An Incivility Analysis of Austrian Political Speeches

Daniel Kapla[1], Matthias Wastian[2*]

[1]Institute of Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8-10, 1040 Wien
[2]dwh GmbH, Neustiftgasse 57-59, 1070 Wien; *mathias.wastian@dwh.at

SNE 30(1), 2020, 1-6, DOI: 10.11128/sne.30.tn.10501
Received: September 15, 2019; Revised: January 10, 2020;
Accepted: February 10, 2020
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The state of the art technology in natural language processing (NLP) is dominated by neural networks. On the one hand, neural networks are used to learn how to calculate word embeddings, some of which are particularly well-suited for representing meaning and relations between words. On the other hand, neural networks make use of these word embeddings as input for additional NLP-related tasks such as sentiment analyses. The goal of this paper is an incivility analysis of Austrian parliamentary speeches. Therefore, different neural network types in combination with different word embeddings are compared in terms of performance and suitability. The best model was chosen to classify the given data set and analyze how incivility changes over time.

Introduction

The state of the art technology in natural language processing (NLP) are neural networks. The goal of this article is to compare different embeddings and network structures in order to find a well-suited neural network with a word embedding, which is particularly well-suited for representing meaning and relation between words, to analyze the political landscape according their incivility. Referring to the work of [1] and [2] two classifiers were trained and used for classifying 56.000 Austrian parliamentary speeches. Finally, building upon the classified speeches two examples of analysis interpretations are given.

1 Data

The text corpus considered is gathered from press releases of Austrian political parties, parliamentary speech transcripts and media reports from 1996 to 2013. In total, these are 56.000 speeches, consisting of more than 2 million sentences.

1.1 Training Set

The training data consists of more than 20.000 labeled sentences with labels from 0 to 4 where 0 means neutral and 4 stands for very negative. The sentences are taken from the previously mentioned text corpus and were labeled via crowd coding. The human crowd coders were asked to classify training sentences without any context, meaning only the sentence itself was presented. To obtain a single score for a sentence the mean of all answers was computed for each sentence. Furthermore, each single coder was checked for cheating or spamming by adding validation sentences with a predefined score into the data the coders were asked to score. Then all results of coders which did not reach an accuracy of over 75% at these test sentences were removed from the calculation of the final score. This led to a continuous score of negativity between 0 to 4 for each sentence in the data set. With these scores the sentences were grouped into three classes, namely neutral C_0, negative C_1 and very negative C_2. The decision boundaries were chosen as $4/3$ and $8/3$, see Figure 1.

In addition to the original training data an augmented set was created which added further labeled sentences to the 20.000 sentences. The amount of added sentences consists of only 20 sentences. They comprise of common phrases in the German language, especially in political speeches, like salutations. The need for these additions arose from a qualitative analysis of classifications performed by trained classifiers. Their analysis showed that almost all salutatory addresses were recognized as negative. Searching the training data set

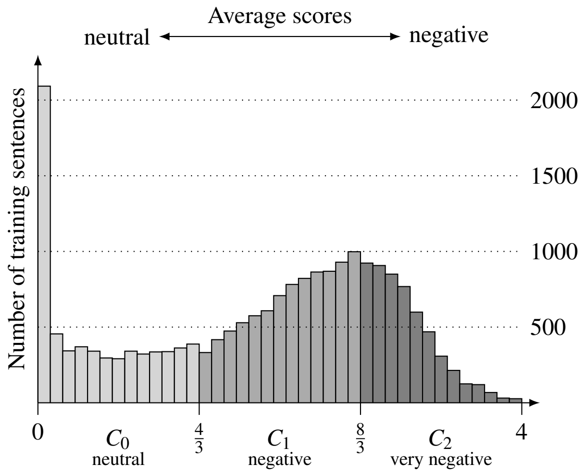

Figure 1: Average score distribution of all training sentences divided into three classes C_0 (neutral), C_1 (negative) and C_2 (very negative).

revealed that there are no positive occurrences of such phrases, but they were part of longer sentences in an ironical manner.

