Capstone Project
Sentiment Analysis for German Twitter

Ariel Brandes

August 20th, 2020

1 Definition

This first chapter of the project report aims to give an overview over the whole project: Which problem do we plan to solve? We will discuss the problem on a high level view and introduce the data we are working on. Subsequently we will define the metrics which are used as evaluation measures for the performance of our solution.

Project Overview

With the advanced technical capabilities and with a huge number of documents from social media which are publicly available the interest in automatic sentiment analysis has increased in the past years. This interest stems from both academia and industry and results in many scientific initiatives and competitions at Kaggle. Indeed, the opinion on companies in social media is crucial for their success. Shitstorms arise frequently can have a huge impact. This makes it necessary for companies to monitor this sentiment on social media platforms to be able to react as fast as possible. To be able to extract sentiments from social media like Twitter, well-trained machine learning models are needed.
In general, sentiment analysis is classic problem in the area of natural language processing (NLP). Doing sentiment analysis on platforms like Twitter has, however, its very own challenges. People use emoticons, hashtags and hyperlinks in their tweets, which makes it more difficult to decide which elements of the text are actually useful for the analysis and which ones are noise. NLP is very specific for the language it analyzes. German, my mother tongue, is a more complex language than English is and it is not as widely used as English is. This is the reason that less scientists concentrate their studies on the German language and less data and fewer benchmarks are available. It also results in fewer pre-trained models, which makes it necessary to train self-made models more often.
For sentiment analysis it is essential to have a collection of texts which are labeled in advance. For many some widely used language like English, Arabic or Chinese there exists a lot of labeled training data for sentiment analysis. For other languages which are not so widely used the amount of publicly available labeled data is very limited. For German, the language to be analysed in this project, only very few corpora of labeled tweets exist. Four

corpora are worth mentioning: the DAI data set, which contains 1800 German tweets with tweet-level sentiments [2]; the large MGS corpus, containing 109,130 German tweets [3] and the PotTS corpus, which contains 7992 German tweets [4]. Unfortunately, the DAI corpus is too small for the training of machine learning mdoels for sentiment analysis, the MGS corpus has a very low inter-annotator agreement, indicating low-quality annotations, and the PotTS corpus is on phrase level rather than sentence level. These circumstances led the authors of [1] to the creation of the SB10K corpus, the fourth corpus worth mentioning. The corpus originally featured almost 10,000 tweets labeled with sentiment (positive, neutral, negative). The selection of tweets followed a sophisticated pattern. Since the majority of tweets in Twitter do not contain any opinion at all, selecting a random set of tweets for manual annotation would result in an unbalanced set, with mostly neutral tweets. So the authors found a way to find tweets with potentially different sentiments: For each tweet they counted the number of positive and negative polarity words in per tweet, using a German polarity clues lexicon.
The corpus is publicly available and hence a good source for projects and benchmarking. Since the authors were only allowed to provide a list of tweet ids, not all of the originally used tweets are still available. Nevertheless more than 6,500 tweets are still there which is enough for most machine learning models and therefore the best data source for this project.
We will tackle this problem with Support Vector Machines (SVM) which is the most commonly used machine learning model for sentiment analysis. To be able to work on the corpus with the machine learning model, some steps of preprocessing are necessary. While one part includes transforming each tweet into a vector of numbers, one part is about deciding which text elements contain information i.e. sentiments. We will introduce a Twitter specific routine for that and discuss each step.
Finally we will test and evaluate our machine learning model on the SB10K corpus. We will use standard measures and metrics for sentiment analysis as accuracy and F1 scores. These solution metrics will be compared to a benchmark working on the same SB10K corpus which is used in this project. This detailed comparison will also discuss the strength and weaknesses of the proposed solution.

