Statistical Consulting

Topic-specific sentiment analysis for tweets by German MPs

Department of Statistics
Ludwig-Maximilians-Universität München

Munich, month dayth, 2021

Authors Asmik Nalmpatian
Lisa Wimmer

Project partner Prof. Dr. Paul Thurner
Department of Political Science

Supervisors Matthias Aßenmacher, Ph.D.
Prof. Dr. Christian Heumann
Department of Statistics

Abstract

Contents

List of Abbreviations

ABSA	aspect-based sentiment analysis
BERT	Bidirectional Encoder Representations from Transformers
BOW	bag of words
DL	deep learning
FN	false negative
FP	false positive
LDA	latent Dirichlet allocation
ML	machine learning
MLM	masked language modeling
MP	Member of Parliament
NLP	natural language processing
NSP	next sentence prediction
POS	part of speech
RNN	recurrent neural network
STM	structural topic model
TN	true negative
TP	true positive
TSSA	topic-specific sentiment analysis

List of Figures

List of Tables

1 Introduction

The advance of social media has sparked a fundamental change to public debate. Political coverage and discourse has spilled from traditional outlets over to online locations whose accessibility has lowered entrance barriers to welcome a wide audience (Bode, 2017). Social media exhibit certain properties that make them attractive to politicians seeking to broadcast their message. From a supply-side perspective it is easy to publish content: posting on social media is cheap, does not require approval of any authority, and allows for full control over the intended presentation. In an environment that spins information at enormous speed the resulting ability to react to events in real-time offers a distinct advantage over the inertia of traditional channels (Stier et al., 2018). On the receiving end, social media grant unprecedented access to target audiences. The industry's oligopolistic structure sees people from heterogeneous backgrounds convene on few global platforms to exchange their views. Research suggests that the low cost of online engagement causes political content to also circulate among users with less political affinity (Jost et al., 2018). A contrary but equally important aspect is the evolution of echo chambers. Users view content according to their perceived preferences and thus often end up in community niches populated by like-minded people. This clustering process creates groups disproportionately receptive to certain messages and has played an important role in large-scale propaganda. Echo chambers are particularly suited for the direct communication among users social media offer: they enable a dialogue between politicians and their electorate that is hard to achieve via traditional channels, and allow to deploy the power of emotion to shape opinion (Hasell and Weeks, 2016).
These opportunities have propelled internet platforms to a position at least level with the former hegemons of political debate. Twitter, in particular, has emerged as a medium for political information. In what might have been inconceivable a few years ago politicians actively convey messages to the public via tweets (van Vliet et al., 2020). The impact of this change in the political environment is complex and most certainly has positive as well as worrisome aspects. Yet, from a purely scientific point of view, activity on social media creates on its way a vast amount of publicly accessible data that benefits the research community with a constant source of information.

A question frequently posed in political analysis is the assessment of public opinion toward a particular matter. However, the textual data gathered from social media that might hold the answer command the use of specific tools subsumed under the field of *natural language processing (NLP).* NLP has gained much traction with the rise of deep learning methods and become virtually ubiquitous, techniques ranging from simple heuristics to gigantic neural networks powering search engines and the like (Torfi et al., 2020). Statistically speaking, the above problem translates into the classification of texts into instances of certain sentiments (typically, *positive* and *negative*). Building upon the assumption that sentiment may be expressed differently in varying contexts, such *sentiment analysis* is often combined with some form of *topic modeling* (see, for example, Ficamos and Liu (2016)).
It is the goal of this project to make analysis of social media texts in a political context more easily accessible to researchers. We focus on the analysis of public sentiment in a topic-aware manner for texts collected from Twitter posts by German Members of Parliament (MPs). Our contribution is two-fold:

1. We explore how topic-specific sentiment analysis can be implemented, considering (1) standard machine learning (ML) techniques and (2) more complex deep learning (DL) models.

2. We provide extensive teaching material on both approaches, composed as a coherent online course, to educate researchers on addressing NLP problems in their own work.

The remainder of this report is organized as follows. First, we outline the project in more detail in section 2. Section 3 provides some theoretical context for topic modeling and sentiment analysis from which we derive what we call *topic-specific sentiment analysis (TSSA)*. We proceed in section 4 by sketching our data collection and cleaning process and then present our proposal for conducting TSSA, laying out for both approaches the underlying methodology and discussing the results of applying them to the data at hand. Section 5 outlines how the findings from this analysis translate to the proposed teaching material. Afterwards, in section 6, we critically assess the findings and limitations of the project, and conclude with a brief summary in section 7.

2 Project outline

Before we outline the scope of our work it should be noted that parts of it are based on a predecessor project. Schulze and Wiegrebe (2020) studied how German MPs' Twitter data can be modeled with a *structural topic model* (*STM*, Roberts et al. (2013)), and have since engaged in follow-up research on the STM (Schulze et al., 2021). Much of the data procurement and topic modeling process is adopted from their work. The project at hand encompasses two subsequent steps. In a pioneering mode we first explore the overall feasibility of TSSA with both basic and more advanced statistical techniques from the NLP toolbox, and then, based on our findings, propose a collection of material to support fellow researchers in conducting similar studies.

Topic-specific sentiment analysis. Regardless of how the downstream task is solved, the first challenge to address is data collection. The idea here is to retrieve information from the web in an automated and resource-efficient manner that results in a suitable data structure. Afterwards, we pursue two fundamentally different approaches toward performing TSSA.

1. The first approach applies standard ML tools which require the input data to be of tabular form. Obviously, texts are complex constructs and not arbitrary sequences of interchangeable characters that can simply be cast into tabled variables (section 4.1.3 will address the challenges arising from Twitter data in more detail). It is nevertheless possible, as general research and also or own results suggest, to obtain fairly good performance with this reduction of complexity.

2. State-of-the-art approaches, by contrast, avoid such blunt simplification and attempt to teach the entire concept of language to machines. This comes at the expense of large computational requirements but achieves promising results in a variety of NLP tasks. We therefore build a deep bidirectional Transformer architecture (BERT, Devlin et al. (2019)) as a second approach and examine whether the additional complexity is justified by better performance.

The basic approach is implemented in `R` (R Core Team, 2021) and thus easily integrated with statistical education at LMU. For the BERT solution we resort to `Python` (van Rossum and Drake, 2011) which is all but standard for deep (NLP) modeling.

Knowledge transfer. Based on the process of this exploratory analysis we propose teaching material devised to support research in similar applications. The acquired collection is organized as a coherent and self-contained tutorial composed of basic theory, code demonstrations, and exercises (including solutions). While the course materials are primarily aligned to solve the TSSA task, the covered components are certainly also instructive for other types of applications. We have made the material available for both live teaching purposes and self-study on a public website. First experiences from a live workshop held in April/May 2021 will be discussed in section 6.

3 General theoretical context

3.1 Terminology

In this section we briefly review general theoretical concepts; the actual methods we employ are described in chapter 4. Throughout the report we will make use of the following terminology:

Word. Words w are sequences of characters. While they typically represent the smallest unit of text considered, BERT may decompose documents into even smaller parts.
Vocabulary. The aggregate of unique terms present in a collection of text constitutes a vocabulary of length $V \in \mathbb{N}$ from which a one-hot encoding for words can be derived: for the v-th instance of the vocabulary, $v \in \{1, 2, \dots, V\}$, this is a length-V vector with all but the v-th entry, which is one, equaling zero. Note that pre-processing (discussed in chapter 4) might result in a vocabulary that is smaller than the total number of distinct words occurring across all texts.
Document. Documents $d \in \{1, 2, \dots, D\}$, $D \in \mathbb{N}$, are generally understood to be sequences of $N_d \in \mathbb{N}$ words, and, in our case, tweets.
Corpus. Lastly, the set of all D documents considered makes up a corpus.

3.2 Theoretical concepts

3.2.1 Topic modeling

Idea. Recall that the ultimate goal is the classification of tweets into groups signaling a specific sentiment. It is reasonable to assume that sentiment, and the way of expressing it, is susceptible to context, which suggests potential gains from clustering tweets prior to sentiment analysis (see, for example, Ficamos and Liu (2016), Bhatia and Padmanabhan (2018), or Jang et al. (2021)).
Grouping texts into semantic clusters, or topics, is generally referred to as *topic modeling* and typically an unsupervised learning task. The idea is to uncover latent structures in a corpus along which documents can be characterized. Topic modeling is essentially a means of dimensionality reduction: text analysis requires text to be cast to numerical representation, the simplest form of which is to represent documents by counts of vocabulary instances. The dimension of the resulting document-term matrix increases exponentially in the number of documents and words contained in them, making an urgent case for compressing this dimensionality (Vayansky and Kumar, 2020).
Topic modeling results in two types of output: one that links words with their propensity of occurring within a topic $k \in \{1, 2, \dots, K\}$, $K \in \mathbb{N}$, and one stating the extent to which documents discuss each topic. This projection of texts into a K-dimensional latent space ($K \ll V$) is a purely mathematical operation and the assessment of interpretability is up to human judgment. Usually the resulting topics are then examined with respect to their most characteristic terms, according to an appropriate measure, in the attempt to find a meaningful description. In particular, K is a hyperparameter that must be specified a priori (Aggarwal, 2018).

