Deep Neural Networks for Twitter Sentiment Classification

Alexandru Țifrea, Jinank Jain, Saurav Shekhar, Seyed Rouzbeh Hasheminezhad
Team Name: Beauty and the bits
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—The problem of sentiment classification is a classic one in Natural Language Understanding, with a wide range of tools and techniques available. We analyze the performance of some standard methods on a dataset of 2.5M Tweets, and use the insights from the results to combine and improve the models in a novel way. With the best model, we achieve an accuracy of 88.98% on the public test set.**

I. Introduction

Sentiment classification has gained noticeable attention from the machine learning research community within the recent years. The aim of sentiment analysis is to infer opinion and emotion from written text, which could be considered significantly challenging due to the intrinsic complexity of natural language attributed by the presence of humor, sarcasm and irony. Sentiment classification has numerous applications like, for instance, the automatic assessment of public opinions about a certain topic from tweets, messages, reviews and comments. This can be of significant importance for any company with an online presence, political parties, sociologists etc. The classification in its simplest form can be binary (positive or negative) or more fine-grained. In this paper we focus on the former.

For training purpose we are given a dataset of 2.5 million tweets, split equally between positive and negative samples, we aim to train a sentence-level sentiment classifier with a competitive accuracy compared to the state-of-the-art sentiment classification models. To stand on the shoulders of giants, we briefly mention and discuss pieces of research that we drew inspiration from in II. In IV we describe the details of the task and the tools we have used. In III we introduce our trained models in detail and the baselines against which we compare them. In V we introduce the results acquired from the trained models. Finally we present our final remarks in VI.

II. Related Work

SemEval is a prominent competition for semantic evaluation. According to the results and standings on Task 4 (Sentiment Classification) within recent years, deep neural nets have demonstrated that they capture the salient complexities of this task much better and they significantly outperform other models. The competition winner in 2016 on Task 4 was the SwissCheese model from ETH, Zurich[1] which employs the use of Convolutional Neural Networks (CNNs), building on top of other works that used CNNs for this task. Another recurring idea was the use of ensembles of different models. As an example the architecture explained in [2], which is an ensemble of Recurrent Neural Nets(RNNs) and CNNs, showed a noticeable performance in the sentiment classification subtask at SemEval in 2017, also the best performing method ending up in the first place for all the five english subtasks including sentiment classification on twitter data was based on an architecture using CNNs along with an array of forward/backward Long Short Term Memory (LSTM) units[3].

OpenAI recently released an article on learning to generate reviews [4] where they train a multiplicative LSTM model to generate text from Amazon reviews. The final trained model embeds a sentence into a 4096 dimensional embedding. They also discover that out of the 4096 neurons (which can be seen as features), one specific neuron can be interpreted as a sentiment-controlling neuron. Using a supervised linear classifier of the features produced by this unsupervised model, the authors were capable of outperforming a some state of the art methods on the Stanford Sentiment Treebank while using a lot less data.

Building on the foundation of the ideas from the models mentioned in this section we come up with our own models, and develope basic baselines to compare our developed models to.

III. Models and Methods

We explored the behavior of a number of neural network architectures which will be described in this section. Ultimately, we plugged the models that showed the best results on the validation set in an ensemble. Most of the models that we used required converting the tweets into some sort of vector representation (embeddings) starting either from word level (word embeddings), sentence level or character level. We can treat these embeddings as features and train a machine learning model on them.

A. Preprocessing

We parse every tweet into tokens before applying any processing. We then build the vocabulary consisting of the unique tokens. For efficient processing, we only keep tokens that occur at least 5 times in the corpus. We use the `<UNK>` symbol for tokens that do not appear in the vocabulary.

We analyzed the tweet length distribution (see Figure 1) and picked a sentence lenght of 40. We trim longer tweets

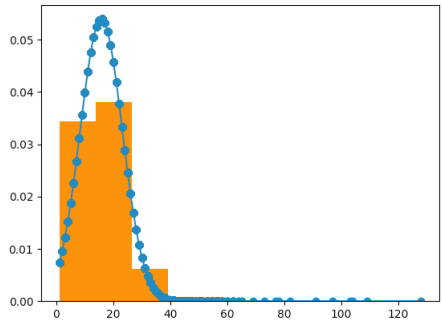

Figure 1: Tweet length distribution in the 2.5M-tweet corpus (histogram and approximation to a normal distribution)

to this length. Shorter tweets are padded using the <PAD> symbol. We did not remove the punctuation marks, because we thought that in the context of tweets, they can convey important information about the feelings of the author and could therefore help for the purpose of this task. We did not perform any stemming because we did not want to affect the way the words were spelled out, as this is also indicative of a number of things in an environment like Twitter.

