Movie Review Sentiment Analysis Kaggle Competition

Feng Gao, Ziyun Chen, Li Cai

November 12, 2018

1 Introduction

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis. This Kaggle competition has been inspired by Socher's work on labeling parsed phrases in the corpus. The goal of the project is to predict the sentiment of phrases using the labeled dataset. The initial phase of the project focuses on achieving the most accurate model using the CNN-RNN architecture, then other architectures will be explored to further improve on CNN-RNN.

2 Related Work

Prior to Socher's work, most sentiment prediction models focused on words in isolation, giving positive points for positive words and negative points for negative words. However, these methods lost important context information by overlooking the order of words. Below are some common methods that have been used for sentiment analysis.

2.1 Semantic Vector Spaces

The most popular approach in semantic vector spaces uses the distributional similarities of single words. Usually statistics of a word and its context are used to describe each word, such as tf-idf.

2.2 N-gram/Bag of words

A lot of approaches used the n-gram presentation. Several works have explored sentiment compositionality through careful engineering of features or polarizing shifting rules on syntactic structures.

2.3 Deep Learning

Sequence models, in particular Convolutional Neural Network (CNN), is widely applied for sentiment classification. Long Short-Term Memory model (LSTM) is also common for learning sentence-level representation, as proposed in 1997 by Sepp Hochreiter and Jrgen Schmidhuber and improved in 2000 by Felix Gers' team due to its capability to model sequential data. Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks introduced in 2014 by Kyunghyun Cho. Their performance on polyphonic music modeling and speech signal modeling was found to be similar to that of LSTM. Socher showed that deep learning is a great method for sentiment analysis. The model in his paper, a recursive network tensor network, achieved performance much better than other traditional language models at the time.

2.4 LSTM

The Long Short-Term Memory (LSTM) was first proposed by Hochreiter and Schmidhuber (1997) that can learn long-term dependencies.

Different from traditional recurrent unit, LSTM unit keeps the existing memory ct R n at time t. The input at time t is xt, ht1, ct1, the output is ht, ct, they can be updated by the following equaions:

$$\begin{gathered}
\mathrm{i}_t = \delta(W_i x_t + U_i h_{t1} + b_i) \\
\mathrm{f}_t = \delta(W_f x_t + U_f h_{t1} + b + f) \\
\mathrm{o}_t = \delta(W_o x_t + U_o h_{t1} + b_o) \\
\mathrm{gt} = (\mathrm{W}_g x_t + U_g h_{t1} + b_g) \\
\mathrm{ct} = \mathrm{f}_t c_{t1} + i_t g_t \\
\mathrm{ht} = \mathrm{o}_t \tanh(c_t)
\end{gathered}$$

2.5 GRU

A gated recurrent unit (GRU) was initially proposed by Cho et al. (2014) to make each recurrent unit to adaptively capture dependencies of different time scales. The parameters can be updated by the following equations

$$\mathrm{r}_t = \delta(W_r x_t + U_r h_{t1})$$
$$\mathrm{z}_t = \delta(W_z x_t + U_z h_{t1})$$
$$\text{ht=}\ \delta(W x_t + U(r_t h_{t1}))$$
$$\mathrm{h}_t = (1 z_t) h_{t1} + z_t h_t$$

2.6 CNN

The convolutional layer applies a matrix-vector operation to each window of size w of successive windows in the sentence-level representation sequence. Let

$$H \in R^{d \times w}$$

be the weight matrix of the convolutional layer, we then add a bias item b to the result of the matrix-vector operation and get a feature mapping

$$C \in R^{1-w+1}$$

. The i-th element of the feature map is:
$\mathrm{c}_i = \delta(\sum(C[, i : i + w]H) + b)$

3 Problem formulation

The train dataset labeled phrases of movie review sentences on a scale of five values: negative (0), somewhat negative (1), neutral (2), somewhat positive (3), and positive (4). The goal of this project is to use the dataset to train a neural network model that can accurately label phrase sentiment. There are many obstacles that make this exercise difficult (e.g. sentence negation, sarcasm and ambiguity).

4 Dataset

4.1 Train/Testing Data

The dataset comprises of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each sentence has been parsed into many phrases by the Stanford parser. Each phrase then has a PhraseId while each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data. For example:

PhraseId	SentenceId	Phrase	Senti
537	20	It's what you'd expect	2
538	20	what you'd expect	2
539	20	you'd expect	2
540	20	expect	3

train.tsv - contains the phrases and their associated sentiment labels. data also contains sentence id of each phrases.

test.tsv - contains just phrases. the project will be to label each phrase.