Approaches. Topic modeling approaches roughly decompose into deterministic and probabilistic, or generative, approaches. The former are based on factorizing the document-term matrix $F \in \mathbb{R}^{D \times V}$ (or a weighted version that takes into account prior probabilities of term occurrence) into two low-rank matrices, $U \in \mathbb{R}^{D \times K}$ and $W^T \in \mathbb{R}^{K \times V}$, whose product approximates F loss-minimally. Probably the most prominent methods from this category are *latent semantic analysis*, which performs singular value decomposition and thus projects the data into a subspace spanned by F's principal eigenvectors, and *non-negative matrix factorization*, a constrained version that often yields better interpretability (Aggarwal, 2018).
Non-probabilistic models suffer from limitations in inference and out-of-sample extension, which is why generative approaches, addressing these issues, have become widely popular. Generative

models hail from the Bayesian paradigm. Loosely speaking, they seek to reverse-engineer the imaginative process of document generation: first, for each document d in a corpus we draw a length-K vector of topic proportions from some distribution; then assign each word position in $\{1, 2, \dots, N_d\}$ to a topic with probabilities according to the sampled topic proportions, and then draw a word from the distribution associated with this topic (Vayansky and Kumar, 2020). *Latent Dirichlet allocation (LDA)* by Blei et al. (2003), employing Dirichlet and multinomial distributions, pioneered this approach to topic modeling. In this, it also provides the foundation for the STM.

3.2.2 Sentiment analysis

Idea. Sentiment analysis refers to the process of establishing the emotive nature of the opinion an author expresses in their text. The types of sentiment that are of interest vary across applications and may be arbitrarily fine-grained. We focus on the simplest form and attempt to determine whether tweets convey positive or negative sentiment, also called *polarities*. In this, sentiment analysis is a standard classification problem (Medhat et al., 2014).
Sentiment analysis occurs at different levels of a document. Depending on its scope we can broadly discern document-level, sentence-level and aspect-level sentiment analysis. The document and sentence level techniques are not fundamentally distinct; rather, document-level analysis follows after sentence-level analysis (it is also possible to drill down even further) and aggregates the sentiments expressed by sentences for the entire document. Aspect-based sentiment analysis (ABSA) has a slightly different angle. It studies sentiment regarding aspects of topical entities (for instance, *carbon taxes* as an aspect of *environmental policy*), meaning sentiment targets are identified with respect to contents and not merely by sentence delimiters. This requires solving the sub-tasks of aspect extraction and aspect sentiment classification (Aggarwal, 2018). ABSA therefore has connections to topic modeling. We will discuss our view on the relation of topic modeling, sentiment analysis and ABSA in the subsequent section.

Approaches. There are two principal ways of approaching sentiment analysis. The first is rule-based and avoids statistical modeling altogether, instead classifying documents by summing the number of terms associated with each sentiment and assigning the class with the highest count. Prior polarities are typically taken from large dictionaries (Sidarenka, 2019). In our binary task this corresponds to identifying whether tweets contain more words with positive or negative connotation. We do incorporate this type of analysis but use the respective numbers of positive- and negative-polarity terms as an intermediate input rather than the sole grounds for classification.
Instead, we perceive sentiment analysis as a classic instance of supervised ML. Under the assumption that the data can be categorized into $g \in \mathbb{N}$ discrete classes we predict for each document its associated class from a set of features. More formally, we find a model $f : \mathcal{X} \rightarrow \mathbb{R}^g$, $\mathcal{X} \subseteq \mathbb{R}^p$ for $p \in \mathbb{N}$, that maps from the space of input features into g-dimensional Euclidean space. Each observation is assigned a vector of continuous class scores or probabilities (depending on the classifier). This mapping must be learned from a set of labeled training data, which marks a fundamental difference to the topic modeling task. The actual class labels $y \in \mathcal{Y}$ are then found by thresholding or an *argmax* operation on the score/probability vectors (Bishop, 2006). In our case the set of labels, with $g = 2$, is typically encoded as $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$.
Once the data are available in appropriate form, we can, in principle, use any type of learner suited to classification. We specifically consider random forests and regularized logistic regression for the standard ML solution and turn BERT into a classifier by fine-tuning it to sentiment analysis; details are given in section 4.

3.2.3 Topic-specific sentiment analysis

With the concepts of topic modeling and sentiment analysis we define a combined task we call *topic-specific sentiment analysis*. There are various ways in which such a combination is conceivable and we ourselves pursue different approaches, which is why we use TSSA as an overarching term. We see two types of solution to the TSSA problem: either perform topic extraction and sentiment analysis jointly, or regard it as a cascading task where sentiment analysis is subsequent to topic modeling. Most publications to date share the second view; recent work, however, has highlighted the advantages of joint modeling (e.g., Ngyuen and Shirai (2018), Tian et al. (2021), Wang et al. (2021)). While theoretical arguments are strong, we decide against simultaneous methods mostly due to their complexity which is at odds with our goal of making text analysis accessible to a broader audience. Also we would like to retain the option of performing both tasks in a stand-alone fashion. The latter has become all the more important in the course of our study as we experience some difficulties in modeling topics for tweets.
We make the general assumption of each document pertaining to exactly one topic. This seems plausible in the light of tweets' brevity (Twitter currently allows for 280 characters, up from 140 originally) and is supported by other research (Zuo et al. (2016), Qiang et al. (2019)). We then explore different options in our proposal. For the standard ML classification we use topic modeling, somewhat implicitly, as a means of feature generation for sentiment analysis. The clusters resulting from grouping tweets by topic serve as environments for computing topic-specific embeddings that are afterwards fed to the classifier. In the DL solution we experiment with sentiments that directly refer to aspects by fine-tuning BERT to an ABSA task. However, we observe in both approaches that the topical component does not aid the sentiment classification task and even tends to worsen results. We will discuss this finding in section 6.

4 Analytical proposal

4.1 Data

4.1.1 Data collection

The subject of our analysis are tweets by members of the German parliament (*Bundestag*) issued after the last federal election in September 2017[1]. Twitter makes these publicly accessible via its official API and the number of retrievable tweets per user can be exploited generously, so data supply is almost unrestricted. However, with sentiment classification as ultimate goal of analysis, we face a major bottleneck in the need for labeled data. Lacking the resources for large-scale annotation we did the labeling by hand. The resulting data set, from a vast amount of available data, consists of 1,215 observations, imposing some practical limits on the analytical scope.

Web scraping. For data collection from the Web we rely on the scraping procedure developed in the predecessor project, with minor modifications. The process entails four steps: first, gather MPs' names and basic information (such as party affiliation and electoral district) from the official Bundestag website; second, find Twitter account names (using individual party websites as additional sources); third, acquire socioeconomic information for the time of the last federal election on a per-district level (available at the official website of the federal returning officer); and, lastly, scrape actual tweets along with some additional variables like the number of retweets. We use a `Python` code base and mainly employ selenium webdrivers as well as the `BeatifulSoup` library (Richardson, 2007) for parsing HTML content and the `tweepy` library (Roesslein, 2020)

[1]The 2017 Bundestag is comprised of 709 seats and seven political parties: the right-wing AfD, the Christian Democrats (CDU/CSU), the Liberals (FDP), the Greens, the Left Party, and the Social Democrats (SPD). CDU/CSU and SPD as ruling parties co-exist in a grand coalition.

for accessing the official Twitter API. For more details on the procedure and the large data base assembled in the predecessor project please refer to Schulze and Wiegrebe (2020); the code is fully submitted in our electronic appendix and a somewhat more compact demonstration may be found among the teaching material.

Data labeling. In the data annotation phase we extracted a set with some tens of thousands of observations according to the above process and manually selected what we deem informative examples. For these we assigned polarities, i.e., predicates *positive* or *negative*, and also topic descriptions required for BERT's ABSA task. We noted in the process that a large number of tweets do not appear to carry sentiment at all. The resulting 1,215 training observations, originated by a total of 256 MPs, date from the period of October 2017 to January 2021. In figure 1 we detect both periodical fluctuations in the number of tweets over time and a general upward-sloping trend.

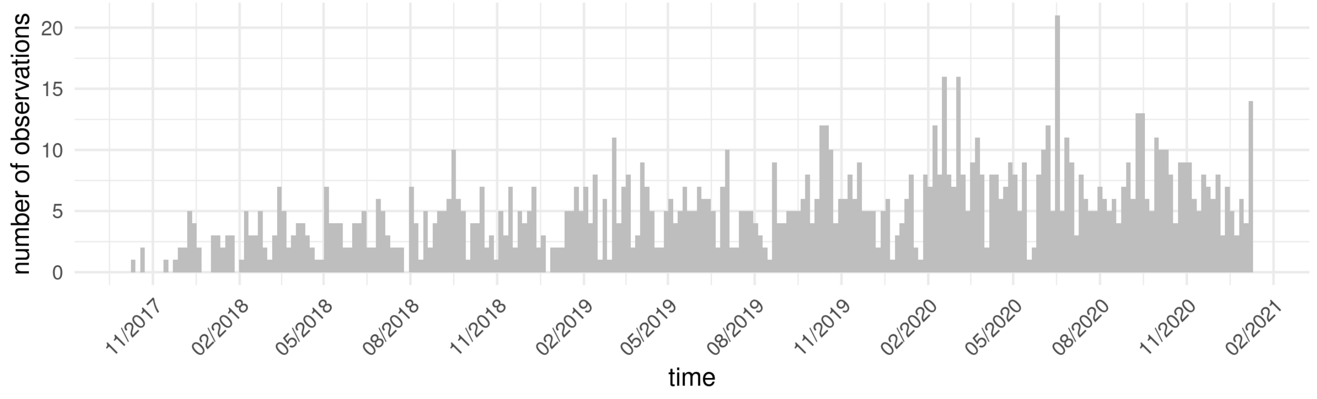

Figure 1: Observations over time.

