SENTIMENT ANALYSIS OF TWEETS USING NEURAL NETWORKS

Team Name: VIP_CIL, Members: Benjamin Gallusser, Daniel Haziza, Rayhaan Jaufeerally

Department of Computer Science
ETH Zürich
Zürich, Switzerland

ABSTRACT

The automatic detection of a sentiment that a sentence or a short text passage conveys is one of today's major challenges in the field of Natural Language Understanding. It is a task that requires proficient semantic understanding of natural language but can yet be simplified to a straightforward classification problem with only a few different categories. In this work, we focus on the base case, a binary sentiment classification task, that aims to label microblog entries as generally positive or negative. To tackle this supervised learning task we introduce a model based on Recurrent Neural Networks and train it on a dataset of 2.5 million tweets. We evaluate it's performance and compare it to two standard approaches to message polarity classification. With our best model, we achieve a classification accuracy of about 87 percent.

1. INTRODUCTION

On an average day, 500 million messages are sent on Twitter. It has been proven that diverse meaningful information can be extracted from these messages, from the detection of earthquakes in real-time [1] to the prediction of stock market [2], or even the prediction of the outcome of an election [3]. Being able to classify the sentiment of a tweet as positive or negative is a first step in understanding complex thoughts of human beings, but is also required to go beyond the simple keyword based analysis. In marketing for example, traditional customer surveys can be replaced by *opinion mining* based on the sentiment analysis of Tweets related to a certain product [4]. Even though one can easily determine the buzz around the product, it's much harder to understand how positive potential customers are about it. Without explicitly telling a customer to give e.g. a one-to-five-star rating, we would like to be able to infer such a rating from comments about a product that have been written in natural language.

Tweets have several simplifying properties compared to natural language that make analyzing them a reasonable first step towards complete understanding of natural language. Most important of all, due to their limited length, they usually contain only one main idea. That means that the separating of several notions in one piece of text can be ignored for Twitter sentiment analysis.

However, Tweets also often contain spelling mistakes, intentional alterations of specific words or informal language. This poses a challenge for creating the dictionary of words that is used for representing words as one-hot encoded vectors in order to feed them to a neural network. We will try to address this problem with architectural choices concerning our neural network model.

In this paper, we introduce a novel method to classify the overall sentiment of a Tweet as either positive or negative. We first present two baseline approaches: a SVM classification based on word embeddings [5], and a Convolutional Neural Network (CNN). After that we introduce our novel method: the words of a tweet are embedded and processed sequentially by a Long Short-Term Memory (LSTM) cell.

2. RELATED WORK

Twitter sentiment analysis has been addressed by numerous publications in the past years. A good overview of the most recent successful approaches can be found in the SemEval-2016 report [6]. Specifically, task 4.A is very similar to what we are trying to achieve in this work. The major difference is that said task 4.A aims to classify tweets into three categories: positive, negative and neutral. For that reason, the results of this challenge are not directly comparable to our work.

Still we want to highlight the paper "SwissCheese at SemEval-2016 Task 4" [7], which seems to be the state-of-the-art approach for Twitter sentiment analysis with very few classes. The authors use distant supervision on a two layer CNN and then use a random forest classifier to combine the outputs.

An approach quite similar to ours is presented in [8]. The authors also use LSTM cells in the context of sentiment analysis. This approach turns out to be particularly useful for sentiment analysis due to the LSTM's ability to understand contextual relationships between words in the input.

3. MODELS AND METHODS

3.1. Data preprocessing

Tweets and similar online user-generated content differ slightly from traditional text corpora in the sense that they contain other semantic information such as hyperlinks, hashtags and other identifiers which carry a special meaning. For example, a hashtag represents a category which the text refers to. These characteristics have been accounted for in the preprocessing of the used dataset. Concretely, this means that all non-natural-language elements have

been replaced by generic tags such as <url> or <user>. Also, all letters are lower-case and all tokens including special characters are separated by whitespace.

In order to make the process of analyzing the sentiment of tweets more uniform and straightforward, we first remove words from the data that are not very frequent. This step eliminates words such as proper names and misspellings which can otherwise skew the results. In our model we used an occurrence threshold of 50 to filter out rare words. That is to say, any words which were seen less than 50 times in the corpus were discarded.

