topics LDA should create and `passes`, which is the number of training passes through the model[39].

[39]https://groups.google.com/d/msg/gensim/ojySenxQHi4/jjX1RbbHERgJ

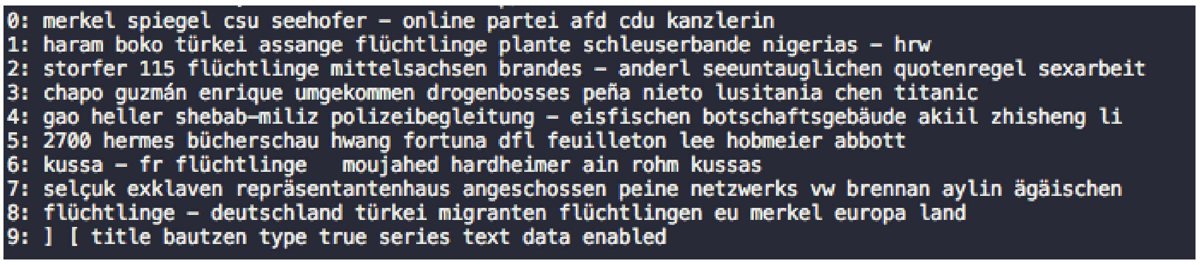
5.5.1. First Model

Text Preprocessing The preprocessing in this step was fairly simple. The first step was to remove all punctuation using Pythons `string.punctation`. Then I tokenized the text and removed stop words by matching the words with a stop word list.

Corpus To be able to adapt parameters efficiently I started out with 1,000 documents. To exclude unwanted articles, such as sudokus, and to get a more coherent set of data, I chose articles from the category *Nachrichten*. I then created a dictionary of all words in the dataset, using all 581,744 articles. Then I created a corpus of the 1,000 documents using the bag-of-words model.

LDA Model For the LDA model I changed the parameters to achieve higher quality results

Interpretation At this time I did not know about LDAvis. To interpret the model I printed out the ten most relevant words in each topic, as can be seen in figure 30



```
0: merkel spiegel csu seehofer - online partei afd cdu kanzlerin
1: haram boko türkei assange flüchtlinge plante schleuserbande nigerias - hrw
2: storfer 115 flüchtlinge mittelsachsen brandes - anderl seeuntauglichen quotenregel sexarbeit
3: chapo guzmán enrique umgekommen drogenbosses peña nieto lusitania chen titanic
4: gao heller shebab-miliz polizeibegleitung - eisfischen botschaftsgebäude akiil zhisheng li
5: 2700 hermes bücherschau hwang fortuna dfl feuilleton lee hobmeier abbott
6: kussa - fr flüchtlinge   moujahed hardheimer ain rohm kussas
7: selçuk exklaven repräsentantenhaus angeschossen peine netzwerks vw brennan aylin ägäischen
8: flüchtlinge - deutschland türkei migranten flüchtlingen eu merkel europa land
9: ] [ title bautzen type true series text data enabled
```

Figure 30: LDA model displaying the ten most relevant words of each topic

We can see some coherence, such as in topic 0: *merkel, csu, seehofer, partei, afd, cdu, kanzlerin* all have a logical connection and the topic can be categorized as *domestic policy* and *CDU/CSU*. We can also see that there are some words, such as *spiegel* and *online* that should be filtered out. Additionally, the - punctuation hints that the punctuation removal is still suboptimal. Going further down we find less coherent topics. Furthermore, there are various nonsensical words, such as *fr, hwang*, numbers and names that hold no information, taken out of context. The human interpretation shows us the lack of coherence in this topic model, and for the most part gives us little information about the topics.

5.5.2. Second Model

Text Preprocessing To optimize the results I added a filter using regex, removing a list of special characters that `string.punctation` could not remove due to the encoding it is using, such as German quotation marks „“.
Corpus I increased the corpus size to 10,000 documents. This should make sure there are enough documents to contain various linguistic features and latent topics to be analyzed and discovered.
LDA Model To improve the coherence and distribution over topics I increased the training passes to 50.
Interpretation After discovering LDAvis it was a lot easier to inspect the model. Not only does LDAvis offer an interactive visualization, resulting in more intuitive inspection possibilities, but it also offers an overview of topic distribution, size and similarity. The left side shows a similarity distance map. Topics are represented as circles. The size of the circle visualizes the distance, i.e. similarity between the words of a topic, while the similarity between topics is visualized by the distance between the circles. The right side displays the words that make up the topic, sorted by their salience. Due to limited space I will only show screenshots without a selected topic, showing only the overview of the model and it's top-30 most salient terms.

Figure 31 shows the LDAvis visualization. On the left the ten topics are displayed on the intertopic distance map. The model created four main topics, with topic one being the biggest, while the other six topics are small and almost completely overlap. These topics usually contain words with little association to a topic and low coherence.

Topic one mostly contains words regarding politics, such as *spd, merkel, union, eu, cdu, bundestag, afd.* We can see there is strong association to a topic. Interesting is also the high rank of AfD, while other oppositional parties do not occur.

The words in topic two already have a lower association to a topic, containing words such as *film, tiere, musik, tesla* and *kryptowährungen.* The topic appears to be containing a multitude of topics, showing close to no visible coherence.

Topic three seems to contain a multitude of topics. While words such as *kim, pyeongchang*

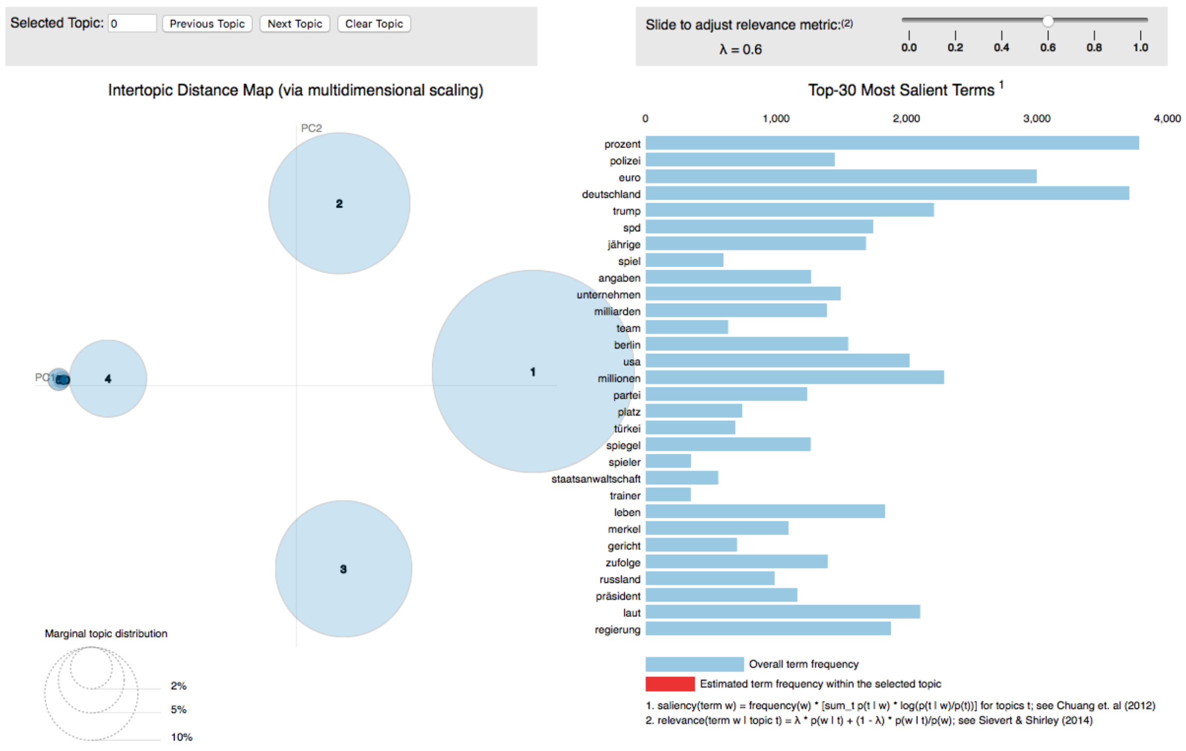

Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 0.6
0.0
0.2
0.4
0.6
0.8
1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
Marginal topic distribution
2%
5%
10%
Top-30 Most Salient Terms 1
0
1,000
2,000
3,000
4,000
prozent
polizei
euro
deutschland
trump
spd
jährige
spiel
angaben
unternehmen
milliarden
team
berlin
usa
millionen
partei
platz
türkei
spiegel
spieler
staatsanwaltschaft
trainer
leben
merkel
gericht
zufolge
russland
präsident
laut
regierung
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 31: LDA model visualization using LDAvis

and *nordkorea* could be titled *North Korea*, *erdogan, syrischen, afrin, demonstranten, ypg, kurdischen, ankara* seem to cater to one topic as well. Another theme could be *justice*, with words such as *staatsanwaltschaft, ermittlungen, festgenommen, täter, mord, angeklagten, mutmaßlichen, untersuchungshaft, freilassung*. If we add *zschäpe*, the theme could also be *NSU*. While some words are coherent among each other, there is no one topic connecting all.

