INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Analysis of User Profiles in Comment Areas of News Websites

Filip Kajzer

Bachelorarbeit

Beginn der Arbeit:	21. Juni 2017
Abgabe der Arbeit:	21. September 2017
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Gunnar W. Klau

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 21. September 2017

_____

Filip Kajzer

Abstract

The comment sections of news sites are an important outlet for users to talk about different topics, express their personal opinions and discuss them with other users. This thesis aims to investigate whether it is possible to determine on which news site a comment has been written, solely based on the comments contents, by using machine learning techniques.

This thesis is part of a longer-term research project, led by Dr. Ines Engelmann at the institute for communication science at Jena University. The research project aims to investigate interaction and deliberation of users on German news sites.

For the purpose of our experiment, we will first choose suitable news sites, crawl their comments and make a first evaluation of the data. Following the selection of suitable news sites for our experiment, and crawling their comments, a short analysis of the data set will show that news sites exhibit unique and distinctive characteristics in their statistics. News sites which allow for a login using Facebook have almost double the monthly comments than sites which do not. An investigation of the top 20 most contributing users of each site has shown that Süddeutsche Zeitung has an extremely active top 20 of users, with them being the authors of roughly 30 percent of all comments which we have crawled from the site. When further investigating a news site's structure, especially the comment section, we found that most of the statistics could be explained through it.

Our main focus, however, is on the classification problem at hand. The machine learning experiment is split into two parts. First, we will make the experiment, trying to classify with five different sites. Afterwards, we will repeat the experiment with only three sites to see if observations change. Both experiments are run with a number of different classifiers and features, and by using three-fold cross validation and grid search to get optimal results.

Our experiments show that the classification with all five news sites had best results when using a linear Support Vector Machine with a combination of all features, which yielded a macro-averaged F_1 score of 0.72. Same scores could be reached when using an RBF-kernel SVM, however the runtime of our experiment was almost 400 times higher compared to the linear SVM. We also conducted the experiment with Random Forest and K-Nearest-Neighbors classifiers, which both showed worse results. When repeating the experiment with only three of the five chosen sites, results change only marginally. However, performance of the classification did increase by roughly 10 percent compared to having all five news sites as labels.

Contents

1 Introduction

1.1 Motivation and Goal

In today's society, news are both consumed via print media as well as being accessed online, primarily through mobile devices. Newspapers are cross-media brands which reach consumers over all channels. In Germany, newspapers reach more than 86 percent [Rei16] of the population across all age groups over their printed and online presences.

One central aspect for many of the frequenting users of such a news paper's website is the comment section. Here, users can post comments anonymously to discuss each news article, express their feelings, and debate about a given topic. By being anonymous, these comments are generally more honest opinions and people do not shy away from expressing negative feelings if they have any [San14]. Since fewer and fewer people are accessing news over print media these days [Sch13, Pim16], more and more people are consuming news over the Internet [Rei16], meaning they can instantly express their opinion about topics.

In this Bachelor thesis, we will find distinctive characteristics of these anonymous users in a number of different German news sites and train a system to ultimately find out our thesis goal: *whether the origin of unknown comments can be predicted based on user comments used as training data*, origin meaning the news site where the comment was posted. To achieve this, we will use data mining techniques to crawl the desired data in the first step, and then analyze this data with various machine learning techniques, including natural language processing.

1.2 Outline

The remainder of the thesis is structured as follows. We will start off the thesis with an introduction on infrastructure of news sites' comment sections as a whole by examining their core characteristics in Section 2. Following this, we will decide on which news sites to investigate by examining which news sites are particularly popular in Germany with respect to their user base. Afterwards, we will discuss aspects which have to be considered for the data mining process of each of the chosen sites. A thorough examination and analysis of the collected data, based on numbers only, will be given. We will then try and find similarities between the chosen news sites based on the results of our descriptive analysis approach and also look for distinctive user behavior which separates one from another, if possible at all.

Based on key characteristics of the gathered data, we will discuss suitable classifiers and parameters for the given task at hand in Section 4. After that, we will use machine learning to reach the thesis goal. Finishing with a thorough discussion of the presented works, we will summarize the findings in a conclusion.

2 Dataset

In this section we will discuss the news sites and ultimately the dataset which we will use for our machine learning experiment in detail. In a later chapter we will then talk about machine learning techniques that can be used to test out the main hypothesis of this thesis, whether news sites can be distinguished solely on text.

2.1 Description

To make an experiment based on data, we first need a set of data. In this subsection we will discuss a number of German news sites which could be considered for the experiment and we will try to make an evaluation of the amount of data we would be able to access on each of them. Afterwards, we will look at points which need to be addressed to scrape the data from the chosen sites. Finally, we will look at some descriptive statistics based on the gathered comments.

2.1.1 Discussion on Suitable News Sites

Let us now take a look at a number of popular German news sites. Since the process of getting and processing the data can be a difficult task depending on the infrastructure, we then need to filter out some of the suboptimal candidates. For this, we will consider a statistic made by AGOF digital facts [Zei16] which shows the digital reach of websites in Germany for people above the age of 14. We will choose popular news sites based on the number of unique users. Figure 1 shows the amount of monthly unique users of German web sites of July 2016. This top 15 of digital reach per site shows that the interest in news sites is still incredibly high. Eight out of the top fifteen are news sites. After filtering out the non-news sites, we are left with following sites for consideration: Bild.de, ntv.de, n24, FOCUS ONLINE, SPIEGEL ONLINE, DIE WELT, Süddeutsche.de, ZEIT ONLINE, stern.de and FAZ.NET.

Now, we will have to investigate each of the comment sections to find viable candidates among the above mentioned ten candidates. As it shows, Bild.de and stern.de, the two tabloid news papers from the above list, do not include a comment section which disqualifies them for our analysis. More surprisingly, ntv, one of the more serious and quality news oriented sites, does not allow comments under their news as well. Important to note is that since the release of the statistics in July 2016, DIE WELT and n24 have merged their web presence into a single website in the month of September 2016. However, this does not have a negative influence on our analysis. This leaves us with the news media and online presences as seen in Table 1. The listed sites seem to be appropriate candidates for our analysis.

In the next step, we have to take a deep look at each comments section. Everyone of the above mentioned sites uses user names as pseudonyms for their users. These user names can be uniquely identified which allows us to do profiling. focus.de is the only website among the few that requires users to specify their real first and last names. focus.de and welt.de only allow to extract user's comments of the past month. spiegel.de is the only

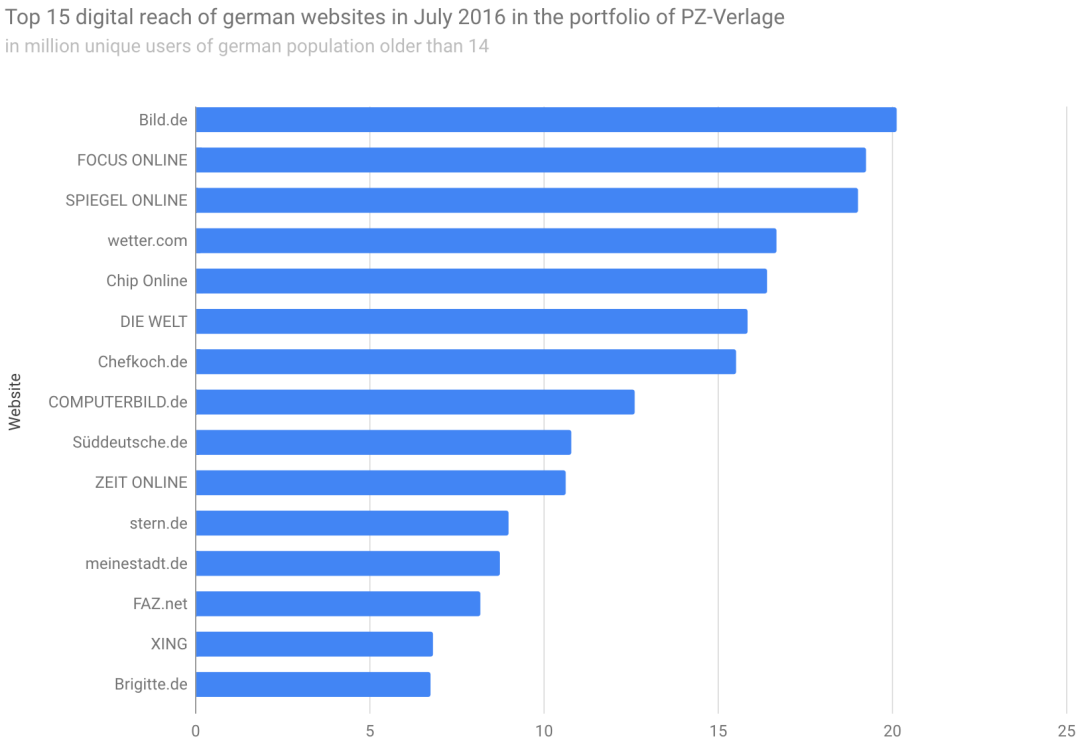

Figure 1: Top 15 reach of news sites July 2016. Source: AGOF [Zei16]

site that does not have a comment upvoting system. Among the sites focus.de is the only one that offers its users the possibility to downvote a comment. On every mentioned news site, the replies to a comment are also available. Finally, zeit.de is the only website on which the editorial staff can chose to highlight comments they think are exceptional by starring them. Table 2 summarizes these findings about the available data of each comment on the respective site.

As already mentioned, some sites only allow access to comments made in the past month. Based on this information and since the process of getting the data is rather time consuming, we will scale down the data we are aiming for to the past month for all of the sites. However, this is not true for Süddeutsche as it was the only website providing a dedicated, clear API to request comments directly in contrast to scraping them. However, comments on Süddeutsche cannot be made on all of the articles. Instead, there is an extra section called "Leserdiskussion"[1], where users can comment on a subset of articles only. We will be mining all comments from this section.

A further look at the sites has shown that the part of each news site with the most active users are the politics section. They show a broad set of different political orientations and are also the place where most discussions take place which makes them more interesting, in contrast to the fashion or weather section. We will thus focus this thesis on the politic

[1] `http://www.sueddeutsche.de/thema/Leserdiskussion`

Media	Website
Focus Online	focus.de
Spiegel Online	spiegel.de
Die Welt	welt.de
Süddeutsche Zeitung	sz.de
Die Zeit	zeit.de
Frankfurter Allgemeine	faz.de

Table 1: Popular German news media and their online presences

site	user name	first / last name	content	up votes	down votes	replies	starred
focus.de	yes	yes	month	yes	yes	yes	no
spiegel.de	yes	no	yes	no	no	yes	no
welt.de	yes	no	month	yes	no	yes	no
sz.de	yes	no	yes	yes	no	yes	no
zeit.de	yes	no	yes	yes	no	yes	yes
faz.net	yes	no	yes	yes	no	yes	no

Table 2: Comparison of news site comment sections

sections of the sites. Furthermore, since the user profiles on news sites have little to no information about the user, we will base our analysis solely on the comments which have been done over this one month period instead of the user profiles.

With all this information in mind, we are now ready to go into detail about the process of mining the comments.

2.1.2 Data Mining

Now that we have a good understanding about what data we want, we have to talk about the process of mining the data.

We will be using a web *crawler* to iterate over articles and *scrape* the data. While scraping is the process of extracting (downloading) texts from a web site, a crawler simply iterates over websites. We will use the Python framework *scrapy*[2] which provides us with such functionality. Since our focus will be on a one month period, we will make use of the filter function for news articles where it is possible. Faz, Spiegel Online and Die Zeit allow for such functionality, which simplifies the scraping process.

Süddeutsche on the other hand uses an external central discussion platform provider, named disqus[3], and simply embeds those comments on their website. Since disqus comes with an API, we can simply request and save the comments. Since this process is exceptionally fast compared to the process needed to get data from other news sites,

[2] `https://scrapy.org/`
[3] `https://disqus.com/`

we decided to download all comments which were available from this site.

Lastly, it is important to note that during the scraping process, we found that focus.de is not a suitable candidate for our analysis. The site does not allow for a simple view of comments and their respective answers, instead every comment of an article has to be requested individually to get the user profile and the comment replies. However, those comments and replies would not include up and downvotes. These would have to be requested additionally from a different endpoint. However, for request throttling reasons, one could only make a small number of concurrent requests before our scraper got banned. The number of requests that would have been necessary to get all the data was simply too high to be feasible. Because of this, we decided to exclude focus.de from our analysis because the scraping process was simply too time-consuming and costly.

3 Data Analysis

After having described the data set of our study in Section 2, let us now look at the dataset analysis in detail.

First, we will discuss our approach on comparing the news sites. Afterwards, we will present, analyze and interpret some descriptive key statistics about the underlying data in our experiment. In the analysis we will sometimes refer to Süddeutsche Zeitung as SZ, Spiegel Online as Spiegel, Die Zeit as Zeit, Die Welt as Welt, and finally Frankfurter Allgemeine as FAZ.

3.1 Descriptive Data Analysis

As we can see, the nature of our chosen news sites results in a broad mix of data. Nevertheless, this will make the evaluation of the data more interesting because different news sites have different user bases, that react to different articles written by different authors. Therefore, the reactions could vary drastically from site to site. After scraping the comments, we will analyze the raw data to find out key statistics and characteristics of each respective news website's users solely based on the data.

As described in Section 2, different kinds of statistics can be gathered from different sites. For this reason, we will compare our sites based on the smallest common denominator, i.e. we have to drop downvotes as well as starred comments from our analysis as they are only available different sites. Instead, we will make an analysis of the comments in respect to the amount of words used in each comment, the replies, and the upvotes. Even though spiegel.de does not offer upvotes on its website, we will still consider them for every other site as they are the best indicator for a comments quality and acceptance among the respective website's user base. Thus, upvotes are too important of a characteristic to be ignored. Furthermore, we will look at the most active users for each of the sites. We will investigate if particular sites have "power users", users who make a lot of comments.