2 Word Embeddings

A *word embedding* is basically a vector representation for words. Their intention is to capture semantic and syntactic information within the relations between embedded words.

A basic principle within most word embeddings is well explained through a quote from John Rupert Firth ($*$ Keighley 1890; † Lindfield 1960);

> "You shall know a word by the company it keeps."

The following two embeddings are reviewed: the `word2vec` embedding introduced in [6, 7] and the `fastText` embedding from [8].

The *Continuous Bag-of-Words* (CBOW) variant of the `word2vec` model is basically a log-linear classifier. The main objective is to train the classifier to predict a word given its context within a text corpus. The context of a word is the set of all c previous and following words. The second variation is the *Skip-Gram* model which is structurally similar except that it is trained to predict context words given one word. They are intended to be very simple and efficient in training while having high quality representations. To get the efficiency in perspective the `word2vec` models are created to be able to be trained on multiple billions of words for vocabularies of millions of words. The quality is measured on different word similarity tasks. One such task is simple word similarity; for instance, `Hund` and `Katze` (In English: dog and cat) are close in the resulting vector space (according to cosine distance). Also more complex similarity tasks like if `groß` is similar to `größer` in the same sense as `klein` to `kleiner` (In English: big to bigger as small to smaller) were considered. To validate these similarities the authors used algebraic operations on the word vectors and computed $v(\texttt{größer}) - v(\texttt{groß}) + v(\texttt{klein})$ and searched for the closest vector which is $v(\texttt{kleiner})$. Even more remarkable is that when trained on huge data sets even more complex relations are encoded in the embedding structure like

$$v(\texttt{USA}) - v(\texttt{Obama}) + v(\texttt{Putin}) \simeq v(\texttt{Russland}).$$

The `fastText` embedding comes in the same two flavors and is very closely related to the `word2vec` embedding. In comparison the `fastText` embedding reaches higher qualitative measures. This results from a combination of multiple well-known techniques for training word vectors. In addition, the capability of embedding *out of vocabulary* words is added by using subword information for computing vector representations.

3 Neural Networks

This chapter gives a short overview of neural networks required in the following. For more details see [2, 3, 4].

A *Neural Network* is a network of neurons, where each neuron is a simple model of an actual neuron in the brain. One such model of a single neuron is the *Rosenblatt's Perceptron*. It assumes multiple inputs and a single binary output. The model computes a weighted sum plus a bias and then applies the Heaviside function. Learning is done by adapting the weights and the bias. For a neuron in a neural network this model is slightly altered by replacing the Heaviside function through a more general function ϕ called *activation function*. Among the most common activation functions are nowadays the *rectified linear unit* (ReLU) $x \mapsto \max(0,x)$, the *hyperbolic tangent* and the *logistic function* $\sigma(x) = (1+e^{-x})^{-1}$. Let $w \in \mathbb{R}^n$ be a vector representing the weights for each of the n inputs, $b \in \mathbb{R}$ the bias and ϕ an activation function. Now a single neuron is modeled by

$$x \mapsto \phi(w \cdot x + b).$$

A *Multi Layer Perceptron* (MLP) is a simple neural network which is constructed by stacking together multiple layers of parallel neurons. For every layer each neuron gets all the outputs of the previous layer as inputs.

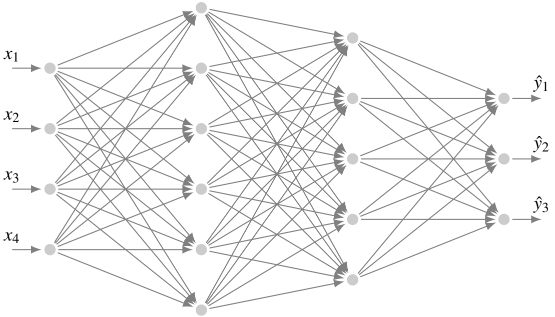

Figure 2: A three layer MLP (not counting the input layer). Each gray circle represents a single neuron.