An exemplary extract from our training data with some of the most important variables is shown in table 1. Exactly which features enter sentiment classification is documented in the subsequent chapters.

username	party	created_at	text	followers	unemployment_rate	label
karl_lauterbach	spd	2019-12-01 09:44:00	"Die Wahl ..."	337001	8.5	negative
Martin_Hess_AfD	afd	2018-08-17 07:15:00	"Vor den ..."	6574	3.5	negative
BriHasselmann	gruene	2019-09-25 15:35:00	"Ich finde ..."	20299	8.6	positive
danielakolbe	spd	2020-05-12 06:05:00	"Aber verpflichtend ..."	8158	8.3	negative
JuergenBraunAfD	afd	2020-08-13 22:05:00	"Panik-Latif + ..."	3188	3.4	negative

Table 1: Training data extract for selected variables.

Figure 2 depicts the number of observations per party both for our labeled training data and the larger sample from which the training data have been selected (containing just over 31,000 tweets). We notice two things. First, in either case, the share of tweets (blue) does not mirror the share of seats in the Bundestag (gray); most notably, the Christian Democrats tweet rather little, whereas the Greens are disproportionately active on Twitter. Second, the right-wing AfD and the Greens are over-represented in our training data at the expense of the other groups. This is simply because these two parties, in our personal experience from the annotation process, more often issue tweets that are strongly opinionated.

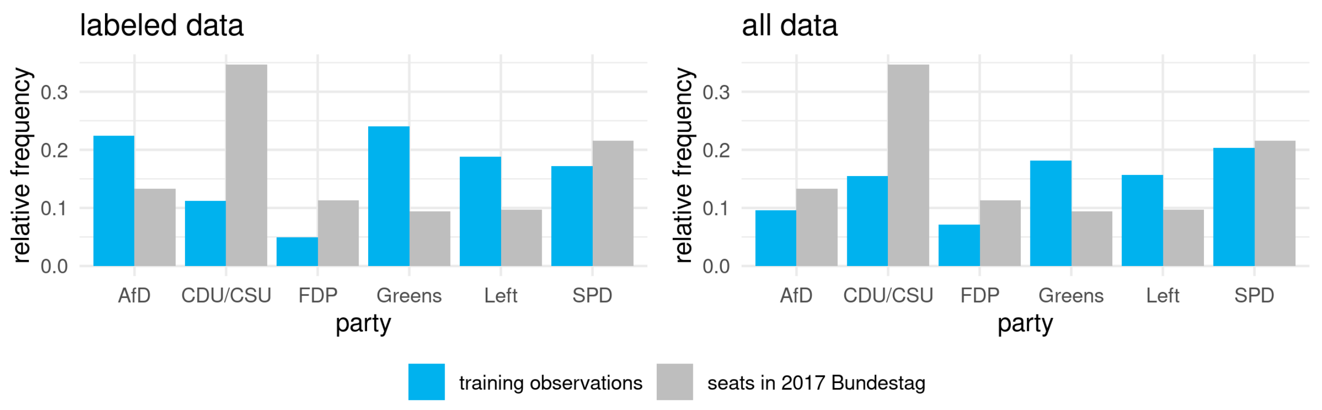

Figure 2: Observations per party in labeled training data (*left*) and entire scraped data example (*right*), both depicted against seat distribution in current parliament.

Lastly, when we inspect the class label distribution in the training data, an imbalance favoring the negative class becomes immediately visible: some 72% of tweets have been marked as negative. This reflects our general impression that most tweets which do carry sentiment express negative opinions and might be partly appropriated to the fact that the majority of authors belong to opposition parties.

4.1.2 Data pre-processing

The standard ML approach requires a lot more feature engineering than BERT does, but some general pre-processing steps are applied for both. In an initial step all tweets in non-German language are excluded from the data. We proceed with basic text cleaning, namely transcription of German umlauts and ligature s into standard-Latin characters and removal of non-informative symbols (such as those induced by ampersand conversion). The next block of operations is specific to Twitter data and includes the identification, separate storage and subsequent removal of special characters such as hashtags, emojis and user tags. By this we ensure the data are available for explicit analysis but do not introduce noise in the text. We finish the pre-processing procedure by assigning a unique identifier to each tweet.

4.1.3 Challenges

Text data come with many idiosyncrasies to begin with: language is highly diverse, irregular, and subject to constant change. Contextual dependencies and complex constructs such as colloquialisms or sarcasm pose serious obstacles in NLP, besides which profanities like spelling or translation mistakes must be handled (Mohammad, 2017). Some particular properties of the data at hand add to these challenges.

Language-specific. Not surprisingly, most work in NLP is concerned with the analysis of English documents. Although German is not a low-resource language and attracts its own share of research, many analyses and tools are predominantly tailored to English. German grammar is another aspect that needs to be considered. Syntax is heterogeneous, and inflections due to cases and genera result in many variations of lexical lemmata (Rauh, 2018).

Twitter-specific. Tweets' brevity is arguably the most critical issue for analysis. The limit of 280 characters means that words rarely appear more than once and we cannot expect many indicators of topics or sentiments in each document. It also prompts the use of abbreviations. On a similar note, we observe that Twitter posts often refer to certain events or topical entities without explicitly mentioning them, which is probably both due to the character limit and the real-time character of publications. The message may be clear for an informed human annotator then but will be hard to grasp for machines. Furthermore, tweets tend to be of rather informal style and use language that appears almost exclusively in social media, enlarging the vocabulary the classifier must understand.

Context-specific. The context of our data lessens the degree of informality somewhat; the issued documents are mostly political statements and as such more akin to written texts from other sources. Still, the political domain introduces new vocabulary yet again and makes the transfer of knowledge from other contexts harder. We also note that sarcasm and irony are frequently applied. Lastly, as mentioned before, we find many tweets to be solely informative and detect an imbalance toward negative sentiment in those that do convey opinion.

4.2 Standard machine learning solution

4.2.1 Concept

A number of components make up a typical workflow in supervised ML that adheres to the principle of *empirical risk minimization* (figure 3).

Starting from a *task* which consists of (initial) features and one or multiple target values, we train a *learner* on parts of the data. The learner encapsulates our beliefs about the feature-target relationship (*hypothesis*) and uses some *optimization* method to produce a model that incurs minimum training loss, or *risk*, according to some metric. Applying this usually parameterized model to the observations set aside for testing yields predictions we can compare to the ground truth for *performance evaluation*, again with respect to a given metric. With this, we estimate the true *generalization ability* of our model toward unseen data from the same data-generating process.

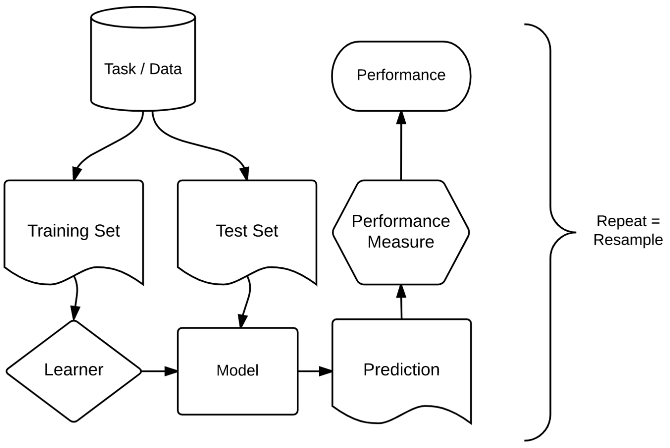

Figure 3: Typical ML workflow (Becker et al., 2021)

As performance might depend rather strongly on the given training data, especially if there are few observations to begin with, we will typically conduct multiple train-test splits (*resampling*). Various resampling strategies are available, such as bootstrapping or cross-validation. When the training process requires internal validation steps itself, e.g., for hyperparameter tuning, we need a *nested resampling* procedure to ensure that no information from training leaks into the test predictions (Japkowicz and Shah, 2011).

As figure 3 suggests, the workflow is of quite modular nature. We therefore build an automated ML (AutoML) pipeline for sentiment classification that allows to repeat training procedures and exchange components easily. Following a pre-training feature extraction phase (referred to as *static features*), the pipeline roughly serves the purpose of computing topic-specific embeddings (*dynamic features*), selecting an appropriate learning algorithm with included tuning step, and returning a trained model. Designing the process this way is beneficial for several reasons:

1. **Dichotomy between train and test sphere.** Topic-specific embeddings are computed globally across several observations. In order to actually enable predictions for unseen data, these dynamic calculations must be based solely on the training data. Since unbiased learning requires strict separation between train and test sphere, we must be careful to wall off predictions at test time from these computations exacted during training. The pipeline approach ensures this for each sequence of training and prediction.

2. **Automatization.** Automating (parts of) algorithm selection and hyperparameter tuning helps with crucial design decisions by attaching them to the given data.

3. **Transferability.** The possibility to plug in alternative components makes the approach instructive also for different applications.

In the following, we explain in more detail the two different stages of feature extraction, introduce the learners we propose to include, and show how the final pipeline is assembled.