Problem Statement

The problem we will work on in this project is to predict a sentiment from a clearly defined space of labels (positive, neutral, negative) for a text stemming from the micro blogging platform Twitter. This problem is referred to as sentiment analysis and is a classification problem. Since the data for this project is labeled, the problem is a supervised learning problem and the result is measurable and quantifiable. This problem can be tackled by either neural networks or classical machine learning models. The approach of this project will be the latter which makes it necessary to have a good routine for data preparation and preprocessing, since tweets usually contain a lot of noise and parts that are not text like emoticons. The model which will be used will have to make use of human language specific characteristics. This makes this problem part of the NLP problems.

Technically, to be able to process the data with a machine learning model, each tweet will be cut into its individual text elements. This is called tokenization. Subsequently this sequence of text elements will be encoded to a vector of numbers displaying which subset of text elements from the whole vocabulary used in all tweets is used. Having each tweet represented as a vector of this kind one the one hand and the corresponding sentiment labels on the other, the model can train which text elements are beneficial for which sentiment. Crucial before the transformation to the vector representation is a preprocessing which reduces noise from the tweet. It will also be necessary to bring words originating from one word stem back to the stem, to be able to treat them as one. This makes it easier for the model to delegate the same sentiment to words from the same word stem. Since we will use classification models, we have many options for choosing a classifier. The most commonly used approach for sentiment analysis are SVM so this will be the model of choice. The final goal is to predict a sentiment for a text originating from a tweet with a accuracy as high as possible.

Metrics

For evaluating a model for sentiment analysis, the most common metric used is the F1 score (also F-score or F-measure). This is a measure of a test's accuracy. The calculation base is the precision p and the recall r of the test. The precision p is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly. The recall, r, is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive. Having calculated these two metrics, the F1 score is the harmonic mean of the precision and recall. The best value of F1 is 1, which means perfect precision and recall. Since we have three sentiment labels (positive, neutral, negative), we can calculate $\mathrm{F1}_{pos}$, $\mathrm{F1}_{neutral}$ and $\mathrm{F1}_{neg}$, the F1 scores for each of the three sentiments. Very common in the field of sentiment analysis is the F1 score macro-averaged from the $\mathrm{F1}_{pos}$ and $\mathrm{F1}_{neg}$. This is also a metric used by the authors of [1] in their paper which will serve as a benchmark for the model of this project. Additionally, we will use the actual accuracy (correct predictions/all predictions) and a confusion matrix. In a confusion matrix each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the model commonly mislabels one label as another.

2 Analysis

This chapter of the report will firstly give insight to the data we are working on. How is the data distributed? What are specialties of the data? Subsequently we will discuss algorithms and techniques which can be used to tackle the problem and the data. After that a benchmark solution is introduced which works on the same corpus. The metrics and evaluation measures are elaborated.

Data Exploration

The SB10K corpus of German tweets we work on includes the tweet ids of 9,939 tweets. The selection of tweets followed a sophisticated procedure and can be found in [1]. The authors also published a python script for the download of tweets from Twitter via the Twitter api. After downloading the tweets, the data set we will work on has five columns in total: the tweet id, the sentiment, an md5 hash, alternative ids, and the tweet as a string. For our analysis the only attributes we will need is the actual text of the tweet and the corresponding sentiment as labeled by the creators. As mentioned before, not all tweets were still available to the time of the project creation. Tweets can be deleted or corresponding accounts could have been suspended. After filtering deleted tweets the data set still contains 6,539 labeled German tweets which should be sufficient for the purpose of this project.

Looking at the distribution of sentiments shows that there are 1,480 positive tweets, 4,078 neutral tweets, and 981 negative tweets as can be observed in Figure 1. This statistic shows, that the vast majority (approximately 60%) of tweets have neutral sentiment. This was intended by the publishers of the corpus, due to the fact that most tweets published on Twitter do not contain sentiments.

