Sentiment Analysis of Slack Messages

Julius Hietala & Juha-Pekka Puska

***Abstract*—In this project we use an LSTM based RNN to label messages sent in a Slack discussion group based on their positive or negative sentiment and evaluate the distribution of the labels. The network is trained on a dataset consisting of 1.6 million labeled tweets from Twitter.**

I. Introduction

On the Web today, a constantly increasing amount of data is being generated. A large part of this data, such as social media, blogs, and comments is human generated text. Following this trend, there has risen a need to automatically process and analyze the contents of these messages to gain an understanding of the subjective content in the texts. A specific example might be a political party wanting to get feedback on the general atmosphere among their votes, or consumers looking for reviews on a product or service.

To address this issue, the field of *sentiment analysis*, also known as *opinion mining*, has developed. Sentiment analysis could be defined as "automatic analysis of evaluative text and tracking of the predictive judgments therein". It can therefore be considered a subfield within *natural language processing*, with the key focus on identifying subjective statements within the text. A closely related field is *information extraction*, where the goal is usually to determine the topic of a text, and provide a summary of it, whether in natural language, by keywords or other means.[1]

The main difficulty in assessing opinion from a text is that subjective content can be expressed in quite subtle ways, and a single negation can change the whole sentiment from negative to positive. This means that bag-of-words-approaches and methods based on word frequency can easily fail, especially in shorter texts. More sophisticated tools factoring in context, word order and grammatical structure have been developed to address these challenges. Also, determining an objective label for the sentiment in a text is often impossible. Therefore, the *inter-annotator agreement*, that is, the agreement between multiple annotators annotating the same text, provides a theoretical upper bound to the classification accuracy of any classifier.

Within the framework of sentiment analysis, a number of different targets could be considered. In the most general case, we want to extract the opinion *holder* (or holders), the opinion *type* and the opinion *strength*. In our study, we aim to classify the polarity, that is, negative or positive sentiment of Slack messages. The assumption is then that the holder of the opinion is always the author of the message, and the strength of the opinion is not considered.

As with other machine learning problems, deep learning methods have entered alongside more traditional rule-based approaches. An exampe of this is in [2], where a recursive network is introduced to solve the problem of rating sentences as negative or positive. The unique feature of the network is that the composition of words is formed by a tensor product, where the tensor is determined as a learnable parameter. The idea is that the positivity rating of a word is context specific, depending not only on the word itself, or its position, but also on the words around it. The authors of the referenced paper report that this model manages to classify sentences into positive and negative classes with 87.6% accuracy, and also achieves an over 80% accuracy in a more fine grained, 5-class classification task.

The main goal of this project is to create a state-of-the-art classification method to classify text sentences into two classes. In this case, the two classes will represent positive and negative sentiment. Subsequently, this method is then applied to messages obtained from the CS-E4890 Deep Learning course Slack channel to analyze the distribution of sentiment and qualitatively analyze the accuracy. The data used to train the model used in the method is explained in the next section.

II. Data

The unique aspect of our study is that the dataset used for training and evaluating was collected from different platforms. This presents the question of whether the features learned from our training dataset will be good predictors of sentiment polarity in the evaluation dataset.

The training dataset was downloaded from `www.kaggle.com/kazanova/sentiment140` and contains 1.6 million tweets, each annotated as positive or negative. The data itself has also been labeled using machine learning methods, which might affect the usefulness of the results of this work, although it is not a concern given the scope of this project (introductory course in deep learning). The data is in a tabular format and contains six fields:

- **target**: The polarity of the tweet (0 = negative, 1 = positive)
- **ids**: An identifying number for the tweet.
- **date**: The date of the tweet.
- **user**: Username of the author of the tweet.
- **text**: The text of the tweet.