A *Convolutional Neural Network* (CNN) is a neural network which uses parameter sharing and localization for processing structured data, for example images. Parameter sharing means that all neurons in one layer are using the same weights for processing their input. Localization means that a neuron is only connected to a small neighborhood of neurons of the consecutive layer.

A *Recurrent Neural Network* (RNN) is a network which uses back-references to keep track of previous information to process sequential data. The main working principle uses an internal state to summarize already processed data which is passed forwards while processing a sequence of data.

Assuming a finite sequence $(x_t) \subset \mathbb{R}^n$, for example a sequence of word vectors, a simple RNN layer could be constructed as follows. Let h_t denote the internal state after step t with an initial state h_0 (typically defaults to the zero vector). The weights are represented by three matrices W, U and V. They are for processing the previous internal state, the current sequence input and the current step's output, respectively. With a bias vector b and an activation function ϕ the next internal state is computed by summation of the transformed internal state, transformed input and the bias. Then the activation function is applied in an element-wise way. For computing an output $\hat{y}_t$ at step t the internal state is transformed using V.

$$\begin{aligned} h_t &= \phi(Wh_{t-1} + Ux_t + b), \\ \hat{y}_t &= Vh_t. \end{aligned}$$

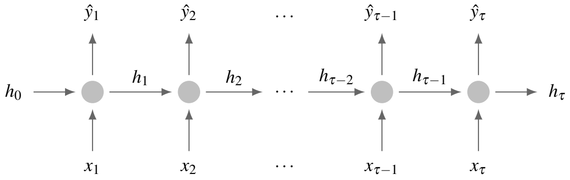

Figure 3: Information flow in an "unfolded" RNN layer for a sequence of length τ. For each time step t the network processes the current input x_t as well as the last (initial) internal state h_{t-1} and computes an output $\hat{y}_t$ while updating the internal state h_t that is passed forward in time to be processed by the same cell for each time step.

These simple RNN constructs are hardly practical for complex tasks because they are extremely hard to train, especially for long-term dependencies. This problem was resolved by gated RNNs. The first presented gated RNN was the *Long-Short Term-Memory* (LSTM) network, see [2, 3]. Years later the *Gated Recurrent Unit* (GRU) was introduced in [4]. It uses two gates, namely a *reset gate* and an *update gate*, to control the information flow while processing a sequence. The reset gate controls how much of the previous information is considered for computing the new hidden state. If the reset gate is closed, the past is ignored by "resetting" the hidden state leaving only the current input. The update gate then decides how much of the previous hidden state shall be propagated forwards in combination with the new hidden state. The recursive definition for the update gate u_t and reset gate r_t with their own weights as well as the hidden state h_t reads as

$$\begin{aligned} r_t &= \sigma(W_r h_{t-1} + U_r x_t + b_r), \\ u_t &= \sigma(W_u h_{t-1} + U_u x_t + b_u), \\ h_t &= u_t \odot h_{t-1} + (1 - u_t) \odot \phi(W(r_t \odot h_{t-1}) + Ux_t + b). \end{aligned}$$

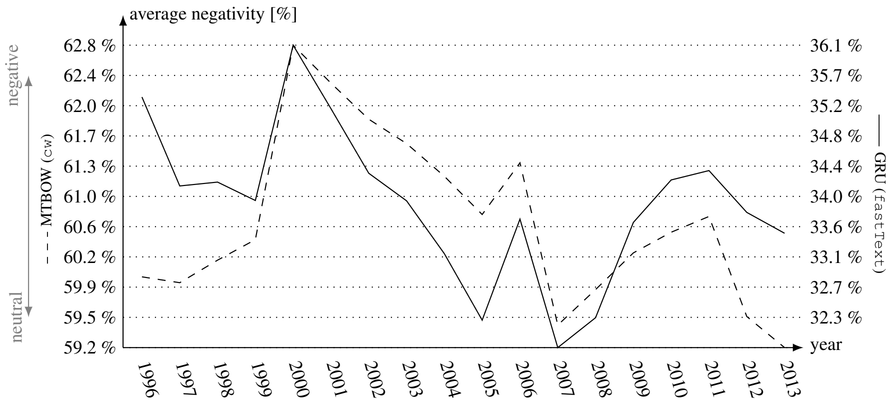

Figure 5: Comparison of Average Negativity of Speeches over Years labeled by MTBOW (`cw`) and GRU (`fastText`).