B. Word Embeddings

We tried two different kinds of word embeddings: GloVe [5] and word2vec [6]. For word2vec, we used 400-dimensional pretrained embeddings trained on 400M tweets and a vocabulary with 3M tokens from [7][1]. For GloVe we used 200-dimensional pretrained embeddings, trained on a corpus of 2B tweets with 1.2M unique tokens. For words that did not have an embedding, a random vector with values in the interval $[0.0, 1.0)$ was used instead. During training, we backpropagated the gradients through the word embeddings as well, updating them accordingly.

We used pre-trained word embeddings due to the fact that they are trained on a corpus much bigger than what we are given for training.

C. OpenAI Features

We extract OpenAI features and try out different classifiers (SVM, Multi Layer Perceptrons etc) on them. We also tried to include these features in ensembles with other models.

To extract OpenAI features from our dataset we used a pretrained model[2]. We transformed the sentences in batches of size 10k and saved them to disk.

D. Baselines

The first baseline that we used was to implement a linear classification model on top of the Glove word embeddings. We average the word-embeddings of all the tokens in a tweet to obtain a sentence-level embedding and then use this to train a hinge loss-based linear classifier. In addition to this, we also implement the same linear classifier over OpenAI features which are known to give good results when used with linear classifiers (see section II). We used an online linear SVM formulation with hinge loss and $L2$ regularization penalty.

E. Recurrent Neural Networks (RNNs)

Previous work on related tasks showed that recurrent architectures lend themselves well to this sort of problems. They manage to capture the relationships between words in the sequence and provide more informed features to embed the tweets. We explored a number of recurrent architectures, but ended up using only two of them in the final ensemble, based on their performance on the validation set. The first model is a gated recurrent network (GRU) with 1024 hidden units followed by a softmax layer. The second one is a bidirectional LSTM (1024 hidden units) followed by a fully connected layer of 512 units and a softmax layer. For both architectures we used the GloVe pretrained embeddings to reduce the memory consumption (because they are smaller than the word2vec embeddings that we used).

F. Convolutional Neural Networks (CNNs)

1) Input: A tweet is represented as a $N \times D$ matrix, where each row represents a D-dimensional word embedding. The number of rows (words) N is fixed to 40. Unlike with image though, here we can only apply 1D convolutions along the vertical axis (see Figure 2). We used the 400-dimensional word2vec embeddings for these architectures.

2) Convolutional Architectures: We used two different convolutional architectures, *SeqConv1* and *SeqConv2*. Details about them can be found in Table I. While *SeqConv1* is somewhat similar to what has been introduced in the SwissCheese paper[1], *SeqConv2* is deeper (i.e. has more convolutional layers and more fully connected layers at the end). It is important to note that using convolutional networks for this tasks brings a lot of the advantages that were noticed on visual computing tasks: the filters will extract data-dependent features that are relevant for the task they were trained on and the pooling layers will provide a sort of positional invariance of these features. Filters in these convolutional networks can be seen as working on different length segments of the sentence. Also, since tweets are short, we don't really need to model long term dependencies.

G. CNN + LSTM

Both a recurrent neural network and a CNN will provide a sentence-level (tweet-level) embedding. We decided to concatenate the features obtained with these two methods and use a fully connected layer of 1024 units to classify the tweets. We used an LSTM with 1024 hidden units and three

[1] http://www.fredericgodin.com/software/
[2] https://github.com/openai/generating-reviews-discovering-sentiment/

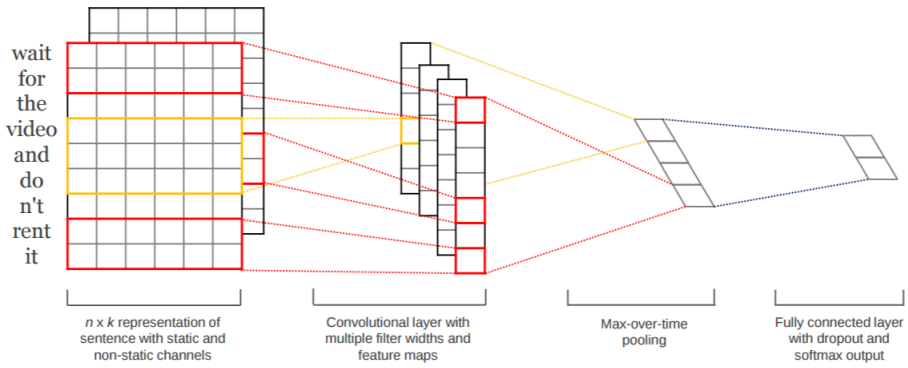

Figure 2: CNNs for sentence classification example, taken from [8]