Topic four mostly contains sports, with words such as *fc, partie, league, liveticker, turnier, halbfinale, bvb, hsv, bundesliga*. This topic has a high cohesion and is easily distinguishable.

5.5.3. Third Model

Text Preprocessing Because I was still having issues removing all special characters from the text I added more specific characters to my regex list and numbers that were not part of a word, since these usually add no further context to the topic.

Corpus While the previous examples had proven that my approach to topic modeling using LDA works, it did not uncover any latent topics or hidden structures. Because I was interested in researching different political topics such as *migration*, *U.S. election* or *Middle East*, I only chose documents from the category *politics*

LDA Model While keeping all values the same, I added the `random_state` parameter. Since LDA seeds the distribution over topics at random, Gensim reproduces this behavior by setting a random state and using the same value, and therefore the same "random" seeds for every new model, resulting in more comparable models.

Interpretation After using a different corpus to compute the models, the new topic model looked less well distributed, even though corpus size and topic number stayed the same. Figure 32 shows two main topics, with topic one containing 73% of the tokens and topic two 17%.

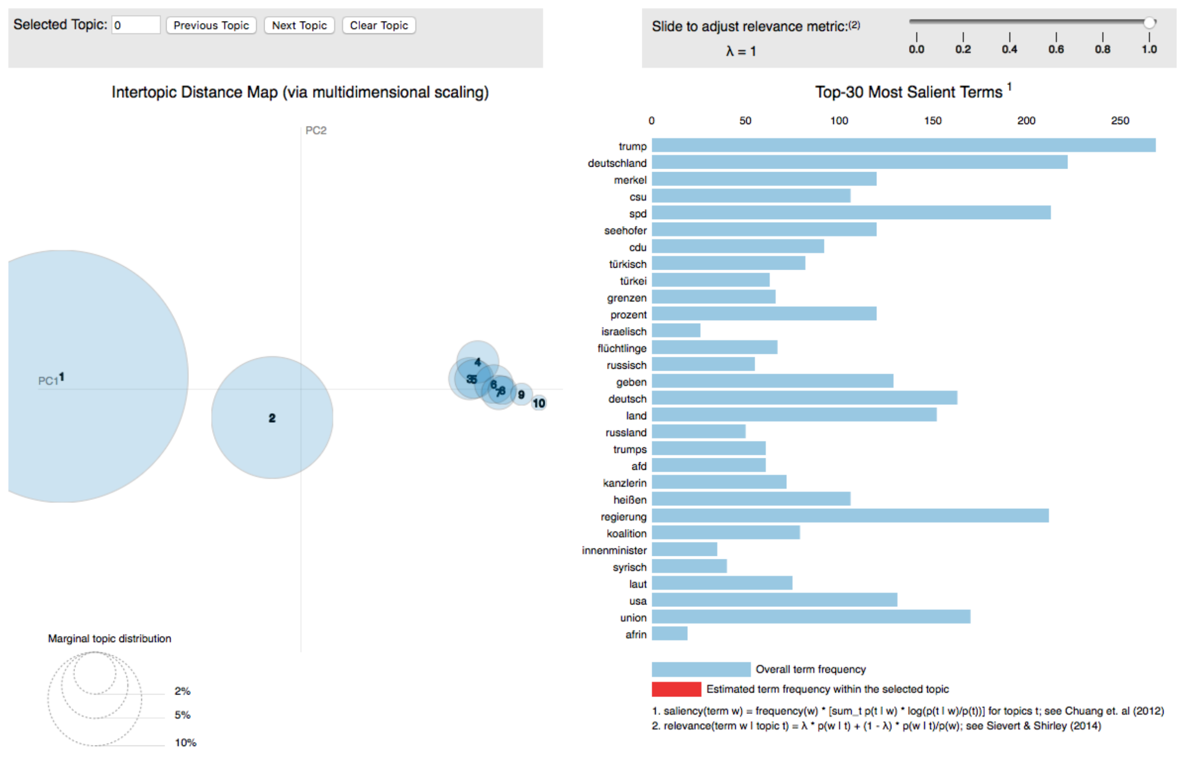

Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
4
3
5
6
7
8
9
10
Marginal topic distribution
2%
5%
10%
Top-30 Most Salient Terms 1
0
50
100
150
200
250
trump
deutschland
merkel
csu
spd
seehofer
cdu
türkisch
türkei
grenzen
prozent
israelisch
flüchtlinge
russisch
geben
deutsch
land
russland
trumps
afd
kanzlerin
heißen
regierung
koalition
innenminister
syrisch
laut
usa
union
afrin
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 32: LDA model using 1,000 documents and ten topics

Topic one contains words describing general politics with words such as *trump, regierung,* *spd, deutschland, union, usa, may, seehofer.* All words in this topic associate to European politics or U.S. relations and are therefore coherent. Because I am using a corpus exclusively containing articles about politics, ideally this topic should have been split up in multiple smaller topics such as *U.S.*, *E.U.* or *Brexit.* That this topic contains too much information can also be seen by the high percentage of tokens it contains.

The second topic contains a more specific theme. While still containing *merkel* and *spd*, showing that the topic model recognized different contexts of a word, it also includes *türkei, grenzen, flüchtlinge, transitzentren* and *syrisch.* This topic talks about migration of refugees, borders, and the the negotiations between Germany and Turkey. Because of the deal Germany and Turkey are discussing about migration to Europe, a second theme can be found in topic two. With words such as *afrin, ypg* and *kurdisch*, the topic also contains the Turkish - Kurdish conflict.

While the other topics still contain some visible themes, such as *North Korea / South Korea, Pegida / Rechtsextremismus* and *Trump / Supreme Court*, the themes are too mixed and the topics mostly include non-sensical or oddly specific words, resulting in low visible coherence.

5.5.4. Fourth Model

Text Preprocessing In addition to the existing preprocessing, I added the IWNLP lemmatizer to the pipeline. This should return the lemma for each word, if one can be found.

Corpus To discover more hidden structures, I decided to increase the corpus size to 10,000 documents, keeping the category *politics.*

LDA Model With a corpus size of 10,000 documents, I also increased the topic number to 50. To increase performance I reduced the training passes to ten.

Interpretation The idea was to use a bigger corpus and increase the topic size to 50, to divide topics and in return get more, smaller and precise topics. In figure 33 we can see the result: three main topics, containing 71% of all tokens, and 47 small, overlapping topics. Topic one was still about German politics and did not change much in regards to terms. Topic two and three increased in precision, with topic two mainly containing words about Brexit and topic three containing words about Turkey, Israel and Syria. The other 47 topics mostly contained words with little association to a topic. It seemed like the overall topic coherence had strongly deteriorated.

