Word embeddings for predicting political affiliation based on Twitter data

Ibrahim Abdelaziz[1], Oliver Berg[1], Angjela Davitkova[1], Venkatesh Iyer[1], Shriram Selvakumar[1], Kumar Shridhar[1], and Saurabh Varshneya[1]

[1]Technische Universität Kaiserslautern

January 17, 2018

1 Introduction

The modern world of social media knows a plethora of means to communicate ones personal opinion and political alignment. With the platform *Twitter*, figures of political interest are expressing their standpoints in small-sized 144 character texts (recently updated to 280 characters), which contain a comprised message specific to the general public. This yields great potential for automated analysis of party affiliations to classify political persons of interest within the overall political spectrum [Biessmann et al., 2017].

Hence, we propose a structured approach of building a deep learning based classification model that utilizes state-of-the-art **word embeddings** [Pelevina1a et al., 2016] to perform **qualitative analysis** on a constructed **social media data set**. Our approach will be helpful in analyzing possible early intuitions and dedicated insights within the German political spectrum.

This is to be seen in context of latest **advances in research**.

2 Related Work

Political motives were shown to be consistently predictable with an accuracy better than chance [Biessmann et al., 2017].

Many existing papers either propose tackling the classification problem of political bias using techniques such as Support Vector Machine (SVM) or Singular Value Decomposition (SVD) [Misra and Basak, 2016], or they focus on comparison of different classifiers to not restrict themselves to a single well-developed approach [Bhanda et al., 2009]. These approaches mainly consider the political affiliation in America, where the political orientation is mostly binary with two parties - republicans and democrats - covering most of the political spectrum. Overall, sentiment classification is mostly covered using recurrent- or convolutional neural networks [Kim, 2014].

In connection to the given focus of working on Twitter data, [Cohen and Ruths, 2013] introduces objections to some of the pre-existing approaches. With standard classifiers for inferring political orientation having greatly lower accuracy compared to what was initially report, it is stated that the classifiers cannot be used for classifying users outside the training data. Thus the contradictory arguments hold true.

3 Proposed Methodology

As our approach aims to analyze text messages of political figures concerning party affiliation, we leverage *word embeddings* to represent words in context. We shall initially restrict ourselves to a pre-trained model as the number of political parties as classes is not very high. A person's political affiliation will be calculated as a combined analysis of all of his Twitter messages.

Subsequently, a neural network architecture then classifies the Twitter profile, consisting of a collection of this person's tweets, concerning party affiliation. It will learn to represent a political figure's affiliation through their expression in short message texts.

3.1 Data Set and Feature Extraction

The data set used in this approach was constructed of German politicians' tweets posted on thier Twitter accounts, the politicians' Twitter accounts were retrieved from the website "https://www.wahl.de/", which has a list of around 1000 German politicians profiles. from each profile politician's name, political party, and Twitter username were gathered.

The retrieved Twitter accounts were filtered to only those which belong to the seven major political parties, namely "CDU", "CSU", SPD", "FDP", "GRÜNE", "LINKE" and "AFD", ordered by age of introduction into German parliament, old to new.

After the filtering stage we had a list of around 700 politicians, for each one we gathered his/her most recent 1000 tweets using the Twitter API, number of gathered tweets may fall below 1000 for some politicians who don't have enough tweets on their accounts. Since some parties had greater number of tweets compared to other parties, and to create a balanced set for training our model we used only 12000 tweets for each party with approximately equal number of tweets for each politician.

To prepare the tweets for the training step, each tweet was preprocessed by:

- Removing URLs, special characters, user names, and mentiones.
- Turn all characters, including German special characters, into lowercase characters.

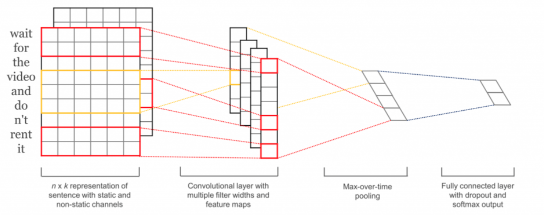
To represent the preprocessed tweet's text into numerical values that can be used to train our model we have used a pre-trained Word2Vec model [Mikolov et al., 2013], trained on 200 million German tweets [Cieliebak et al., 2017], to represent each word of the tweet as a 200-dimensional vector. Using this approach we can now represent each tweet, which consists of n words, as matrix $M \in Rn \times 200$.

3.2 Classification

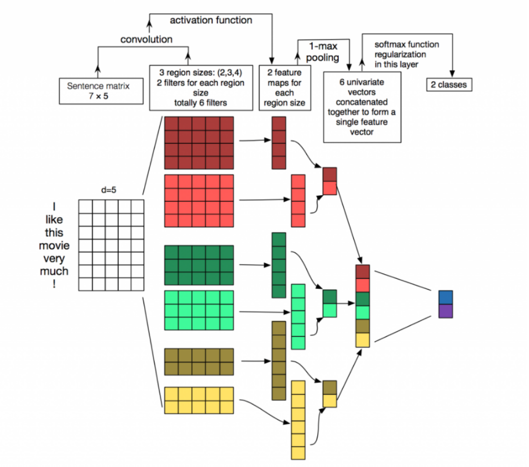
While deciding upon the architecture, RNN(Recurrent Neural Network) looked like an intuitive solution as it is more similar to how humans think and process language: one word at a time and then forming sentence using those words. But we wanted to experiment things in terms of making the training process faster and less computational intensive, so we used a CNN model for our case as convolutions are a central part of computer graphics and implemented on a hardware level on GPUs which make them very fast for training.