In order to use the data in the network, it needed to be preprocessed. First, all the sentences were split into words using `https://spacy.io/`, and each word was converted into lower case, and cleaned from all non letter or number characters except for dots, commas, question marks, and exclamation marks. After this step, all words of all sentences were counted and ranked into an index-word dictionary based on the frequence of their occurence. The resulting dictionary had a size of around 200 000, but after qualitative analysis it was decided to be cut into 2 000 most frequent words to improve computational efficiency. In addition to the most frequent words, the following tokens were added to the beginning of the dictionary: SOS (start of sentence), EOS (end of sentence), CTX (”context specific” values used to replace words not found in the dictionary, and PAD (padding value to account for different lengths of sentences during batch processing of the model).

After completing the dictionary, all the individual sentences were tokenized using the dictionary. The integer index representing the SOS token was added to the start of each sentence, and the integer index representing EOS was added to the end of each sentence. The words of the sentences were converted into integers using the dictionary, and if not found from there, represented using the integer representing the CTX token. Therefore, a processed sentence could be represented as an array of integers, which can then easily be processed by the network. Processing of the whole dataset took approximately 30 minutes on a standard laptop.

The training and evaluation of the model was done utilizing the Twitter dataset, but the interesting dataset in terms of the problem of this project are all the messages sent by students and staff in the Slack discussion groups for the course CS-E4890 Deep Learning. The dataset has been obtained by using a web scraper utilizing Selenium (`https://www.seleniumhq.org/`), link to source code is included in the appendix. The messages contain a lot of punctuation, mostly due to code snippets, and username mentions in the form of ”@username” that needed to be handled in preprocessing. It seems like a fairly safe assumption that any sentiment reflected mainly in the words and not the punctuation. A histogram of the lengths of the messages after filtering is shown in 2.

Both the training and evaluation datasets were subjected to the same preprocessing step. This would, hopefully, make the learned features more consistent across the domains.

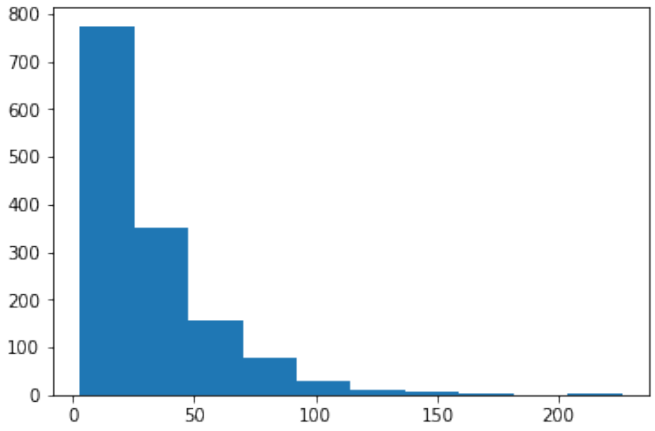

Fig. 1. Histogram of message lengths (words) in evaluation dataset

III. Methods and experiments

The implementation of the network was done with PyTorch [4]. We first implemented a custom dataset Python class, that converts the tokenized sentences into PyTorch tensor format that can be consumed by a custom data loader class. The custom data loader mainly handles converting data into batches and padding of the sentences. This custom data loader class could then be used by the network during training and evaluation. All the source code can be found behind the link in the appendix.

For the classifier, we used a recursive neural network with LSTM cells as defined in [3]. The LSTM architecture is hypothesized to be suitable for this task due to its ability to take into account information from further in the past. This is important for this task, due to, for example the handling of negations discussion in the introduction. LSTM networks were also discussed during lecture 5 of the CS-E4890 Deep Learning course, lecture slides can be accessed on MyCourses (`https://mycourses.aalto.fi`) on the CS-E4890 Deep Learning course page for further details.

The network processes multiple sentences at a time using batch processing. First, all the tokens of the words in the sentences are embedded into a vector space of specific size. Next, LSTM layers take as input the embedding of each word, as well as the output from the LSTM layer at the previous step. Since the sentences in a single batch can be of different lengths, all sentences are padded with a needed amount of 0s, which is done by the data loader class, so that the resulting lengths match in the batch. That length is also the amount of LSTM

cells needed to process a single batch. The output of the last LSTM layer is fed into a linear layer that transforms the final output into two scalars, representing the likelihoods for negative and positive sentiments. The decision function is simply a maximum of these two numbers. A diagram depicting the network layout is shown in the appendix. For training, the ADAM optimizer was used, with Cross Entropy Loss as the loss function. The size of the hidden output of the LSTM layer was set at 128. The model was trained for 10 epochs using minibatches of size 512. Training the model took approximately 5-6 hours on a standard laptop.