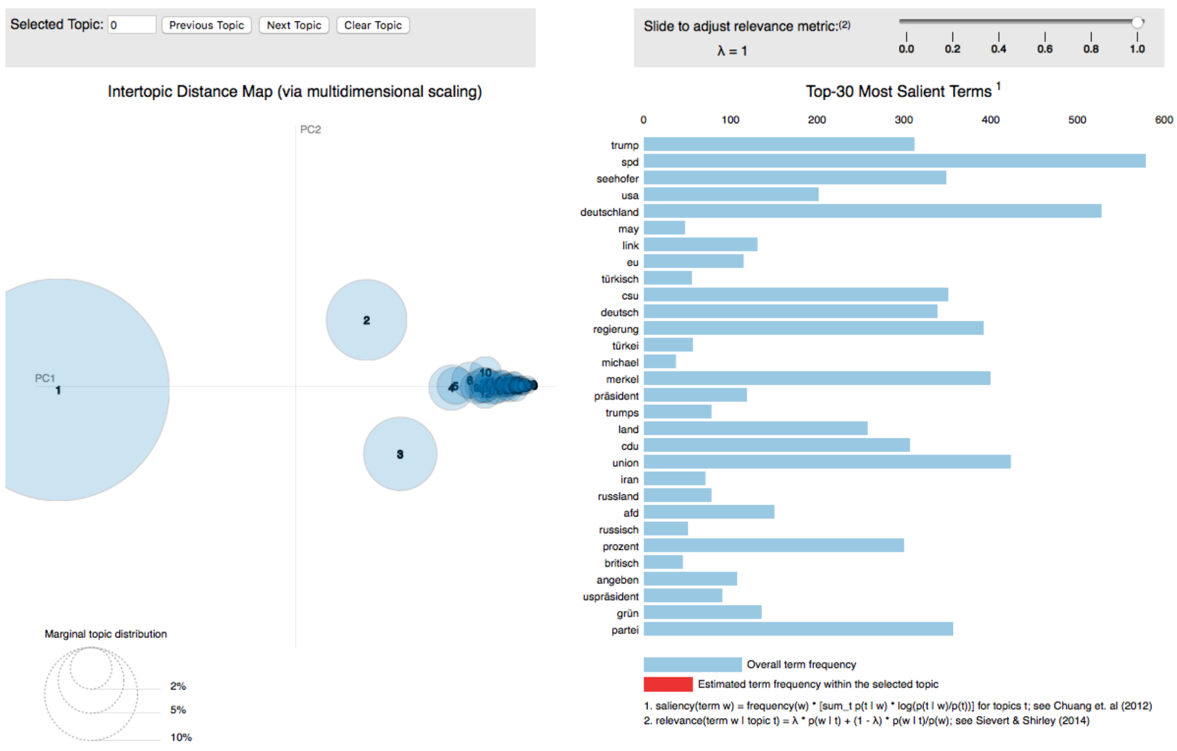


Figure 33: LDA model using 10,000 documents and 50 topics

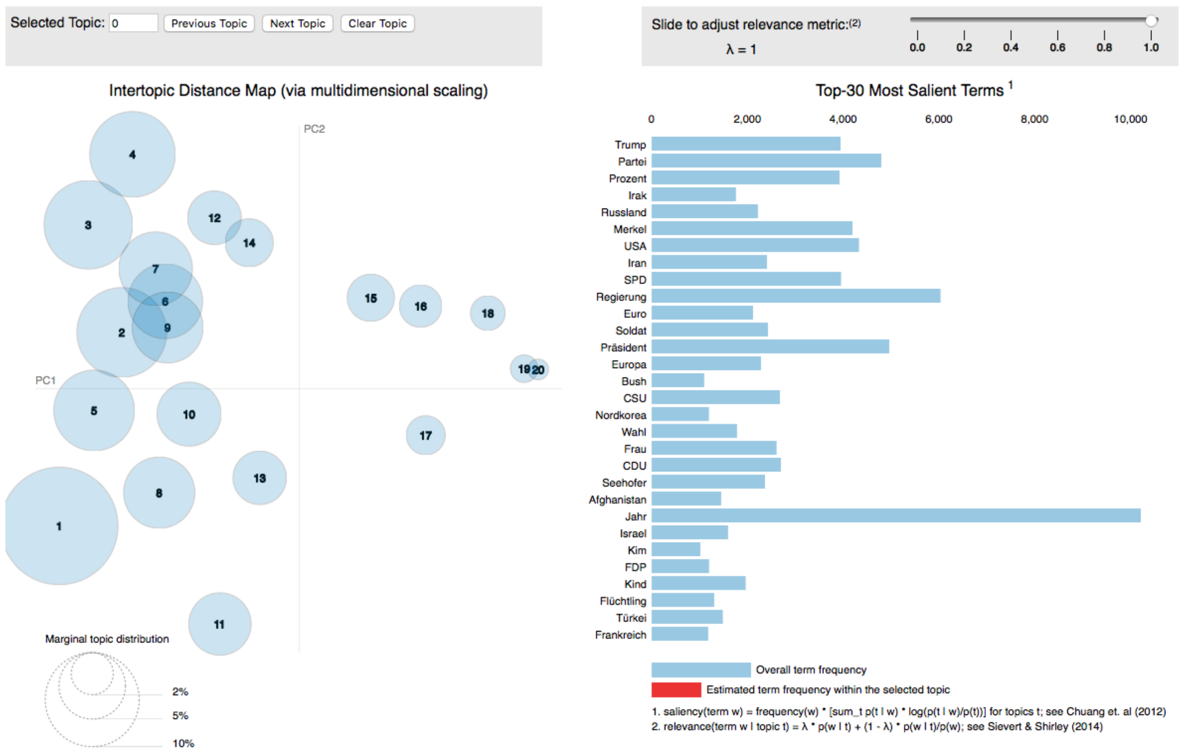
5.5.5. Fifth Model

Text Preprocessing To be able to process a larger corpus while improving the results I had to improve text preprocessing. Text preprocessing is still a heuristic approach. Which steps positively effect the outcome of a topic model are still not completely understood [25]. In their paper, Schofield et al. describe stemming as a process that can even be harmful to the outcome. Lemmatizers however have been found to increase the observed topic coherence by 8% [20]. Since the IWNLP lemmatizer did not deliver high accuracy, I decided to use spaCy. The newest spaCy version, which is still in development, added a set of improvements to the German language model. With a new, larger training set for the model, the lemmatizer outperformed IWNLP by far. In addition, I would now only add Nouns to the corpus, which I could implement using the spaCy POS tagger. Martin and Johnson have found that additionally removing all non-nouns results in less 'junk' topics, meaning articles with low or zero observed coherence, while keeping the improvements of lemmatizing. Furthermore, while this increases processing time for preparing the dataset, it decreases processing time for the model training. I also noted multiple topics containing words with a low term frequency. To reduce noise and declutter the topics, I filtered out all tokens that occurred in less than 20 documents, and all tokens that occurred in more than 35% of the articles.

Corpus I kept the corpus size at 10,000 documents from the category politics.

LDA Model Because 50 topics seemed to be too many topics, I used 20 topics for this iteration. It seemed that 10 passes were not sufficient to gain a good distribution over topics for a vocabulary of 10,000 documents, so I increased the passes to 50 again.