To add to this raw analysis, we will consider a project by Leipzig University called SentiWS [RQH10]. SentiWS is a limited vocabulary for the German language with a sentiment score for each word in this vocabulary. It is made up of 15649 positive and 15632 negative word derivations. The score is within an interval of -1 to 1 with -1 meaning an extremely negative sentiment and 1 an extremely positive. We will investigate the verbs, nouns, adjectives and adverbs used in each comment to make an evaluation of its sentiment. We will then compare the sites to analyze the general direction of opinions. Furthermore, we can expand on this by evaluating a sites average sentiment.

3.2 Raw Data

Let us first take a look at how many comments we have mined from each site, which is summarized in Table 3.

With Süddeutsche Zeitung being the site where we were able to crawl all comments from, it naturally is also the site where we mined the most comments (170000). Bear in mind

News site	Comments
Frankfurter Allgemeine	22,121
Spiegel Online	46,247
Süddeutsche Zeitung	170,341
Die Welt	38,636
Die Zeit	22,352

Table 3: Amount of mined comments per news site

that SZ migrated to the discussion platform disqus on September 1st, 2014. Therefore, comments made prior to this date were not available. Next is Spiegel Online with around 46000 comments and Die Welt with around 39000, followed by Zeit and Frankfurter Allgemeine, which have a similar amount of comments (both around 22000). From this first set of data, one can already see that Spiegel and Welt are far more frequented or at least have a more active user base than Zeit and FAZ.

Figure 2 shows the average number of words per comment for each of the sites. SZ has the longest comments of our data set, with an average amount of 79 words per comment. FAZ and Zeit are in the middle, with 67 and 54 words per comment, respectively. Spiegel and Welt on the other hand have the shortest comments, with Spiegel averaging 43 and Welt averaging 41 words per comment.

To get an idea of how much discussion is taking place on each of the sites, we will look at the share of replies in comparison to the total amount of comments as shown in Figure 3. As we can see in the Figure, Zeit makes first place with 85 percent of all comments made being replies. SZ follows with 65 percent of comments being replies. Die Welt is in the middle with 51 percent. Lastly, Spiegel and FAZ make an almost even last place with Spiegel at 42 percent and FAZ at 40 percent reply share.

Next, Figure 4 shows the average amount of upvotes per comment. FAZ is the site where most upvotes are given by far. With FAZ averaging 33.5 upvotes per comment and the second highest, Welt, only averaging 16 upvotes, the gap is quite big. Zeit follows with 7 upvotes per comment and SZ makes last with only 2.5 upvotes per comment. Spiegel does not provide an upvote function as mentioned in Section 2.1.

Finally, to determine if a site has particular power users, we compare the comment share of the top 20 users in contrast to the whole user base. Figure 5 describes how many of the comments are made by the top 20 commenting users of each site. SZ's bar is notably high, meaning they have particularly active power users. 31 percent of comments on the site are made by only 20 users. Next up, Welt comes second with roughly 20 percent of comments made by the top 20. The other sites, however, do not have quite as active power users, with FAZ showing a comment share of 12.5 percent, Zeit with 10 percent and Spiegel with only 7 percent.

Let us now interpret and draw conclusions from these first statistics. For FAZ, the high amount of upvotes per comment is most striking. When examining FAZ's upvote system explicitly, we made the observation that even unregistered users are able to upvote a

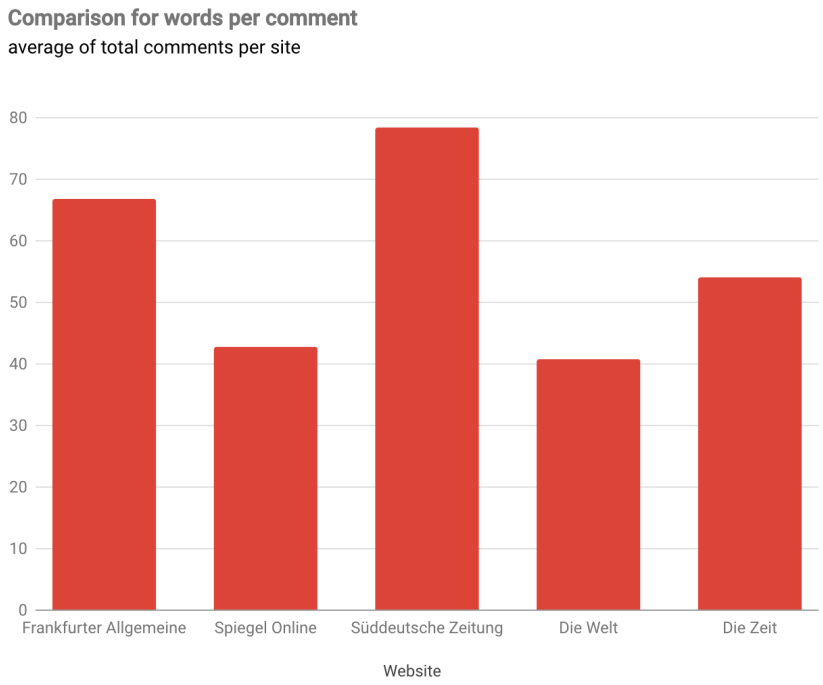

Figure 2: Average number of words per comment

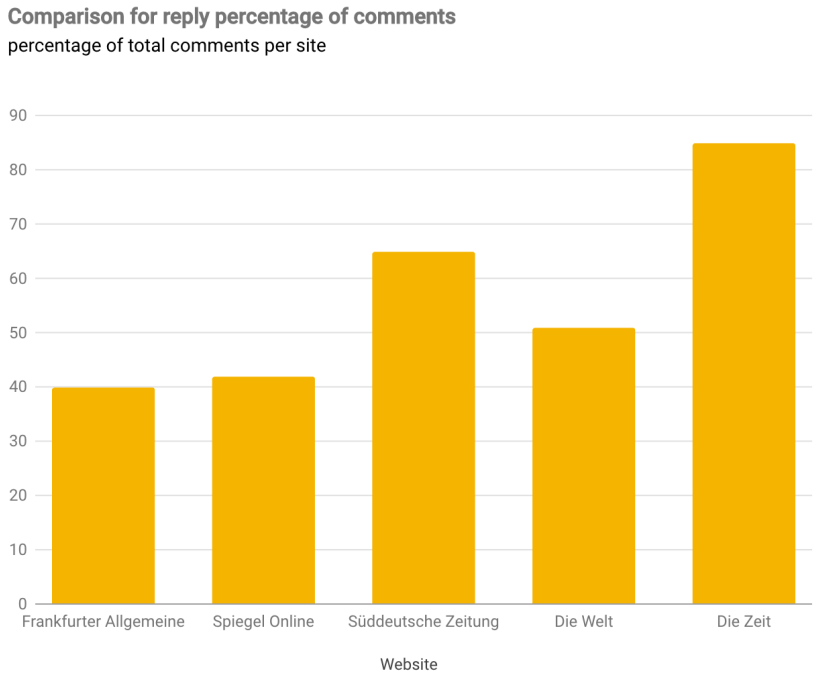

Figure 3: Reply percentage of comments

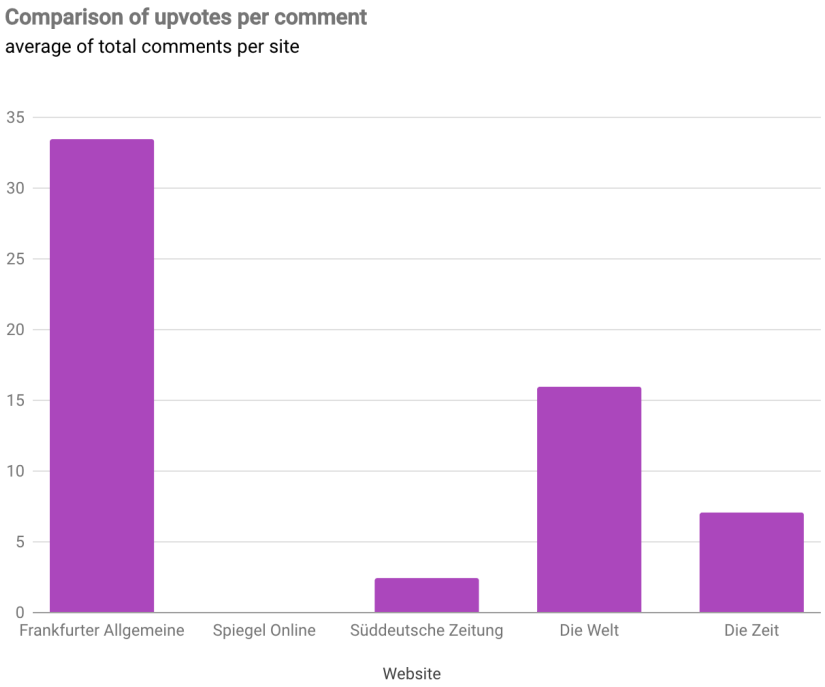

Figure 4: Average number of upvotes per comment

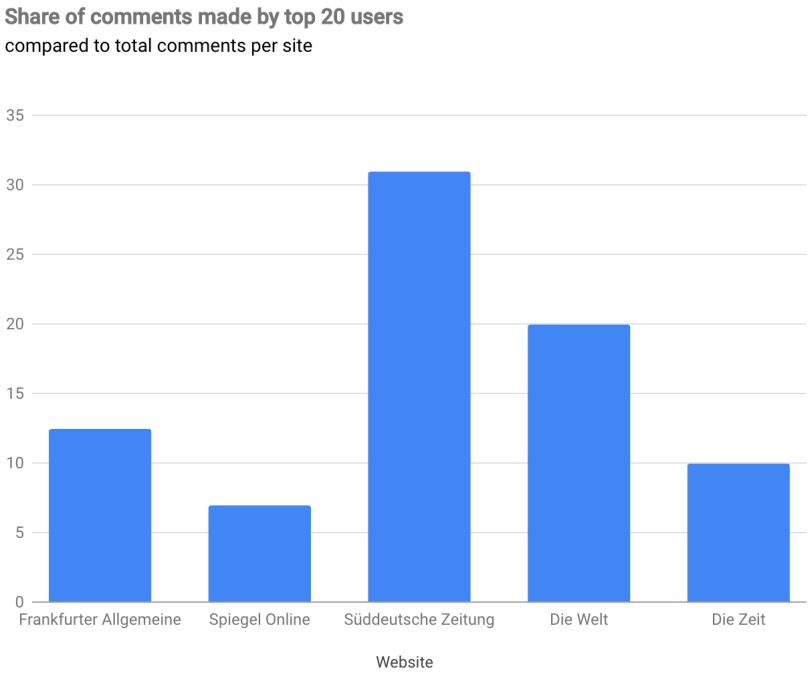

Figure 5: Share of comments made by the top 20 users

comment. The high number of upvotes in comparison to the other sites is most likely a consequence of this. However, to write a comment, a user still needs to be registered. In contrast, the amount of replies on Frankfurter Allgemeine seems very low. We will try to make an assumption as to why by examining the comment section. To read a comment, a user has to scroll down and click on a comment, which is described by a title, to read it. To read answers made to a comment, one has to then click "show answers", then click the title of an answer to read it. The process of reading comments and answers is rather bothersome and does not invite for discussion. Compared to Zeit, which has the highest share of replies with 80 percent, the 40 percent reply rate of FAZ seems rather low.

When looking at Spiegel, the comment share of the top 20 users seems very low when compared to the other sites. A deeper look at the login section for users on Spiegel reveals that a login using one's Facebook identification is possible. This most likely results in a much higher user base in contrast to only allowing a login by being a registered user, ultimately resulting in a high amount of different users commenting and a higher amount of comments overall. Note that Spiegel is also one of the sites where the average word length is one of the lowest with 43 words. On another note, the relatively low amount of replies might be a result of Spiegel's reply system. Users cannot directly reply under a comment, which is the case for every other news site. Instead, a user will need to make a comment with a reference ID to the original comment, which can be distant from each other inside the comment section. Since this seems rather questionable, we assume that users are not as keen to make replies to a comment instead of simply writing a comment with a reference in quotes to the original comment as observed multiple times by us (For example comments like: "Angela Merkel is wrong" - No, Angela Merkel is not wrong because of [...]).

The amount of comments per month for SZ in the timespan of three years would be around 4,700. This is much smaller when compared to the other sites, but due to the fact that users can only comment on small subset of all articles. It seems like SZ users like to discuss topics in more detail, resulting in a high number of words per comment and the moderately high amount of replies. It is very interesting to investigate the amount of comments made by the top 20 users as seen in Figure 5. The most commenting user has made a total of 5,760 comments. Considering the fact that disqus for SZ has only been around for roughly three years, SZ's power users are particularly active. Note that the comment section of SZ is not embedded in the main articles of the news site. Users can only comment on a small subset of articles, which they explicitly have to view in the "Leserdiskussion" section to comment on news and discuss them, but also to upvote them. The low number of upvotes per comment and the high comment share of power users are most likely a consequence of the usage of this scheme, since the exposure of the comment section is not as good as for other sites, resulting in a smaller user base when compared to the other sites.

Welt is one of the sites where the average number of words per comment is the lowest. Note that on Welt, users can login with their Facebook account as well. It is most striking that the word length is so small and the upvote rate (considering you have to be a registered user) is the highest. Users on Welt seem to like the quick-witted, short comments and make more use of the upvote function.

Zeit has a particular high rate of replies per comment, as seen in Figure 3. The reason

for this high number of comments per article within the politics section is immediately obvious at first glance on Zeit. The reply sector of the comment section of Zeit is very well built. With one click, users can see all replies to a comment and choose which one to answer. Users seem to enjoy this feature a lot and have many discussions in the comment section. The number of replies to comments can get incredibly high at times. However, the amount of upvotes per comment however is rather low for Zeit. This might be due to the sometimes extreme numbers of answers to a comment, which no one bothers to read (since they have to click "show answers"). More interestingly, even though Zeit has almost 85 percent replies, the amount of words per comment is not the lowest among the sites. Replies to users seem to be thought out and well structured comments of high quality.