4.2.2 Feature extraction

Static features

Static feature extraction forms the first block of the standard ML procedure and provides tabular data which then enter the AutoML pipeline. We refer to these features as static because they can be calculated prior to any training process, depending solely on single observations. They are based on the so-called *bag-of-words (BOW)* assumption that leads to texts being treated as arbitrary collections of vocabulary instances. In particular, information about grammar and word order is discarded in BOW approaches. This is obviously a strong simplification but hard to avoid entirely with standard classifiers (Cambria et al., 2017).
The static feature extraction steps rely heavily on `R`'s `quanteda` package (Benoit et al., 2021) for organizing documents in corpus objects, tokenizing texts and performing look-ups with dictionaries. We use lists of stopwords in multiple places to exclude uninformative tokens such as determiners or auxiliary verbs. These are compilations from various sources, including `quanteda`'s built-in list, open-source data available on GitHub and some manually appended, domain-specific terms. In addition, we repeatedly apply *stemming* in order to reduce words to their root form (for instance, cutting back both *"works"* and *"working"* to *"work"*).
All of these operations are aimed at compressing document representation from the total of unique original terms to more generalized tokens that co-occur across texts This way we create features that are actually shared by multiple observations and thus help classifiers infer feature-target relations with the ability to generalize. For the static part we exploit insights from other studies (e.g., Baly et al. (2017), Correa et al. (2017), Jabreel and Moreno (2017), Sidarenka (2019)) and include the following features:

1. **Lexicon-based polarity counts.** These comprise two sources of prior polarity, namely words and emojis. We use the aggregate of three large German polarity-term dictionaries, *GlobalPolarityClues* (Waltinger, 2010), *SentiWS* made available online by Leipzig university and a collection by Rauh (2018), for word polarities. The resulting lexicon distinguishes weak and strong sentiment. In each tweet, if a word is part of the dictionary and judged to have either positive or negative connotation (weak or strong), it raises the respective polarity count by one. Emojis are treated analogously, albeit without the discrimination between degrees of sentiment, using an emoji polarity list accumulated by Kralj Novak et al. (2015).

2. **Twitter variables.** We make use of the additional information provided by the Twitter API and note for each tweet the number of likes and retweets as well as the number of users tagged. For hashtags we resort to the same naive measure since we find them to be so heterogeneous across documents in our training set that no meaningful information can be extracted. This certainly marks an opportunity for future improvement.

3. **Syntactic features.** This group attempts to mitigate the simplification introduced by the BOW assumption to some extent. We track the presence of modifiers that indicate sentiment intensification as well as punctuation via the respective number of quotation and exclamation mark. Negation is a delicate problem we currently treat with the brute-force approach of noting whether or not negating tokens are present. German language complicates negation handling by the fact that modifier and modification target often appear in quite separate sentence positions, making it hard to identify the actually negated construct in an automated manner.

4. **Character unigrams.** Unigrams refer to single tokens (whereas n-grams mean the concatenation of n tokens) and we consider the special case of character tokens, meaning we count the occurrence of single letters. While this may not seem too helpful at first glance, character unigrams lead to a very dense document representation as opposed to the sparsity induced by word unigrams or even bi- and trigrams.

5. **Part-of-speech (POS) tags.** The last type of features captures grammatical structure. POS tags are used to categorize sentence parts according to their syntactic role. Including these stems from the idea that the number of certain tags, for example adjectives and adverbs, might indicate sentiment.

Dynamic features

The static features are not in any sense topic-specific. In order to allow for sentiments to vary across topics, we include topic-specific word embeddings. This entails a two-step procedure: first, we group documents with an STM, and then compute word embeddings for each topical cluster.

Topic modeling. The STM (Roberts et al., 2016) is a generative topic model and attempts to recreate the very process that generated the observed documents. As sketched in section 3.2.1, this is achieved by sampling topic proportions for each document from some distribution, assigning word positions to topics with the resulting probabilities, and drawing vocabulary instances from the distribution associated with the respective topic. As opposed to the original LDA approach assuming a single global prior distribution, the STM offers the possibility to incorporate document-level meta data. These may affect the sampling process in two places: first, by influencing the topic proportions for a given document (*topical prevalence*), and second, by altering the distribution the vocabulary assumes over each topic (*topical content*). Our case lends itself more to using topical prevalence variables, presuming that certain meta features on electoral district level, which can be specified in terms of an additive formula, have an impact on which topics are discussed. Defining a topical content variable (a single, categorical one is supported in the STM implementation (Roberts et al., 2020)) allows for topic-word distributions to vary for its different levels. Such an approach is also conceivable but not further pursued here. The STM can be expressed by the following generative process[2] for each document $d \in \{1, 2, \dots, D\}$ (Roberts et al., 2016):

[2]We adopt the notation from Schulze and Wiegrebe (2020) for consistency.

1. Draw non-normalized topic proportions $\boldsymbol{\eta_d}$ from a $(K-1)$-dimensional Gaussian: $\boldsymbol{\eta_d} \sim \mathcal{N}_{K-1}(\boldsymbol{\Gamma}^T \boldsymbol{x}_d^T, \boldsymbol{\Sigma})$, where the K-th component is set to zero to keep the model identifiable. $\boldsymbol{x}_d$ is the vector of $P \in \mathbb{N}$ prevalence covariates associated with document d, and $\boldsymbol{\Gamma} \in \mathbb{R}^{P \times K}$ is a matrix of topic proportion coefficients obtained by means of Bayesian linear regression with Gaussian priors and first-order penalty term. The covariance matrix $\boldsymbol{\Sigma}$ can have arbitrary structure and thus incorporate inter-topic correlations.
2. Normalize $\boldsymbol{\eta_d}$ through a softmax operation, yielding $\boldsymbol{\theta}_d$ with $\theta_{d,k} = \frac{\exp(\eta_{d,k})}{\sum_{j=1}^{K} \exp(\eta_{d,j})}$, $k \in \{1, 2, \dots, K\}$. The $\boldsymbol{\theta}_d$ then effectively follow a logistic normal distribution.
3. For each word position $n \in \{1, 2, \dots, N_d\}$:
   (a) Draw $\boldsymbol{z}_{d,n} \sim \mathit{Multinomial}(\boldsymbol{\theta}_d)$ to assign the n-th position to a topic.
   (b) Draw a word $w_{d,n}$ from the word distribution corresponding to the assigned topic: $w_{d,n} \sim \mathit{Multinomial}(\boldsymbol{\beta}(d, n))$.

For our analysis we include a total of four prevalence covariates, namely the MP's party affiliation and federal district (*Bundesland*) as well as the share of migrant population and overall employment rate. The latter two are modeled as smooth effects with five degrees of freedom each. While we find this formula to work well with our particular training data, the specification can be easily modified.

Word embeddings. We now use the information from topic assignment to compute topic-specific word embeddings. Recall that the BOW assumption results in very high-dimensional document representations disregarding inter-term relationships. Word embeddings, by contrast, seek to capture semantic structures in these representations: words are characterized by their surrounding context and projected into a latent space in which similar words ought to end up in nearby locations. At the same time, embeddings perform dimensionality reduction as the latent space is typically chosen to be of much lower dimension than the observation space.
We use the *Global Vectors (GloVe)* model proposed by Pennington et al. (2014) and implemented in R by Selivanov et al. (2020). Its computation is based on the symmetric *co-occurrence matrix* $C \in \mathbb{R}^{M \times M}$, where $M \in \mathbb{N}$ denotes the number of tokens remaining after such pre-processing steps as stopwords removal. Entry $c_{i,j}$ states how often token j occurs in the context of token i. Context is a local notion controlled via *window size* $\ell \in \mathbb{N}$, where ℓ specifies the number of positions to each side of a token considered to be in its vicinity, the underlying (simplifying) assumption being that the strength of semantic relation decays with increasing distance. The entries of the co-occurrence matrix are weighted by the inverse distance to the target token. We can therefore express probabilities of co-occurrence as fractions of row (or column) sums: the probability of the i-th token occurring in the context of the j-th token is given by $p_{ij} = \frac{c_{ij}}{c_{i.}}$ with $c_{i.}$ as the i-th row sum. The co-occurrence probabilities in turn give rise to odds $\frac{p_{ik}}{p_{jk}}$, for which Pennington et al. (2014) posit that they should be reflected by embedding vector distances. This is formalized as

$$\exp((q_i - q_j)^T \tilde{q}_k) = \exp(q_i^T \tilde{q}_k - q_j^T \tilde{q}_k) = \frac{p_{ik}}{p_{jk}},$$

with $q_i \in \mathbb{R}^m$, $m \in \mathbb{N}$, as the m-dimensional embedding vector representation for word i. $\tilde{q}_k$ is structurally equivalent to q_k for symmetric C and discrepancies arise solely from different random initialization of the model. The authors do not provide a sound theoretical foundation for this double computation but point out that such stochastic variation has been shown to benefit over-parameterized models. In the end GloVe actually uses the sum of the vectors as embeddings. We arrive at $q_i^T \tilde{q}_j + b_i + \tilde{b}_j \approx \log(c_{ij})$, with bias b_i approximately equal to $\log(c_i)$, and can now express the embedding error minimization problem by the following weighted least-squares objective:

$$\min_{\boldsymbol{v},\boldsymbol{b}} \sum_{i=1}^{M} \sum_{j=1}^{M} g(c_{ij}) \left(q_i^T \tilde{q}_j + b_i + \tilde{b}_j - \log(c_{ij}) \right)^2 .$$

$g : \mathbb{R} \rightarrow \mathbb{R}$ is a weighting function for which the authors propose to use

$$g(x) = \begin{cases} \left(\frac{c_{ij}}{c_{max}} \right)^{\delta} & c_{ij} < c_{max}, \\ 1 & \text{otherwise.} \end{cases}$$

$c_{max} > 0$ and $\delta > 0$ are hyperparameters to the GloVe algorithm, as are the embedding dimensionality m and the size ℓ of the context window (Pennington et al., 2014).