Interpretation The visualization of the model, as can be seen in figure 34, shows an obvious improvement over the last model. The topics are well distributed with little overlap and the topic sizes are similar, with most of the topics containing between 5%-10% of the tokens. Among the most salient terms are words such as *Trump, Irak, Russland, Merkel, Nordkorea, Türkei, Flüchtling, Israel* - giving a rough insight into the corpus. We can also see that the words are now capitalized, because the lemmatizer translated them into their dictionary form. On further inspection we can see topic one containing words associated to German politics. Topic two contains words about family, such as *Frau, Kind, Mann, Familie, Eltern.* While this topic has a strong association to a topic, it is not completely clear what the documents containing these topics look like. Topic three seems to be about the relations and interactions between the U.S., Israel, Iran, Syria and Palestine. It seems that most words in a topic have a strong association to a topic and there are only a few low coherence topics, such as topic 20. The topic model has split up topics in more detail subtopics, for example one topic for the U.S. interactions in the Middle East and one topic for Britain's interactions with the Middle East, instead of one combined topic. To improve the model further we could look at the overlapping topics, being an indication of a too small topic number. We can also see that a word like *Jahr* is in the top-30 most salient terms with a very high frequency. This indicates that the preprocessing can still be optimized, e.g. by removing more high frequency terms.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 34: LDA model using 10,000 documents and 20 topics

While the results have clearly improved over the course of five models, obtaining an optimal result is a long trial and error process. Different preprocessing techniques, parameters or corpus selection can change the outcome completely, to the better as well as to the worse.

```
    {[(13, 0.022378843)]
['0.022*"Fluechtling" + 0.009*"Innenminister" +
  ↪ 0.009*"Jahr" + 0.008*"Pauli" + 0.008*"Migranten" +
  ↪ 0.007*"Behoerde" + 0.006*"Fall" +
  ↪ 0.006*"Bundesregierung" + 0.006*"Migration" +
  ↪ 0.006*"grenzen"']}
```

Figure 35: Method displaying the most probable topic for a word. Display error due to technical issues

Word - Topic Lookup

It is also possible to look up which documents are most involved in a topic, or which topic represents a word with the highest probability. Looking up the word *Flüchtling* in our latest model, the following output shows us that it is most present in topic 13, which also includes words such as *Innenminister, Pauli* (St. Pauli most likely), *Bundesregierung, Migration* and *Grenzen*

5.5.6. Topic Number Parameter Optimization

An LDA model has various parameters that influence the outcome of the model, the most obvious being the number of topics it should create. There is no gold standard for how many topics to use and changing the parameters therefore mostly follows a heuristic approach. One idea to find an optimal amount of topics was to measure the coherence of topic models with different topic numbers. Starting with a topic number of two, I increased the topic number for each model by two, with a maximum of 50 topics, resulting in 25 topic models. Figure 36 shows the results of this experiment, with the number of topics on the x-axis and the coherence score on the y-axis, where a higher score means higher coherence.

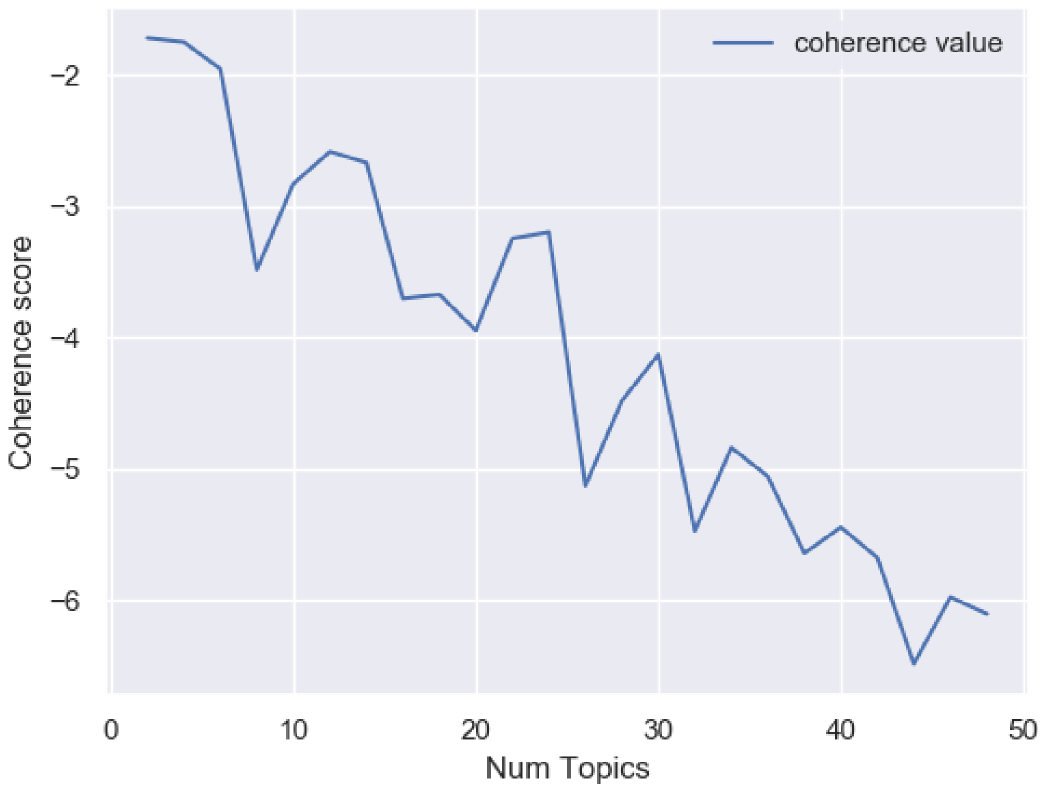


Figure 36: Coherence Scores of LDA models with increasing topic number

The model with only two topics obtained the highest coherence score. With increasing topic number, the coherence score tends to decrease. Before plotting the results I expected an increase in coherence with a higher number of topics, until a certain point is reached after which it would flatten out or decrease again. Interpreting the coherence score requires a certain degree of intuition and knowledge about the corpus. With a corpus of 10,000 documents, it can be assumed that two is not the optimal number of topics. Between 10 and 30 we can see two local maxima, one at 12 and one at 24. It can be assumed that these values will perform better thanother values in that range. Why the results turned out different than expected is not quite clear. One possibility might be that I did not increase the topic number enough, showing only a decline. Another possibility might be intrinsic to the measure. The measure could be relative to the number of topics, meaning that a coherence score of -2 of a model with two topics is very bad, while a coherence score of -3 is good for a model with 24 topics.

Further topic models of Neues Deutschland and Junge Freiheit, using a corpus containing only articles from 2017, can be found in the appendix.

6. Conclusion

6.1. Results

The goal of this thesis was to extract and analyze information that could be extracted from a text, based on examples in the domain of news media. I decided on three German newspapers with different political orientations. In the process I collected my own data, prepared it and analyzed it in various ways. The data collection was more time consuming than expected, but in the end resulted in more interesting data. Data preparation turned out to be a crucial task. Using a German corpus made the preparation more complex, with special unicode handling, lemmatization and POS tagging not typical in the English language literature. Thanks to great tools and available language support, the end results were not notably effected. The statistical analysis led to some insights in publishing behavior and article categories. More extensive analysis showed how increasing migration impacted German news. Furthermore, the development of words used to describe migrants was visualized, hinting at a trend in word choice for different political orientations. The most complex analysis was done using unsupervised machine learning techniques to find topics newspapers write about, and to unveil hidden structures. This process, called topic modeling, was done using LDA. With help of extensive research and trial and error it was ultimately possible to achieve quality topic models.

6.2. Potential

A textual analysis can always be improved upon, with more time. The outcome, for example, can see better results by using n-gram models to group collocations together. Furthermore, the analysis could branch out into fields such as text similarity, which shows and visualizes similarities between different newspapers. Articles, such as those about migration, could be semantically analyzed and compared. All this could result in interesting interpretations. While I restricted myself from further political or sociological interpretations of my results, it is possible to see that the topic model is generally applicable as the technical basis for such research.

References

[1] [Online; accessed 18-August-2018]. May 2016. URL: https://www.quora.com/Why-would-some-use-scrapy-instead-of-just-crawling-with-requests-or-urllib2.

[2] [Online; accessed 08-August-2018]. Aug. 2018. URL: https://www.crummy.com/software/BeautifulSoup/.

[3] [Online; accessed 17-August-2018]. Mar. 2018. URL: https://stackoverflow.blog/2017/09/06/incredible-growth-python/.

[4] [Online; accessed 11-August-2018]. 2018. URL: http://www.ivw.eu/aw/print/qa/titel/1680?quartal%5B20182%5D=20182#views-exposed-form-aw-titel-az-aw-az-qa.

[5] [Online; accessed 11-August-2018]. 2018. URL: http://www.ivw.eu/aw/print/qa/titel/8020?quartal%5B20182%5D=20182#views-exposed-form-aw-titel-az-aw-az-qa.