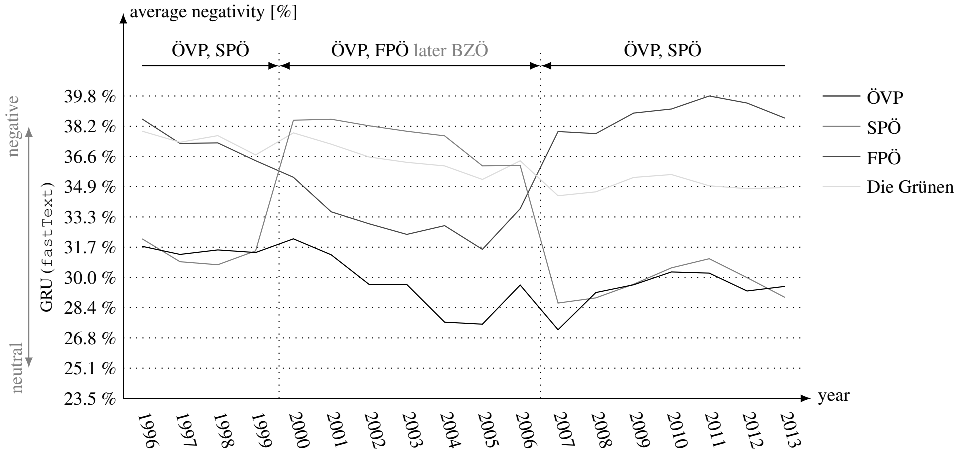

Figure 6: "Average Negativity" of Speeches over Years grouped by a subset of parties with indication of coalition.

4 Models

For the incivility analysis two models were used, denoted as MTBOW (`cw`) and GRU (`fastText`). The MTBOW (`cw`) (More Than Bag Of Words) model was initially considered in [1] and consists of a MLP in combination with the `cw` embedding provided by the `Polyglot` library, where the MLP was compared against conventional NLP approaches like a Naïve Bayes classifier. Building on the work of [1], different word embeddings and more complex neural network types were compared for the same task in [2]. The comparison involved different types of qualitative measures from simple classification accuracy as well as precision, recall and F_1 score on a per class basis. These measures were computed for different embeddings and architectures using a grid search with a 3-fold cross validation. The most significant result was the finding that the quality of the word embedding has a significant influence on the overall quality of the model. Therefore, the `fastText` embedding was chosen which performed best. The best performing model using the `cw` embedding was outperformed by the best model with the `fastText` embedding by 8% in the 3 class classification accuracy. Regarding the neural network type

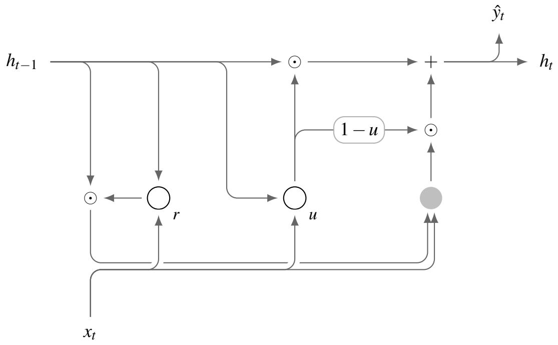

Figure 4: A single GRU cell. Inputs are x_t and the previous hidden state h_{t-1}, The white circles ○ represent the gates where r is the reset gate and u the update gate. The ⊙ operator is the element wise multiplication for applying the gating. Output is the new hidden state $h_t = \hat{y}_t$.

the gated RNN models are superior to other types like MLPs or CNNs over all used embeddings. This led to the choice of the GRU (`fastText`) model for the incivility analysis.