4.2 Embedding Data

Many pre-trained embedding data have been tested, and the data from FastText, crawl-300d-2M.vec, was the best. FastText was developed by Thomas Mikolov's team, who proposed the word2vec framework in 2013. The main improvement of FastText over the original word2vec vectors is the inclusion of character n-grams, which allows computing word representations for words that did not appear in the training data.

5 Methodology

5.1 Understand the Data

This dataset is a bit strange compared to a typical review text so it's important to understand its quirks.

1. On average there are 18 phrases per sentence in the training set. Phrases of sentences are generated at random. As such, sentence tokenization based on punctuation isn't likely to work well.

2. Some phrases start and/or end with a punctuation, which often changes the sentiment label for an otherwise identical phrase. Therefore, it doesn't make

sense to perform the usual data cleaning that would eliminate punctuation.

3. Many common words appear frequently due to random splitting of sentences into phrases. As a result, removing stopwords would also remove a lot of information from the dataset.

4. Words in certain orders appear to be more predictive than simple bag of words. Combined with the fact that the dataset contains many short phrases, deep learning, RNN in particular, seems to be the superior method over vector spaces and n-gram models.

5. Looking at the class distributions, neutral (2) accounts for 51% of the training set.

Sentiment	Count
0	7072
1	27273
2	79682
3	32927
4	9206

5.2 Architecture

We have implemented and compared various models. The models listed below include ones that had the best performance.

1. Architecture 1: Embedding + LSTM/GRU + CNN

2. Architecture 2: Embedding + LSTM + multiple layer CNN

3. Architecture 3: Embedding + LSTM/GRU + multiple layer CNN

Other architectures will also be explored in an attempt to further improve performance

1. Deep CNN

2. Dynamic CNN using Dynamic k-Max Pooling

3. Linguistically Regularization

4. Leaky ReLU and batch normalization

5.3 Github Repository

https://github.com/MangoHaha/SentimentAnalysis/tree/master

6 Preliminary Results

6.1 Evaluation criteria

Submissions are evaluated on classification accuracy (the percent of labels that are predicted correctly) for every parsed phrase. The sentiment labels are:

0 - negative

1 - somewhat negative

2 - neutral

3 - somewhat positive

4 - positive

6.2 Results

In this section, we list the highest accuracy values from models as described in section 5.2.

Embedding + LSTM/GRU + CNN

loss	acc	val_{loss}	val_{acc}
.2742	.8755	.2378	.8928

Embedding + LSTM + multiple layer CNN

loss	acc	val_{loss}	val_{acc}
.7590	.6859	.6695	.7249

Embedding + LSTM/GRU + multiple layer CNN

loss	acc	val_{loss}	val_{acc}
.2772	.8740	.2506	.8871

6.3 Kaggle Leaderboard

Architecture 1, the best model so far, is good for 3rd place on the Kaggle leaderboard.

7 References

1. https://www.kaggle.com/artgor/movie-review-sentiment-analysis-eda-and-models

2. https://github.com/vivanraaj/Sentiment-Analysis-on-Movie-Reviews

3. https://github.com/ovguyo/moviereview

4. Pang, Bo, and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004. http://www.cs.cornell.edu/home/llee/papers/cutsent.pdf

5. Sosa, Pedro M. Twitter Sentiment Analysis Using Combined LSTM-CNN Models. Konukoii.Com, 2018, http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/

6. Zhang, Lei, Shuai Wang, and Bing Liu. Deep Learning for Sentiment Analysis: A Survey. arXiv preprint arXiv:1801.07883 (2018). https://arxiv.org/pdf/1801.07883.pdfmo

7. Joao Carlos Duarte Santos Oliveira Violante. Sentiment Analysis with Deep Neural Networks. https://fenix.tecnico.ulisboa.pt/downloadFile/1407770020544739/METI-ARTICLE-THESIS- 70502-JVIOLANTE.pdf

8. Xingyou Wang, Weijie Jiang, Zhiyong Luo. Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts http://www.aclweb.org/anthology/C16-1229