GloVe returns an embedding vector for each word. Since we are interested in a representation on document level, the word embeddings are aggregated in a final step. This leaves us with a block-like structure of embedding vectors: for K topics and m embedding dimensions we obtain $K \cdot m$ feature vectors, where each document has non-zero entries only for the columns corresponding to its associated topic. At the end of the entire feature-generating process we can thus describe our documents by a number of static covariates plus topic-specific embedding values. This tabular representation of our Twitter corpus eventually forms our classification task.

4.2.3 Sentiment classification

We propose two different classification algorithms, a *random forest* and a logistic regression learner with elastic net penalty (*glmnet*). Both ship with built-in feature selection, which we consider rather helpful in the potentially high-dimensional feature space of text classification, and are fairly easy to tune (Japkowicz and Shah (2011), Probst et al. (2019)). However, the pipeline design would allow to plug in any other suitable learner instead.

Random forests. A random forest is an ensemble of tree base learners (Louppe, 2014). Starting from a root node containing all observations, classification trees perform subsequent binary splits of the feature space, repeatedly dividing it into rectangular partitions along one dimension at a time. Observations are passed along the resulting decision tree and end up in a particular end node, or leaf, where they are assigned the most frequent class label in this node. In the end, this constitutes hypotheses of the form $f(\boldsymbol{x}) = \sum_{t=1}^{T} r_t \mathbb{I}[\boldsymbol{x} \in R_t]$, where T denotes the number of terminal regions $R_1, R_2, \ldots, R_T$ and $r_t \in \{0, 1\}$ is the class label predicted for observations in leaf node Q_t. Each split is constructed such that the empirical risk across all child nodes is minimized, which typically corresponds to minimizing class impurity. To this end, trees perform a greedy search that requires assessing all possible splitting variables and associated thresholds (Breiman et al., 1984).
Without regularization, i.e., constraining tree depth, classification trees will perfectly fit the patterns they observe and may thus change strongly with small alterations of the training data. Random forests exploit the low bias of trees and assemble $B \in \mathbb{N}$ of them to reduce variability. Predictions are obtained by aggregating predicted class labels of all base learners (typically, a majority vote is used). Randomness enters in two ways: first, the subset of data used to fit a single tree within the ensemble is chosen at random via bootstrapping, and second, each base learner is allowed to use only a random subset of features as splitting criteria. This procedure enables decorrelation and enhances ensemble stability (Louppe, 2014).

Tunable hyperparameters of random forests include ensemble size, the number of candidate features to be considered at each split, and various attributes associated with single trees' complexity (Probst et al., 2019). Random forests easily handle all types of features, including arbitrary interactions, and perform natural variable selection. They mitigate much of single trees' instability and still achieve good performance in many applications (Louppe, 2014).

Regularized logistic regression. Logistic regression is a special case of generalized linear regression where the predictor is linear in the parameters but transformed by a possibly non-linear link function (Lindsey, 1997). Consider the feature matrix $\boldsymbol{X} = [\boldsymbol{x}_d^T]_{d=1,2,\dots,D} \in \mathcal{X} \subset \mathbb{R}^{D \times p}$ and let $\boldsymbol{y} \in \{0,1\}^D$ denote the binary sentiment target. Logistic regression models the probability of $y_d = 1$ for a given observation $\boldsymbol{x}_d$ as $\pi(\boldsymbol{x}_d) = \mathbb{P}(y_d = 1 \mid \boldsymbol{x}_d) = \frac{\exp(\gamma_0 + \boldsymbol{\gamma}^T \boldsymbol{x}_d)}{1+\exp(\gamma_0 + \boldsymbol{\gamma}^T \boldsymbol{x}_d)}$, with regression coefficients $(\gamma_0, \boldsymbol{\gamma}) \in \mathbb{R}^{p+1}$ (Lindsey, 1997). Prediction of class labels requires thresholding the posterior probabilities, usually at a cut-off value of 50%. By application of the logit transformation we obtain the linear predictor $\gamma_0 + \boldsymbol{\gamma}^T \boldsymbol{x}_d = \log \frac{\pi(\boldsymbol{x}_d)}{1-\pi(\boldsymbol{x}_d)}$. Here, $(\gamma_0, \boldsymbol{\gamma})$ is estimated by minimizing a penalized version of the negative log-likelihood. The additive elastic net penalty balances absolute and quadratic regularization in a hybrid of LASSO and Ridge regression. Regularization causes shrinkage in the coefficients (typically not affecting the intercept) and leads to the following optimization problem, where $\lambda \geq 0$ and $\alpha \in [0,1]$ are hyperparameters (Hastie et al., 2021):

$$\min_{(\gamma_0, \boldsymbol{\gamma}) \in \mathbb{R}^{p+1}} -\frac{1}{n} \sum_{d=1}^{D} y_d(\gamma_0 + \boldsymbol{\gamma}^T \boldsymbol{x}_d) - \log(1 + \exp(\gamma_0 + \boldsymbol{\gamma}^T \boldsymbol{x}_d)) + \lambda \left((1-\alpha) \|\boldsymbol{\gamma}\|_2^2 \cdot \tfrac{1}{2} + \alpha \|\boldsymbol{\gamma}\|_1 \right).$$

4.2.4 Automated machine learning pipeline

We assemble all components presented above in an AutoML pipeline. The pipeline takes care of

1. computing the dynamic features, i.e., topic-specific embeddings,
2. selecting and training the best learner, including hyperparameter optimization, and
3. predicting sentiment labels for new observations.

We implement the algorithm as a graph learner in the `mrl3` ecosystem (Lang et al., 2019). Graph learners are composed of nodes that perform operations on the data. Multiple options for branching and re-uniting enable arbitrarily complex constructions. The design allows to use the entire pipeline just like any other learner object in standard ML procedures such as training, prediction, tuning or benchmarking. `mrl3`'s internal structure ensures the dichotomy between train and test sphere in every step and makes our implementation compatible with other routines.

Pipeline components. The proposed pipeline takes a static feature representation of our annotated training data and starts by assigning topic labels, afterwards computing topic-specific embeddings. The next block conducts the entire tuning process, where the selection of a learner is directly included, such that the optimal classifier is found in joint optimization of algorithm and associated hyperparameters. We rely on existing functionalities for much of the required steps. However, we contribute custom implementations for the STM and the subsequent embedding computation with GloVe. Embeddings can also be calculated without a preceding topic modeling step. Furthermore, an alternative topic modeling approach is available. It is a non-stochastic procedure that accepts manually defined keywords and finds all documents associated with the terms or derivatives thereof. If necessary, this can be combined with a stratification step that approximately retains the ratio of documents associated with a keyword in each resampling step. All these novel operations are constructed as so-called *pipe operators* in `mlr3`. Eventually, the pipeline assumes the structure depicted in figure 4.

Settings. Our design choices are partly inspired by Probst et al. (2019) and otherwise reflect what we find to work best. The decision to compute topic-specific embeddings and the selection of learners are such choices already. We further fix some hyperparameters in advance to save on computation time. This includes capping the maximum number of iterations in regularized regression at 1,000 and limiting tree growth in the random forest (leaf nodes must contain at least 5% of observations) as well as allowing for parallel computations. Other hyperparameters are left to tuning. The first two are the number of topics with $3 \leq K \leq 6$ and the dimensionality of embeddings $10 \leq m \leq 50$, where we exploit the fact that these inherently unsupervised problems are input to a supervised learning task: we can simply have the algorithm choose the configuration with the best predictive performance.

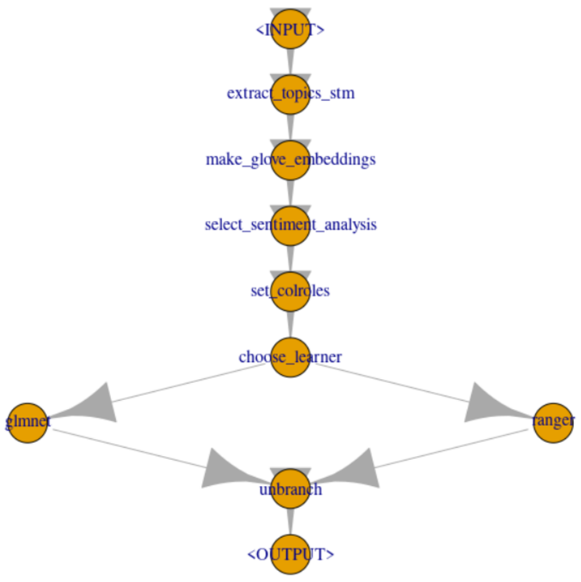

Figure 4: Schematic representation of proposed AutoML pipeline

For the actual classification part we allow for optimization of the penalty coefficients in logistic regression ($0 \leq \alpha \leq 1$ and $0 \leq \lambda \leq 0.2$) and the number of candidate features for split computation in the random forest (limited to 50% of features at most), as well as the fraction of observations to be sampled for each base learner (between 0.25 and 1). Note that the search space remains rather small; more tuning parameters, larger ranges and/or more configurations could be explored with enough computational resources.
We use three-fold cross validation (3-CV) as resampling strategy in both the outer (performance evaluation) and inner (hyperparameter tuning) training loop. 3-CV divides the training data into three roughly equal-sized partitions, each of which serves as test set once while the other two provide the training data (Japkowicz and Shah, 2011). Tuning is conducted via random search with a total budget of 50 evaluations (again, this could be extended if affordable). Random search is based on the simple idea of drawing from the space of possible configurations uniformly at random. While implementation is easy and readily parallelizable, the number of evaluations required to sample the space adequately rises exponentially with the number of dimensions, with no possibility to learn from prior draws (Feurer and Hutter, 2019). More efficient strategies might improve the tuning process here. Lastly, in the inner tuning loop, we evaluate performance by means of accuracy (i.e., the share of correctly classified instances, see section 4.2.5), as we identify no need to emphasize one side of misclassification or the other in any particular way.