3.3 Sentiment

Now that we analyzed the raw data, let us take a look at the sentiments on the sites with the help of SentiWS. In the following illustrations, we will be looking at the sentiment of all comments which have been made on a site, grouped up in intervals.

First of all, we have to consider that all of the following results have been computed by using SentiWS [RQH10]. Since SentiWS is a very limited dictionary, of all the words written in the comments that we want to analyze (nouns, verbs, adjectives, and adverbs) only around 15 percent had a match in the dictionary. Of course, word misses were not considered for the average sentiment of a comment. With that out of the way, let us look at the results.

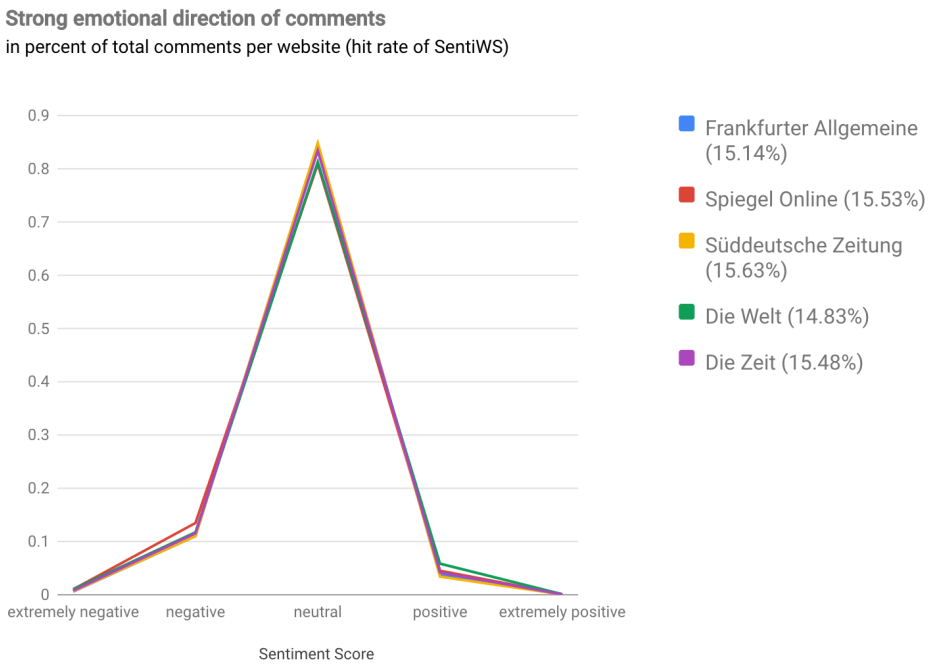

Figure 6: Rough sentiment for each site

Since it is not defined how to label sentiments, we experimented with different sentiment score intervals to put comments in categories. To classify the sentiment of a comment we will start off by using the following five categories: extremely negative, negative, neutral, positive, extremely positive. In Figure 6, we see the sentiment of the each site's comments if we only consider these categories. Here, extremely negative means a comment has an average sentiment score < -0.6, negative means < -0.2, neutral comments have a sentiment score $-0.2 \leq x < 0.2$, positive meaning ≥ 0.2 and extremely positive meaning ≥ 0.6. If we look at the sentiments of comments this way, it seems like all sites are equal. Most comments will be neutral, however the amount of negative comments is three times higher (≈ 12 percent) than the amount of positive comments (≈ 4 percent) across all sites, with extremely negative comments being even 10 times as frequent as extremely positive ones. However, the amount of extremely negative comments is still very low ($0.5 - 1$ percent).

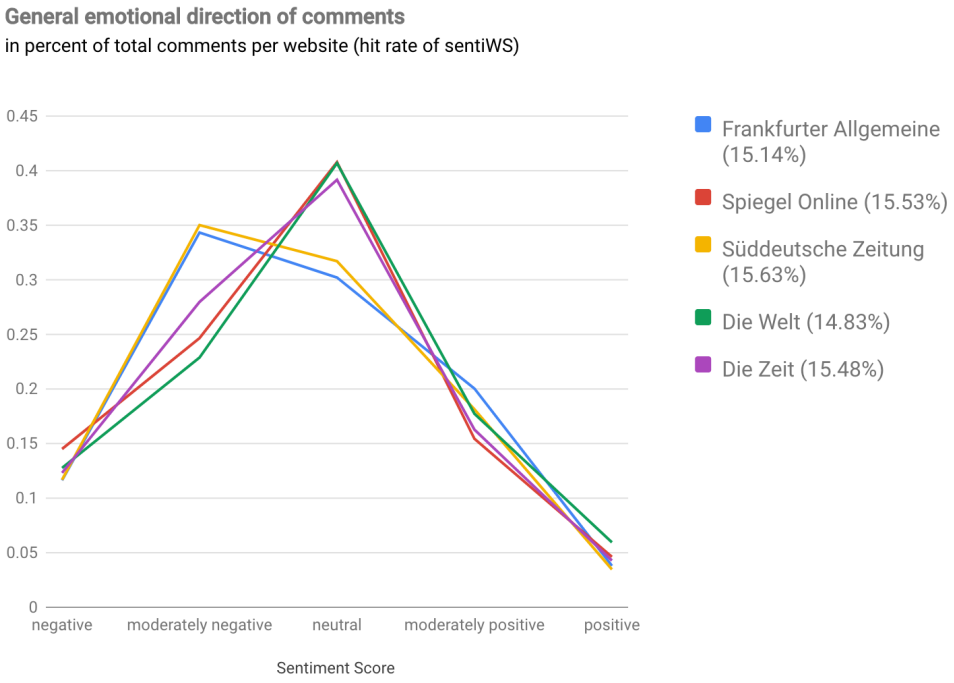

Figure 7: General sentiment for each site

Since we want to get more information from the sentiment analysis, we will now try to divide the neutral part of Figure 6 into a less coarse partition. In Figure 7, negative comments are comments with sentiment scores < -0.2, moderately negative comments have a score < -0.02, neutral is $-0.02 \leq x < 0.02$, moderately positive ≥ 0.02, and positive ≥ 0.2. With this partitioning, differences between sites become visible. It seems like the news sites can be partitioned into two groups based on their comment sentiment. Frankfurter Allgemeine and Süddeutsche Zeitung have more moderately negative than neutral comments, while the rest of the sites still has more neutral than negative comments. The amount of moderately positive comments however is a fair amount lower (≈ 17 percent) compared to moderately negative comments for Spiegel Online, Die Welt and Die Zeit

($\approx$ 26 percent). The amount of negative comments ($\approx$ 12 percent) however is almost three times as high as the amount of positive comments ($\approx$ 4 percent). This is even further illustrated in Figure A in the appendix, where we look at the sentiment scores in 0.1 intervals. However, the findings do not change.

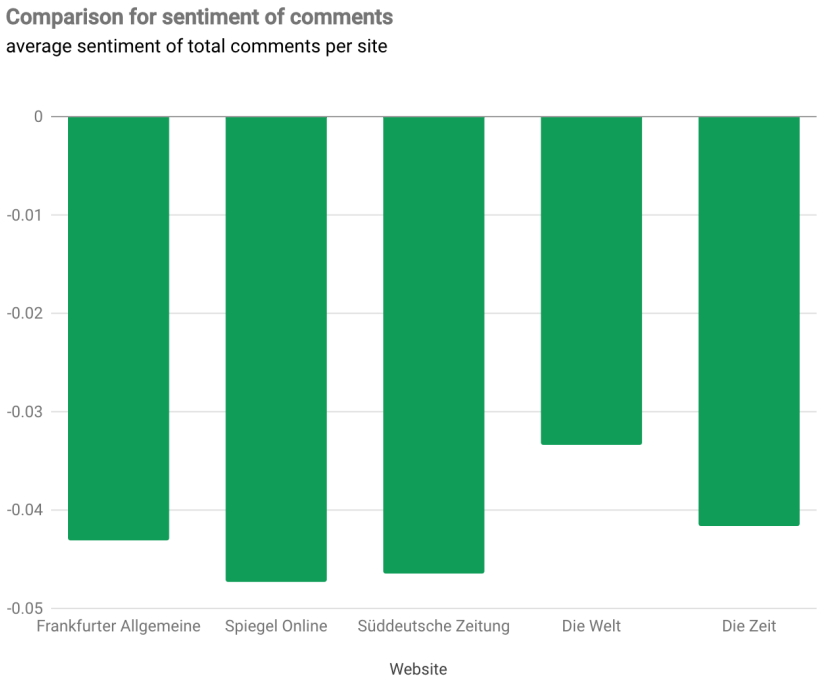

Figure 8: Comparison of average sentiment

Lastly, let us investigate the average sentiment of all comments on the sites. First of all, when looking at Figure 8, the scores which we are investigating are all extremely low and close to each other. However, Welt seems to be a little less negative in comparison to the other sites, who are relatively close. This might be due to users on Die Welt having to register with their Facebook account which shows one's first and last name.

It seems like all news sites roughly share the same sentiment. Since the nature of anonymity in comment sections generally results in a less civil behavior [San14], this was to be expected. However, when taking a deeper look, Süddeutsche and Frankfurter Allgemeine seem to be a little bit more negative than the rest. It is important to note that, since this evaluation was made by using SentiWS and the hit rate of words in the dictionary was only about 15 percent, these results have to be taken with a grain of salt.

4 Classification of News Comments

In Section 2.1, we focused on the dataset and how we are going to mine data. In the following, we will discuss what techniques are necessary to test the main hypothesis of this work, namely whether it is possible to determine the origin of a comment solely based on its contents. Before presenting the results of our experiments, we will discuss those techniques in depth and we will also learn why they are necessary.

4.1 Experiment Setup

For our analysis we will use *scikit-learn* [PVG+11], a simple and efficient machine learning framework for the programming language Python. It is one of the most common and widely distributed frameworks for machine learning and provides us with everything we need for the analysis of the news site comments.

To prepare our dataset for the classification process there are basically four necessary steps.

1. **preprocessing** of raw text from the comments, such as tokenization
2. **splitting** the whole dataset into training and testing data
3. **extracting** of features from text
4. **choosing** an appropriate classifier and its parameters

With fine tuning of features and parameters, a much better accuracy and ultimately much better results from our classifier can be achieved. We will experiment with a multitude of combinations of features and classifiers to get a detailed look at the efficiency of our features in regards to the classifier.

4.1.1 Preprocessing

To get features from raw text it is absolutely necessary to transform the text into a list of tokens. There are many definitions of what a token might be, however the most common approach is to transform each word from a text separated by a whitespace into a token. This way, it is much easier to iterate over a text, e.g. to count the occurrences of each token within a given text and ultimately to compare samples based on this feature. The scikit library already comes with this functionality of tokenization, however because this library tokenizes in its own way and it is specialized for the English language, we will instead use natural language processing (NLP) to extract tokens for the mined comments which are in German.

We will use *spaCy*[4] for preprocessing our data, a powerful and very fast NLP framework. Compared to other solutions, spaCy provides a model for German, which is of course

[4] `https://spacy.io/`

mandatory for our experiments. For us, the most interesting feature of spaCy is the part-of-speech (POS) tagging provided by the framework. With the help of the POS tagging of spaCy, we additionally have the information which part-of-speech belongs to each token. This will be very helpful for creating our own features, as it will be described in a later paragraph.

4.1.2 Training and Test Data

Our ultimate goal in machine learning is applying an algorithm to create a *model* from our data which then can make predictions on new data. *Validation* is the process of assessing how well our model performs against real world data that is previously unknown to the system. Validation is necessary to understand the characteristics of our model before making real predictions.

To achieve this, we first have to split our data into two sets: a training and a test set. On each of the five scraped data sets (one for each website) the following procedure was used. A random 20 percent of the data was set aside for the test set. We then further halve the training set four times using random subsets of the remaining split until the data set reaches 1/16th of our data per site for training. This way we can investigate how much influence a bigger set of training data has on the prediction quality of the test set. We will still benchmark against the 20 percent of the whole data set that we set aside as a test set in the beginning. We will also take a look at our model when only classifying three of the given five sites (Frankfurter Allgemeine, Spiegel Online and Die Zeit), since classifying gets more complex and harder the more labels (classification targets) there are.

To properly evaluate a model before making real predictions, the training set will be used for training and validating (tuning) of our model while the test set will be saved for a later, final evaluation of the model.

To compare benchmarks of our models, we will need a scoring function which in our case is the *macro-averaged* F_1 score. This scoring considers both precision and recall as shown in Figure 9. Precision tells us what proportion of results, which we classified as a certain label, actually belong to that label. Recall tells us what proportion of data that actually belong to a certain label were classified by us to that label.

The F_1-score, also F-measure, is the harmonic mean of precision and recall and is computed as in equation 1.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1}$$

Since our dataset is a multi-class case, the F_1-score would be the weighted averaged F_1 scores across all classes. Since we have a balanced data set, meaning all labels have an equal number of samples, we will use the macro-averaged F_1 since it computes the unweighted mean.

