Party2vec

Taras Horbatiuk
Philipps-Universität Marburg
horbatiuk@⋆

Sven Löchte
Philipps-Universität Marburg
loechte@⋆

Jakob Leander Müller
Philipps-Universität Marburg
muelle5t@⋆

⋆students.uni-marburg.de

Abstract

Party2vec is aiming to correctly classify tweets into different German political parties. To achieve this, we will take a look at different approaches and take a combination as the final classification.

1 Problem

Tweets are often short messages with less than 50 words. To classify these belonging to a party brings some difficulties. Our project 'Party2vec' will tackle the task to correctly classify a tweet to the six biggest German parties. We want to publish our results on a website, where any user can copy or compose a tweet and our three different models will each give a prediction, which the user can then see.

2 Motivation

Over 70% of the German Bundestag members have a twitter account. Even if we only look at the 15 most popular politicians (based on followers) of each party they average over 200 Tweets per day. Our motivation for this project is, that we are interested if it's possible to precisely classify all of these tweets. We would like to test, if parties with a more extreme position, like *'Afd'*, *'Die Linke'* and *'FDP'* are easier to classify than the more central parties, like *'Bündnis 90/Die Grünen'*, *'SPD'* or *'Union'*(*CDU/CSU*). Figure 1 (appendix) shows where parties have common or opposed views, based on each parties' statements to the most recent Bundestag election (Friedrich-Ebert-Stiftung, 2017).

3 Data

To achieve our goals, we need some data. Since there aren't sufficient data sets online, we decided to build one ourself. We want to analyse tweets of politicians which belong to one of the larger German parties. To assure, that there would be enough tweets for each class, we decided to take all parties that have at least 20 seats in the 19th Bundestag into account. One of the challenges was the distribution of tweets: If all politicians were considered, the 'Union' (CDU and CSU) would have thrice the amount of potential twitter users than 'Afd', 'B90/Grüne' or 'Linke'. Ideally, we would want to have an equal amount of tweets for each class, so the 15 politicians of each party, which have the most followers (20.05.2021) were used in the data set.

3.1 Scraping

Using SNScraper (JustAnotherArchivist, 2021) we gathered data of 105 politicians from 24.09.2017 (date of the most recent election for the Bundestag) to 20.05.2021. To ensure that our results can't be affected by the scraping order we cut off all data after the 16th of May 2021. This leaves 190 weeks of data with over 300k tweets.

3.2 Pre-Processing and Sorting

Along with the number of collected tweets, the quality of the data set is equally important. Most of the raw tweets are very unstructured and full of informal content, such as emoticons, links, hashtags etc, which can harm the machine learning algorithms' performance. In order to train models, it's necessary to conduct training with a clean data set. Our pre-processing stages, carried out on the raw tweets, are as follows:
1) Replacing German umlauts with the corresponding English letters (e.g. ä → ae, ß → ss).
2) Replacing all user mentions (@) with default token 'user'.
3) URL's and symbols, like points, hashtags or commas, which do not contain any predictive information, were were removed.

4 Classification Approaches

To get an idea of how difficult it is to classify tweets correctly, a set of 48 tweets (8 per party) was selected and a total of 5 people tried to guess the parties. As expected, the results were mediocre at best. On average, they could classify at little less than 35% correctly. Of course, this is a very small sample size, but it highlights the difficulty of the problem.

4.1 Vectorizer and Classifier

Before we used some complex algorithms for machine learning, we started by implementing some basic models, in order to compare their effectiveness and the result. Logistic Regression, Naive Bayes and Linear Support Vector Classifications (LinearSVC) were trailed. To convert the tweets to numeric feature vectors, both a CountVectorizer and a TfidfVectorizer were used to see the differences between these two. Additionally, a training was carried out with LinearSVC and CountVectorizer, but with the use of word lemmatization with the help of a large pretrained model for German (*de_core_news_lg*) (Honnibal et al., 2020).

4.2 Neural Network Model

To have an understanding of how well our other methods perform, we decided to benchmark them against a pure out-of-the-box method. To vectorize the tweets, the pretrained model *de_core_news_lg* (Honnibal et al., 2020) was used. For the vectorization, spaCy will return the mean of all token vectors of a given tweet. That vector has 300 dimensions and represents the input of a neural network. After testing, we settled on a network with three hidden layers. The first two being 150 dimensional and the third being 75 dimensional, all using a relu activation function. Lastly, the output layer consists of six neurons, each representing one party. The final layer uses a softmax activation function. That way we have an array of probabilities as result.

4.3 Word Embedding Matrix and RNN

4.3.1 Embedding Matrix

For this approach, firstly, an "Embedding Matrix" was created from the collected and preprocessed data set through a Word2Vec-Algorithm using skipgrams with negative sampling (NEG) (Mikolov et al., 2013a,b).