4.2.5 Results

We evaluate the performance of two variants of our pipeline, a full version including topic-specific embeddings and a topic-agnostic alternative that omits the STM node. Performance metrics include, besides the elements of the confusion matrix, the following quantities (e.g., Japkowicz and Shah (2011)):

1. Accuracy: $\rho_{\text{acc}} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}}$, and

2. F1 score: $\rho_{\text{F1}} = 2 \cdot \dfrac{\frac{\text{TP}}{\text{TP} + \text{FP}} \cdot \frac{\text{TP}}{\text{TP} + \text{FN}}}{\frac{\text{TP}}{\text{TP} + \text{FP}} + \frac{\text{TP}}{\text{TP} + \text{FN}}}$,

where TN = true negatives, TP = true positives, FN = false negatives and FP = false positives.

We see from table 2 that both algorithms achieve rather low cross-validated performance in the realm of 72% accuracy. The meaning of this observation becomes clear when considering the fact that these values are remarkably close to the share of negative labels in the data: essentially, neither of the learners is able to outperform a random classifier assigning classes according to the empirical relative class frequencies. We have included such a featureless learner in our benchmark experiments and the results confirm our findings:

	learner with topic modeling	**learner without topic modeling**	**featureless learner**
accuracy	0.723	**0.729**	0.724
F1 score	0.839	**0.840**	0.840
TN	**293.000**	288.000	**293.000**
TP	0.000	**7.333**	0.000
FN	111.667	**104.333**	111.667
FP	0.333	5.333	**0.000**

Table 2: Results for standard ML approach (non-integer values in TN, TP, FN and FP are due to aggregation over cross validation folds). Best performance achievements are marked bold; overall best learner is marked gray.

It is, moreover, worth noting that the learner without topic clustering performs slightly better (if only by a very small margin) than the one using topic-specific embeddings. We will see how the BERT solution exhibits similar behavior, leading to the conclusion that topic modeling for this kind of data seems to complicate the task rather than aiding it. All in all, it appears that the handcrafted features by which we represent our texts are not helpful in sentiment classification for these data.

4.3 Deep learning solution

4.3.1 Deep transfer learning with BERT

As an alternative to casting TSSA as a standard ML classification task we now consider a solution rooted in deep learning. This second part of our work is based on the Bidirectional Encoder Representations from Transformers (BERT) network (Devlin et al., 2018) and its variations. The name draws from the core architecture as a multi-layer bidirectional Transformer encoder. BERT models are extremely large, their parameters numbering in the millions, and considered state of the art for many NLP tasks today.
A thorough introduction to deep learning is beyond the scope of this report and the BERT models' architectures are particularly complex at that. We will therefore only point out two distinctive properties of Transformer models and refer to, e.g., Goodfellow et al. (2016) for further reading on deep learning in general and to Vaswani et al. (2017) for the detailed Transformer workflow. Transformer-based models, which have gained much traction in recent research, rely on two key principles, namely *transfer learning* and *attention*.

Transfer learning. In the standard supervised learning approach, where we ideally have a large number of labeled training instances, we assume the test data to follow the same underlying distribution as the training data, a belief we exploit in evaluating the generalization ability of our learner. When this can no longer be plausibly claimed and domain shift occurs, for example in applying our model to documents with different topical context, we cannot expect our model to generalize as the test performance suggests. Still, the tasks might not be so fundamentally different, and some of the gathered information on the feature-target relationship will be of use nonetheless. Much like in human learning processes we can presume a certain amount of abstraction. It is precisely this knowledge migration that is at the heart of transfer learning (Murphy, 2021).

As discussed before, the abundance of data is in stark contrast to the scarcity of labels in many applications, such as in ours. The large (manual) annotation effort this entails poses severe constraints on the practicability of supervised learning. Therefore, it is reasonable to accumulate as much existing knowledge as possible to aid the given task. Transfer learning represents the approach of statistical learning in which we first train a model on an original task and domain and then transfer the acquired knowledge to the target task and domain (Ruder, 2019). This is mainly achieved in the pre-training task (see below) of a BERT model on a large corpus in the relevant language with the help of neural networks. In the fine-tuning step, this knowledge (i.e., the parameter values learned during training the source model) is then applied to a new, purpose-specific context to learn a certain task and vocabulary (Devlin et al., 2019).

Attention mechanism. Moreover, Transformer-based architectures avoid processing text data sequentially, as a *recurrent neural network (RNN)* would. Obviously, text is inherently sequential and RNNs have been designed to handle precisely this property (as opposed to, for example, the BOW approach). Disadvantages of recurrent modeling, however, include a lack of parallelizability and the tendency to favor recent input. The attention mechanism allows the network to access any past modeling state by allocating dedicated attention weights and therefore achieves much better parallelization and reduced training times (Vaswani et al., 2017).

4.3.2 Input pre-processing

Very roughly speaking, a Transformer architecture consists of an encoder and a decoder component: the former processes inputs and creates an intermediate representation the latter then maps to predictions (Zhang et al., 2020). In order to apply BERT, the textual input to its encoder must be pre-processed in a specific way. To this end, we split word into tokens or word fragments based on a given vocabulary which is determined a priori by the pre-trained language model. Each sequence of tokens is then converted into a numerical vector, with some additional tokens: the beginning of the first sequence is signed by a “[CLS]” token, the end by a “[SEP]” token.
Consider the following example to illustrate this procedure, where the first statement is the original text and the second its pre-processed form:

Die Ausgrenzung von MigrantInnen von der #EssenerTafel ist inakzeptabel und rassistisch. Wir dürfen nicht zulassen, dass die Ärmsten gegeneinander ausgespielt werden.

[CLS] Die Ausgrenzung von MigrantInnen von der #EssenerTafel ist inakzeptabel und rassistisch. Wir dürfen nicht zulassen, dass die Ärmsten gegeneinander ausgespielt werden. [SEP]

A marker assigning it to sentence “A” or “B” helps to indicate for each token in a sequence which sentence it belongs to. A positional embedding further stands for the position of each token in the sentence. In short, the document contents and structure are translated into a numerical representation. The inputs have to be of identical length, where the maximum length of an input sequence can be 512 tokens. Shorter inputs are filled with padding tokens and longer ones are truncated (Devlin et al., 2019).

4.3.3 Pre-training

In its originally proposed form, BERT was pre-trained on a huge corpus of Wikipedia entries for two types of task: *masked language modeling (MLM)* and *next sentence prediction (NSP)*. These basic strategies shall enable the network to learn the fundamental structure of a language from a vast amount of training data in a self-supervised manner, without any need for labels.

Masked language modeling (MLM). By masking a random word, BERT tries to predict it without considering its positioning in the sequence. As BERT is able to work in a bidirectional way, that is, to condition the predictions for the masked word on the co-occurring words on both sides, it is able to capture varied and flexible information about the context of a certain word. After randomly choosing 15% of the token embeddings, 80% of these will be replaced by the "[MASK]" token, whereas a further 10% are swapped for a random token, and in the 10% remaining cases, the masked tokens are left unchanged. Afterwards, these sequences are fed into the BERT model (Devlin et al., 2019). Revisiting our example, the masking process could result in something like:

[CLS] Die Ausgrenzung von [MASK] von der #EssenerTafel ist inakzeptabel und [MASK]. Wir dürfen nicht zulassen, dass die [MASK] gegeneinander ausgespielt werden. [SEP]

Next sentence prediction (NSP). Furthermore, to capture the relationship between two sentences, BERT learns to predict whether or not the second sentence in a pair is the subsequent one in the original document. During training, half of the inputs are comprised of original pairs, while the other half has a random sentence from the corpus as second sentence. The goal is to have BERT distinguish between real and fake pairs in a reliable manner.

Sentence A: *[CLS] Die Ausgrenzung von [MASK] von der #EssenerTafel ist inakzeptabel und [MASK]. [SEP]*
Sentence B: *Wir dürfen nicht zulassen, dass die [MASK] gegeneinander ausgespielt werden. [SEP]*
Label: *IsNextSentence*

Sentence A: *[CLS] Die Ausgrenzung von [MASK] von der #EssenerTafel ist inakzeptabel und [MASK]. [SEP]*
Sentence B: *Freue mich sehr für ihn und auf die Zusammenarbeit. [SEP]*
Label: *NotNextSentence*

These two strategies are applied jointly in order to minimize an additive loss function.