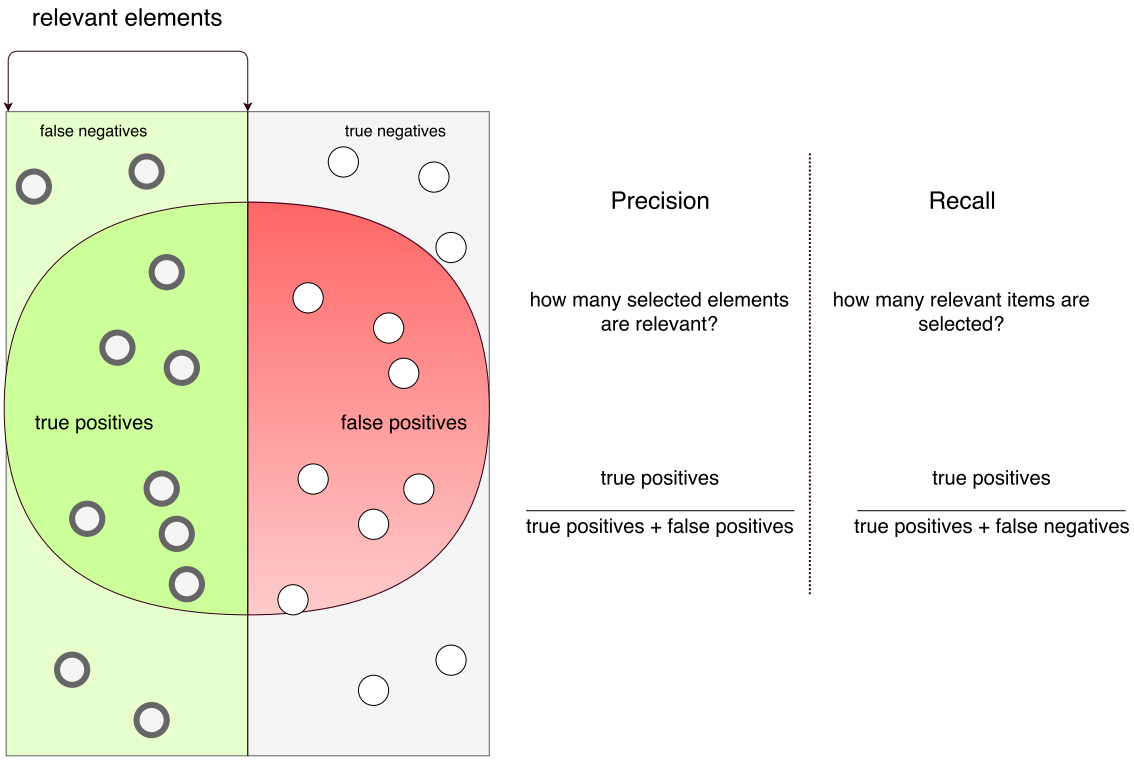

Figure 9: Precision and Recall. Own graphic, based on [5]

4.1.3 Features

The choosing of the right features for a classification problem is often the decisive factor for success. Since classifiers cannot handle our raw data, the sequence of symbols (words), we first have to transform our data in numeric vectors with fixed length as described below.

4.1.3.1 Bag of Words The most intuitive way to reach our goal would be a so-called *Bag of Words* approach using a *Vectorizer*. The Vectorizer creates a vocabulary from our training data in which each token gets assigned a fixed id. The extraction of a Bag of Words from our raw data happens as follows.

1. **tokenization** which we already did while preprocessing the data
2. **occurrence counting** for each given token in a text
3. **normalization** of the resulting feature vector belonging to the whole text

In this scheme, each occurrence of a token is feature. In the end, we obtain a numeric vector with all features for each text. This process is called *vectorization*.

Since we still do not consider how many words a text has with the above approach, we will transform our vectors into a **term-frequency inverse document-frequency** representation. The goal of this representation in contrast to simply using term frequency is to scale down the impact of tokens which occur very often in the training set since they often have little to no meaning for the classification process such as articles and pronouns. This step comes into play after we have vectorized our texts.

Unfortunately, a simple Bag of Word comes with a multitude of limitations. A collection of unigrams which a Bag of Words representation ultimately is, can not grasp phrases or word contexts. Word order, spelling mistakes or derivations which are to be expected in our dataset are ignored. Our solution for this problem will be *bigrams* and *character-n-grams*. As an example consider these three **unigrams** synthesized from preprocessing a German sentence:

ich, lese, gerne

If we consider each of those unigrams only by themselves the context is not preserved. **Bigrams** consist of two immediately consecutive unigrams and therefore allow features to hold more information about the context of our tokens. The phrases

ich lese, lese gerne

have a lot more information about the context of the tokens but are both built up from the same unigrams.

Character-n-grams are a representation which is resilient towards spelling mistakes and word derivations. Consider the following example of somebody misspeling the word "ich". Using a 2-gram the following features will be created.

i, ic, cc, ch, h

In contrast, the correct spelling would have the following features.

i, ic, ch, h

As one can see, the n-gram representation contains the same features for the correct and wrong spelling and most of the features will still match. This is also true for word derivations which is why we will use character-n-grams as a feature. Word derivations are especially important in the context of language dialects.

4.1.3.2 Part-of-speech Even if we can preserve some of the word context with the above mentioned techniques which add upon the Bag of Words approach, most of the structure of a sentence will still be lost. Whereas we have more information about our tokens thanks to the preprocessing, we will build our own feature which counts part-of-speech instead of counting tokens themselves. This way, we can examine the structure

of sentences in a given comment. The idea here is to use the POS tagging generated by spaCy and transform it into a feature. To reach this goal, we will use spaCy's subset of the universal part-of-speech tagset by google [PDM11] as POS tags[6]. We will create one feature for each of the 17 kinds of part-of-speech that exist in spaCy for German sentence construction and count their occurrences in a given comment.

With help of our newly created feature, we can try to find similarities in sentence structure in comments from each news site and use them to identify a comment's origin. After we generated our vector with the occurrences of POS features, we normalize the vector in respect to the token count.

4.1.3.3 Feature Union We already got to know a multitude of ways to extract features from our data. It can often be advantageous to combine features which can increase performance of a classifier. For our analysis, we will take a look at each feature individually and also a combination of our features to compare their performance in the evaluation.

4.1.4 Classifiers

The choice of the right classifier can also be a deciding factor in success or failure. The classifiers each differ greatly in functionality and field of use. We will take a look at some of the more traditional and common classifiers, their parameters and the effect that they have. At the end, we will take a further look at how we can fine tune those parameters.

4.1.4.1 Support Vector Machine Support Vector Machines [CV95] are based on the concept of hyper planes that define decision boundaries between a set of objects that belong to different classes. A schematic example is shown in Figure 10.

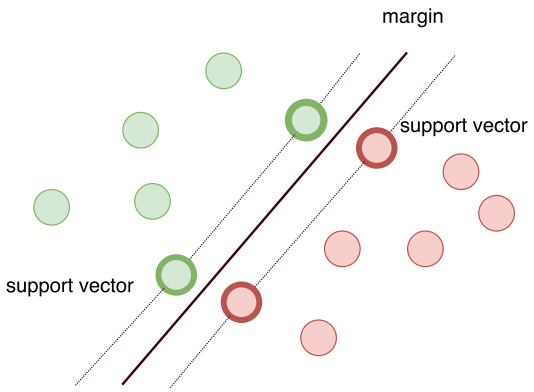

Figure 10: Linear SVM demonstration

In this two dimensional example, the objects belong either to class green or red. The separating line defines a boundary that divides the green objects from the red ones. Any

[6] https://spacy.io/docs/usage/pos-tagging#pos-tagging-german

new object falling to the bottom right is labeled (classified) as red. If it falls to the top left it will be classified as green.

The classifier shown in Figure 10 is the basic example of a linear SVM. However, sometimes a trade off between the margin and the number of mistakes on the training data has to be made to get the best results.

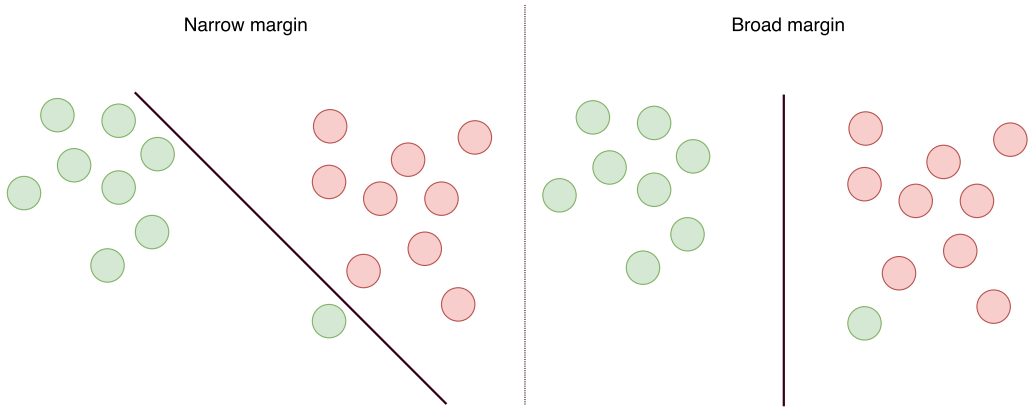

Figure 11: Trade off between the margin and the number of mistakes on the training data

Figure 11 shows, that in a hard margin solution such as on the left side, the points can be linearly separated but there is a very narrow margin which might result in more test data getting misclassified. Possibly the large margin solution on the right is better. Even though one training input is not correctly separated, allowing the violation of a constraint might result in better test results, due to the broader margin. The standard approach is to allow the decision margin to make a few mistakes (some points such as outliers and noisy examples are inside or on the wrong side of the line) as demonstrated on the right side of Figure 11. In such a *soft-margin SVM*, we can state a preference for margins that classify the training data correctly, but soften the constraints to allow for some training mistakes. Here, C is the regularization parameter. A small C allows constraints to be easily ignored, resulting in a large margin. A large C makes constraints hard to ignore, meaning a narrow margin. A C close to ∞ enforces all constraints, meaning a hard margin. We will be testing out a number of different values for C and evaluate our model with the optimal margin.

Some classification tasks however can not be solved with such a linear method. More complex structures are needed to make the best separation of classes for all samples (training data).

On the left side of Figure 12, we see a classification problem which is not linearly separable. The Figure as a whole shows the *kernel trick* used in Support Vector Machines. Here, we see the original training data input being mapped using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as transformation. Note that in this new setting, the transformed objects are tried to be made linearly separable and thus, after applying the kernel trick, we do not have to construct a complex curve. All we have to do now is to find an optimal line that can separate the green and the red objects.

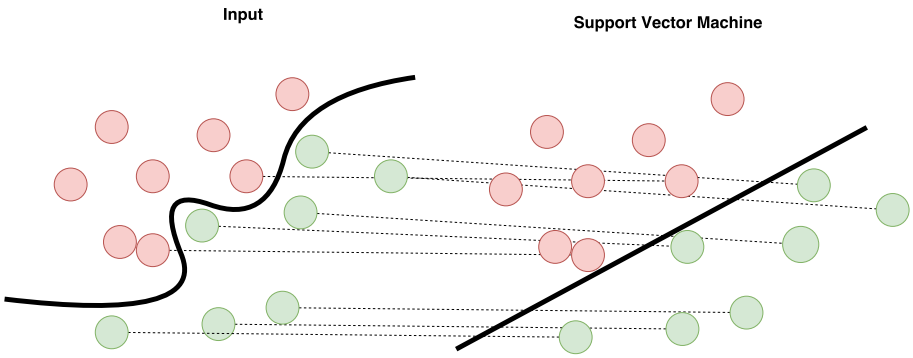

Figure 12: Support Vector Machine Transformation

The classic example for classification with a Support Vector Machine (SVM) is the SVM with a Radial Basis Function (RBF) kernel. The RBF-kernel measures similarities between two points with the Gaussian Function. When two inputs are close to each other the kernel returns a value near 1 and if they are far from each other a value near 0 will be returned.

The RBF-kernel comes with the additional γ parameter. The behavior of the kernel function is deeply connected to the γ. If γ is chosen too high, the radius of influence of the support vectors will be so small that it only includes one input node, and hence will never be able to correctly classify the input. If γ is too small the radius will reach over the whole training set. In this case, two points can be considered similar even if they are far from each other and a classification is not possible anymore.

Our task will be to balance the parameter of the RBF-kernel and the soft-margin SVM so that the model will return optimal classification results. A wrong choice of the parameters can result in the model being too specialized towards the training data, and thus will fail at predicting the test data. This phenomenon is called *overfitting*.

Since our problem is a problem of text classification and the amount of features will be relatively high compared to other classification tasks, it may not be necessary to map our data into higher dimensions as an RBF-kernel would do, since it should not be able to increase performance of the model [HCL03]. To test this hypothesis, we will additionally test out model with a linear kernel which does not map the data into hyperplanes resulting in generally way faster training and test speed when having a huge number of features (which is true for our case). Since a linear kernel is a degenerate version of the RBF-kernel, the performance of the model should not increase [CP17]. More importantly however, the time for training and testing our data will be reduced drastically for higher number of data [CP17], which is why it is interesting to take a look at both of the kernels and compare them. If the performance does not differ at all or just slightly, it is a huge win for the linear kernel because of the much faster running time.

4.1.4.2 Nearest Neighbors Neighbor-based classification is conceptually a different kind of classification which is why it is also interesting to take a look at. Nearest Neighbors is an instance-based learning algorithm where there is no internal model which tries

to learn from the training data. Instead, every training instance is simply stored.

To demonstrate k-nearest neighbors classification, let us consider the task of classifying a new object (test set) among a number of known samples (training set).

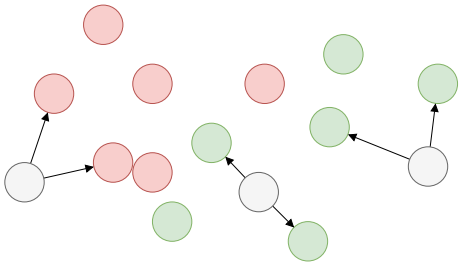

Figure 13: Nearest Neighbor Classification

The classification process as explained in Figure 13 is made in sense of a majority vote, which in this example is $k = 2$. To classify a new node from the test set, the two nearest neighbor's classes are considered. Based on those, the new node will get a class assigned.

For our classification problem, we will vary the parameter k which defines how many neighbors are considered for classification to find out the best results. If we choose our k to low, there is the danger that noise in our training data results in bad performance. If k is too high, points with high distance to the new node will be considered for classification even though they should not. This happens often if there is a small training set or the training data is not equally distributed, e.g. x has three neighbors, 1 is close, 2 are very far, and $k = 3$.