Name	SeqConv1	SeqConv2
initial	word2vec-400	word2vec-400
conv1	128 kernels of size 3	256 kernels of size 3
conv2	-	128 kernels of size 3
maxpool1	pool size 2	pool size 2
conv3	-	64 kernels of size 3
conv4	-	64 kernels of size 3
maxpool2	-	pool size 2
fc1	1024 units	1024 units
fc2	-	512 units
softmax	yes	yes

Table I: Convolutional Architecture details

CNNs, each with one convolutional layer (128 filters each, with a kernel size of 3, 4 and 5, respectively), followed by a max pooling layer. This can be seen as a simple ensemble of a CNN and a LSTM model followed by a classifier. The classification based on this mixed sentence-level embedding gave better results than each of the individual sentence-level embeddings that were used to create it (i.e. simple LSTM or simple CNN).

H. Ensemble

After picking the most promising models in terms of prediction accuracy, they were all plugged into a simple, majority voting system. The final prediction of the binary classification problem is the label that is predicted by the majority of the five models that we selected: the GRU, the bidi LSTM, *SeqConv1*, *SeqConv2* and the CNN+LSTM model.

I. Other models

Using convolutional networks, we tried using two input channels for the embedding layer. One input channel was constructed using word2vec and the other layer was constructed using GloVe word embeddings. We also tried to use a two-layer LSTM or to use a mixed sentence-level embedding created using an LSTM and the OpenAI features mentioned above. None of these methods showed convincing results on the validation set. We also looked into well known topic modeling algorithms like LDA but could not develop a convincing sentiment classification algorithm out of it.

IV. Experiment setup

Two training sets were given, a small one, with 200k tweets, and a big one, with 2.5M tweets. All the results come from models trained on the large dataset. The dataset was split into a training set and a validation set. The validation set was composed of 50,000 samples.

Training was done on the Euler ETH cluster and on Azure GPU machines. Code was written primarily in Python, using the Keras framework [9] with a Tensorflow backend [10]. We used Scikit-learn [11] for linear classifiers and SVM.

All the neural networks are trained using Adaptive stochastic gradient descent, on mini-batches of 64 samples. We use Adam [12] with a learning rate of 0.0001 as the update rule. We also tried RMSProp and other values for the learning rate, but Adam showed the best results across the board for the models that we used. For regularization we use dropout [13] after every fully connected layer and for all the LSTM and GRU cells. The dropout rate is set to 0.2 (20% of the units were dropped). After every 200,000 iterations we save a checkpoint of the model and the evaluate the validation accuracy. We use an empirical way of validation-based early stopping: we interrupt the training when the validation accuracy stops showing improvement for a significant number of iterations.

V. Results

A. Baselines

The accuracy scores of the baseline classifiers are **0.676** for GloVe averaging and **0.8334** for the linear classifier on OpenAI features. This confirms the hypothesis that a sentence-level embedding obtained as averaging the individual word-level embeddings fails to capture complex relationships between the words in the sequence. It also shows that a linear classifier is not able to get good results out of the unsupervised, high-dimensional OpenAI features.

Model	Accuracy	
	Train	Validation
Avg. Glove + Lin. Classif. (baseline)	n/a	0.676
OpenAI + Lin. Classif. (baseline)	n/a	0.833
SeqConv1	0.9000	0.8807
SeqConv2	0.9009	0.8831
CNN + LSTM	0.9061	0.8883
GRU	0.9039	0.8909
Bidirectional LSTM	0.8987	0.8908

Table II: Accuracy comparison of the different models

B. *Static vs Trainable word embeddings*

After the first few (fixed) number of batches, fixed word embeddings gave an accuracy of **0.79** while trainable word embeddings gave an accuracy of **0.83**. Since trainable word embeddings were clearly superior, we did not look further into using static embeddings. Training word embeddings from scratch on our own dataset did not bring any improvement.

