Text Sentiment Classification

Iveri Prangishvili
ETH Zurich
iverip@student.ethz.ch

Rishu Agrawal
ETH Zurich
ragrawal@student.ethz.ch

Neerav Karani
ETH Zurich
nkarani@student.ethz.ch

Abstract

In this paper, we look at the problem of sentiment analysis of tweets. The goal is to classify a given tweet as either positive or negative. This can be quite valuable for gauging the aggregate opinion of a large group of people regarding a product, a company or a socio-economic event like an election. Such information can further be used to predict, for instance, movie revenues or stock exchange behaviour. We use two approaches based on an aggregate of long short term memory (LSTM), gated recurrent units (GRU) and convolutional neural networks (CNN). We also introduce a novelty of using two dynamic channels with different word embeddings, namely word2vec and GLoVe in one of our models which gives a higher classification accuracy as compared to using either one of the word embeddings alone.

Keywords sentiment analysis, word embeddings, word2vec, GLoVe, GRU, LSTM

1. Introduction

Twitter is a rapidly growing micro-blogging and social networking website. On this website, users interact with each other and express their opinions via tweets, which have a character limit of 140 characters. Unprecedented amount of data gets exchanged on a daily basis. As of 2016, it has 310 million monthly active users [1].

Potentially, a lot of useful information such as individual and aggregate opinions about certain products or companies, election trends, etc. can be extracted by carrying out a sentiment analysis on this data. Traditionally, this kind of work has been done for text data from websites that review products or movies[2]. However, the natural language processing of twitter data is generally considered to be more difficult that processing larger texts. One reason for this is that tweets generally contain unstructured text. Many users use unusual acronyms in order to comply with the 140 character limit. Additionally, it is not uncommon for tweets to contain multiple negations, sarcasm or irony, so that they convey a negative opinion while seeming to convey a positive one, or vice versa.

In this paper, we describe the method that we have implemented to solve this twitter sentiment classification problem. We adopt two methods for this. The first method involves a convolutional neural network based approach and introduces a novelty in that we use two dynamic channels with separate word embeddings. In our experiments, this leads to a remarkable improvement in the classification accuracy over either using one dynamic channel only or using one static and one dynamic channel.

The second method involves the construction of a model that first learns sentence representation with long short-term memory. Afterwards, the semantics of sentences and their relations are adaptively encoded in document representation with gated recurrent neural network.

The rest of the paper is organized as follows. In section 2, we briefly review the important approaches for sentiment analysis and point out their pros and cons. In section 3, we provide a detailed description of our approach to the problem. This is followed by a presentation of the results on test data in section 4. Finally, in section 5, we discuss the strengths and weaknesses of the suggested approaches as well as some of the general issues we faced.

2. Literature Review

A simple naive approach to sentiment analysis may be to consider the individual words of a tweet independently. However, the meaning of a word can depend on the context and the sentence structure. Therefore, a model like a bag-of-words model that ignores sentence structure does not work well, especially for small texts like tweets, where we typically have a single sentence to classify the tweet as positive or negative. [3]

Many machine learning algorithms such as Naive Bayes, Entropy Maximization and Support Vector Machines have been used for sentiment analysis of twitter data [4]. They claim that SVM performs better than the other approaches. In [5], it has been proposed to use, along with prior polarity of words and part-of-speech features, twitter specific features such as hashtags and retweets. They report that these specific additional features help marginally for sentiment classification of twitter data.

More recently, approaches based on recurrent neural networks and convolutional neural networks (CNNs) - that have experienced tremendous success in computer vision - have been shown to be very effective in natural language processing tasks such as semantic classification as well [6,14]. In these approaches, words are represented using word embeddings [7]. A word embedding is a mapping that maps a word to a real-valued vector. The size of this vector is typically of the order of a few hundreds - significantly smaller than the size of the vocabulary. Two of the most successful approaches of learning these word embeddings are word2vec [8] and GLoVe [9]. While word2vec uses a 2-layer neural network to learn the word embedding, GLoVe adopts an unsupervised learning approach. In both these representations, words that are used in a similiar context in the training data are located close to each other in

CIL '16 July 2016, Zurich, Switzerland

ETHZ . . . $15.00

the mapped space. In [6], they perform sentiment classification by training a CNN on top of word2vec vectors. We use this approach as a reference and then add the novel modification of using both word2vec and GLoVe channels in the CNN.