Viewed from a modeling perspective, both models are trained in a semi-supervised manner. The word embedding was trained unsupervised in an unsupervised way, then parts of the word embedding model were frozen and transferred to the actual classification for providing embedded word vectors as input. This is also known under the term *transfer learning*. Then the classifier was trained in a supervised setup, using these embedded word vectors as input.

If the process of embedding words is considered as a part of the pre-processing, it can be argued that the training of the classifiers is completely supervised, especially when all used word embeddings are pretrained and provided through external sources. The `Polyglot` library [5] is the source of the `cw` embedding and the `fastText` embedding was provided by Facebook AI Research [8].

5 Incivility Analysis

Both the GRU (`fastText`) and the MTBOW (`cw`) models are used to classify all of the 56.000 speeches.

For a change of incivility over time see Figure 5 where the average incivility of all speeches per year is computed by both models. The average negativity is the average of a sentence negativity from all speeches in a specific year. The sentence negativity is set to 0% if the sentence is labeled to be C_0 (neutral), 50% if C_1 (negative) and 100% for C_2 (very negative).

There are two main points that have to be mentioned. First, there is a huge difference in the bias. Regarding who the average negativity is defined it is not reasonable to assume that the average negativity is around 60%. The range of 32% to 36% seems a bit high but not unreasonable. Second, despite the bias difference both models capture a course which is quite alike. Both models agree that 2000 was a very rude year in politics. Surprisingly, from 2000 on the incivility in parliamentary speeches went down and never grew back to the tone in early 2000.

Another interesting comparison is to group the parliamentary speakers according to their party affiliation. Therefore, four parties were chosen, namely the ÖVP, SPÖ, FPÖ and the Green Party called Die Grünen. Now the average incivility for all speeches in one year grouped by the party affiliation was computed and visualized in Figure 6. Political science experts suggests that the opposing parties use a rougher tone in general which is supported by the results as follows: The ÖVP was a governing party while the Green Party were in opposition throughout the entire analyzed time period. Both were very stable in their tone like their parliamentary position and as stated in the hypothesis by political scientists the ÖVP used fewer sentences classified as containing uncivil content throughout that period than the Green Party. On the other hand the FPÖ and the SPÖ are switching between being a member of a governing coalition and opposition. In the case of the FPÖ, their incivility lowers as they gained power and finally managed to be a governing party in the beginnings of 2000. They got more and more polite until they reached the point of losing mandates to the newly founded party BZÖ and finally votes and governing power in the elections of 2007, which lead them back to a more frequent use of incivility in their parliamentary speeches. In contrast, the SPÖ was governing until the year 2000, suddenly losing in the elections and being left in the opposition. That led to an increase in the use of rude language until they got back to governing power in 2007.

References

[1] Rudkowsky E, Haselmayer M, Wastian M, Jenny M, Emrich Š, Sedlmair M. More than Bags of Words: Sentiment Analysis with Word Embeddings. *Communication Methods and Measures, Special Issue on Computational Methods*; 2017

[2] Kapla D. *Comparison of Different Word Embeddings and Neural Network Types for Sentiment Analysis of German Political Speeches*. [Diplomarbeit]. [Analysis und Scientific Computing]. Technische Universität Wien; 2019. URL: `http://katalog.ub.tuwien.ac.at/AC15493712`

[3] Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016 URL: `http://www.deeplearningbook.org`

[4] Cho K, van Merrienboer B, Gülçehre Ç, Bougares F, Schwenk H, Bengio Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*. 2014; abs/1406.1078 URL: `http://arxiv.org/abs/1406.1078`

[5] Rami A, Bryan P, Steven S. Polyglot: Distributed Word Representations for Multilingual NLP. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*; Aug. 2013; Association for Computational Linguistics: 183–192.

[6] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. *CoRR*. 2013; abs/1301.3781 URL: `http://arxiv.org/abs/1301.3781`

[7] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*. 2013; abs//1310.4546 URL: `http://arxiv.org/abs/1310.4546`

[8] Tomas M, Edouard G, Piotr B, Christian P, Armand J. Advances in Pre-Training Distributed Word Representations. *LREC*. 2018; abs/1712.09405 URL: `http://arxiv.org/abs/1712.09405`