Word embeddings for predicting political affiliation based on twitter data

IBRAHIM ABDELAZIZ[1], OLIVER BERG[1], ANGJELA DAVITKOVA[1], MD ASHRAF HOSSAIN[1], CHARU JAMES[1], SHRIRAM SELVAKUMAR[1], KUMAR SHRIDHAR[1], and SAURABH VARSHNEYA[1]

[1]Technische Universität Kaiserslautern

November 21, 2017

1 Investigated Problem

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, 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 political spectrum.

As an attempt to create this classification of political alignment/affiliation, a corresponding approach will be discussed. In addition, finding of the patterns or types of jargon/language used per party will be detected. The following discussed approach will focus on the previously mentioned things, but with a slight variation it may answer additional interesting questions, when considering the different german parties and the people who follow them. This topic is interesting for everyday users, which tend to browse through different social media content and to read different blogs, and thus being affected by different political opinions without knowing whether the certain content is biased or not. Also, this type of analysis also can be used for statistical information by organizations and politicians.

[.. #taskDescription & why do this in the first place ..]

2 Literature Research

A thorough introduction to political affiliation analysis was provided in *"Predicting political party affiliation from text" by Biessmann et. al.*, where political motives were shown to be consistently predictable with an accuracy better than chance.

Bla blab la other approaches and why we differ from them[I don't know why do we differ, only that we are creating for germany]. For sentiment classification most people use RNN and CNN ex.[https://arxiv.org/pdf/1408.5882.pdf]

Existing papers[5] tackle this problem using various techniques, such as SVM, SVD, LSM etc.. Mainly these approaches consider just the political affiliation in America, where the orientation is rather simpler, since there are only two sides and the users are mainly biased towards one side. Some of the already existing papers[4], tend to focus on comparison of different classifiers when trying to attack this problem, and thus not proposing a complete well-developed approach.

But unlike the previously mentioned approaches, a rather different report 'Classifying Political Orientation on Twitter: It's Not Easy!' by Raviv Cohen and Derek Ruths, tends to contradict some of the already exiting approaches. According to them, standard classifiers for inferring political orientation have greatly lower accuracy from the accuracy that they report, by stating that the classifiers cannot be used for classifying users outside the training data. Thus the following research approach tends to improve the prevoisuly mentioned approaches and contradict the Raviv and Derek statement[3].

[.. papers for methods deployed in example paper and/or our later approaches ..]

[.. papers for methods deployed in example paper and/or our later approaches ..]

3 Considerations towards Data

According to the presented task description of analyzing political affiliation based on Twitter-data, the final comparison occurs on the basis of **tweets** made by political figures on the platform Twitter.

Along the ground work established in *"Predicting political party affiliation from text" by Biessmann et. al.*, the learning models are also deployed against **parliament discussion data** as well as **party manifesto data** to establish a categorical groundwork.

For the purposes of this specific research, the main portion of data would be the Twitter data. In order to include all of the relevant politicians from the parties, their corresponding Twitter accounts will be collected from websites [1], where the data is offered as open source data. For each of the Twitter accounts of the politicians, a corresponding party is kept, for which the politician is working. Having these accounts, the same number of tweets will be crawled for each of the parties. 80 percent of the collected data would be used from training the classifier and 20 percent of it would be used for testing. Additionally, as testing data the previously mentioned sources, which incorporate parliament discussion data and party manifesto data, can be used.

[.. more specifically: which data should be used how ..]

[...]

4 Deployed Methods

As mentioned, in order to provide semantic classification for detecting political affiliation data with the corresponding text and the associated party will be used. For the purpose of this research, a Neural Network will be used, such as CNN or RNN(using TensorFlow).[THIS PART IS MY SUGGESTION] But in order to train a certain neural network, the corresponding texts/words need to be converted into numerical values, more specifically vectors, such that those which are similar will end up having similar values. And as seen in the previously mentioned, already implemented techniques, one way to do it is to create a word embedding for each od the words from the tweets. As described, word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers [2]. Thus, this means that the similar words will have similar vectors and they are going to reside in relatively the same area in the vector space since they both have similar definitions and are both used in similar contexts. While constructing/learning/tuning this word embeddings appropriate dimensionality reduction will be applied, as well as a convolution layer will be included, with specified window size. The Window size will be chosen accordingly such that the needed nearby words/inputs, which we consider relevant, are included in the construction of a word embedding. [I prupose Word2Vec or TensorFlow]

For the appropriate classification RNN or CNN can be used. Brief explanation of one of them, we can do it when we decide or just talk in general.

[.. which algorithms / categories of algorithms to test first/second/third ..]

[.. which approaches may yield which types of results; generally: what is to be expected? ..]

AND A GANT CHART[SOMEBODY NICE ENOUGH SHOULD CREATE THIS]

MY PROPOSED APPROACH AND ACCORDINGLY GANT CHART

0) Gathering data 1) Training a word vector generation model (such as Word2Vec) or loading pretrained word vectors 2) Creation of Matrix for our training set 3) RNN or CNN for classification which takes as input word vectors 4) Training 5) Testing