We also implement the approach shown in [14], that uses a RNN based approached instead of CNN, as it proved to be more effective and produced better results. The idea behind RNNs is to make use of sequential information. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. But for language modelling, this is not a feasible assumption as to predict the next word in a sentence, the model needs to be aware of the context, i.e. the previous words and the sentences it succeeds. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNNs is that they have a memory which captures information about what has been calculated so far. In theory, RNNs can make use of information in arbitrarily long sequences. But a vanilla RNN isn't capable of maintaining long term dependencies because the gradients tend to either vanish or explode. This makes gradient-based optimization method struggle because the effect of the long-term dependencies is then altered by the effect of short-term dependencies.

On the other hand, an LSTM unit is capable of deciding whether to keep the existing memory via the usage of gates instead of overwriting its content at each time-step, as in the case of a traditional or vanilla RNN. If the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information over a long distance, hence, capturing potential long-distance dependencies. A GRU is a more simpler form of an LSTM unit that adaptively captures dependencies of different time scales and is capable of maintaining long term dependencies.

3. Models and methods

3.1 Data Preprocessing

We were provided with a large training set of tweets. Each tweet was labelled as either positive or negative. These tweets had been pre-processed such that all words were separated by a single whitespace. Also, the smileys - that were used to label the tweets as positive or negative - had been removed.

Furthermore, we did the following processing of the training data.

- We padded all the tweets such that they all contained the same number of words as the 99th percentile of the lengths of tweets in the training dataset.
- We constructed a vocabulary that consisted of every word that appeared at least 5 times in the training dataset. From this, we created a look-up table that mapped each word in the training dataset to an integer. Each tweet is a vector of integers.

3.2 Text Representation

In our model, we are using two methods for word embeddings: GLoVe and Word2vec. We computed the GLoVe word embedding for each word in the vocabulary. As this step was taking too long, we used only 10 percent of the total training dataset to compute the GLoVe word embeddings. We also computed the word2vec word embedding for each word in the vocabulary.

To optimize the quality of the word embeddings, we carried out several experiments to determine the best parameters.

Our implementation of GLoVe embeddings uses the adaptive gradient descent algorithm and includes bias variables. To assess the quality of GLoVe word embeddings in the earlier stages of the project, we used two methods: intrinsic and extrinsic. In the intrinsic method, we evaluated the quality of embeddings on specific intermediate tasks. For instance, we created a list of semantically similar words that should have been clustered into groups, and based on these words, we tried to understand which parameters had a positive effect on the quality. To get a graphical representation of the words, we needed to transform embeddings into two dimensions. In order to achieve this, we used PCA to determine the principal dimensions and visualized the resulting embeddings. These graphs gave us good bases to judge the effects of tuned parameters in the early stages.

For the next stage we evaluated GLoVe word embeddings for the real task at hand. We coupled the optimization of the GLoVe parameters with simple classification methods like logistic regression and random forests and judged the effects based on the classification errors. Once we found the optimal parameters and weren't able to increase the classification accuracy through the use of GLoVe embeddings and simple classification models, we moved on to exploring more complex approaches that led us to Convolutional Neural Networks and Recurrent Neural Networks (LSTM and GRU). Thereafter, we ran several experiments to see the effect of tuning the dimension parameter of GLoVe embeddings. Through several experiments we determined that optimal dimension was 40.

Our word2vec implementation had a shorter running time, so we could run it on the complete dataset, while using the same vocabulary as we did with the GLoVe method. To find the optimal dimension of embeddings, we ran several experiments with CNN using one channel of word2vec to see the isolated effect of the dimensions. The optimal value of the embedding dimension for word2vec method was found to be 100.

