Sentiment Analysis for Yelp Reviews based on Attentive CNN

Congrong Zhou **Jiazhao Li** **Royce Hwang**
Tianyu Zhang
{congrong, jiazhaol, royceh, tyrozty}@umich.edu

Abstract

In this project, we aim to build a novel attentive convolutional neural network model that can understand the latent sentiment of text data on social media. In social media, there is a great deal of opinion-rich text data without clear sentiment labels. It can be costly to label a large number of unlabeled text data manually, which makes using an automatic sentiment classifier necessary. However, as dataset sizes increase over time, it becomes increasingly challenging to use traditional natural language processing models based on lexicon. To solve this problem, we propose a new model that combines design elements of convolutional neural networks along with attention mechanisms in order to learn sentiment information carried by text data. After training on user written reviews from the Yelp Dataset Challenge, our model was able to achieve sentence level accuracies of 69.85 percent, which is a significant improvement over baseline models such as linear and decision tree models.

1 Introduction

Many services such as Yelp, YouTube and Amazon allow users to write reviews on their platforms. These reviews can be extracted and analyzed to better understand users and how they relate to the company's services and products. For example, the feelings conveyed by comments on a Youtube video can reflect that video's popularity among users. Based on the analysis of the users' comments history, we can understand the users' preferences on different categories of videos on YouTube, which can potentially be applied to provide more accurate recommendations for users in the future. Thus, understanding opinions contained in user-generated text can be an essential task for many businesses and organizations.

Inspired by a recent paper (Vaswani and Polosukhin, 2017), which introduced an attention-based mechanism, we propose a novel model to learn sequential data for sentiment analysis combining a convolutional neural network with attention mechanisms. The attention mechanism allows key elements in a sentence to play a pivotal role in sentence representation. This attention mechanism will hopefully allow us to achieve better results than other widely used methods like recurrent neural network (RNN) (Mikolov et al., 2010) models. Also, to reduce the computational complexity, we built our model based on a convolutional neural network architecture. Compared with an RNN model, convolutional neural networks do not depend on the output of previous steps, which allows computations to be efficiently completed in parallel, such as on a GPU. By combining the CNN architecture with an attention mechanism, we hope to achieve both good accuracy while improving the performance with regards to training and prediction speed. As of this writing, we do not know of any published research showing results for sentiment analysis using a convolutional neural network integrated with an attention mechanism.

Methods for automatically classifying sentiments expressed in reviews have been heavily studied. One traditional way to classify sentiment is based on word level analysis. This method extracts the key words for each review and discounts the stop words effect by applied term frequency (TF) and inverse document frequency (IDF). Also, it smooths the model to address the problem of introducing new words into corpus. However, this statistical method is too simple to capture sequence information in reviews by itself.

To improve the ability of the language model, word embedding methods and deep learning models have been introduced into sentimental analysis. There are many techniques to convert words into a computer understandable format, such as one-hot embedding and word2vector embedding (Goldberg and Levy, 2014). However, the one-hot embedding has limitations when it comes to representing the relationship between words and process a relatively large corpus. Thus, the word2vec embedding was proposed and became the most popular preprocessing method. After embedding a sentence into a matrix combined by words embedding vectors, more and more elaborate deep learning models are introduced to capture the continuous information contained in such data. For our project, we aim to train an attention-based model to learn to emphasize the key features in the sentence, resulting in better performance in sentiment analysis.

2 Related Work

2.1 Previous State of the Art

Much work has been previously done in sentiment analysis over the years on a variety of datasets. One example of this is the Yelp Dataset Challenge, which is a competition sponsored by Yelp in which the company provides user-written reviews, among other data, for researchers to use in various tasks like sentiment analysis. It has already been running for many years, and over the years, many of the winners have used a variety of methods involving neural networks. One example of this comes from the most recent champion of the Yelp Dataset Challenge. While their main focus was on evaluating techniques to understand and diagnose the inner workings of recurrent neural networks, they also trained a RNN in order to classify whether Yelp reviews were positive or negative, managing to achieve accuracies of 91.52 and 91.91 percent for the validation and test set, respectively (Ming and Qu, 2017). These results, combined with other ongoing work, have recently established RNNs as a state of the art technique for sentiment analysis as well as other natural language processing (NLP) tasks (Zhang and Liu, 2018).

2.2 Convolutional-Based Methods

Newer techniques are emerging, however, that promise the possibility of further performance gains. One approach recently introduced by Facebook AI Research involves using convolutional neural networks (CNNs) rather than recurrent models in order to perform sequence to sequence learning for machine translation (Gehring and Dauphin, 2017). By using a fully convolutional model rather than RNNs, computations can be fully parallelized since they do not depend on the previous time step, resulting in potentially significant performance gains. Additionally, the authors report that their fully convolutional model outperforms comparable RNN models in machine translation tasks such as the WMT'16 English to Romaninan translation. Previous work has shown that machine translation techniques can often be adapted to sentiment analysis, making this a possible area of exploration for our project (Hiroshi and Hideo, 2004).