4.3.4 Fine-tuning

BERT can be applied to various language tasks, e.g., *question answering* or *named entity recognition*, and sentiment classification in particular. In order to have the pre-trained network solve a specific problem, we need to fine-tune it on the target task. Crucially, at this point, the need for labeled data comes in. For fine-tuning we simply need to alter the output layer that adapts to the target task. In our analysis, we use the `huggingface PyTorch` implementation `BertForSequenceClassification`[3], which adds a single linear network layer on top of the existing network for classification. During fine-tuning, parameters in all layers are updated (Devlin et al., 2019). We set the following parameters recommended to use for fine-tuning by the authors: a minibatch size of 16 sequences, a global Adam learning rate of 2e-5, and 4 as number of epochs.

[3]`https://huggingface.co/transformers/model_doc/bert.html#tfbertforsequenceclassification`

4.3.5 Aspect-based sentiment analysis

The ABSA approach is a more sophisticated task than standard text-level sentiment analysis. Apart from classifying a given text, in this case a tweet, it focuses on extracting aspects mentioned in a given text together with the tokens contained in the text in the hope of extracting the most relevant information from the data. For our data situation, the most appropriate approach is therefore the methodology described by Xu et al. (2019) in fine-tuning BERT to ABSA.

Post-training. The basic pre-trained BERT model used in our work is the `bert-base-german-cased` model as we deal with tweets in German language and may expect to have advantages from letter casing (all nouns beginning with capital letters in German). This model was originally pre-trained using the latest German Wikipedia texts, news articles and open legal data sets of German court decisions and citations. This data set has a size of approximately 12 GB in total (`https://deepset.ai/german-bert`, Ostendorff et al. (2020)).
In order to improve both the domain and task knowledge, the authors recommend to apply a so-called post-training task based on contextually related data because fine-tuning BERT directly with a limited amount of labeled data may end up with domain and task challenges. Post-training on domain knowledge is applied using the pre-trained weights as initialization and leveraging both pre-training tasks (MLM and NSP). The weights are updated based on the summed losses from both tasks.

Aspect extraction. The aspect extraction sub-task is supposed to find aspects in a given text that the author has commented on. The idea is to use a supervised technique and label each token from a sequence with one of three labels: *B – beginning of an aspect*, *I – inside of an aspect*, or *O – outside of an aspect*. Afterwards, for each position of the sequence, a dense layer with subsequent softmax operation is applied to predict one of the three labels for all positions of a sequence. As shown in the original paper, the aspect extraction task requires, perhaps unsurprisingly, exhaustive domain knowledge (Xu et al., 2019).

Aspect sentiment classification. Lastly, the aspect sentiment classification sub-task attempts to classify the sentiment polarity of a given text, generally into positive, negative or neutral (omitting the neutral category in our case). The two inputs for this task are an aspect and a sentence or tweet containing this aspect, where the aspects are either extracted automatically with the above methodology or are made available beforehand by another technique. After application of the softmax activation and training with cross-entropy loss we obtain probabilities predicted for each of the sentiment categories that are eventually thresholded to yield a unique class label. (Xu et al., 2019).

4.3.6 Implementation

All of the models and methods are applied in `Python` (version 3.7.10) using the computational resources of Google Colaboratory[4], where free GPU power is available to researchers. As mentioned above, we adopt the `bert-base-german-cased` model as the basis for our experiments.
The overall application can be divided into two general parts: first, document-level sentiment analysis, and second, ABSA. For document-level analysis we discern four variations. Besides using the basis model by directly fine-tuning it on our train set of 972 tweets (that is, using 80% of our labeled data), we additionally enrich our training instances by 6,444 customer reviews about the German public train operator *Deutsche Bahn*, made available in the course of the GermEval

[4]`https://colab.research.google`

coding challenges. The other 20% of labeled data are used for evaluation. Furthermore, we develop our basic model by post-training it on domain knowledge, namely on 29,715 unlabeled and previously unused tweets collected in the scraping process. This model is then fine-tuned. Finally, a fourth variant is given by another German-language model named `bert-base-german-dbmdz-cased`[5] and pre-trained on a larger data source than the basic model. The data set has a size of 16 GB and over 2 billion tokens, suggesting the ability to produce better results.
For ABSA we additionally post-train the basic model on the GermEval data set but do not use these data as part of training instances, as the aspects deviate substantially from those detected in our corpus. The methodology for ABSA is widely used in sentiment analysis of review texts and is applied to tweets with political context in this project, which is arguably a quite different domain. Perhaps due to this fact, abstract extraction is not successful for our data. Our personal presumption is that this happens because of the lack of clear contextual and semantic delimitation of aspect-containing words we observe in the training data. We see plenty of opportunities for other use cases of this implementation, and therefore include it in our submitted code, but will not discuss it any further as part of this report. For the ABSA task we thus resort to the manually assigned aspects for both training and evaluation procedures.

4.3.7 Results

We assess each model's performance on the test set we have set aside, containing 243 observations. The performance metrics are the same as for evaluating the standard ML solution in chapter 4.2.5. Table 3 shows the results for the total of eight models implemented in both approaches (with SA short for sentiment analysis and overall best learners per task shadowed in gray):

	ABSA				**SA**			
	GC	**GC-G**	**GC-T**	**GCD**	**GC**	**GC-G***	**GC-T***	**GCD***
accuracy	0.893	0.905	**0.918**	0.889	0.889	0.901	0.905	**0.926**
F1 score	0.803	0.816	**0.851**	0.791	0.794	0.821	0.827	**0.864**
TN	164.000	**169.000**	166.000	165.000	164.000	164.000	165.000	**168.000**
TP	53.000	51.000	**57.000**	51.000	52.000	55.000	55.000	**57.000**
FN	14.000	16.000	**10.000**	16.000	15.000	12.000	12.000	**10.000**
FP	12.000	**7.000**	10.000	11.000	12.000	12.000	11.000	**8.000**

Table 3: Results for BERT variants, where
GC = `bert-base-german-cased`,
GC-G = `bert-base-german-cased` post-trained with GermEval data
GC-T = `bert-base-german-cased` post-trained with unlabeled tweets, and
GCD = `bert-base-german-dbmdz-cased`.
Asterisks indicate additional fine-tuning with GermEval data; best performance achievements are marked bold; overall best learner per task is marked gray.

The BERT-based classifiers clearly outperform the standard ML solutions (and thus the random baseline) and also achieve fairly good results in absolute terms. For the document-level sentiment analysis, `bert-base-german-dbmdz-cased` with additional fine-tuning on the GermEval data emerges as the overall best classifier with top results across all metrics and 92.6% accuracy. Regarding the ABSA task, `bert-base-german-cased` post-trained on the pool of unlabeled tweets performs best with only slightly lower accuracy of 91.8%.

[5] `https://huggingface.co/dbmdz/bert-base-german-cased`

The experiments confirm what the standard ML results have already suggested: the additional consideration of aspects, or topics, leads to a marginal increase of predictive power at best and does not quite justify the procedural and computational overhead. However, we can conclude that the application of BERT and its variants results in satisfactory outcome in classifying tweets into positive and negative categories for both document-level sentiment analysis and ABSA tasks.

5 Knowledge transfer

In the course of our analysis we have employed a multitude of techniques that find application in NLP tasks. Many of these are universal and not limited to Twitter data or sentiment analysis, making them part of a basic toolkit any NLP researcher might use at some point. We have therefore assembled a broad collection of teaching material that shall help practitioners acquire the basic methodology of text processing. Everything is available on a public website[6]. The material is composed as a coherent online course, largely following the analytical steps reported above, with some omissions so as not to overload the scope.

Scope. The course is organized in three blocks: an introductory part, NLP with `R`, and NLP with BERT. In the first block we give a general overview on NLP and point out key challenges, present the working data – the same corpus we have worked on for the above analyses – and introduce the most important text processing objects in `quanteda`, which we use heavily for the `R` part.
The `R` block comprises three sub-chapters. In the first of these, we begin by teaching basic web scraping, as this is often preliminary to the actual analysis, and proceed with fundamental text processing tools, such as handling regular expressions and normalization techniques. Static feature extraction, comprising most of the quantities described in section 4.2.2, concludes this first hands-on part. Topic modeling is discussed in the subsequent sub-chapter, where the focus is mostly on the STM. The last part is concerned with collecting the previously extracted features into a classification task and performing the actual sentiment analysis. We also include some general background on ML and demonstrate visualization options for results. Lastly, we show how to use BERT for sentiment analysis. This block briefly touches upon theoretical foundations of deep learning and Transformer models, but focuses more on practical implementation.

Media. We use a mixture of different media in the attempt to to strike a balance between theoretical and practical contents. All blocks contain, to varying extent, the following materials:

1. slides introducing the topics to be covered in the respective part,
2. code demonstrations giving first examples,
3. exercises (with solutions) offering the opportunity for practical application, and
4. additional sources for more in-depth study.

A first two-day live workshop already took place in April/May 2021. We were careful, however, to design the material in a self-contained way, with exhaustive explanations, so it may be used without additional instructions. With this, we hope to support researchers that are new to NLP to get familiar with a basic foundation of relevant techniques.

[6]`https://lisa-wm.github.io/nlp-twitter-r-bert/`

6 Discussion

Possible explanation for topics worsening results: make task more complex rather than aid it maybe pre-defined word embeddings would have helped

7 Conclusion

foo

A Appendix

B Electronic appendix

Data, code and figures are provided in electronic form.