3.3 CNN Architecture

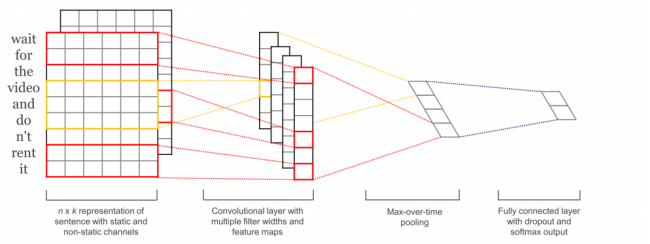

Figure 1. CNN architecture with two channels for an example sentence.

We use the same CNN architecture as [6]. This is shown in Figure 1.

The first layer consists of the two channels - constructed using the two word embeddings. In the second layer, convolutions are performed with filter sizes of 3, 4 and 5 for each channel. This is followed by a max-pooling layer, where only the maximum response to each filter is preserved. The result of the max-pooling is concatenated to form a long feature vector for each channel. This vector for both the channels is added and finally, the weighted sum of this feature vector is finally passed through a softmax function to obtain the probability of this tweet being positive or negative.

S No	Model configuration and parameters	Training Accuracy	Training loss	Validation Accuracy	Validation loss
1	LSTM+GRU, word2vec embedding	0.8794	0.2807	0.8763	0.2823
2	CNN+GRU, word2vec embedding	0.8754	0.2895	0.8704	0.2956
3	CNN, GLoVe + word2vec embedding	0.8792	0.2747	0.8654	0.3166
4	CNN, GLoVe embedding	0.8522	0.3375	0.8109	0.4523
5	TF-IDF with Logistic Regression	0.8310	NA	0.8215	NA
6	Logistic Regression, GLoVe embedding	0.67	NA	0.67	NA
7	Random Forests, GLoVe embedding	0.69	NA	0.67	NA

Table 1. Results obtained using different models on the given dataset. Models 1 and 3 are used as the final submission.

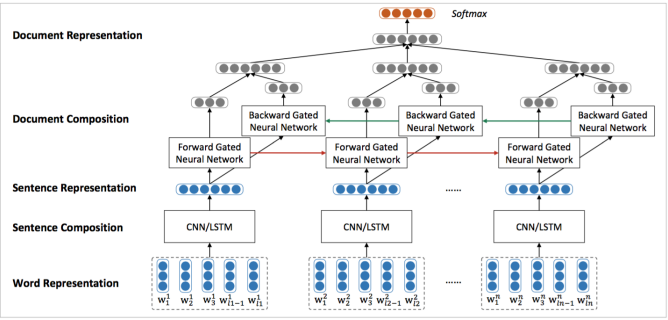

Figure 2. LSTM + GRU architecture for an example sentence.

3.4 LSTM + GRU

We use the same architecture as [14] for our final model. This is shown in Figure 2.

This model is based on the principle of compositionality, which states that the semantics of an expression is dependent on the semantics of its constituents. Each word in the corpus is represented as a word2vec embedding. The model utilises a LSTM to produce sentence representations from word representations since a vanilla recurrent neural network is not very capable in capturing long-term dependencies. LSTMs are also preferred as they are not plagued by the vanishing gradient problem. Then, a gated recurrent neural network is utilised to adaptively encode semantics of the tweets and their inherent relations in the corpus. These representations are finally used as features to classify the sentiment label of each tweet, as seen in the CNN architecture.

3.5 Regularization

We implemented dropout regularization [10] after the max-pooling layer - that is, in each step of the CNN training, we randomly drop some of the elements of the feature vector in this layer and update the weights only of the other neurons. This is a well-known technique for preventing over-fitting.

We implemented l2-regularization to combat overfitting and applied l2-norm constraint on weight vectors in the last layer of the network.

Another regularization approach is to make one of the channels static, while evolving the other. However, we initialized both our word representation channels randomly and kept them dynamic (allowed to be learned by the network) as we got better validation performance using this approach.

4. Results

In order to address the problem of text sentiment classification, we implemented and experimented with several models, varying the different parameters that influence the outcome. We report the results obtained by the various models, in their best configuration, using the most optimum parameters according to our findings, and tabulate them as shown in Table 1. We can see that the LSTM + GRU model provides the best accuracy compared to other models, which is expected, as it captures long term dependencies that ensures that tweets that have a similar meaning or belonging to a particular context belong to the same class. This becomes important in classifying tweets that exhibit traces of sarcasm or satire to express an incongruous sentiment.