9. Mihaela Sorostinean, Katia Sana, MOHAMED Mohamed, Amal Targhi, Sentiment Analysis on Movie Reviews, March 1, 2017. http://www2.agroparistech.fr/ufr-info/membres/cornuejols/Teaching/Master-AIC/PROJETS-M2-AIC/PROJETS-2016-2017/main(Amal

10. Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. https://nlp.stanford.edu/ socherr/EMNLP

11. http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/

12. Xingyou Wang1, Weijie Jiang2, Zhiyong Luo3, Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts

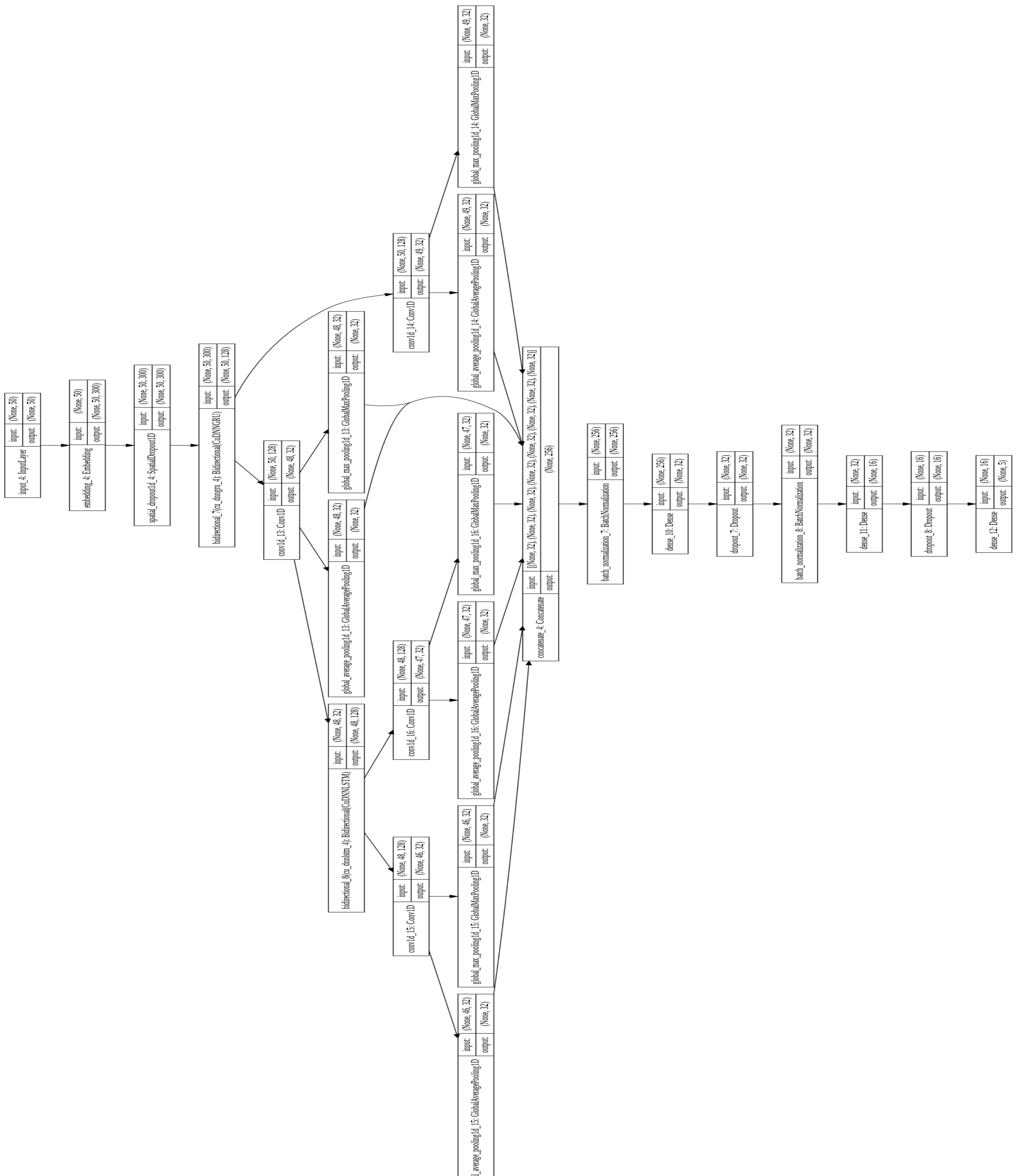

Figure 1: Model 1 Summary