Rather than eliminating stop words from the given tweets, a sampling table was used, so that skipgrams with frequent words are mostly discarded, while each skipgram with a rare word is used. Additionally, an embedding is also generated based from the lemmatized data set (see Vectorizer and Classifier), so that multiple tokens are summarized into a single lemma, so there are more context for each lemma. As for the lost grammatical information (e.g. genus, numerus, tempus), those might not add a lot of information for the classification and can therefore be discarded for then possibly more precise vectors.

4.3.2 Classification with Embedding Matrix and bidirectional LSTM RNN

The embedding matrix is used to train a bidirectional - Long-short-term memory model (Figure 2, appendix). A LSTM system (Figure 2, appendix) is able, as a recurrent neural network, to contain information through a feedback loop, but it has additionally two gates, the forget gate and the input gate, which lets the cells in the LSTM layers forget and learn, respectively, based on the input (the output of the previous time step), effectively changing the cell state. This state is then used, in combination with the aforementioned input to generate a new output (through the output gate), which is used in the next time step. In a bidirectional LSTM-model (BLSTM), the input is traversed twice, once in reverse, used for short text classification (Nowak et al., 2017).

For the structure (after testing), the data was padded/truncated to a vector of length 50 (length ≤ 50 for 99.71% of the data, see 3 (appendix)), and for each word, the corresponding weight vector of the embedding matrix was inserted into the bidirectional LSTM (256/128 Neurons, "tanh"-activation) and the results were used in 3 sequential densely connected layers (128 Neurons with 10% dropout (relu activation), 64 Neurons (relu activation), 6 Neurons (softmax)). As result, the maximum of the 6-neuron output layer was used as predicted label.

5 Results and Discussion

5.1 Word Embedding Matrix

For the embedding matrix, multiple parameters were tested (dimension of word-vector, skip gram size, number of negative sample), and the best performing parameters (Table 1) are comparable to the

one described by (Helaskar and Sonawane, 2019). Through the usage of word tokens, a embedding matrix (with 82.23% accuracy for each context - target skip-gram pair) was achieved, while using lemma yielded 76.50% and using the no-party-matrix resulted in an accuracy of 80.10% (see figure 4 (appendix)).

5.2 Classification Problem

5.2.1 Classification Problem

For the classification problem, each proposed model was trained with a portion of the collected and preprocessed twitter data (70% train) and then tested with the remaining data points. For the vectorizer + classifier approach, the highest accuracy was achieved with the LinearSVC and TfidfVectorizer while using word tokens, the results were slightly worse with lemmatization. In table 2 the results (in different metrics) of each model can be seen.

The benchmark model achieved a combined accuracy of 48.40%, meaning it did not perform particularly well, but still performing better than the human classification. Reason for this could be the fact, that the input vectors are means of the vectors of all words. Here, there is a clear improvement for LSTM-RNN (3) in comparison to the NN (2) visible in all metrics. While the LSTM-RNN also performs better than the combination TFidf-vectorizer and linear support vector classifier (LinearSVC) (1), there is only a slight difference ($< 1\%$), which could be caused by unclean data, which still contains tweets, which might not contain any political information, and have a bigger impact on the LSTM-Model than on the LinearSVC.

5.2.2 Classification of specific parties

While the broader differences between the three different models were discussed, there is a sizable difference in the prediction of each party (table 3). For both the *AfD* and the *FDP*, a great difference to the macro avg. F_1-score is observed in all three models, reaching a $11 - 13\%$ difference for using the LSTM-model. On the other hand, both the *Union* and the *Grüne* show the adverse effect, reaching even a F_1-score under 50% in (1) and (3). A reason for the better predictability (for *AfD*/*FDP*) might be, that both those party represent are relatively static and isolated point in the political spectrum (see 1, (Friedrich-Ebert-Stiftung, 2017)), so that the tweets of the corresponding politicians convey those standpoints, while not changing in the analyzed time frame. On the other side, the political standpoint of, e.g. the *Union*, move more towards other parties, like the *AfD* or the *SPD*, therefore having a broader, more diffuse standpoint, which might be shared by those overlapping parties, resulting in less precise predictions. Therefore, the system might predict, that a certain opinion belongs to a party with a solidified standpoint rather than to a party, where there might be different opinions about the topic, e.g. the refugee crisis, where the opinion of the *AfD* is set, while there are differences between *CDU* and *CSU*, which make up the *Union* (Börnsen, 2020).

Another reason, which interferes with the predicting, could be data in the used data set, which might not actually care any political motivation. So, those kind of tweets then both lower the accuracy in the training and the testing, since they might be from any party, and could therefore be labelled in the testing as any party.