4.1.4.3 Random Forest A Random Forest is an ensemble of uncorrelated Decision Trees [Bre01]. Those trees consist of various sub-samples of the dataset and use averaging to improve the predictive accuracy and control overfitting. A classification will then be made in sense of a majority vote from the forest as shown in Figure 14 below.

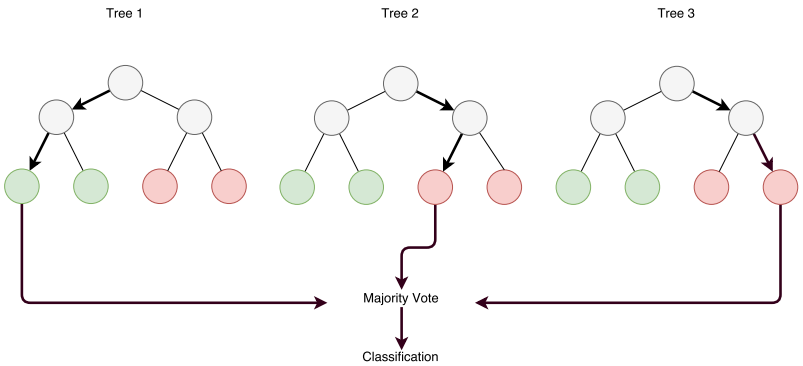

Figure 14: Demonstration of a Forest Classification

When making a classification, each tree in the ensemble will perform a classification and

each trees prediction will be considered for the majority vote from the forest. Since two of the three trees voted for red in our example, the new sample will be classified as red as well. Each Decision Tree in the forest grows in the training phase with a certain degree of randomness. A huge benefit from using such an ensemble of decision trees is that overfitting, the drawback which a single decision tree has, will not occur [Bre01].

In random forests, the number of estimators (trees) can be varied. Moreover, the depth for all trees in the forest can be set to fixed amounts or go infinitely deep. Additionally, the minimum number of samples required to split an internal node can be varied as well as the minimum number of samples required for a node to be a leaf node. We will try out various combinations of those parameters to make sure we use the best one for our data set.

4.1.5 Grid Search

To test out the right parameters, the training of a classifier with certain parameters and the following evaluation of those parameters on the same set of data is a methodical error. A model would have a very high performance since it already knows the exact data, hence overfitting will occur. Therefore, we have to use validation.

4.1.5.1 Cross Validation To prevent overfitting, it is common to separate the training set into two smaller subsets, a training and a validation set. One would then proceed and train with the subset of training data and make a first evaluation of the model with help of the validation set. If this process is a success, one can then start the final evaluation with the test data. A disadvantage of this approach is that we reduce the amount of data that we can use to train our model. Since we do not want to "waste" any of our data and the results you get can be extremely reliant on the part of the training set you chose for learning we will use *cross validation* to resolve this problem instead. Here, the test data will still be kept for the final evaluation but the validation set is no longer needed when using cross validation. In k-fold cross validation, the training set is cut in k equal parts (folds) and the following process will be done for each of the folds: First, the model is trained with $k - 1$ folds. Then, the remaining fold is used to validate the model. This is repeated for each combination of folds. Finally, we take all results from the cross validation and compute the average across those results. This process is illustrated in Figure 15. This process is rather expensive but it does not "waste" any of the data we collected as a validation set would do, which is a huge advantage.

4.1.5.2 Tuning hyper-parameters To find the right non-learnable parameter of each classifier, the so-called *hyper-parameters*, we use threefold cross validation on the training set as in Figure 15. Different models require different hyper-parameters, some simple algorithms even require none. In our case however, we repeat the tuning of hyper-parameters for each of the training set and subsets and for each feature. After testing out a multitude of parameters for each classifier our grid search will return us the best parameters from the cross validation.

However, since the RBF-kernel computing time is extremely high with the number of

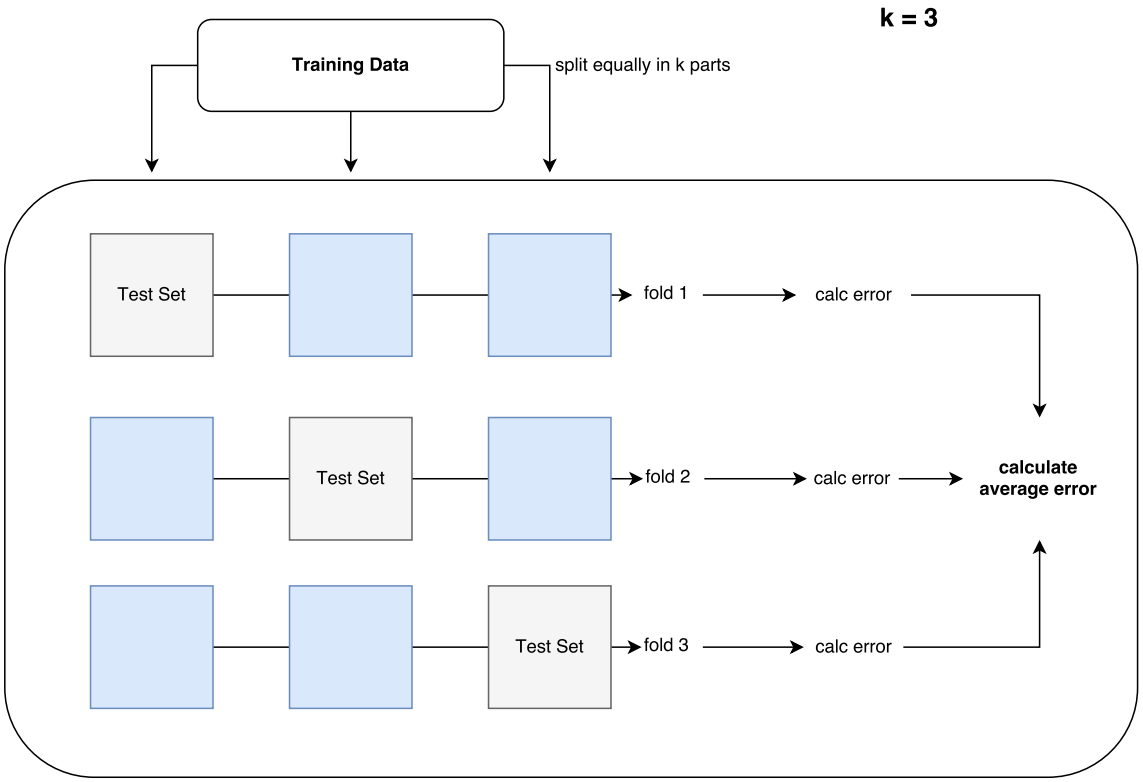

Figure 15: Cross Validation Example

features that we have, we will further cut the slowest running parameter combinations, since they increase runtime of our grid search by a huge amount. We even had to drop the grid search for the biggest data sets, because the running time was simply to high (around 1 month run time on our available machine). We will use the parameters which worked best for the biggest possible subset of training data for the higher ones.

Finally, we will pass those hyper-parameters to the classifier and start the final evaluation on the test set.

4.2 Classification

Let us now switch over to the results for our main hypotheses. This subsection presents, analyzes and interprets all of the results from our classification experiment.

As explained in Section 4.1.2, we will split this analysis into two parts. First, we will analyze the results from classifying the data with five labels. Afterwards, we will take a look at the results from classifying with only three labels (Frankfurter Allgemeine, Spiegel Online, Die Zeit) and compare the results for the different amount of labels.

4.2.1 Five label classification

This section covers all results for classifying with five labels. We will first take a look at the results for the smallest subset of the training data, and then compare it to the next higher one. At the end we will look at the increase in performance over the growing data set. A look at the confusion matrix for particularly interesting examples will also be included.

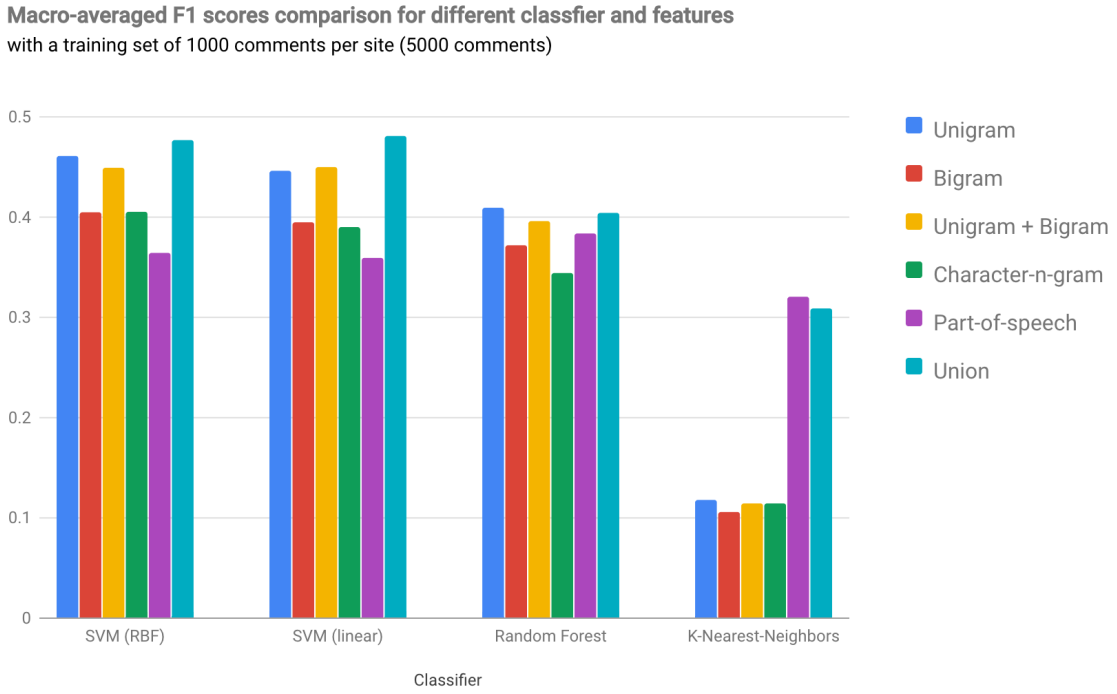

Figure 16: Training with 5000 comments - Five sites

Figure 16 shows results for the experiment with the smallest subset of data: 1000 comments per site. Each bar represents one of the features which we did the experiment with. The F_1 score for each feature was computed with three-fold cross validation and grid search. Since our training data is very small here, the results are subjective to training errors. The best results were achieved by combining all features with the linear SVM classifier, followed by the RBF-SVM. The linear SVM was marginally better here (0.01). If we now closely compare the linear SVM to the RBF-SVM, it is pretty obvious that they almost got the same results. However, this was to be expected [HCL03]. Both classifiers worked best with union, followed closely by the combination of unigrams and bigrams. Bigrams and character-n-grams with $n = 4$ performed worse, with the part-of-speech feature performing the worst for both classifiers. When looking at the Random Forest, the results do not vary as much as for the SVM's. Here, feature union and unigrams performed equally well, but worse than the SVM's. They are then followed closely by the combination of unigrams and bigrams. Note that for Random Forest, the part-of-speech feature performed better than bigrams, with character-n-grams performing the worst. The K-Nearest-Neighbor classification however had the worst results. Unigrams,