We also observe that the model utilising a dual channel embedding performs distinctly better than the model utilising a single channel embedding. In particular, if we observe the training and validation losses, we can see a stark difference in the range of the loss values. The obtained results suggest that the model utilising a single channel embedding is prone to overfitting the data and is more likely to produce a lower accuracy compared to the model utilising a dual channel embedding.

In order to demonstrate the novelty and benefits of using a dual channel based embedding over a single channel embedding, we also performed several experiments on the given dataset using the CNN model. As seen in Figure 3, the validation loss for a dual embedding based model consistently decreases along with the training loss, before reaching a steady state, and is also lower than the loss obtained by a model trained using a single embedding.

It is also worthwhile to note that even a theoretically elementary model such as TF-IDF, that simply stores a count of the term frequency and the inverse document frequency of the words in the corpus, with logistic regression, is extremely effective in the task of classifying positive and negative tweets compared to classification using random forests and logistic regression with GLoVe embeddings.

5. Discussion

Using the various methods discussed in this paper and the given Twitter dataset, we complete the task of sentiment classification of tweets with an accuracy of about 87% in the public dataset at Kaggle using the GRU+LSTM model and an accuracy of about 86% using the CNN model with dual channel word embedding. Superficially, these results are at par with the results obtained by the state of the art sentiment classifiers. However, the scope of the results presented in this paper is trained and tested on a very small

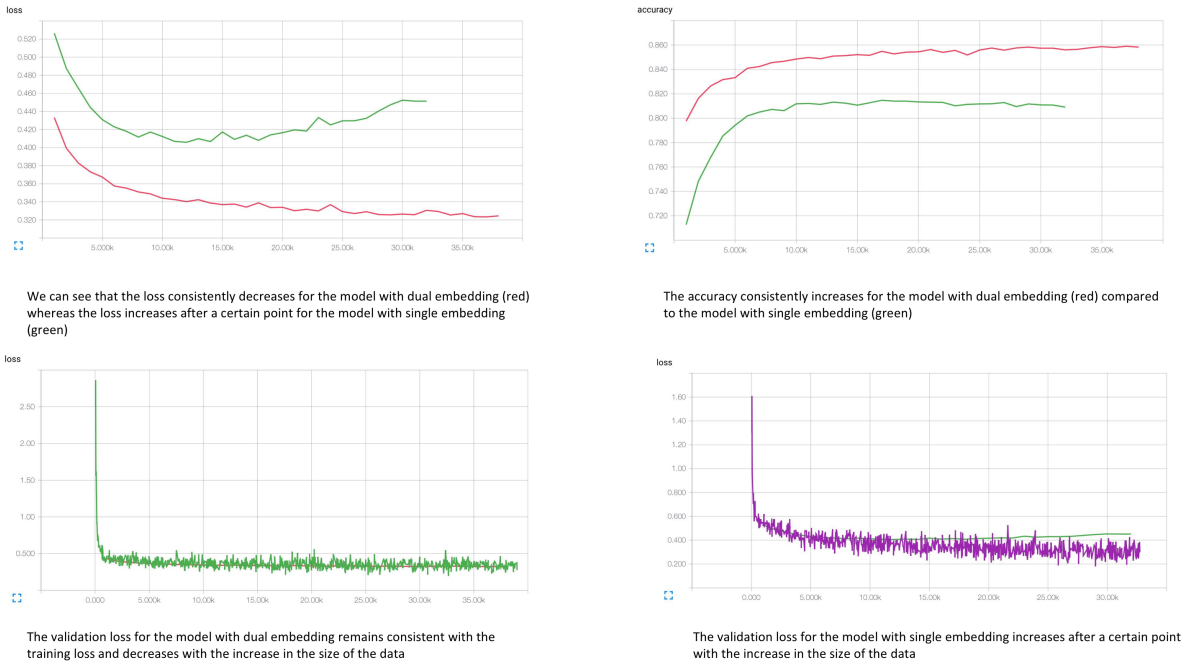

Figure 3. Graphs measuring accuracy and loss for models with single embedding channel(GLoVe) and dual embedding(GLoVe + word2vec) channel

amount of tweets (in millions) compared to the billions of tweets that are sent out everyday. As cliche as it sounds, in order to obtain a more robust and reliable classifier, it is quintessential to train the models on a larger dataset.