Another point in favor of CNN is being very efficient in representations. With a large vocabulary, computing anything more than 3-grams can quickly become expensive. Convolutional Filters learn good representations automatically, without needing to represent the whole vocabulary and we can put a bigger filter size of 7 or more depending on the use cases as different filter sizes learn different representations.

3.2.1 Model Architecture


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Our model architecture is taken from [Kim, 2014] which is used for sentiment analysis. The first layers embeds words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, we max-pool the result of the constitutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer.


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

The whole classification process can be subdivided into two elementary parts

1. Feeding input data to Neural Network
2. Classifying data to correct class

The following steps were taken in order to feed data to the Neural Network:

1. 7 raw text files were prepared each containing the tweets for respective parties collected from Twitter.

2. The dataset was cleaned and tokenized for building a vocabulary.

3. Each sentence was padded to a max length. We append special <PAD> tokens to all other sentences to make them equal of 35 words. Padding sentences to the same length is useful because it allows us to efficiently batch our data since each example in a batch must be of the same length.

4. For feature extraction from twitter data we needed a pretrained word-embeddings to represent every word in vector form. We use word-embeddings [Mikolov et al., 2013] pretrained on 200 million German Tweets using 200 dimensional vector representation of each word [Cieliebak et al., 2017].

5. Finally we built a vocabulary based on our collected data and map each word to an integer between 0 and 109933 (the vocabulary size). Each sentence becomes a vector of integers.

For classifying the user tweets to the correct political party, the following two steps were performed:

1. We feed the batches of twitter data along with their correct political party one-hot-encoded labels and trained above defined CNN

2. To optimize the network we used the cross entropy loss defined as:

$$H_{y'}(y) = -\sum_i y'_i \log(y_i) \tag{1}$$

Where y is our predicted probability distribution, and y' is the true distribution (the one-hot vector with the true-class party labels).

4 Quantitative Analysis

For quantitative analysis of the results, every tweets were taken as sentences and a sentence vector was constructed of the same dimension as the word vectors used for training (i.e. 200 dimensional vector). In doing so, the tweet was tokenized into words and every word corresponding token vectors were taken from the word embedding space and a sentence embedding vector was constructed by adding the word vectors and dividing them with the number of words. Only those tweets were considered whose all words were present in the word embedding space and any tweets with UNK token were ignored for better analysis. This was done to remove any tweets that were not completely related to politics as a user tweets on a variety of backgrounds and we only want the tweets related to political spectrum.

The sentence embedding was used to visualize the vectors based on their 200-dimensional vector representations. The results were pretty mixed up with no clear distinct clusters for any specific party. There could be several reasons for the mixing up of results and no clear clusters. Since a user re-tweets other parties tweets with his opinions, it could be one of the reason for mixing up of clusters.

Another reason for such mixed clusters is the fact every party is tweeting about the same issues with some in favor of it and some against of it and this leads to creation of same tokens for different party supporting user/group tweets. Further, these tweets with some/very less distinct tokens like together in the cluster prohibiting a distinct cluster for each class.

So a direct embedding as above did not proved to be a good measure to visualize the result. So we used a political compass [PoliticalCompass, 2018] for visualizing the result further.

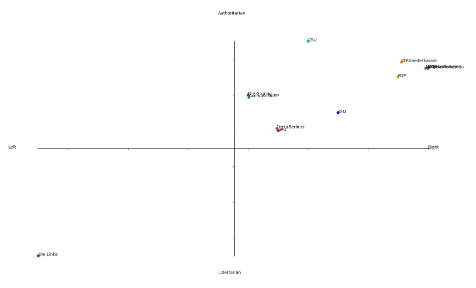
For the political compass representation, we use a four way graph plot with Left, Right on X axis and Authoritarian, Libertarian on Y axis. On the four axis, we used an one hot encoding to plot all the seven classes on the compass. This way we can see all the political classes plotted.

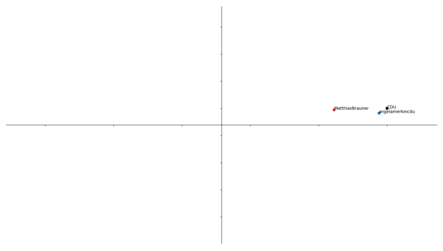
4.1 Visualization

Now, we need to plot all the users on the compass to see which users corresponds to what cluster and following steps were taken in order to do so:

1. We take a user and scrape all the tweets of the user from twitter.
2. Next we pass the tweets of the user to the classification model as specified above in the model architecture.
3. We get the class classification and probability score of all the tweets for all the seven classes.
4. We average all the probability distribution to get one vector consisting of a probability distribution for all seven classes.
5. We plot the user on the political compass to visualize the results and to see if the user has been classified to it's correct class.