IV. Results

Before training, we split the Twitter dataset into training and test sets with an 80/20 split. On the training dataset, the model ahieved a 92% accuracy and on the test dataset, a 83% accuracy, which is close to the accuracy of the recursive network in [2].

Since the evaluation dataset of Slack messages is not labeled, the model is evaluated in a qualitative way, by looking at rating of the sentences manually and also reporting the distribution of the positive and negative messages. Below we have listed sentences with the most positive and negative ratings, respectively. As can be seen, some of the sentences do express some sentiment, but mostly the ratings seem quite arbitrary.

+ set the channel purpose for finding project partners
+ yes the exercise sessions . thanks !
+ in CTX the type torch CTX is actually float doing your CTX CTX will convert torch CTX to torch CTX
- are there any tutorial sessions this week ? the booking option is not available on CTX edited
- when i fetch data and click on it it says empty . i can t seem to download the data .
- i didn t get any feedback ?

In Fig 2 we also show the distribution of positive and negative messages according to the model. 51% are deemed positive, and 49% negative. Intuitively from the author's perspective this seems incorrect when looking at the evaluation messages, the percentage of negative messages is unreasonably high.

V. Conclusions

In this study we aimed to use labeled Twitter data to train a model suitable for processing text and use it to evaluate the sentiment contained in short messages. The model performed well on the training and test datasets, but when evaluated on the Slack message dataset, the outcomes were not very useful. The main hypothesis

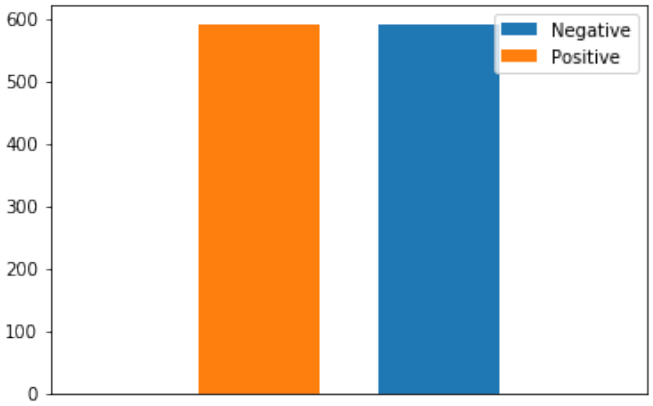

Fig. 2. Histogram of label distributions in the evaluation Slack dataset

is that the limited scale of positive-negative is not very useful when trying to analyze text that is more or less objective (university course communications) and does not contain proper sentiment within this scale. Also, the training dataset of tweets most likely suffers from the same issue. This issue could most likely be solved by using a more suitable and more robustly trained dataset with a larger multi class scale. Also, the Slack messages contained a lot of specialized terms and domain specific language, which meant the model probably did not generalize particularly well.

References

[1] Pang, B., & Lee, L. *Opinion mining and sentiment analysis.* Foundations and Trends in Information Retrieval, 2(12), 1-135, 2008

[2] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. *Recursive deep models for semantic compositionality over a sentiment treebank*. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642), 2013

[3] Argawal, S. (19.2.2019) *Sentiment Analysis using LSTM (Step-by-Step Tutorial)* Retrieved from https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948

[4] Chintala, S., Gross, S., Paszke, A., *Pytorch github* from: https://github.com/pytorch

[5] Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q.V. and Ng, A.Y., *On optimization methods for deep learning.* InProceedings of the 28th international conference on machine learning (ICML-11)(pp. 265-272).