Overall, both models (1) and (3) manage to predict the correct label of a tweet with 56%/57%, raised from $1/6$ with random predicting, so these models represent a good starting point for further improvement in predictability, e.g. through a process to filter out "non-political" tweets.

6 Outlook

6.1 Future

For the Future of this project we can see multiple branches. One of them might be the classification of twitter profiles. Analyzing user profiles in that way could give a direction, which party the user sympathizes most with. An advert agency could then use that information to place personalized ads. If this particular application of our project should be done is of course another discussion.

A brighter use of our work could be election predictions. A group could research if there is a correlation between the amount of tweets for a party and their election result.

6.2 Problems

The main problem of this project is its relevance at the moment, since the tweets were received for those parties and politicians who are now in the Bundestag and the word embedding that was created includes a set of words that are relevant today and should be updated in the future.

6.3 Publishing

To make the project useful to others, we decided to build a website. You can paste a tweet inside a textbox (or write something yourself) and the page will visualize the result of each approach as well as giving you a combined result. As of writing this report only the locally hosted page works, since we have some permission problems with the remote hosting service. Hopefully, that will be fixed till the presentation, so that other students can try the classification themselves.

References

Wenke Börnsen. 2020. Wer will was im flüchtlingsstreit? *Tagesshow*.

Jingjing Cai, Jianping Li, Wei Li, and Ji Wang. 2018. Deeplearning model used in text classification. In *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 123–126.

Friedrich-Ebert-Stiftung. 2017. Strategiedebatten.

Mukund N. Helaskar and Sheetal S. Sonawane. 2019. Text classification using word embeddings. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–4.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

JustAnotherArchivist. 2021. *SNScraper*. "Github".

Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. pages 1–12.

Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

Jakub Nowak, Ahmet Taspinar, and Rafał Scherer. 2017. Lstm recurrent neural networks for short text and sentiment classification. In *Artificial Intelligence and Soft Computing*, pages 553–562, Cham. Springer International Publishing.

Christopher Olah. 2015. Understanding lstm networks.

A Graphics & Tables

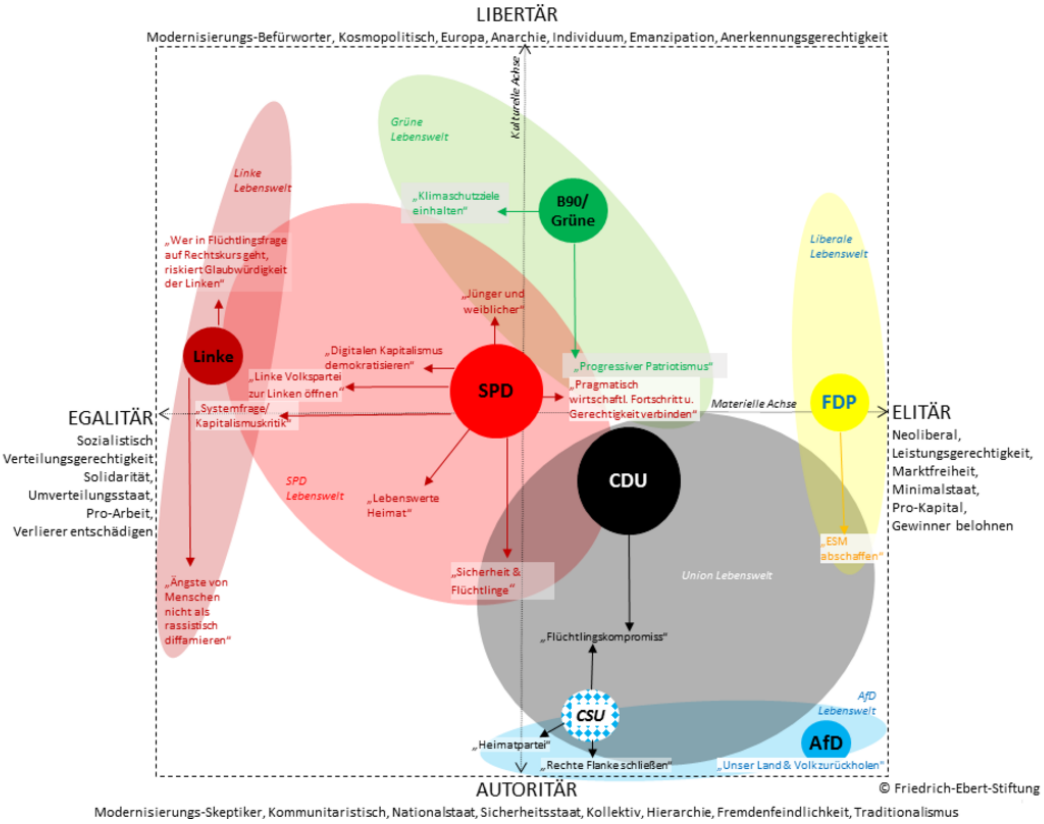

Figure 1: Political alignment of the considered parties (Friedrich-Ebert-Stiftung, 2017).