[6] [Online; accessed 11-August-2018]. July 2018. URL: https://de.statista.com/statistik/daten/studie/154154/umfrage/anzahl-der-visits-von-nachrichtenportalen/.

[7] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 1st. O'Reilly Media, Inc., 2009. ISBN: 0596516495, 9780596516499.

[8] David M. Blei. "Probabilistic topic models". In: *Communications of the ACM* 55 (Apr. 2012), pp. 77–84.

[9] David M. Blei and John D. Lafferty. "Correlated Topic Models". In: *Proceedings of the 18th International Conference on Neural Information Processing Systems*. NIPS'05. Vancouver, British Columbia, Canada: MIT Press, 2005, pp. 147–154. URL: http://dl.acm.org/citation.cfm?id=2976248.2976267.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[11] Jonathan Chang et al. “Reading Tea Leaves: How Humans Interpret Topic Models”. In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio et al. Curran Associates, Inc., 2009, pp. 288–296. URL: http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf.

[12] Kristina Chodorow. *MongoDB: The Definitive Guide*. Ed. by Ann Spencer. Second. O’Reilly, 2013.

[13] Jinho D. Choi, Joel R. Tetreault, and Amanda Stent. “It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool”. In: *ACL*. 2015.

[14] Despoina I. Christou. “Feature extraction using Latent Dirichlet Allocation and Neural Networks: Acase study on movie synopses”. MA thesis. University of Macedonia, 2015.

[15] Scott Crossley, Mihai Dascalu, and Danielle McNamara. *How Important Is Size? An Investigation of Corpus Size and Meaning in Both Latent Semantic Analysis and Latent Dirichlet Allocation*. 2017. URL: https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15441.

[16] David Kriesel. *SpiegelMining*. [Online; accessed 2-July-2018]. 2016. URL: http://www.dkriesel.com/_media/blog/2016/spiegelmining-33c3-davidkriesel.pdf.

[17] Scott Deerwester et al. “Indexing By Latent Semantic Analysis”. In: 41 (Sept. 1990), pp. 391–407.

[18] Thomas Hofmann. “Probabilistic Latent Semantic Indexing”. In: *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’99. Berkeley, California, USA: ACM, 1999, pp. 50–57. ISBN: 1-58113-096-1. DOI: 10.1145/312624.312649. URL: http://doi.acm.org/10.1145/312624.312649.

[19] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0-262-13360-1.

[20] Fiona Martin and Mark Johnson. "More Efficient Topic Modelling Through a Noun Only Approach". In: *ALTA*. 2015.

[21] David Mimno et al. "Optimizing Semantic Coherence in Topic Models". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Association for Computational Linguistics, 2011, pp. 262–272.

[22] Washington Post. *Mike Huckabee attacks CNN reporter on Twitter after he tangles with his daughter at the White House*. [Online; accessed 08-July-2018]. Aug. 2018. URL: https://www.washingtonpost.com/politics/mike-huckabee-attacks-cnn-reporter-on-twitter-after-he-tangles-with-his-daughter-at-the-white-house/2018/08/03/b0fcde7a-971d-11e8-a679-b09212fb69c2_story.html.

[23] *PYPL PopularitY of Programming Language Index*. Mar. 2018. URL: http://pypl.github.io/PYPL.html.

[24] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the Space of Topic Coherence Measures". In: (Feb. 2015), pp. 399–408.

[25] Alexandra Schofield et al. "Pre-Processing for Latent Dirichlet Allocation". In: 2017.

[26] C Sievert and K.E. Shirley. "LDAvis: A method for visualizing and interpreting topics". In: (Jan. 2014), pp. 63–70.

[27] Statista. *BAMF. n.d. Anzahl der Asylanträge (insgesamt) in Deutschland von 1995 bis 2018*. [Online; accessed 3-September-2018]. 2018. URL: https://de.statista.com/statistik/daten/studie/76095/umfrage/asylantraege-insgesamt-in-deutschland-seit-1995/.

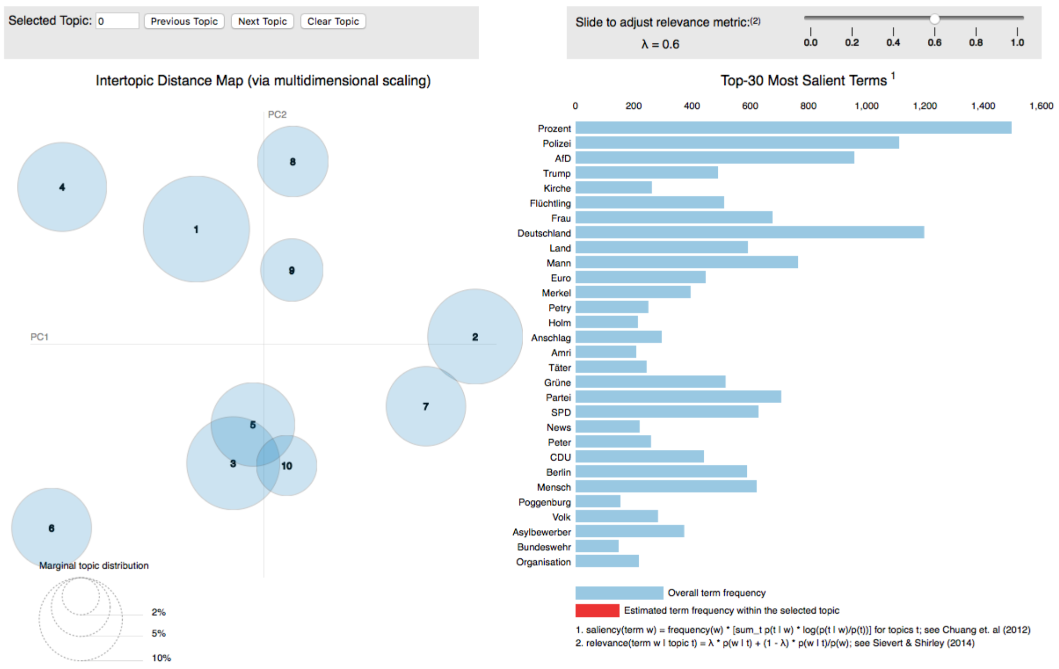
[28] *Which languages do developers prefer by age?* [Online; accessed 17-August-2018]. Nov. 2017. URL: `https://research.hackerrank.com/developer-skills/2018/`.

[29] Wikipedia contributors. *Cluster analysis — Wikipedia, The Free Encyclopedia.* [Online; accessed 2-September-2018]. 2018. URL: `https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=855220862`.

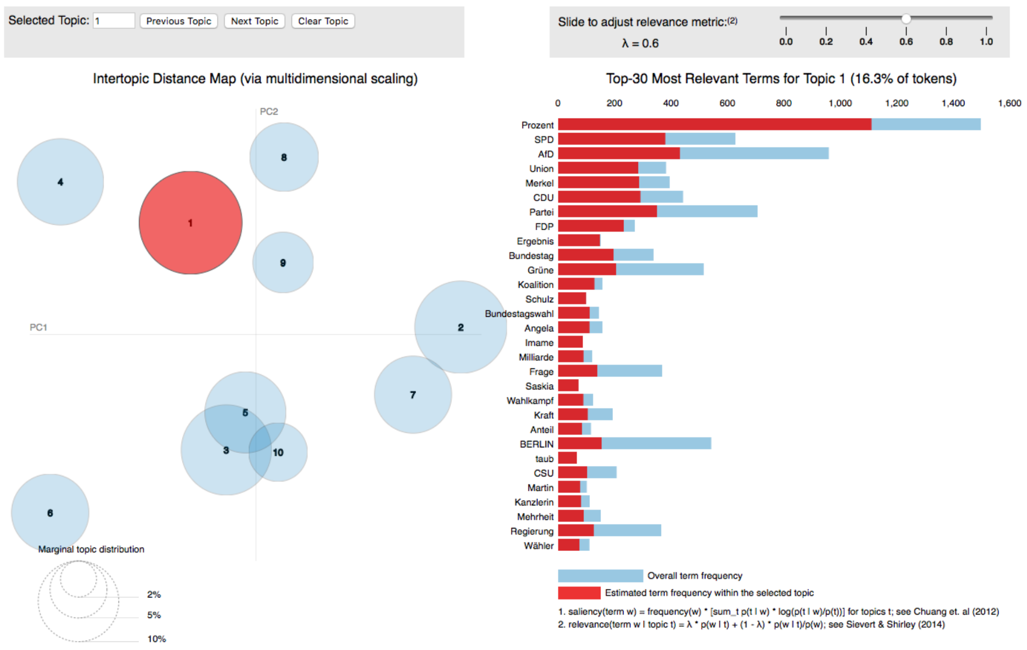
[30] Wikipedia contributors. *Tf–idf — Wikipedia, The Free Encyclopedia.* [Online; accessed 3-September-2018]. 2018. URL: `https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=856884778`.