[6] Goodfellow, I., Bengio Y., Courville, A., *Deep Learning*, MIT Press, 2016 from:http://www.deeplearningbook.org

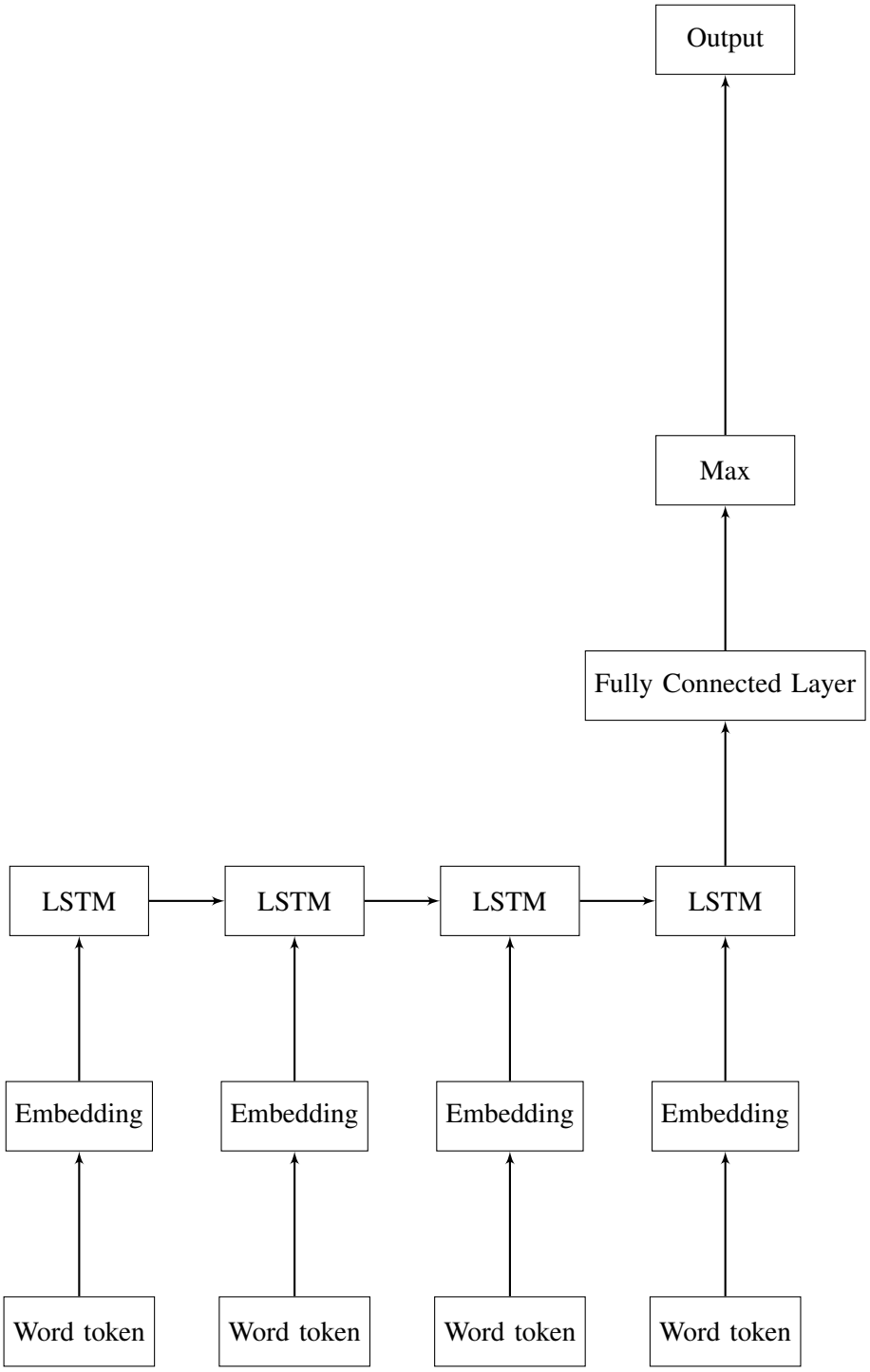

Fig. 3. Network architecture for a four word sentence

APPENDIX

Source code repository of the project in GitHub:
`https://github.com/hietalajulius/deep_learning_aalto`