2.3 Attention-Based Methods

Additionally, Google Brain, in cooperation with other research institutions, has released a paper detailing a novel architecture called the Transformer that is based primarily on an attention mechanism rather than recurrence to draw dependencies from input to output (Vaswani and Polosukhin, 2017). Besides training significantly more quickly than RNNs, the Transformer outperforms all other state of the art methods (even the fully convolutional model by a small margin) in machine translation tasks. Given that researchers have often adapted models optimized for machine translation to sentiment analysis, we believe we can do something similar in incorporating the best aspects of the fully convolutional and attention-based models and using them for our Yelp Review sentiment analysis task that is the basis of our project in order to achieve both improved performance and results.

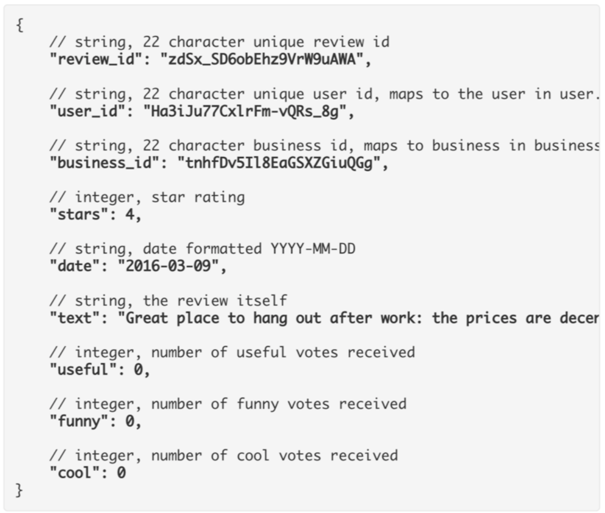
3 Data

In this section, we introduce the source of dataset and methods we used to preprocess the data instances.

3.1 Dataset

We used data from the Yelp Dataset Challenge. The challenge is designed for students and academics to conduct research on Yelp's data in innovative ways and share discoveries. Specifically, the dataset given by the challenge contains information on users, businesses, user-written reviews, and other data collected by Yelp over time.

In this project, we mainly performed sentiment analysis on review text which is stored in a 4 GB JSON file (`review.json`) containing 19,564,819 reviews. One example of an entry in this file is shown below:



```
{
    // string, 22 character unique review id
    "review_id": "zdSx_SD6obEhz9VrW9uAWA",

    // string, 22 character unique user id, maps to the user in user.
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",

    // string, 22 character business id, maps to business in business
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // integer, star rating
    "stars": 4,

    // string, date formatted YYYY-MM-DD
    "date": "2016-03-09",

    // string, the review itself
    "text": "Great place to hang out after work: the prices are decer

    // integer, number of useful votes received
    "useful": 0,

    // integer, number of funny votes received
    "funny": 0,

    // integer, number of cool votes received
    "cool": 0
}
```

Figure 1: One example review in the `review.json` file

From the `review.json` file, we extracted two kinds of records, review text as target of analysis and star rating as label. For each line in the csv file, there is one review along with its corresponding rating. For the ratings, we classified them into three categories: positive (5 stars), negative (1 star) and neutral (2, 3, or 4 stars).

3.2 Data Preprocessing

Before feeding data into our model, we have to take several steps for data preprocessing. The first step is data cleaning. For each review instance, we need to perform sentence splitting, word tokenization and conversion of characters to lowercase. Generally, to reduce inflectional forms and derivationally related forms of a word to a common base form, it is useful to do stemming and lemmatization. Also it is common to remove stop words during the data preprocessing step. However, we skipped these two steps and keep the complete raw sentences in order to analyze the relationship and logic between words.

The second step is data filtering. We set up the length range of 20 to 100 as valid size for review text. The reason is that if the review text is too short, the valid information it contains is too little and it is not valuable for our model to learn. If the review text is longer than 100, it tends to contain too much irrelevant information which has bad influence on training the model. Thus these reviews should be filtered.

The third step is sample generating. For our A-CNN model, each input feature should have the same size. Thus we applied the sliding window method witha window size of 20 on the review text and stored the sequences generated as training samples.

The fourth step is sample balancing. Since originally the numbers of samples labeled with 5, 4, 3, 2, and 1 stars are 1118927 (55.2%), 410178 (20.2%), 167368 (8.3%), 112727 (5.6%), and 218068(10.7%), respectively, the model tends to learn much more information given by the samples with 5 stars and ignore valuable information in the samples with lower stars. Thus we performed upsampling for the samples with 4, 3, 2, 1 stars and get a new distribution of 1118927 (29.4%), 820356 (21.5%), 334736 (8.8%), 225454 (5.9%), and 1308408 (34.4%), respectively.