Example

"Aber wenigstens kommt #Supernatural heute mal wieder um 22 Uhr - ein schwacher Trost"

Translating to:
"But at least #Supernatural will be back at 10pm today - a small consolation"

Taking a closer look at individual tweets, reveals that many tweets contain, in contrast to e.g. news articles, many twitter specific text elements. People use many punctuation marks, emoticons, hyperlinks and mentions of other users. Also the differentiation between small and capital letters does not to be applied in many tweets. This is actually crucial in the German language, since all nouns are capitalized. The fact that sometimes multiple emoticons are written directly after another makes it also difficult for a machine to distinguish between emoticons and other punctuation marks.

Algorithms and Techniques

Neural networks especially CNN (Convolutional Neural Networks) and RNN (Recurrent Neural Networks) gain more and more popularity for sentiment analysis. Yet one of the most widely used algorithms is the classic machine learning model SVM. It is very similar to logistic regression in terms of how the optimize a loss function. It uses *kernel functions*, i.e. functions that transform a complex, nonlinear decision space to one that has higher dimensionality, so that an appropriate hyperplane separating the data points can be found. The classifier looks to maximize the distance of each data point from this hyperplane using *support vectors* that characterize each distance as a vector. Many natural language

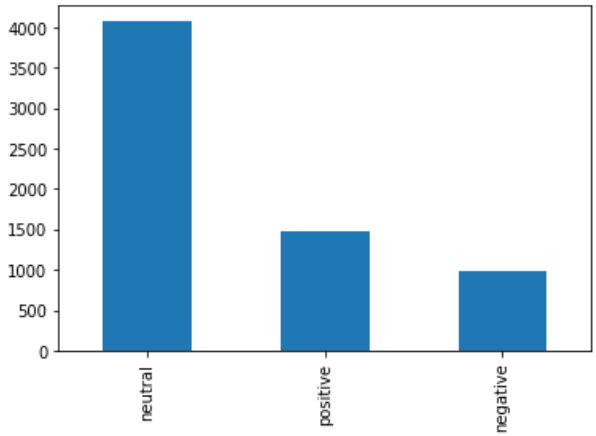

Figure 1: Distribution of sentiments

competitions concerning sentiment analysis have been won using SVMs, hence SVM is the classifier of choice for this project.

Benchmark

The creators of the SB10K corpus for German tweets created the corpus with the goal of doing sentiment analysis, too [1]. The main purpose was to check the performance of a CNN they created. To have a comparison for the performance of their neural network, they also performed sentiment analysis on the corpus using a SVM. The SVM was chosen because it is, as discussed before, the machine learning model of choice for sentiment analysis. This is very convenient, because it gives us a two benchmarks to compare the performance of our own sentiment analysis. Both, the CNN and the SVM, were trained and tested on the SB10K corpus. The results can be observed in Table 1.

Classifier	**Training Corpus**	**Test Corpus**	**F1**$_{pos}$	**F1**$_{neutral}$	**F1**$_{neg}$	**F1**
SVM	SB10K	SB10K	0.66	0.81	0.47	0.57
CNN	SB10K	SB10K	0.71	0.81	0.59	0.65

Table 1: Benchmarks for sentiments for German Twitter. The SVM were trained on fixed split of the corpus (90%), and then tested on the remaining texts. F1 score here is macro-averaged from F1$_{pos}$ and F1$_{pos}$.

It is important to mention that the authors of this benchmark used a corpus with more tweets then used in this project. This is due to the deletion of some tweets and discussed in a previous section. This makes the benchmark not as good as it could be, since the corpus we use is a subset of the corpus used for the benchmark. It is to expect that the loss of training data also goes in hand with a loss of performance.

3 Methodology

During the project, different machine learning classifiers were tested as well as different approaches for preprocessing the data. The changes made in the choice of classifier and in the preprocessing routine were always evaluated with a newly trained model. The metrics used for this were the macro F1 score. This empiric process gave a lot of insight which preprocessing steps were able to improve the result, which did not make a difference and which made the result even worse. This chapter will give an in depth description of the preprocessing and the implementation of the mode.