Appendices

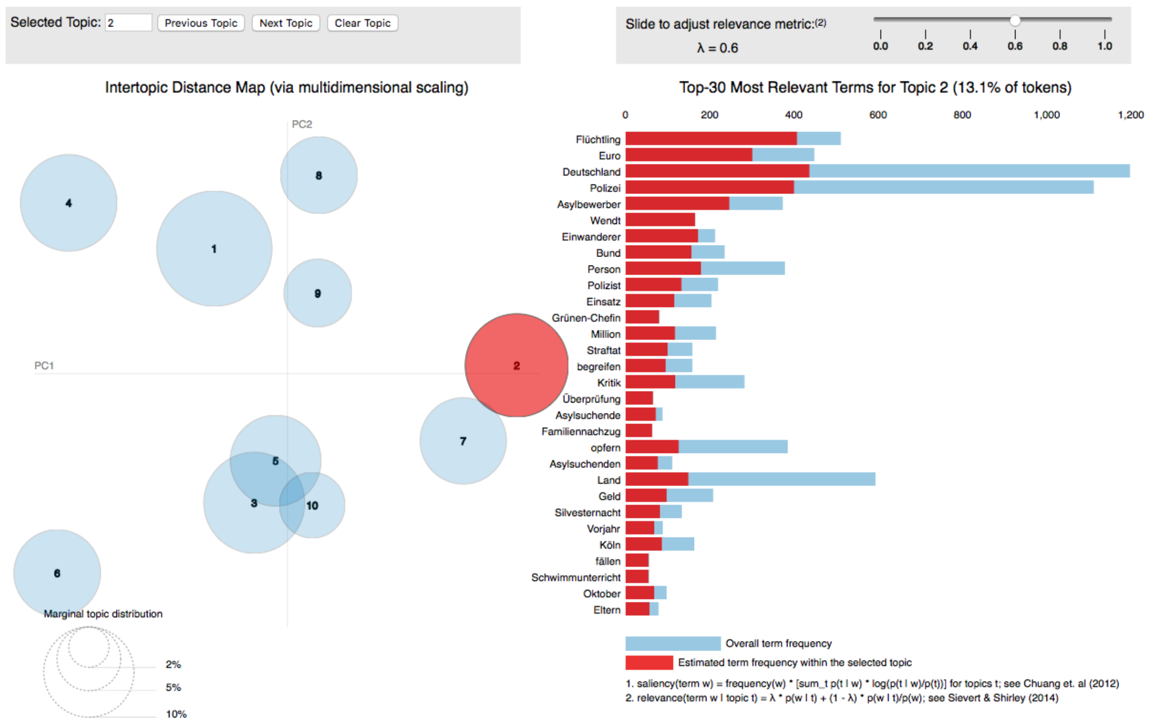
A. Junge Freiheit



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 2

Slide to adjust relevance metric:(2) λ = 0.6

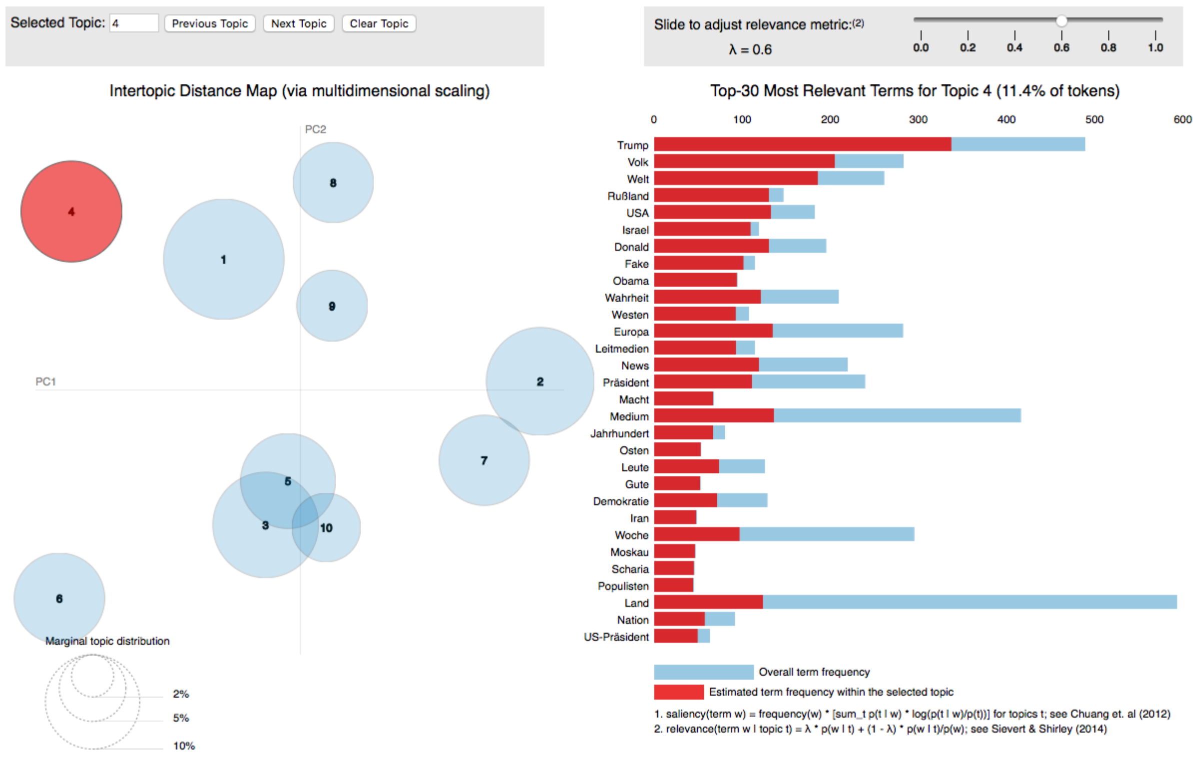
Intertopic Distance Map (via multidimensional scaling)

Top-30 Most Relevant Terms for Topic 2 (13.1% of tokens)

🟦 Overall term frequency

🟥 Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 4

Slide to adjust relevance metric:(2) λ = 0.6

Intertopic Distance Map (via multidimensional scaling)