References

Aggarwal, C. C. (2018). *Machine Learning for Text*, Springer.

Baly, R., Badaro, G., Hamdi, A., Moukalled, R., Aoun, R., El-Khoury, G., El-Sallab, A., Hajj, H., Habash, N., Shaban, K. B. and El-Hajj, W. (2017). Omam at semeval-2017 task 4: Evaluation of english state-of-the-art sentiment analysis models for arabic and a new topic-based model, *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, p. 603–610.

Becker, M., Binder, M., Bischl, B., Lang, M., Pfisterer, F., Reich, N. G., Richter, J., Schratz, P. and Sonabend, R. (2021). mlr3 book.
URL: *https://mlr3book.mlr-org.com*

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., Matsuo, A., Lowe, W. and Müller, C. (2021). *quanteda: Quantitative Analysis of Textual Data.* R package version 3.0.0.
URL: *https://CRAN.R-project.org/package=quanteda*

Bhatia, S. and Padmanabhan, D. (2018). Topic-specific sentiment analysis can help identify political ideology, *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 79–84.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, Springer.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation, *Journal of Machine Learning Research* **3**: 993–1022.

Bode, L. (2017). Gateway political behaviors: The frequency and consequences of low-cost political engagement on social media, *Social Media + Society* **3**.

Breiman, L., Friedman, J. H., Olshen, R. J. and Stone, C. J. (1984). *Classification and Regression Trees*, Chapman & Hall/CRC.

Cambria, E., Das, D., Sivaji and Feraco, B. A. (2017). Challenges in sentiment analysis, *in* E. Cambria, D. Das, Sivaji and B. A. Feraco (eds), *Affective Computing and Sentiment Analysis*, Springer.

Correa, E. A., Marinho, V. Q. and dos Santos, L. B. (2017). Nilc-usp at semeval-2017 task 4: A multi-view ensemble for twitter sentiment analysis, *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, p. 611–615.

Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding, *CoRR* **abs/1810.04805**.
URL: *http://arxiv.org/abs/1810.04805*

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization, *in* F. Hutter, L. Kotthoff and J. Vanschoren (eds), *Automated Machine Learning. Methods, Systems, Challenges*, Springer, pp. 3–34.

Ficamos, P. and Liu, Y. (2016). A topic based approach for sentiment analysis on twitter data, *International Journal of Advanced Computer Science and Applications* **7**.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Hasell, A. and Weeks, B. E. (2016). Partisan provocation: The role of partisan news use and emotional responses in political information sharing in social media, *Human Communication Research* **42**: 641–661.

Hastie, T., Qian, J. and Tay, K. (2021). An introduction to glmnet.

Jabreel, M. and Moreno, A. (2017). Sitaka at semeval-2017 task 4: Sentiment analysis in twitter based on a rich set of features, *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, p. 694–699.

Jang, H., Rempel, E., Roth, D., Carenini, G. and Janjua, N. Z. (2021). Tracking covid-19 discourse on twitter in north america: Infodemiology study using topic modeling and aspect-based sentiment analysis, *Journal of Medical Internet Research* **23**(2).

Japkowicz, N. and Shah, M. (2011). *Evaluating Learning Algorithms. A Classification Perspective*, Cambridge University Press.

Jost, J. T., Barbera, P., Bonneau, R., Langer, M., Metzger, M., Nagler, J., Sterling, J. and Tucker, J. A. (2018). How social media facilitates political protest: Information, motivation, and social networks, *Advances in Political Psychology* **39**(1).

Kralj Novak, P., Smailović, J., Sluban, B. and Mozetič, I. (2015). Emoji sentiment ranking 1.0, `https://www.clarin.si/repository/xmlui/handle/11356/1048`.

Lang, M., Binder, M., Richter, J., Schratz, P., Pfisterer, F., Coors, S., Au, Q., Casalicchio, G., Kotthoff, L. and Bischl, B. (2019). mlr3: A modern object-oriented machine learning framework in R, *Journal of Open Source Software* .
URL: *https://joss.theoj.org/papers/10.21105/joss.01903*

Lindsey, J. K. (1997). *Applying Generalized Linear Models*, Springer.

Louppe, G. (2014). *Understanding Random Forests. From Theory to Practice*, PhD thesis, University of Liege.

Medhat, W., Hassan, A. and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey, *Ain Shams Engineering Journal* **5**: 1093–1113.

Mohammad, S. M. (2017). Challenges in sentiment analysis, *in* E. Cambria, D. Das, Sivaji and B. A. Feraco (eds), *A Practical Guide to Sentiment Analysis*, Springer.

Murphy, K. P. (2021). *Probabilistic Machine Learning: An Introduction*, MIT Press.

Ngyuen, H. and Shirai, K. (2018). A joint model of term extraction and polarity classification for aspect-based sentiment analysis, *Proceedings of the 10th International Conference on Knowledge and Systems Engineering (KSE).*

Ostendorff, M., Blume, T. and Ostendorff, S. (2020). Towards an open platform for legal information, *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, Association for Computing Machinery, New York, NY, USA, p. 385–388.
URL: *https://doi.org/10.1145/3383583.3398616*

Pennington, J., Socher, R. and Manning, C. (2014). GloVe: Global vectors for word representation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543.
URL: *https://www.aclweb.org/anthology/D14-1162*

Probst, P., Boulesteix, A.-L. and Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms, *Machine Learning Research* **20**: 1–32.

Qiang, J., Qian, Z., Li, Y., Yuan, Y. and Wu, X. (2019). Short text topic modeling techniques, applications, and performance: A survey, *Journal of Latex Class Files* **14**(8).

R Core Team (2021). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Rauh, C. (2018). Replication data for: Validating a sentiment dictionary for german political language, `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/BKBXWD`.

Richardson, L. (2007). *Beautiful Soup Documentation.*

Roberts, M. E., Stewart, B. M. and Airoldi, E. M. (2016). A model of text for experimentation in the social sciences, *Journal of the American Statistical Association* **111**(515): 988–1003.

Roberts, M., Stewart, B., Tingley, D. and Airoldi, E. (2013). The structural topic model and applied social science, *Advances in Neural Information Processing Systems Workshop on Topic Models*, pp. 1–20.

Roberts, M., Stewart, B., Tingley, D. and Benoit, K. (2020). *stm: Estimation of the Structural Topic Model.* R package version 1.3.6.
URL: *https://CRAN.R-project.org/package=stm*

Roesslein, J. (2020). *Tweepy: Twitter for Python!*

Ruder, S. (2019). *Neural Transfer Learning forNatural Language Processing*, PhD thesis, National University of Ireland, Galway.

Schulze, P. and Wiegrebe, S. (2020). Twitter in the parliament - a text-based analysis of german political entities, *Technical report*, Ludwig-Maximilians-Universität, Munich.

Schulze, P., Wiegrebe, S., Thurner, P. W., Heumann, C., Aßenmacher, M. and Wankmüller, S. (2021). Exploring topic-metadata relationships with the stm: A bayesian approach.

Selivanov, D., Bickel, M. and Wang, Q. (2020). *text2vec: Modern Text Mining Framework for R.* R package version 0.6.
URL: *https://CRAN.R-project.org/package=text2vec*

Sidarenka, U. (2019). *Sentiment Analysis of German Twitter*, PhD thesis, University of Potsdam.

Stier, S., Bleier, A., Lietz, H. and Strohmaier, M. (2018). Election campaigning on social media: Politicians, audiences, and the mediation of political communication on facebook and twitter, *Political Communication* **35**(1): 50–74.

Tian, Y., Yang, L., Sun, Y. and Liu, D. (2021). Cross-domain end-to-end aspect-based sentiment analysis with domain-dependent embeddings, *Comlexity* .

Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N. and Fox, E. A. (2020). Natural language processing advancements by deep learning: A survey, *CoRR* .
URL: *https://arxiv.org/abs/2003.01200*

van Rossum, G. and Drake, F. L. (2011). *The Python Language Reference Manual*, Network Theory Ltd.

van Vliet, L., Törnberg, P. and Uitermark, J. (2020). The twitter parliamentarian database: Analyzing twitter politics across 26 countries, *PLoS ONE* **15**(9).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). Attention is all you need, *CoRR* **abs/1706.03762**.
URL: *http://arxiv.org/abs/1706.03762*

Vayansky, I. and Kumar, S. A. (2020). A review of topic modeling methods, *Information Systems* **94**.

Waltinger, U. (2010). Germanpolarityclues: A lexical resource for german sentiment analysis, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, electronic proceedings, Valletta, Malta.

Wang, X., Xu, G., Zhang, Z., Jin, L. and Sun, X. (2021). End-to-end aspect-based sentiment analysis with hierarchical multi-task learning, *Neurocomputing* .

Xu, H., Liu, B., Shu, L. and Yu, P. S. (2019). Post-training for review reading comprehension and aspect-based sentiment analysis, *Proceedings of NAACL-HLT*, Minneapolis, USA, p. 2324–2335.

Zhang, A., Lipton, Z. C., Li, M. and Smola, A. J. (2020). *Dive into Deep Learning*. `https://d2l.ai`.

Zuo, Y., Wu, J., Zhang, H., Lin, H., Wang, F., Xu, K. and Xiong, H. (2016). Topic modeling of short texts: A pseudo-document view, *Proceedings of KDD '16*.