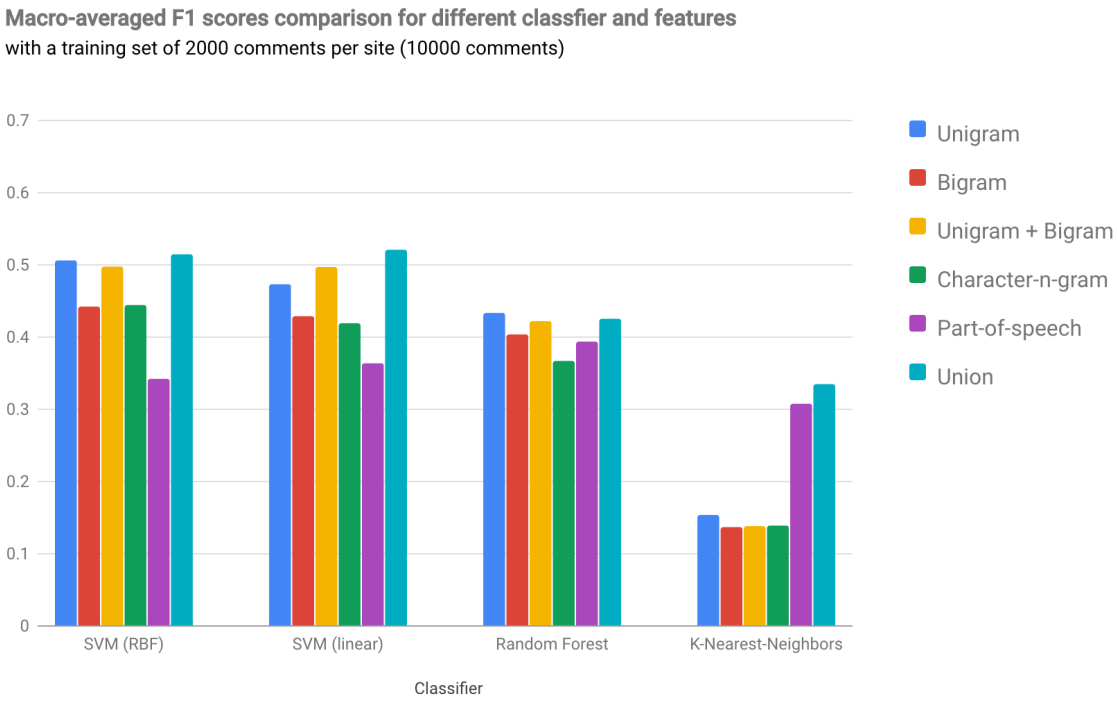

Figure 17: Training with 10000 comments - Five sites

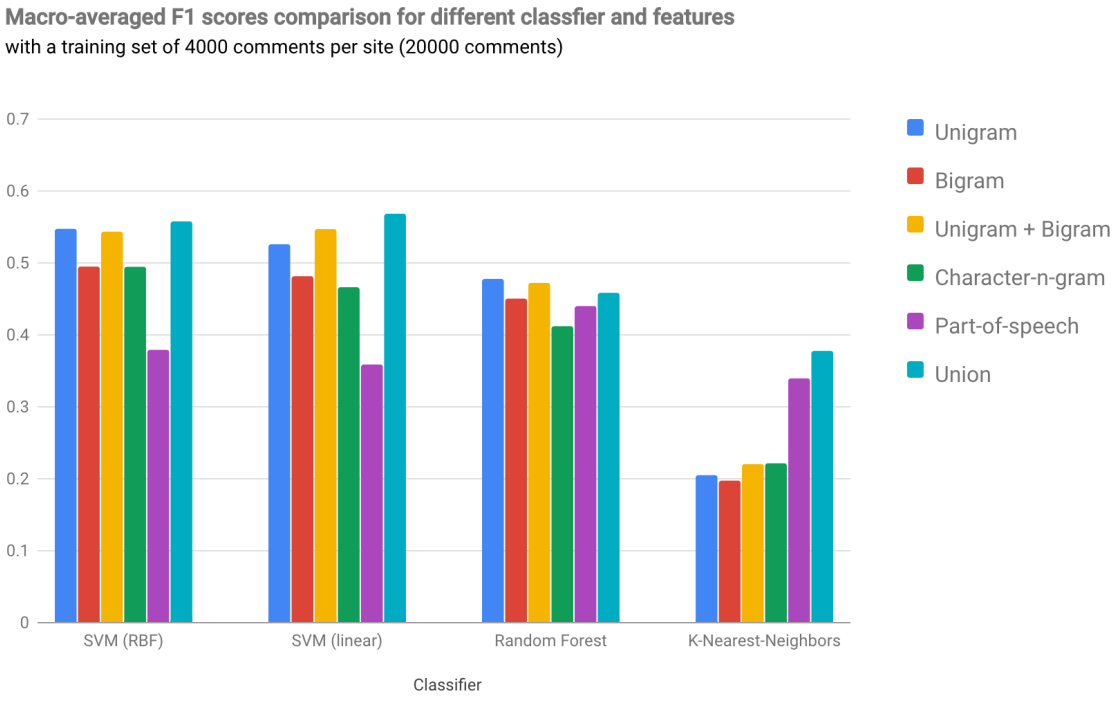

Figure 18: Training with 20000 comments - Five sites

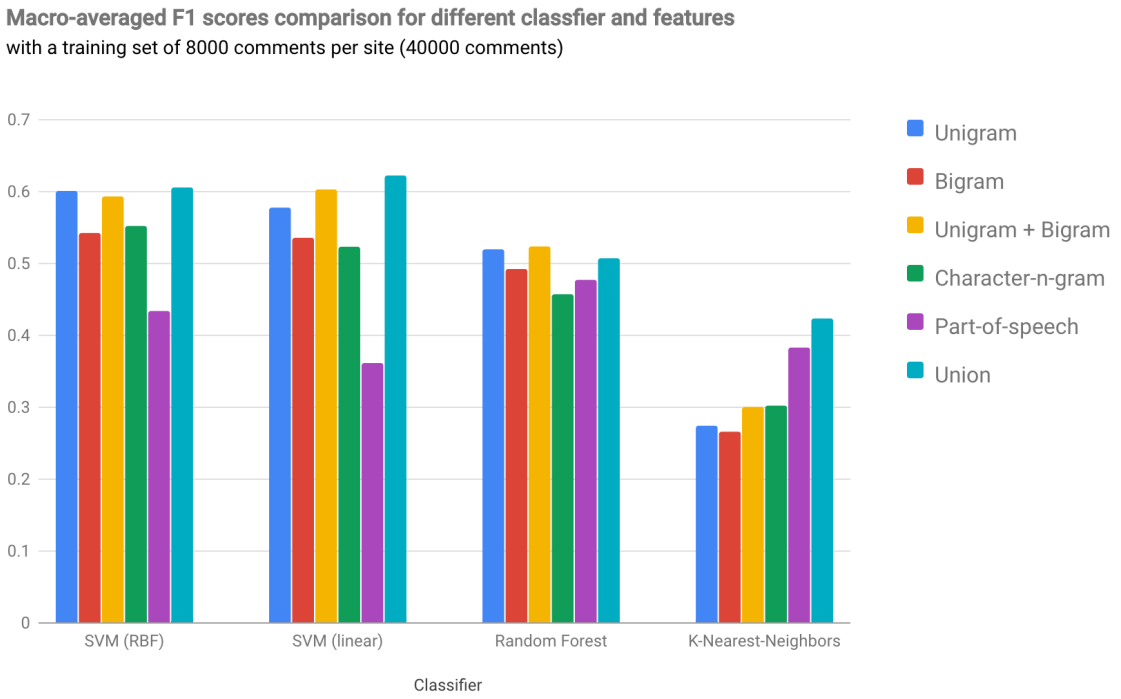

Figure 19: Training with 40000 comments - Five sites

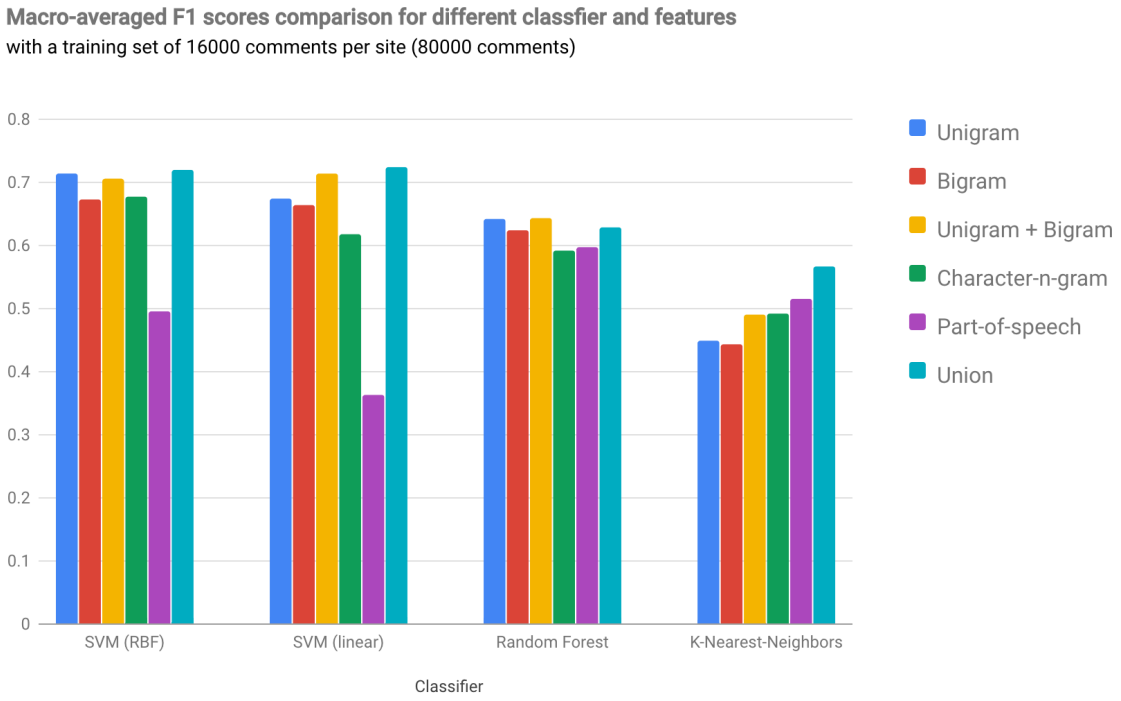

Figure 20: Training with 80000 comments - Five sites

bigrams, unigrams + bigrams, and character-n-grams were all equally bad.

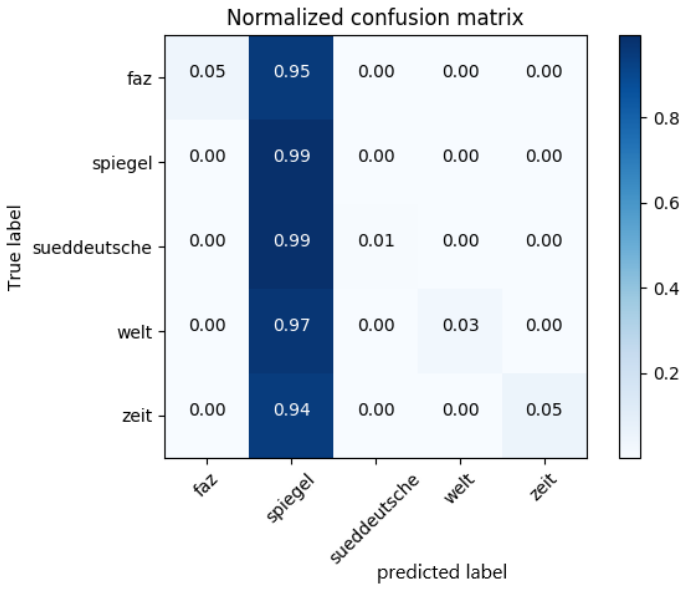

Figure 21: Confusion matrix KNN with unigram using 5000 comments

The confusion matrix for KNN using unigrams is shown in detail in Figure 21. Here, almost all inputs were labeled as Spiegel Online when using unigrams. The other three bad performing features show similar results, the classifier seems to choose a random label and simply labels almost all inputs to it. The low F_1 score is a consequence of this.

However, when using part-of-speech or union as our feature, K-Nearest-Neighbors gets a huge boost in performance. The results are not up to par with the other classifiers, but K-Nearest-Neighbors now does not classify all inputs to one label anymore. It seems like the part-of-speech feature enables this. Since union also includes the part-of-speech feature, it also has (relatively) good performance.

Next, let us compare these observations to results obtained when doubling the training set size. Figure 17 shows the results for our next bigger subset of data. Now that we use 10000 comments for training, let us see how much the performance changes.

First, let us compare the previously best performing classifier and feature to the new best one. The doubling of our training set does only increase performance for the linear SVM by 0.04 points, an increase of the F_1 score by 8 percent. The rest of the results shows a similar increase. The performance for almost all classifiers simply increases by roughly 8 percent. K-Nearest-Neighbors however does not get an increase. Instead, performance drops a little for the part-of-speech feature. At first glance, more data does not seem to bring a boost in performance for KNN.

Next up, we will double the training set again and take a look at the results. Now having 20000 comments as training data, the results are still quite similar as can be seen in Figure 18. The best performing feature (union) on the linear SVM does yet again get a rather small increase in score by 0.05 points (9.6 percent), but the gap between the best perform-

ing feature and the worst one seems to widen. Part-of-speech does still have a rather bad performance for both SVM's with roughly 0.36 for both. If we compare that to the results from our experiment with 1000 comments per site, we see that the performance in fact did not increase at all. With Random Forest however, the performance for all features increases steadily. K-Nearest-Neighbors also gets a boost in performance when using 20000 comments as training data, while the poor results for most features still remain.

When doubling the training set yet again to 40000, things get more interesting. Here, runtime of our classifiers started to vary extremely. As mentioned in Section 4.1.4.1, the RBF-SVM's run time can get quite high with a huge number of features. This started to show in this experiment. When we look at the numbers in Figure 19, performance of a linear SVM and an RBF-SVM are still quite similar. However, while the linear SVM's runtime was around one minute for feature union, RBF took almost four hours to complete. Bear in mind that we skipped grid search because of this for the RBF-SVM. The performance increase is still quite small. Our best performing feature with a linear SVM did yet again only increase by 0.05 (8.7 percent), which seems to be the case across all classifiers. The same gaps between features are still visible. However, note that the gap between part-of-speech and the rest of the features got even bigger for the linear SVM, while RBF-SVM starts to have noticeably better performance than linear SVM (with part-of-speech). Interestingly, K-Nearest-Neighbors starts to work better with this amount of data. Performance is still pretty bad for the four features, but instead of only labeling one class, K-Nearest-Neighbors more and more starts to label the correct ones. It still labels most inputs to one class though, as illustrated in the confusion matrix in Figure 28 in the appendix.

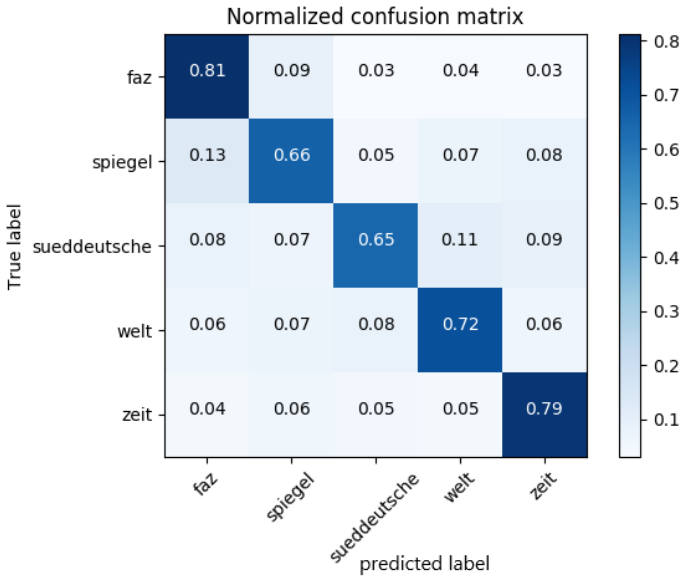

Figure 22: Confusion matrix linear SVM with union using 80000 comments

Finishing with our biggest training set, one would assume best performance here. This is exactly the case, as illustrated in Figure 20. Best performance was yet again achieved

by using either linear SVM or RBF-SVM. Both show very similar performance, with a top score for our whole experiment for five sites of 0.72. Note that here the linear SVM took around two and a half minutes for running the experiment with feature union, while the RBF-SVM ran more than sixteen hours. While linear SVM still performs badly with part-of-speech, RBF-SVM seems to handle it a little better. Interesting to note is that the gap between the classifiers has gotten smaller. With the best performing feature of Random Forest reaching a score of 0.64, followed by K-Nearest-Neighbors with a score of 0.57. More interestingly, K-Nearest-Neighbors seems to start working relatively well with the previously bad performing features. Let us investigate this by looking at KNN's confusion matrix in Figure 23.