C. *Deep learning models*

The results are presented in Table II. It is easy to notice that deep learning models show much better performance over the baselines. This is because deep learning models have very low bias and can represent a fairly complex data space given enough data. Moreover, supervised models like RNNs and CNNs give better performance than the unsupervised feature extraction + linear classification pipeline of OpenAI features. This shows that unsupervised models, even if trained on massive datasets, still lag behind supervised models. *SeqConv2* performs better in terms of classification accuracy than *SeqConv1*, probably because it is deeper (i.e. has more convolutional layers and more fully connected layers at the end).

Recurrent architectures gave slightly better results than the convolutional ones. One possible reason for this is the limited size of the training set. It is well-known that CNNs require vast amounts of data to converge (e.g. there are examples in the literature where 90M tweets have been used on a similar task to train a rather shallow convolutional network). However, the 2.5M tweets that we used as training set were enough for the recurrent models to capture the connections inside the sequences of tokens. Both the bidirectional and the GRU outperform the CNN+LSTM model, which in turn gives better results than the simple LSTM model (which is not reported in the table) and the convolutional models.

Figures 3 and 4 show the evolution of the training and validation accuracy as more samples are processed. From the validation chart, it can be noted that the convolutional networks, especially *SeqConv1* start overfitting over the dataset. This can be attributed to the shallowness of the network and to the limited size of the training set. The recurrent networks, however, take much longer to saturate on this data set. One of the reasons for this is that we use dropout regularization for the recurrent units as well. For our final submission, we select two models. First the GRU model as it gave the best accuracy on our validation set. Secondly we select the ensemble of the five models as described in III-H.

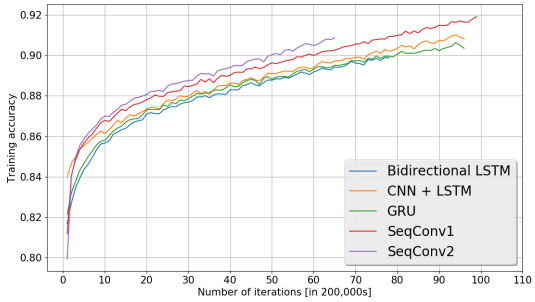

Figure 3: Training accuracy

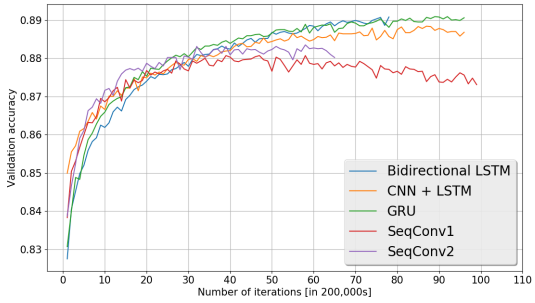

Figure 4: Validation accuracy

VI. Conclusion

We started from some established techniques on the sentiment classification task and came up with our own architectures that use these general ideas. As expected, the deep learning models have a clear advantage over other models, when fed with a large enough data set, because of their complexity. However, convolutional networks performed almost as good as recurrent networks showing, once again, that they can be effectively used for text analysis as well, despite the fact that they require larger data sets for good performance. Lastly, we noticed that factors like regularization, ensembles and mixtures of features obtained through different sources also improve performance significantly.

REFERENCES

[1] J. Deriu, M. Gonzenbach, F. Uzdilli, A. Lucchi, V. De Luca, and M. Jaggi, "SwissCheese: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, S. Bethard, D. M. Cer, M. Carpuat, D. Jurgens, P. Nakov, and T. Zesch, Eds. S.l.: The Association for Computer Linguistics, 2016, pp. 1124–1128.

[2] A.Deshmane and J.Friedrichs, "An ensemble of deep learning architectures including lexicon features for twitter sentiment analysis," http://nlp.arizona.edu/SemEval-2017/pdf/SemEval135.pdf, 2017.

[3] M. Cliche, "Bb_twtr at semeval2017 task 4: Twitter sentiment analysis with cnns and lstms," 2017.

[4] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv preprint arXiv:1704.01444*, 2017.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[7] F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, "named entity recognition for twitter microposts using distributed word representations," *ACL-IJCNLP*, vol. 2015, pp. 146–153, 2015.

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

[9] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

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):

DEEP NEURAL NETWORKS FOR TWITTER SENTIMENT CLASSIFICATION

Authored by (in block letters):

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

Name(s):	**First name(s):**
Tifrea	Alexandru
Jain	Jinank
Shekhar	Saurav
Hasheminezhad	Seyedrouzbeh

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, 04/07/2017

Signature(s)

jinankjain

Seyedrouzbeh Hasheminezhad

Alexandru Tifrea

Saurav Shekhar

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