Data Preprocessing

Finding a good routine for data preprocessing was the most crucial step in this project. A couple of questions occurred and led the way for preprocessing. Which text elements contain information? Which do not? Do some similar text elements have similar sentiments can can be unified? The initial approach was a very basic one, which was then iteratively extended always with regard to the performance. It just split each tweet into the individual text elements using python's `string.split()`, converted each word to lowercase and processed each resulting list of text elements using NLTK's `TfidfVectorizer`. This is basically the same as using NLTK's `CountVectorizer` followed by `TfidfTransformer`. The former converts a collection of text documents to a matrix of token counts, while the latter transforms a count matrix to a normalized tf-idf representation. Tf-idf means *term-frequency times inverse document-frequency*. This is a common term weighting scheme in information retrieval, that has also found good use in document classification.

Subsequently the preprocessing routine was improved. The string split method was replaced by an explicit tokenizer for tweets provided by NLTK. The advantage of this tokenizer is that is recognizes Twitter specific text elements, e.g. the single punctuation marks of emoticons are not split. This makes a huge difference as emoticons obviously carry sentiments. Also hashtags are recognized. Another feature that was implemented and increased the performance was a stemmer. Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. The goal is to classify words not only for the exact expression, but also for the other possible forms of the words we used. A stemmer tries to bring each word back to its word stem. For this end the usage of German specific stemmers was implemented. Three stemmers were tested: `PorterStemmer, GermanStemmer` and `CiStem`. From which the latter provided the best results. The next step was to reduce noise by replacing and removing text elements of the tweets by hand using regular expressions. Since number per se do not carry sentiments, all numbers were removed if they were not part of an alphanumeric string. One challenge was to only remove actual numbers and not parts of emoticons which sometimes also include numbers. When people use consecutive periods ("...") the number of periods has shown to vary a lot. We assume that the sentiment is the same, whether someone uses 2 periods ("..") or 4 periods ("...."). That is why each sequence of consecutive

periods from 2 to n periods was unified to a sequence of three periods ("..."). Additionally, some Twitter specific preprocessing steps were applied. The "RT" standing for retweet at the beginning of some tweets was removed. It is assumed that it is not relevant for the tweet's sentiment whether it was a retweet or not. Also the "#" in front of hashtags was removed to be able to process the actual word of the hashtag. A final step involved unifying Twitter specific phenomena such as @-mentions and hyperlinks by replacing these entities with special tokens that represented their semantic classes. This means that all hyperlinks are treated as if they were the same. The same holds for all mentions of other users. One preprocessing step applies not to the tweet but to the labels: the sentiment labels were replaced with integers in the pattern: (positive:2), (neutral:1) and (negative:0).

This is a summarized list of the preprocessing steps of the final solution:

- Convert all strings to lower case
- Replace hyperlinks and mentions with token representing each class of text element
- Unify consecutive periods
- Remove number not being part of an alphanumeric string
- Remove "RT" for retweets and "#" from hashtags
- Tokenization with `TweetTokenizer`
- Stemming of words using (German specific) `CiStem`
- Using `TfidfVectorizer` for vocabulary and vector representation of tweets
- Replace the sentiment labels with integers

Implementation

After downloading the tweets of the SB10K corpus with a script provided of the creators of the corpus [1], all implementations were done in a notebook instance hosted on Amazon SageMaker, a cloud machine-learning platform. The first step of the implementation was the loading of the data from the files sotring the tweets. The next step was to get insight to the data, which included taking a look at samples an visualizing the distribution of sentiments as to be seen in Figure 1. After removing deleted tweets, the data was split into tarining and test data with 90:10 ratio. Subsequently a routine for preprocessing was implemented. This includes all steps described in the preprocessing section of this report. Since we use custom classifiers from `sklearn` a training file was implemented featuring a model function and a classifier. The actual training evaluation process includes the following steps: upload of the training data to a Amazon S3 bucket, creating a SageMaker estimator using the implemented training script, training the model and finally deploying the model. The deploymed model is then ready to be used with e.g. the test data.