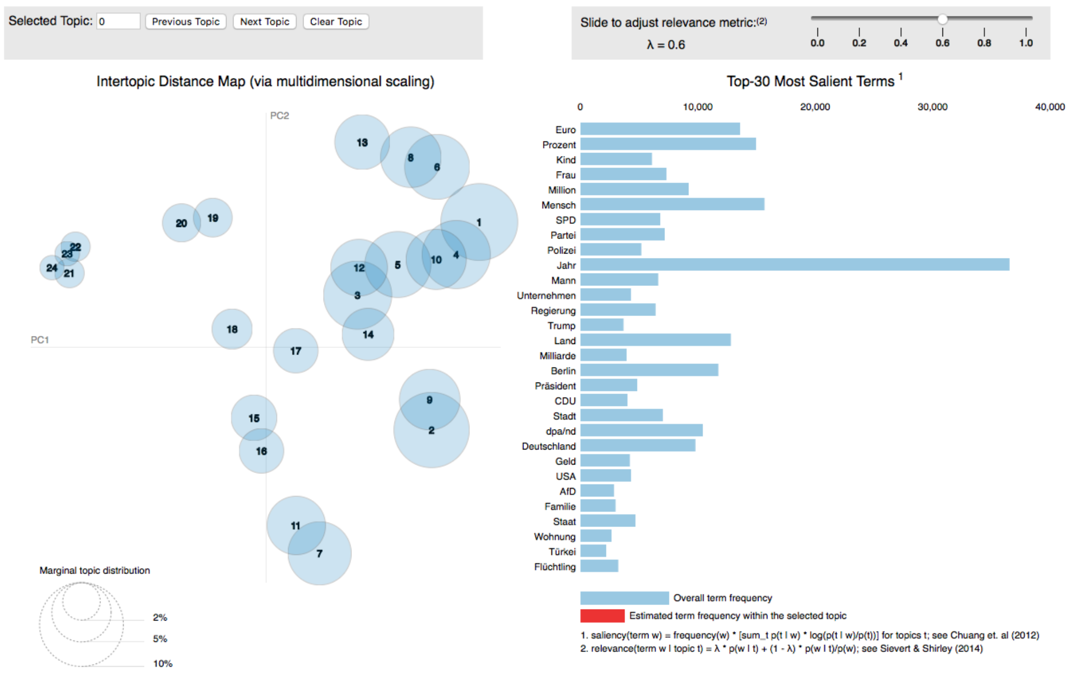
Top-30 Most Relevant Terms for Topic 4 (11.4% of tokens)

🟦 Overall term frequency

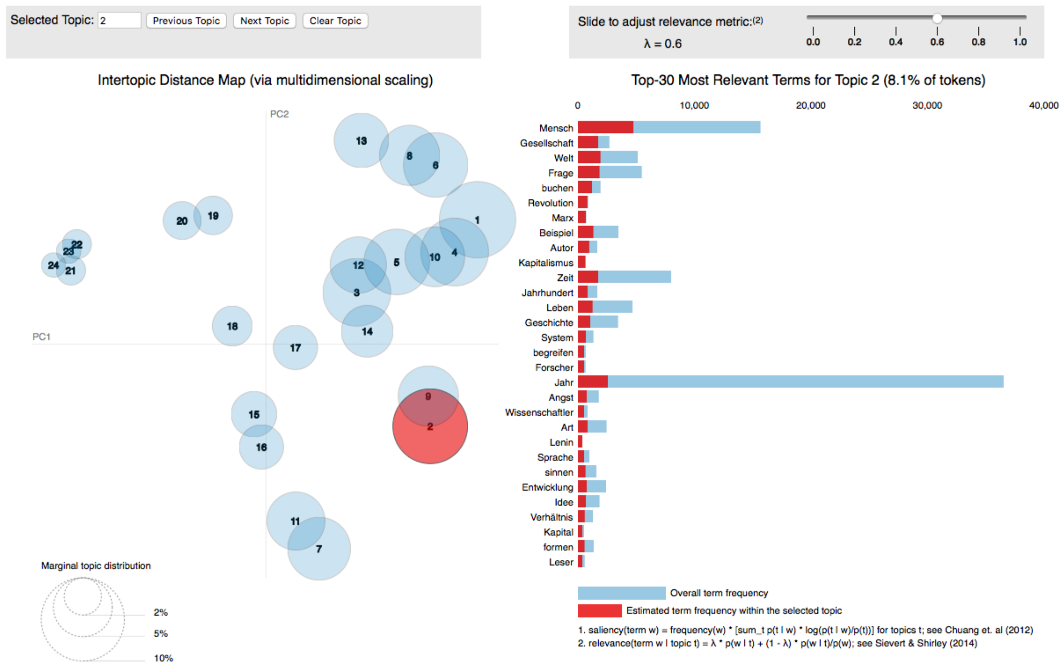
🟥 Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