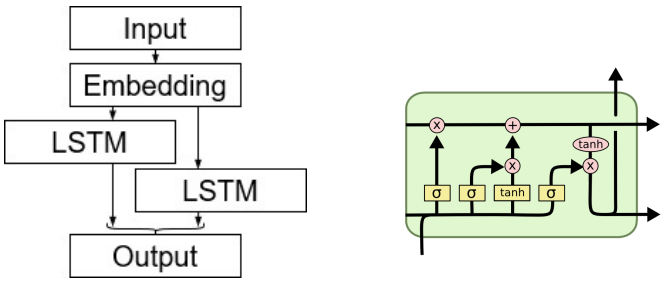

Figure 2: Left: Bi-bidirectional LSTM-RNN - Model (Nowak et al., 2017; Cai et al., 2018)
Right: LSTM - Cell (Olah, 2015)

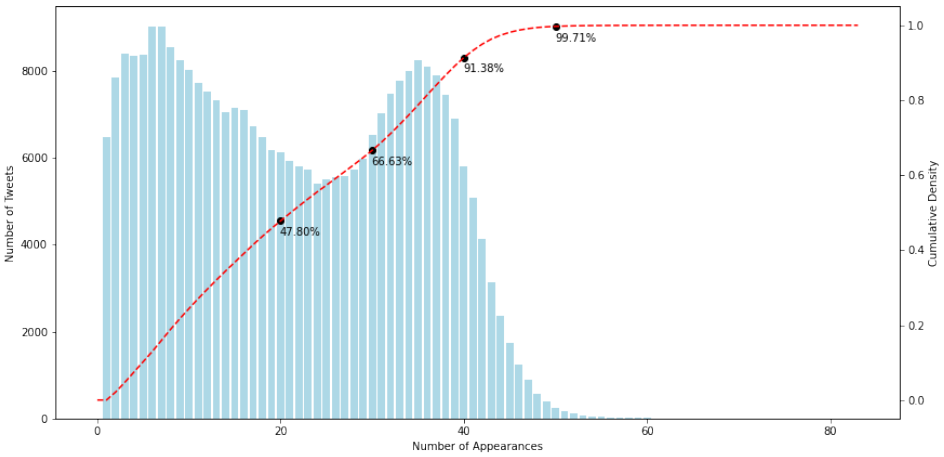

Figure 3: Length of tweets in Dataset, Percentage for length 20, 30, 40, 50.

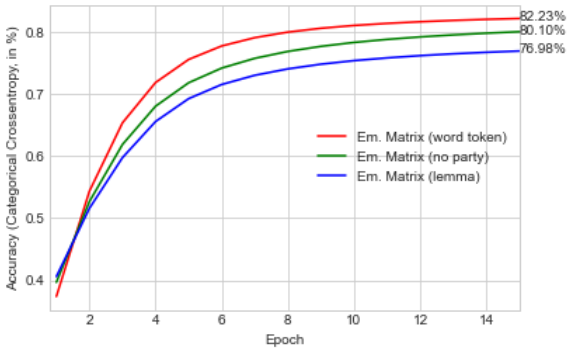

Figure 4: Accuracy curve for embedding matrices.

Figure 5: Wordcloud for most common words in our dataset

Table 1: Parameter for embedding matrix

Parameters	
Skipgram window	3
Negative Samples	5
Embedding dimensions	300
minimal occurrence	3
Unique words	75077
Unique lemma	64724

Table 2: Evaluation (macro avg.)
(1) TFidf + LinearSVC, (2) NN, (3) BLSTM RNN

Approach	Acc.	Prec.	Recall	F_1
(1)	55.94	55.76	55.40	55.51
(2)	48.40	48.18	48.40	48.05
(3)	56.77	56.70	56.36	56.38

Table 3: F_1-score, broken down per Party
(1) TFidf + LinearSVC, (2) NN, (3) BLSTM RNN

Party \ Model	(1)	(2)	(3)
AfD	65.21	58.76	69.95
Union	47.07	40.81	44.54
FDP	59.76	52.88	67.79
Grüne	51.41	37.61	44.99
Linke	57.62	48.59	53.43
SPD	52.00	47.68	53.38

Table 4: Results using different vectorizer and classifier

classifier \ vectorizer	Count	Tf-idf
MultinomialNB	52.87	47.84
Logistic regression	55.25	55.31
Linear SVC	53.41	55.94
Using lemmatization		
Linear SVC	53.37	55.77