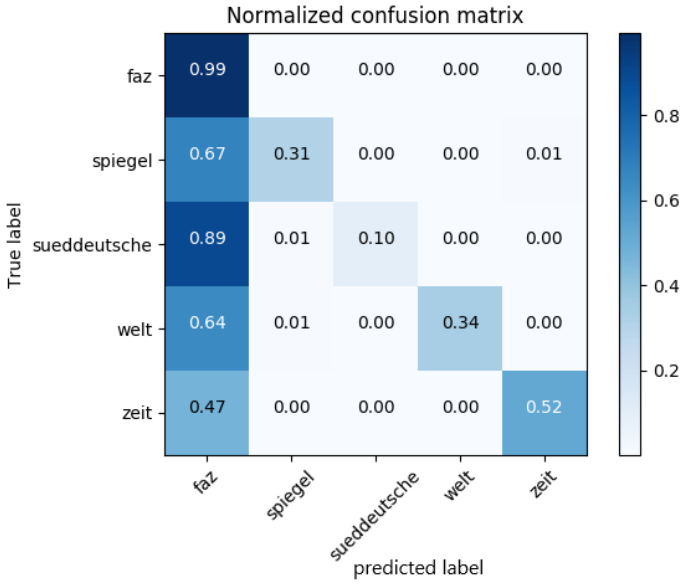

Figure 23: Confusion matrix KNN with unigram using 80000 comments

Even though the F_1 score does now reach around 0.42 points, performance is still not good. The classifier reaches a relatively high score by labeling almost all classes to a single label and correctly labeling a smaller fraction of inputs. When we compare that to confusion matrix from the best performing feature (Figure 29 in the appendix), it proves that performance is indeed still bad.

In the figure we can also see that the classification performance is way better for feature union. However, the F_1 scores are not that far apart. In this case, the F_1 score is not that good of a performance indicator. Lastly, let us look at the confusion matrix for feature union with linear SVM. Figure 22 shows the performance of the classifier at its best in detail. Frankfurter Allgemeine and Spiegel Online were correctly labeled the most number of times, with roughly 80 percent correctly labeled samples. Die Welt follows with 72 percent of correctly labeled samples. Spiegel and Süddeutsche make last place with 66 and 65 percent correctly labeled samples. Note that Spiegel Online and Frankfurter Allgemeine are getting confused the most (in both ways). Other than this, the other labeling errors seem to happen at random, since the score of wrong labeled samples is

roughly equal for all labels. This observation can be made across all results of the subsets of training data.

To better illustrate the growth of performance over the different amount of training data, let us take a look at Figure 24, which shows the performance across our experiments for each classifier when using feature union.

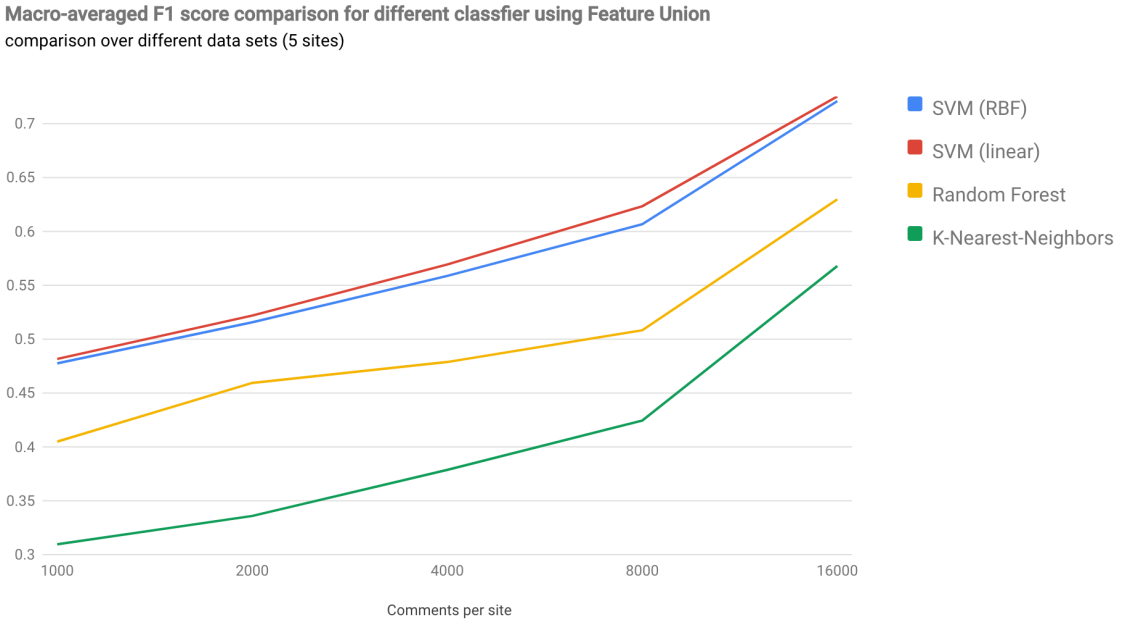

Figure 24: Score progression of different training sets with five labels - Feature Union

Here, we can see the gradually increasing macro-averaged F_1 score across all of the training sets. Note that each step in the graph is doubling the training set. K-Nearest-Neighbors last bump in performance from 8000 to 16000 leads to believe that its performance scales best with larger amounts of data. Unfortunately, we can not further test out this hypothesis due to data set constraints. However, for our evaluation, this hypothesis is true. The results for feature union are similar to the results of the other features, with feature union almost always giving the best performance, albeit only with a small lead over the other features.

More interestingly, if we look at the progression of performance of the part-of-speech feature (Figure 25), we find that the part-of-speech feature seems to be working best with Random Forest and K-Nearest-Neighbors. Note that performance does not gradually increase here as much, it even decreases at 2000 comments per site for RBF-SVM and KNN. The score of the Random Forest classifier with part-of-speech is only 0.04 points lower than for feature union (same for K-Nearest-Neighbors), in contrast to the 0.23 and 0.36 drop in score for RBF and linear SVM respectively when compared to feature union. Note that K-Nearest-Neighbors seems to scale best with larger amounts of data. On another note, linear SVM does not work well with our part-of-speech feature at all.

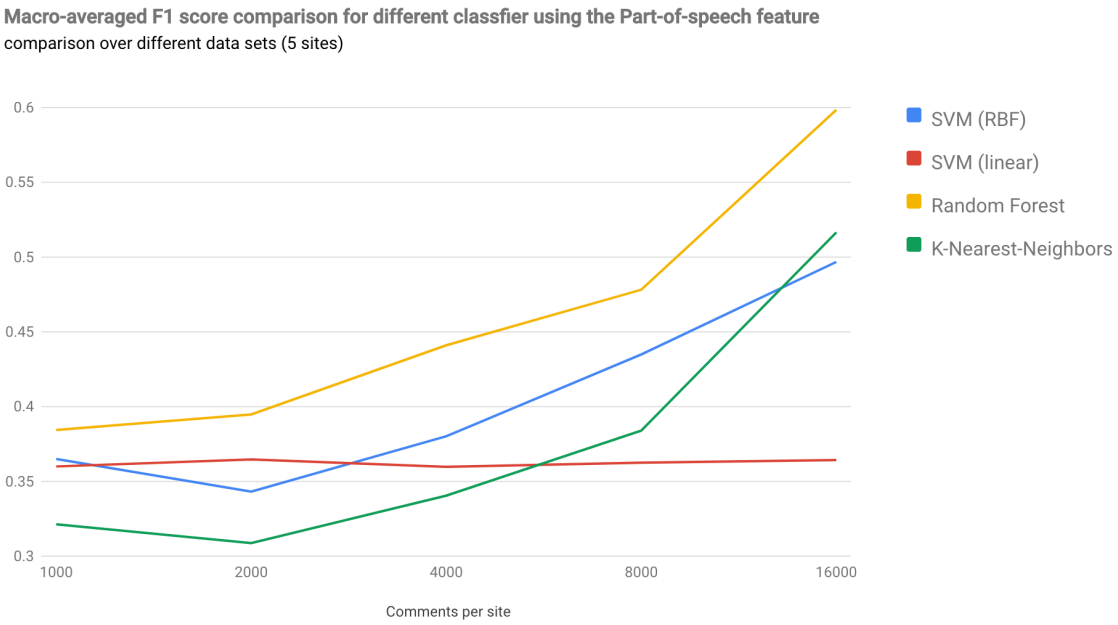

Figure 25: Score progression of different training sets with five labels - Part-of-speech

4.2.2 Three label classification

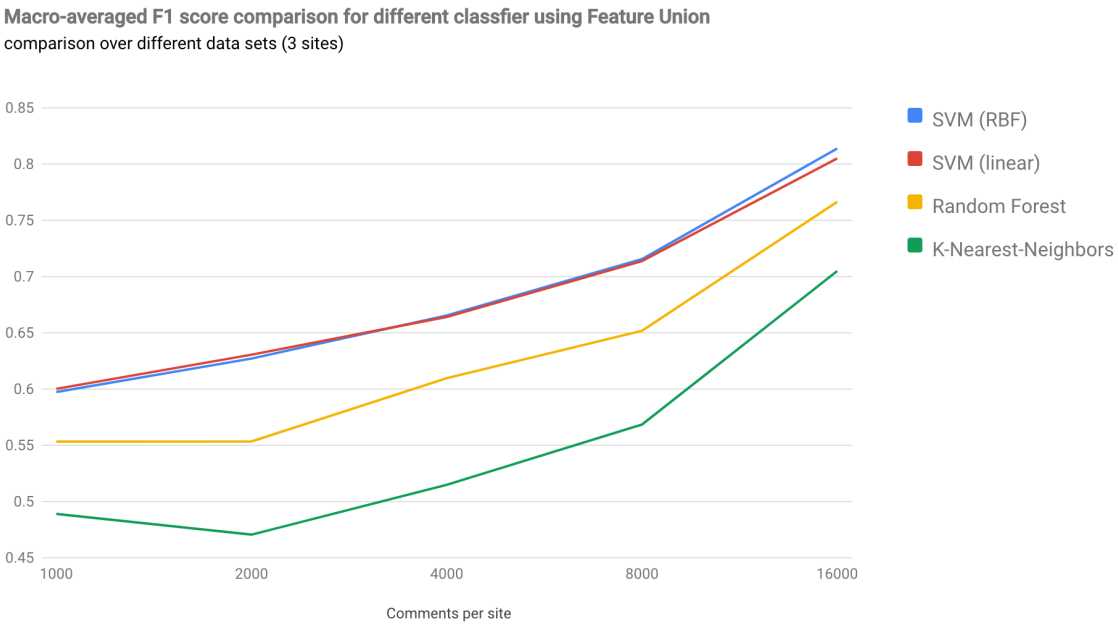

Figure 26: Score progression of different training sets with three labels - Feature Union

To avoid getting repetitive, we will not repeat the same structure as in the above section.

Instead, we will only take a look at the growth over the different data sets with feature union and part-of-speech and compare them to the results from Section 4.2.1. We will take a more detailed look at the more interesting sections of the analysis though.

Figure 26 shows the results for feature union. Compared to the results from Figure 24, results do not change too much. Both SVM classifier still get roughly the same results, with a top F_1 score of 0.81. Random Forest will now have a top score of 0.76. K-Nearest-Neighbors will still make last place with a score of 0.71. The gradually increase when doubling the training set is very similar to Figure 24. However, K-Nearest-Neighbors has worst performance when training with a 10000 comment data set, different than when classifying with five labels as shown in Figure 24.

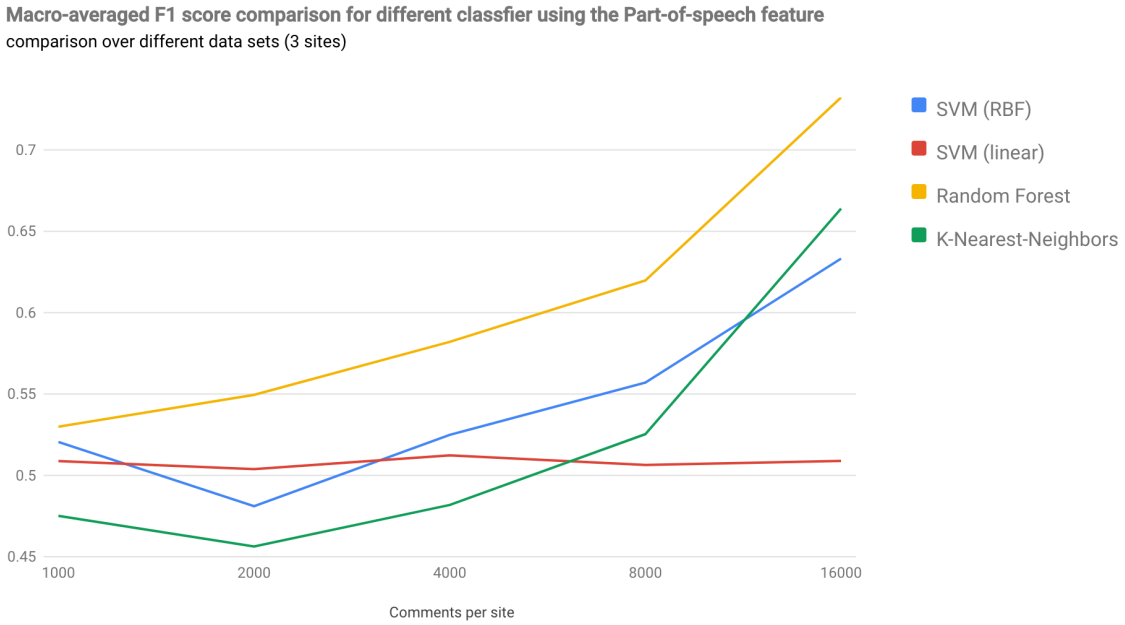

Figure 27: Score progression of different Training sets with three labels - Part-of-speech

In Figure 4.2.2, we have the results for the part-of-speech feature when labeling three classes. The results are very similar to the results described in Figure 25. Random Forest is the top scorer with a macro-averaged F_1 score of 0.73, only 0.03 points lower than feature union. For KNN, the top score is 0.66, dropping by only 0.04 points in respect to the results of feature union.