Despite the ever growing prowess of modern computations, one glaring limitation of the models presented above is the computer hardware, or lack of it. We trained all our models on a dual core 2.6 GHz i5 CPU with 8 GB RAM using the Tensorflow/Theano backend and the Keras library. Each model took about 5 hours of training time for one epoch on the complete dataset. This is still one of the major challenges of using neural network based models for classification as they require powerful hardware and abundant computation power to provide acceptable results.

5.1 Strength and weaknesses using two dynamic channels

The rationale behind using two channels was to battle the conundrum of overfitting, at the early stages of training the model, and boost the prediction accuracy. Traditionally, in a two-channel architecture, the same embedding layers would be used, however, one would be held static and the other's weight would be modified during the training. In the scenario where non-static channel would start overfitting the weights in the embedding layer, the static channel would potentially neutralize this effect since its embeddings are not adapting. This architecture would make sense if the pre-trained embeddings are of good "quality". However in our case, the "quality" of embeddings is suffering due to several factors, one of which would be the size of the data. Furthermore, we trained GLoVe embeddings only on 10% of the original data due to the computational limitations.

The weakness of this standard two channel architecture in our case would come from the relative poor quality of the embeddings. To overcome this challenge, we decided to utilize two channel architecture to our advantage. Instead of relying on the effectiveness of one embedding model, we introduced two individual ones: GLoVe and Word2Vec, and made both of them non-static. Each of the embeddings counteract their strengths and weaknesses, and by allowing their weights to adapt during training, their respective performances increase. Moreover, making both channels non-static also allowed us to employ the provided data in a more effective manner. Word2Vec embeddings were trained on the complete dataset, thus, increasing the representation of the data in the model. The distinctiveness of the two embeddings used in our architecture, and their individual weight adjustments, result in the evasion of overfitting in the early stages of the model training. However, the later stages of overfitting, due to having two non-static channels, is not avoided. Thus, it becomes crucial to pay attention to the validation metrics at the late stages of training (in our model, after 110000 steps) to detect potential overfitting.

6. Acknowledgement

We would like to thank Prof. Thomas Hofmann for his fantastic lectures and valuable advice during the project. We would also like to express our gratitude to the excellent blogs [11], [12] and [13], which we extensively followed to understand many topics during the course of the project. All in all, we learned that pursuing such problems, in general, is an art which requires a blend of creativity and perseverance to lead to an optimum result, which begs the question: Is an optimum result ever optimum or is it just an illusion?

7. References

1. Twitter website. https://about.twitter.com/company

2. K. Dave, S. Lawrence and D.M.Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th International Conference on World Wide Web (WWW), 2003, pp. 519528.
3. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of EMNLP, pp. 79–86, 2002.
4. Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. Technical report, Stanford. 2009.
5. Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 3644, 2010.
6. Yoon Kim. Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882v2. 2014.
7. Yoshua Bengio, Holger Schwenk, Jean-Sbastien Sencal, Frderic Morin, Jean-Luc Gauvain. Neural Probabilistic Language Models. Volume 194 of the series Studies in Fuzziness and Soft Computing pp 137-186. 2003.
8. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781v3. 2013.
9. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. 2014.
10. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-19.
11. http://www.wildml.com/about/
12. http://colah.github.io/about.html
13. http://ataspinar.com/about/
14. Duyu Tang, Bing Qin, Ting Liu. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification, EMNLP 2015
15. J. Chung, Caglar G., K. Cho, Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Text Sentiment Classification

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	First name(s):
PRANGISHVILI	IVERI
AGARWAL	RISHU
KARANI	NEERAV

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Zurich, 1st July, 2016.

Signature(s)

Rishu Agrawal

[Neerav Karani]

Iveri Prangishvili

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.