The benefit of using SageMaker is to execute the model training and deployment of the machine learning model on Amazon Web Services. This makes it easy to perform all steps on a cloud, which makes the computation and training independent from individual computation power. Since we SageMaker in the Nanodegree course before, this did not cause any complications during the development. Two classifiers were tested: the `RandomForestClassifier` and the `SGDClassifier` a SVM both provided by `sklearn`. The latter provided the better performance. This was to expect, since SVM are the classic choice for machine learning models used for sentiment analysis.

4 Results

This chapter will give an detailed overview over the achieved results measured in specific metrics. Subsequently the results are compared with the chosen benchmark.

Model Evaluation and Validation

The results of the trained SVM can be observed in Table 2. Our model is best in correctly predicting neutral sentiments, second best in predicting positive sentiments and third best in predicting negative sentiments. These results match the distribution of the training data, as tweets with neutral sentiment were most represented in the training data, followed by positive and negative tweets. This suggests that a better distributed set or at least more data for especially negative tweets would have been beneficial for the performance. The total accuracy was tested to be 77%. For a better insight into the model, the confusion matrix is displayed in Table 3. It shows that many actually negative tweets got classified as neutral tweets. The prediction of neutral tweets works pretty well. Looking at the last row of the table one can see that some positive tweets got labeled as neutral tweets, although the classification works way better than the one of negative tweets. Generally the models seems to use the neutral label to often.
The characteristics of the performance (best for neutral tweets, worst for negative tweets) is shared by our chosen benchmark. Their results can be seen in Table 1. The performance also shows that we could almost keep up with their SVM [1] taking the F1 score macro-averaged from $F1_{pos}$ and $F1_{pos}$ as evaluation metric (F1 score of 0.54 vs 0.57). Our model did better predictions for positive and neutral tweets, model did better predictions for negative tweets. Their CNN, however, performed even better with an F1 score of 0.65. One reason for their better performance could be the bigger training data. Due to the deletion of some tweets of their corpus, our model could only be trained on a subset of their corpus.

Classifier	Accuracy	$\mathbf{F1}_{pos}$	$\mathbf{F1}_{neutral}$	$\mathbf{F1}_{neg}$	F1
SVM	0.77	0.69	0.85	0.39	0.54

Table 2: Results for sentiments on German Twitter. The SVM were trained on fixed split of the corpus (90%), and then tested on the remaining texts. F1 score here is macro-averaged from $\mathrm{F1}_{pos}$ and $\mathrm{F1}_{pos}$.

		Predicted		
		Negative	**Neutral**	**Positive**
True	**Negative**	0.27	0.62	0.11
	Neutral	0.17	0.92	0.06
	Positive	0.03	0.33	0.64

Table 3: Confusion matrix. The confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

5 Conclusion

We saw that our SVM model with a multi step, Twitter specific preprocessing routine performs well on neutral and positive tweets. One thing to work on would be have the model predict positive and negative sentiments more often. Although there is still room for improvement, especially for predicting negative tweets, the performance of an overall accuracy of 77% and F1 score of 0.54 is quite good. Since the dataset included a couple of thousand tweets, the solution is significant enough to have solved the problem.

References

[1] Cieliebak et al. (2017) *A Twitter Corpus and Benchmark Resources for German Sentiment Analysis*, Association for Computational Linguistics, Valencia, Spain.

[2] Sascha Narr, Michael Hulfenhaus, and Sahin Albayrak (2012) *Language-independent Twitter sentiment analysis*, Knowledge Discovery and Machine Learning (KDML), LWA, pages 12–14.

[3] Igor Mozetic, Miha Gr car, and Jasmina Smailovi c. (2016) *Multilingual Twitter Sentiment Classification: The Role of Human Annotators*. PloS one, 11(5):e0155036.

[4] Uladzimir Sidarenka (2016) *PotTS: The Potsdam Twitter Sentiment Corpus*. In Proceedings of LREC2016, pages 1133-1141, Portoroz, Slovenia.