4.3 Summary

The best results for both machine learning experiments were achieved by using either a linear or an RBF-kernel SVM with feature union. When investigating results for SVMs however, part-of-speech was the worst feature by a big margin, even more so for linear SVM. Unigrams, bigrams, unigrams + bigrams, character-n-grams, and feature union always had quite similar results across all training sets, with feature union always having

a small lead. When considering the runtime for the SVM classifier however, we have to note that RBF took roughly 400 times as long on our biggest data set to finish for the biggest feature (feature union with five sites), making a grid search almost impossible (on our available machine) since the runtime would have been around one month. For training set sizes smaller than our 8000 comments per site experiment, the runtime is not as big of a concern. Part-of-speech works particularly well with Random Forest and KNN, even though feature union always worked best. Note that character-n-grams did not work as well as the other features (only a few points decrease) for all classifiers. Random Forest was the most balanced out classifier without huge spikes in performance. The classifier performed equally well across all training sets and showed a gradual increase in performance when increasing the training set.

5 Conclusion and Future Work

Let us now conclude the thesis by summarizing the results of our experiments.

By analyzing our data set, we found that sites have a number of distinguishable characteristics in the statistics which we analyzed. When revisiting the structure of respective sites comment and login sections, we found that the most striking statistics can be explained by investigating associated features of the sites. Note that the news sites where a Facebook login was possible were more active and had generally less words per comment.

Our machine learning experiment has indeed shown that the linear SVM is better suited for a huge number of features, as our hypothesis at the end of Section 4.1.4.1 suggested. Since using an RBF-kernel increases runtime of the experiment by a huge margin while not significantly improving the classification performance, one could argue that it is not suitable for our experiment. This is especially true when considering that our linear SVM had the same results at almost all times, while runtime was incredibly low. The results for a linear SVM were better than for RBF at times, which should not be possible in theory [CP17]. However, those results were only minimally better, which is within margin of error. Since SVM had the best results, it leads to believe that for text classification, SVM is the best choice. However, when we solely look at part-of-speech results, they show that with smaller feature vectors other classifier might be better suited, like Random Forest and K-Nearest-Neighbors.

In our future work, one could try to scale up the dataset and look for the threshold of when our classifiers are saturated with training data for our experiment. Since our classifiers show a gradual increase in performance for more data, it would be interesting to investigate how well our classifiers scale and when this saturation is reached. This was not possible in the scope of this thesis, especially considering the large amount of time necessary to process the data on our available machine. Also, one could try to test out this experiment with comments, which did not even exist when crawling our data sets, and compare test results to the observations which we made in this thesis. When considering that news sites have many different categories of articles, it would also be interesting to redo our experiment, but this time using categories as labels and look how well classifiers can separate categories instead of news sites. For this experiment however, one has to repeat the scraping and preprocessing process as well. Another interesting experiment would be an investigation where one would try to predict the number of upvotes for a comment on a given site using machine learning techniques. One could also try to add features which increase performance of the classifiers to the experiment.

A Appendix

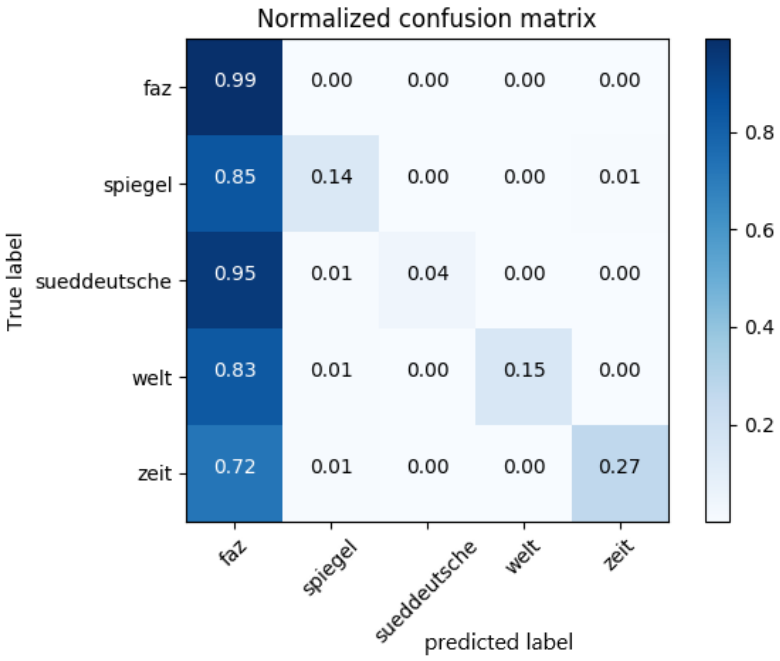

Figure 28: Confusion matrix KNN with unigram using 40000 comments

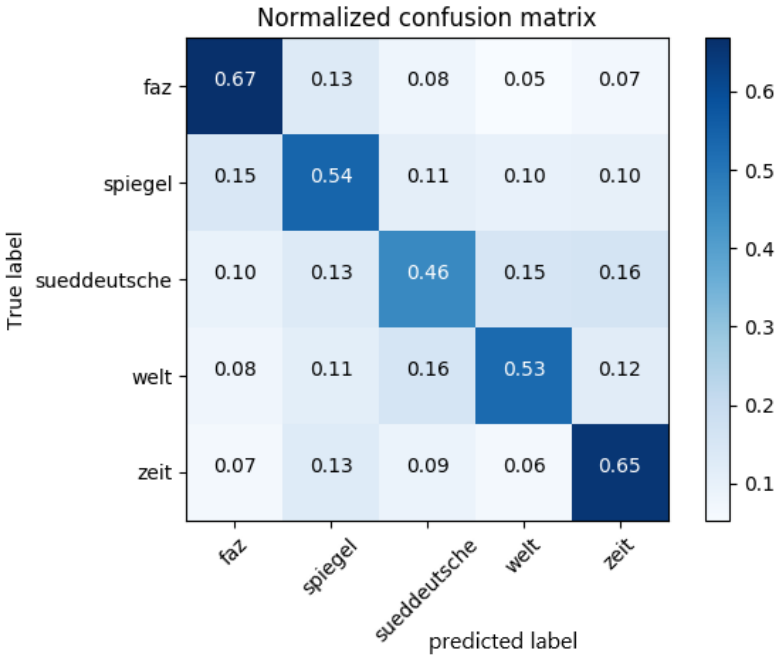

Figure 29: Confusion matrix KNN with union using 80000 comments

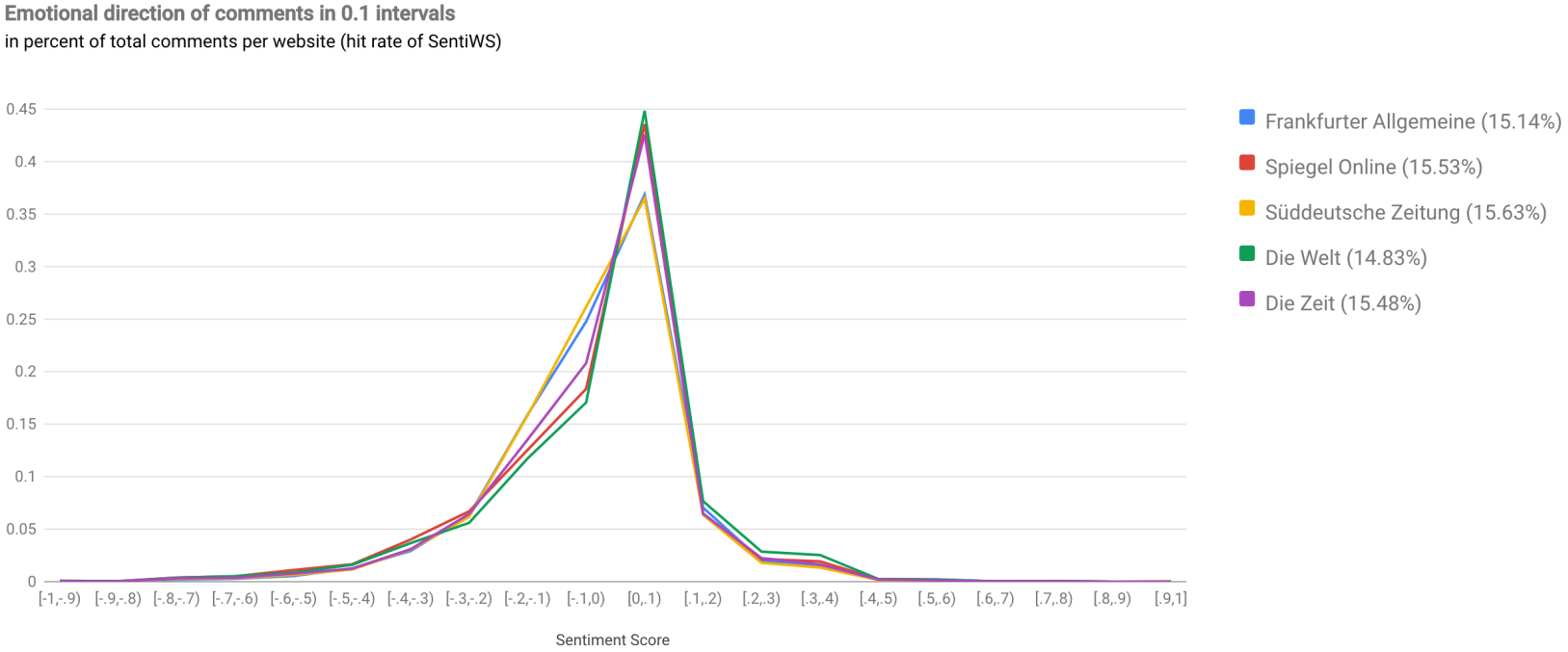

Figure 30: Sentiments in 0.1 intervals for each site

References

[Bre01] BREIMAN, Leo: Random Forests. In: *Machine Learning* 45 (2001), Oct, Nr. 1, S. 5–32

[CP17] CHOUBEY, Dilip K. ; PAUL, Sanchita: GA_SVM: A Classification System for Diagnosis of Diabetes. In: SHANDILYA, Shishir K. (Hrsg.) ; SHANDILYA, Smita (Hrsg.) ; DEEP, Kusum (Hrsg.) ; NAGAR, Atulya K. (Hrsg.): *Handbook of Research on Soft Computing and Nature-Inspired Algorithms*. IGI Global, 03 2017 (Series: Advances in Computational Intelligence and Robotics), Kapitel 12, S. 359–397

[CV95] CORTES, Corinna ; VAPNIK, Vladimir: Support-Vector Networks. In: *Machine Learning* 20 (1995), September, Nr. 3, S. 273–297

[HCL03] HSU, Chih-Wei ; CHANG, Chih-Chung ; LIN, Chih-Jen: A Practical Guide to Support Vector Classification. (2003), November

[PDM11] PETROV, Slav ; DAS, Dipanjan ; MCDONALD, Ryan T.: A Universal Part-of-Speech Tagset. In: *CoRR* abs/1104.2086 (2011)

[Pim16] PIMPL, Roland: *Auflagen von "Spiegel", "Welt" und "Bild" brechen zweistellig ein*. http://www.horizont.net/medien/nachrichten/IVW-I2016-Auflagen-von-Spiegel-Welt-und-Bild-brechen-zweistellig-ein-139882, April 20, 2016. – Accessed: 2017-09-19

[PVG+11] PEDREGOSA, F. ; VAROQUAUX, G. ; GRAMFORT, A. ; MICHEL, V. ; THIRION, B. ; GRISEL, O. ; BLONDEL, M. ; PRETTENHOFER, P. ; WEISS, R. ; DUBOURG, V. ; VANDERPLAS, J. ; PASSOS, A. ; COURNAPEAU, D. ; BRUCHER, M. ; PERROT, M. ; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[Rei16] *REICHWEITE DER ZEITUNGEN STEIGT AUF 86,3 PROZENT*. http://www.zmg.de/presse/presse-artikel/reichweite-der-zeitungen-steigt-auf-863-prozent/, Oktober 19th, 2016. – Accessed: 2017-09-16

[RQH10] REMUS, R. ; QUASTHOFF, U. ; HEYER, G.: SentiWS – a Publicly Available German-language Resource for Sentiment Analysis. In: *Proceedings of the 7th International Language Resources and Evaluation (LREC'10)*, 2010

[San14] SANTANA, Arthur D.: Virtuous or Vitriolic. In: *Journalism Practice* 8 (2014), Nr. 1, S. 18–33

[Sch13] SCHNIBBEN, Cordt: *Newspaper Crisis Hits Germany*. http://www.spiegel.de/international/germany/circulation-declines-hit-german-papers-a-decade-after-america-a-915574.html, August 13, 2013. – Accessed: 2017-09-19

[Zei16] *Zeitraum für Medien: Einzelmonat Juli 2016*. https://www.agof.de/download/Downloads_digital_facts/Downloads_Digital_Facts_2016/Downloads_Digital_Facts_2016-07/07-2016_df_Ranking_Gesamtangebote_Digital_14+.pdf?x87612, July 2016. – Accessed: 2017-09-08

List of Figures

List of Tables