3.2. Baselines

3.2.1. Baseline 1: Tweet embeddings based on Word embeddings

The straightforward representation of words in a corpus is a binary vector with as many entries as there are words in the corpus. However, basic concepts of a vector space such as proximity do not appear in the obtained space. Furthermore, the high dimensionality of the vectors can lead to a computational bottleneck for large datasets. A low dimensional representation for words, a so called word embedding, aims to make simple operations such as averaging word representations more semantically meaningful. We choose an approach commonly known as *word2vec* [5] and obtain the word embeddings matrix by training on our dataset of Tweets that also serves as the training set for the actual sentiment classification task. The next step is going from word embeddings to embeddings of an entire tweet, which we realize by simple averaging of the embeddings for all single words in the tweet. Note that for this purpose an unknown word is represented as the zero vector. The obtained tweet embeddings and their corresponding labels are then fed into a binary linear Support Vector Machine classifier. In this approach, the order of the words in the text is not considered at all.

3.2.2. Baseline 2: Convolutional Neural Network

The second baseline that we propose is a standard Convolutional Neural Network (CNN) as proposed in [9]. Tweets are considered as a list of words, each word being embedded as a vector of dimension 128. 3 convolutional layers are fed from this input with a filter of size respectively 3, 4 and 5, and produce an output of size 384 (128*3). After a maximum pooling and dropout, a softmax layer predicts the sentiment. Dropout was used in this baseline as it prevents ove

3.3. Model description

3.3.1. LSTM

The Long Term Short Memory (LSTM) cell is a fundamental building block in the approach that we have taken. As tweets are limited to 140 characters and therefore only contain about 30 words[1] if the 140 characters are used, one could argue that all patterns occurring in the Tweet could be detected with a CNN. But one of the main properties

[1] with the empirical word length being around 5 characters

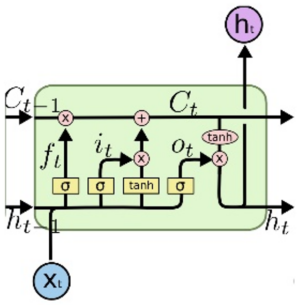

Fig. 1. Detailed data flow graph of an LSTM cell as it appears in an unrolled network. Figure from [11]

of CNN, namely translation invariance of the pattern detection, is not given in natural language. The position of a word or respectively a combination of words can change both the syntactic and semantic meaning of a sentence. We therefore need to use a model that accounts for the positional dependencies of the words in a Tweet, such as a Recurrent Neural Network. We decide to use the Long Term Short Memory (LSTM) cell [10], as it is able to detect very subtle long term dependencies. It has been shown that it is superior to a simple RNN cell as it addresses the RNN's vanishing gradient problem by introducing gating functions into the internal state dynamics. Each LSTM cell (as seen in Figure 1) is defined by the following equations

$$f_t = \sigma(W_f \cdot [t_{t-1,x_t}] + b_f) \quad (1)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * \tanh(C_t) \quad (6)$$

where C_{t-1} and h_{t-1} are the cell state and the output value of the previous time step, and C_t and h_t are the cell state and the output value of the current step in the unrolled network.

3.3.2. A variation: Character-based model

Going through the tweets, we noticed that many words were quite rare, because they were misspelled. For example "loveeee" for "love", "meeee" for "me" and so on. Additionally, two or more words may be stuck together ("#onemore"). In these cases, a character-level based approach may help to understand the meaning of the sentence. Furthermore, this is justified by the fact that several words come from the same root: great, greatest, greatly, ... for instance.

Thus, we decided to run the same model as previously described, except that a tweet is no longer a list of words, but a list of characters. Similarly, characters are embedded into vectors, and fed into the LSTM.

However, despite several attempts to fine-tune hyperparameters, we didn't manage to beat our previous word-based model. The character-based model converges to about 85 percent classification accuracy.

3.3.3. Further techniques

Dropout. In order to avoid overfitting the network to the training data, we apply *Dropout* [12], a common regularization technique for neural networks. During training of the network, the architecture is temporarily altered in the following way: a Bernoulli experiment with probability p is performed for each edge in the neural network, which determines if the edge will be used or set to zero for one particular batch. Both forward and backward propagation of the stochastic gradient descent are performed on this thinned-out network for one batch before a new randomly thinned-out network is determined for the next batch. Note that Dropout is only applied for training, whereas the prediction of unknown labels is done with the full neural network. Mathematically, dropout approximates ensemble averaging.

We set the dropout coefficient $p = 0.4$ in our model as it can be commonly found in the literature.

Adam optimizer. We used the state-of-the-art *Adam* algorithm [13] for the stochastic gradient descent optimization of the network's loss function.

4. EXPERIMENTS AND RESULTS

4.1. Input data

The training dataset consists of 2.5 million Tweets, where exactly half of it is labeled as positive $(+1)$ and the other half is labeled as negative(-1). The labeling is based on smileys [2] that were contained in the original Tweets, which were removed from the messages in the training set.