4.2 Inference





The following can be inferred from the above political compass visualization:

1. Most users have been plotted very near to it's political party representation. E.q. Angela Merkel plot is very near to CDU which shows all her tweets were pretty biased for the CDU Party.

2. Plots of FDP is very near to the plot of CDU class. This means the ideology of these parties are quite similar. (Not exactly similar but some similarity is there.) This could also be seen from the usage of similar vocabulary by both the parties.

3. User DerUrBerliner from party CDU is plotted near to SPD. Upon analyzing the data, it can be inferred that there are a lot of English language based tweets present in the scraped tweets data, which led to a lot of <UNK> token generation and hence the classification was not correct. Upon data cleaning we were able to see the user plot moving to the correct class.

5 Appendix - Work packages and distribution

We plan to distribute the workload into the work packages, for each work package we assign a group of people, and an initial estimated deadline.

Building dataset
Beside the Tweets of German politicians that we will get using the Twitter API, we also plan to collect data from other sources such as parliament discussion data and party manifesto data.

Training word vector model
After building our dataset we will convert the data text into numerically creating word embedding of the words in the text using Word2Vec or derived approaches.

Developing classification model
As previously mentioned in the Proposed Methodology we will implement a neural network as our classifier, and train it with our data set.

Training / Testing
After having quantified model accuracy with a dedicated validation split, this separate training- and testing-stage ensures that the obtained results match initial expectations or reject estimates in an understandable fashion. We thereby ensure that the obtained results obey logic and real-world measures.

Quantitative Analysis
To conclude the findings from real-word social media data analysis, we infer statistical and sociological meaning to the modeled results and put them in context to the initially motivated research question of political affiliation and political classification.

References

[Bhanda et al., 2009] Bhanda, M., Robinson, D., and Sathi, C. (2009). Text classifiers for political ideologies.

[Biessmann et al., 2017] Biessmann, F., Lehmann, P., Kirsch, D., and Schelter, S. (2017). Predicting political party affiliation from text.

[Cieliebak et al., 2017] Cieliebak, M., Deriu, J., Egger, D., and Uzdilli, F. (2017). A twitter corpus and benchmark resources for german sentiment analysis. *SocialNLP 2017*, page 45.

[Cohen and Ruths, 2013] Cohen, R. and Ruths, D. (2013). Classifying political orientation on twitter: It's not easy!

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Misra and Basak, 2016] Misra, A. and Basak, S. (2016). Political bias analysis.

[Pelevina1a et al., 2016] Pelevina1a, M., Arefyev, N., Biemann, C., and Panchenko, A. (2016). Making sense of word embeddings.

[PoliticalCompass, 2018] PoliticalCompass (2017 (accessed Jan 10, 2018)). *Using LaTeX for Your Thesis*.

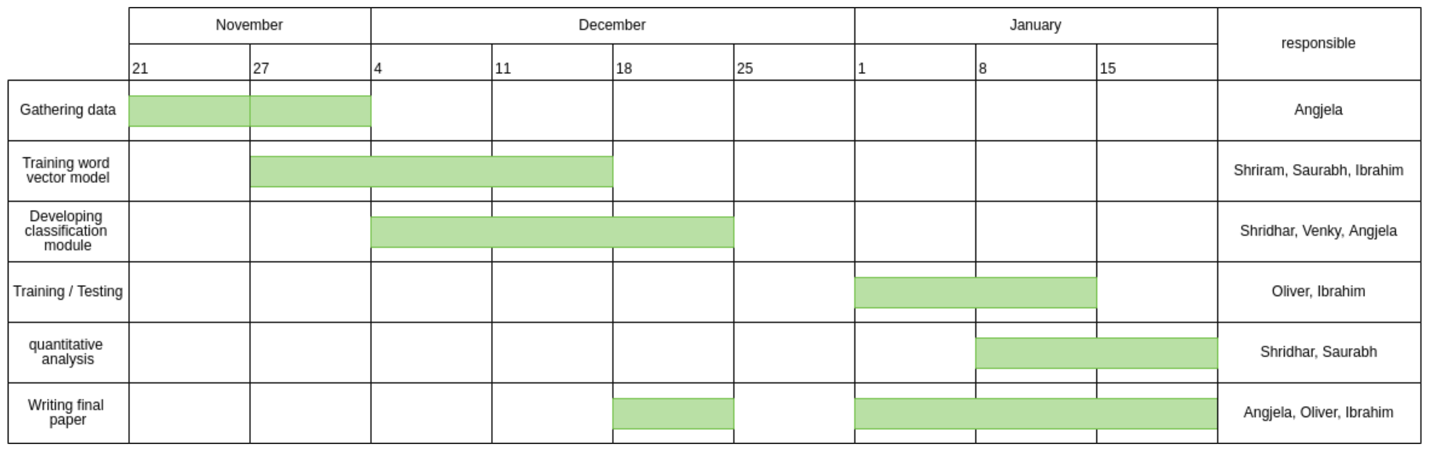


Figure 1: Gantt-Chart displaying workload distribution per team-member