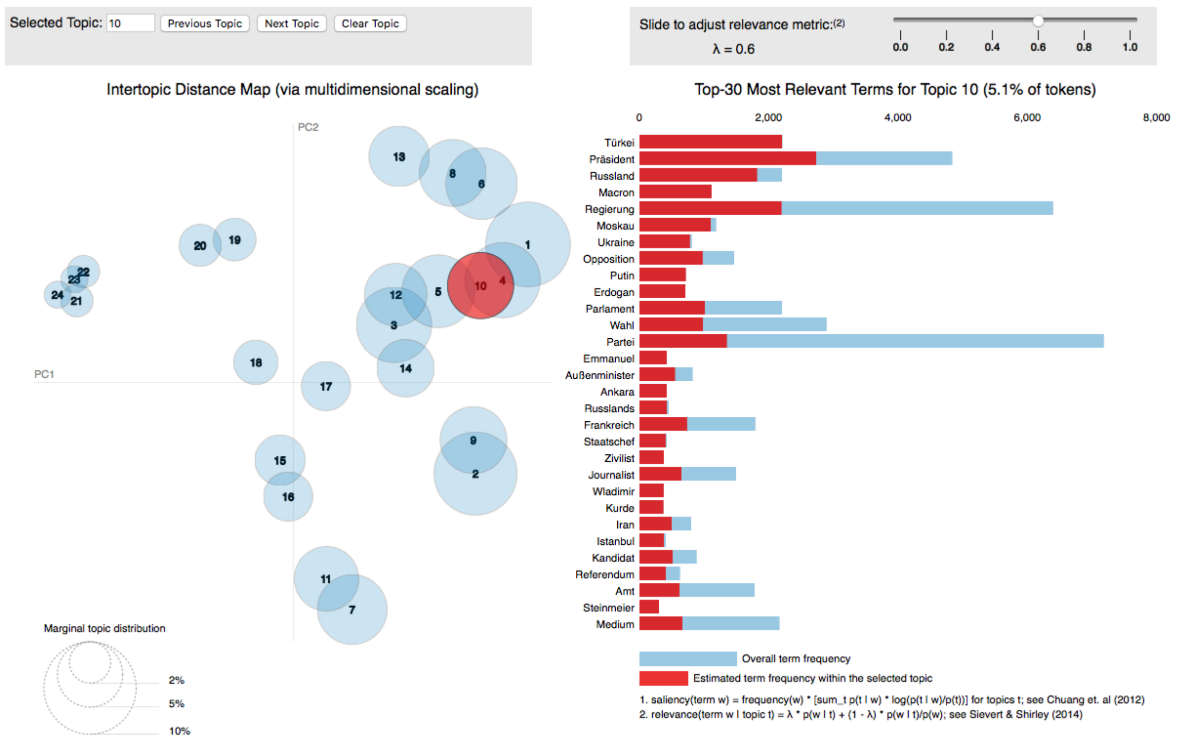
B. Neues Deutschland



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

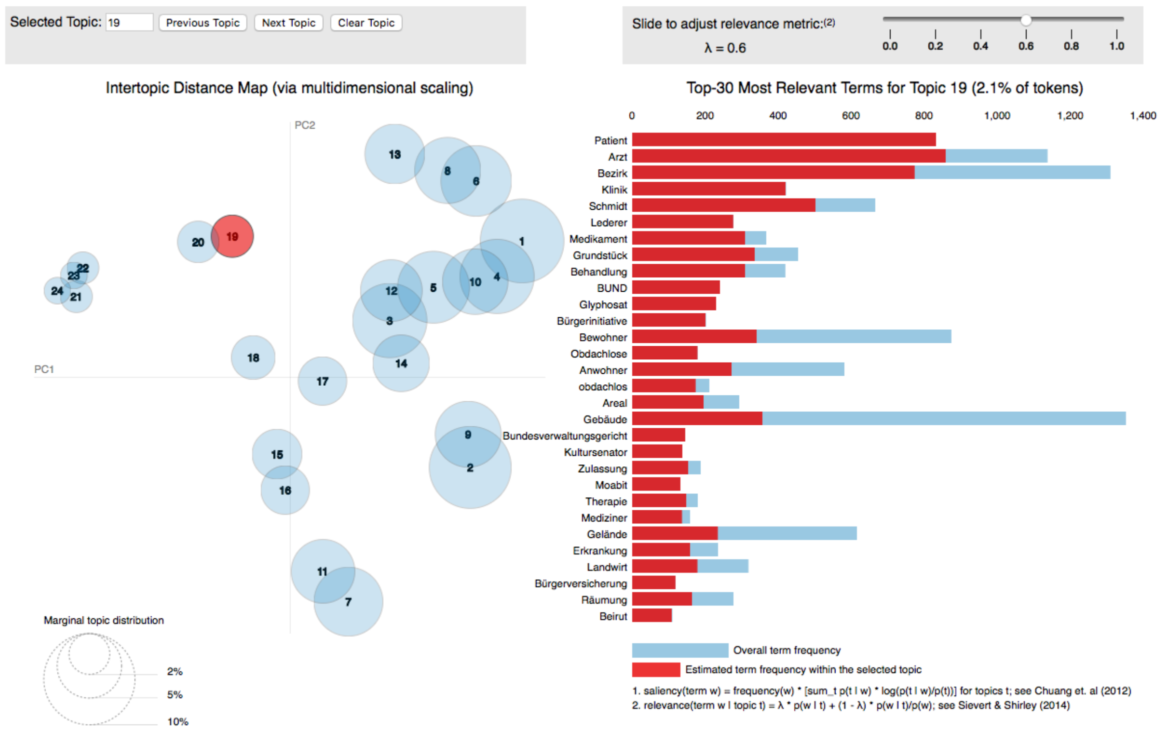


Selected Topic: 10

Slide to adjust relevance metric:(2)

λ = 0.6

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

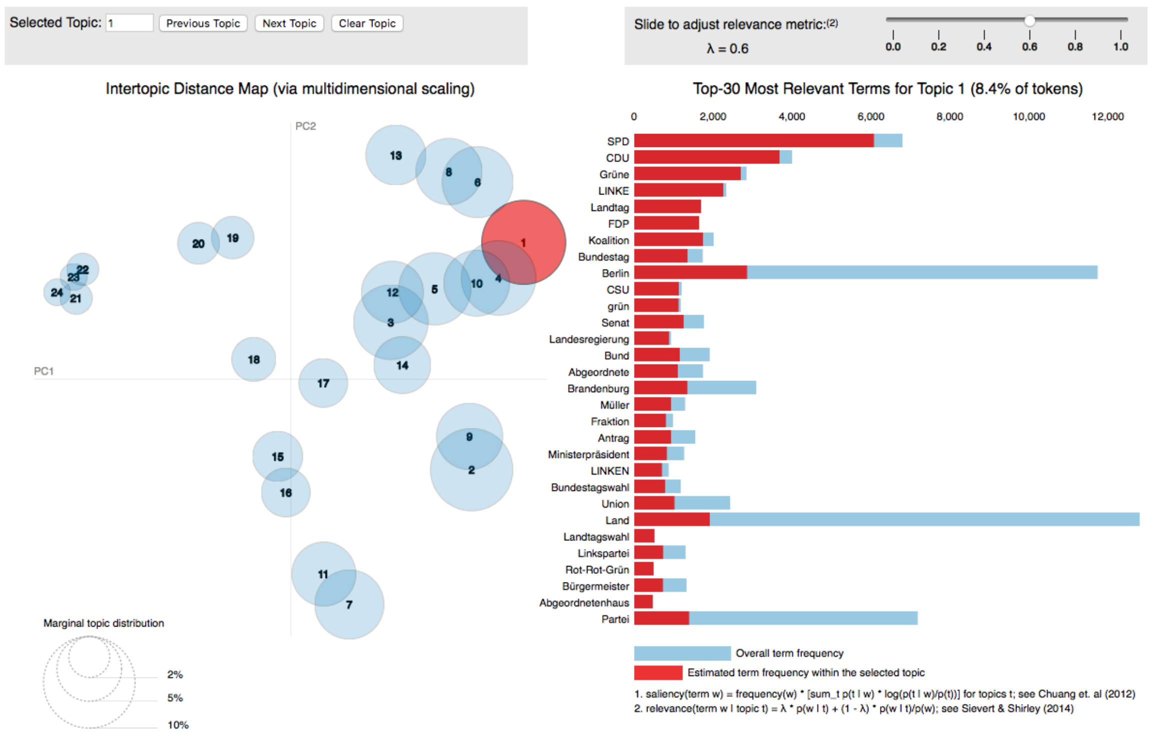


Selected Topic: 19

Slide to adjust relevance metric:(2)

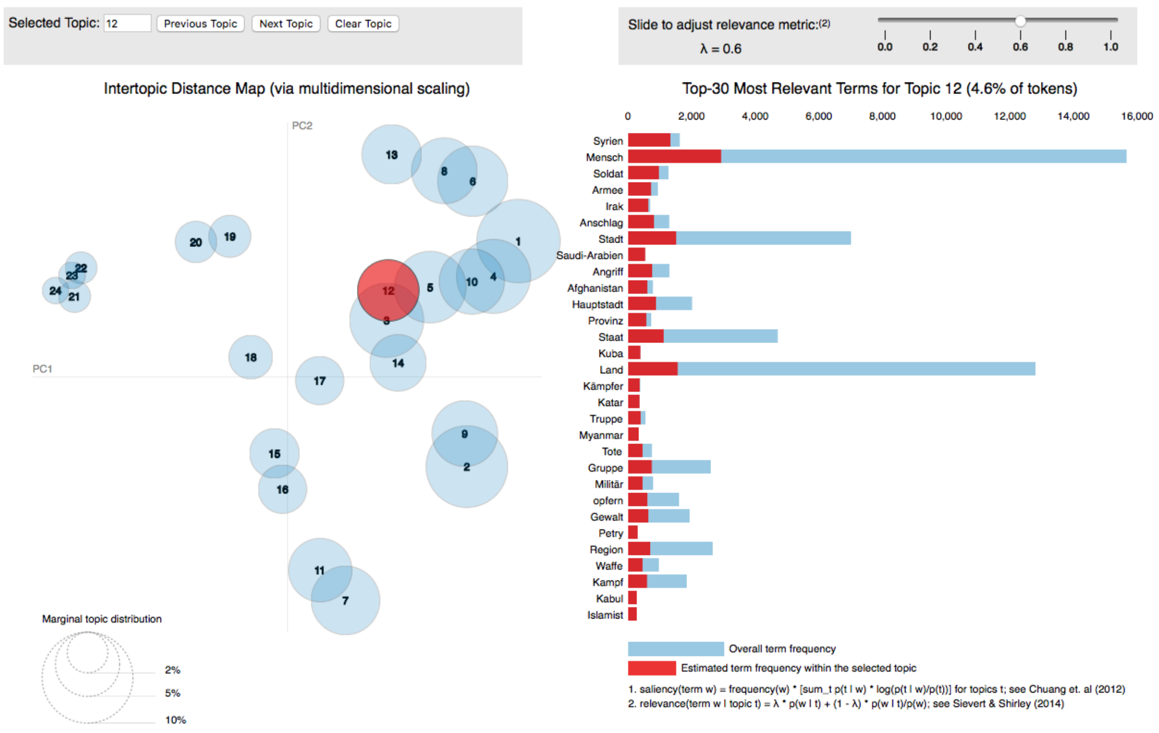
λ = 0.6

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 1

Slide to adjust relevance metric:(2) λ = 0.6

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 12

Slide to adjust relevance metric:(2) λ = 0.6

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass

- ich die vorliegende wissenschaftliche Arbeit selbständig und ohne unerlaubte Hilfe angefertigt habe,
- ich andere als die angegebenen Quellen und Hilfsmittel nicht benutzt habe,
- ich die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe,
- die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfbehörde vorgelegen hat.

Berlin, 04.09.2018: ______________________________

Unterschrift