The test set consists of 10000 Tweets with labels unknown to the authors, but available on the machine learning platform Kaggle[3] to evaluate the performance of our model without any information leakage.

4.2. Setup

For this project, we choose to use Google's TensorFlow library [14] over other Deep Learning libraries for its seamless support of recurrent neural networks (especially LSTM networks) and many helpful features for the training pipeline that it provides out-of-the-box. Experiments were run on different Nvidia GPUs with several thousand cores and 8 to 12 GB of memory.

4.3. Baseline performance

4.3.1. Baseline 1: Tweet embeddings

We validate Baseline 1 with a 10-fold cross-validation and obtain a mean classification accuracy of 0.748. Note that the standard deviation of the classification accuracy is only 0.003, which can be attributed to the convex cost function of the SVM.

[2] such as **:-)**, **;-)**, **:-(** etc.

[3] https://inclass.kaggle.com/c/cil-text-classification-2017

4.3.2. Baseline 2: CNN

The mean accuracy in a 10-fold cross-validation that we obtained with this model was 0.73, with a standard deviation of 0.013 [4].

4.4. Final approach with LSTM

Our final model that we use to generate the predictions we submit to Kaggle has a classification accuracy of 87 percent. For training we pass over the entire training set four times and then do early stopping, as the network seems to have converged for certain by that time looking at the validation loss in Figure 2 and the validation accuracy in Figure 3. The total training time for four epochs is about 1 hour. We compare the accuracy of our final model to the baseline results in Table 1.

	mean	standard deviation
Baseline 1	0.748	0.003
Baseline 2	0.730	0.013
LSTM model	**0.870**	0.0048

Table 1. Classification accuracy for binary Twitter sentiment classification

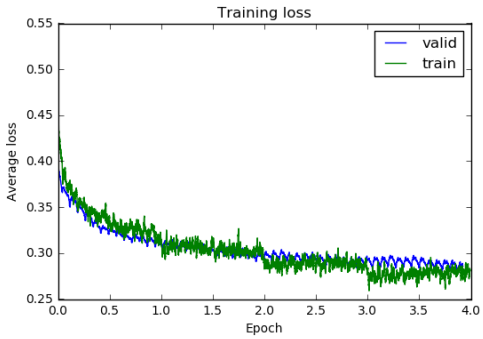

Fig. 2. Average loss during training of our final model

Fig. 3. Accuracy during training of our final model

5. CONCLUSIONS AND FUTURE WORK

In this article, we introduce a novel model and use it to classify tweets as either positive or negative. Compared to the baselines we introduce, the accuracy is improved by about 10 percent. The LSTM model is indeed able

[4] out of the box, without any parameter tuning

to correlate information from spaced words and takes the order of the words into account: this is not the case for our baselines.

We also discuss a possible variation, such as a character based model. Although this approach doesn't improve the performance, it still outperforms our two other baselines in terms of accuracy. As it is quite different from the word based approach, it could be interesting to combine these two approaches in a single predictive model. However, we had to leave this promising research way because of time constraints. With more powerful hardware, it should be possible to optimize this model even further with hyper-parameter fine-tuning.

For future work, we have two ideas which could improve the current method: experimenting with different Recursive Neural Network architectures (Bidirectional LSTMs, variations of LSTM cells, Convolutional LSTMs, Tree LSTM if syntax tree is available), and improving the tweet preprocessing (to identify numbers, split hashtags into words, attempt to fix typos and slang, et cetera). Once the binary classification problem is mastered, it would be interesting to see if the model can be extended to more, more subtle classes that categorizes the sentiment of a tweet without notable losses in accuracy.

6. REFERENCES

[1] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860.

[2] Johan Bollen, Huina Mao, and Xiaojun Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[3] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment.," *Icwsm*, vol. 10, no. 1, pp. 178–185, 2010.

[4] Alexander Pak and Patrick Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining.," in *LREc*, 2010, vol. 10.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

[6] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov, "Semeval-2016 task 4: Sentiment analysis in twitter.," in *SemEval@ NAACL-HLT*, 2016, pp. 1–18.

[7] Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi, "Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision.," in *SemEval@ NAACL-HLT*, 2016, pp. 1124–1128.

[8] James Hong and Michael Fang, "Sentiment analysis with deeply learned distributed representations of variable length texts," Tech. Rep., Technical report, Stanford University, 2015.

[9] Yoon Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[10] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] Chris Olah, "Understanding lstm networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015, accessed: 2017-07-04.

[12] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.