Finally, we implemented word embedding. In order to quantify and categorize semantic features, we need to map the original words to vectors of numbers. Basically, we mapped the words to count vectors with TF-IDF, but as a statistical method, it assumes that words are independent and ignores the relationship between the word and its neighbors. Therefore, instead of simply using the baseline method, we have two optimal solutions: word2vec and GloVe. While both do very well at capturing semantic features, word2vec preserves semantic analogies under basic arithmetic on the word vectors. For example, it keeps the concept that king - man + woman = queen. The simpler implement makes us to choose word2vec as our word embedding method.

Besides, we extracted some discrete features of the sentences, including Part-Of-Speech(POS) tag and position information. The NLTK package is used to label words POS tags and words are labeled with relative positions by one hot encoding for encoding indexes of words in sentences.

3.3 Overview of Dataset

We implemented visualization and analysis of the dataset.

3.3.1 Word Cloud

We plotted the word cloud based on the word frequency of the three classes (positive, neural and negative). The bigger the word is, the more frequently it appears in the class. Examining the word cloud for the positive class, we see that the most frequent words are *great*, *best*, and *love*.

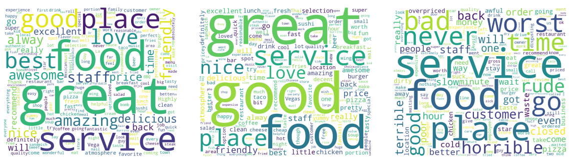


Figure 2: Word cloud of positive, neutral, and negative samples.

In contrast, the most frequent words of the negative class are *bad*, *worst*, *horrible*. These types of key words which frequently convey strong sentiment are the main features that the model is expected to learn.

3.3.2 Label Distribution

Label distributions of 5, 4, 3, 2, and 1 stars are 1,118,927 (29.4%), 820,356 (21.5%), 33,4736 (8.8%), 225,454 (5.9%), and 1,308,408 (34.4%), respectively.

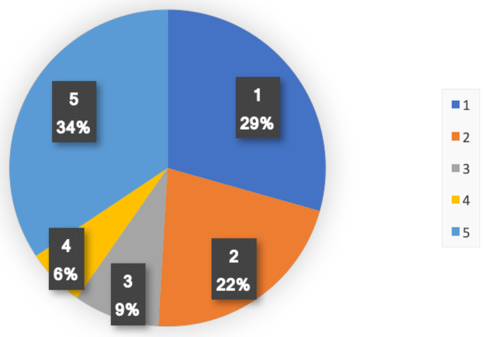


Figure 3: Label Distribution

4 Methods

In this section, we would like to introduce the method we designed, called Attentive Convolutional Neural Network (Attentive CNN, ACNN), which includes three components: embedding layers, N-gram convolutional layers and attentive fully-connected layers. The workflow is shown in Figure 4.

4.1 Embedding Layers

To allow the neural network to understand sentences better, we need to embed each token with more information than simply one-hot encoding. To do so, we embed sentences with pre-trained Word2Vec, Part-Of-Speech (POS) tag and relative position in embedding layers. Each sentence can be split and represented as a matrix shown in Figure 5. Let $\boldsymbol{x}_i \in \mathbb{R}^k$ be the k-dimensional word vector corresponding to the ith word in the sentence, where the word embedding is a concatenation of the word embedding and one-hot encoding over the POS tag and relative position. Consequently, $\boldsymbol{x}_{i:i+j}$ can represent a set of words $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, ...\boldsymbol{x}_{i+j}$. A sentence of length n (padded or segmented where necessary) can be represented as a train patch $\boldsymbol{x}_{1:n} \in \mathbb{R}^{n \times k}$.

4.2 N-gram Convolutional Layers

Inspired by the N-gram model, convolutional kernels are set with various sizes from 1 to N, $\boldsymbol{w} \in \mathbb{R}^{Nk}$, which is applied to a window of N words (padded at the start and end of sentences) to produce a scalar feature. For instance, a feature c_i is generated with a bi-gram kernel on $\boldsymbol{x}_{i:i+1}$:

$$c_i = f(\boldsymbol{w} \cdot \boldsymbol{x}_{i:i+1} + b) \tag{1}$$

where b is a bias term and f is the activation function that maps each N-gram context window of a single word to a feature scalar. In order to learn M different features in each N-gram level, we apply M kernels for a given kernel size. After applying M kernels on each word, the word can be embedded into a feature vector $\boldsymbol{c}$:

$$\boldsymbol{c} = [c_1, c_2, c_3, ..., c_M] \tag{2}$$

with $\boldsymbol{c} \in \mathbb{R}^M$.

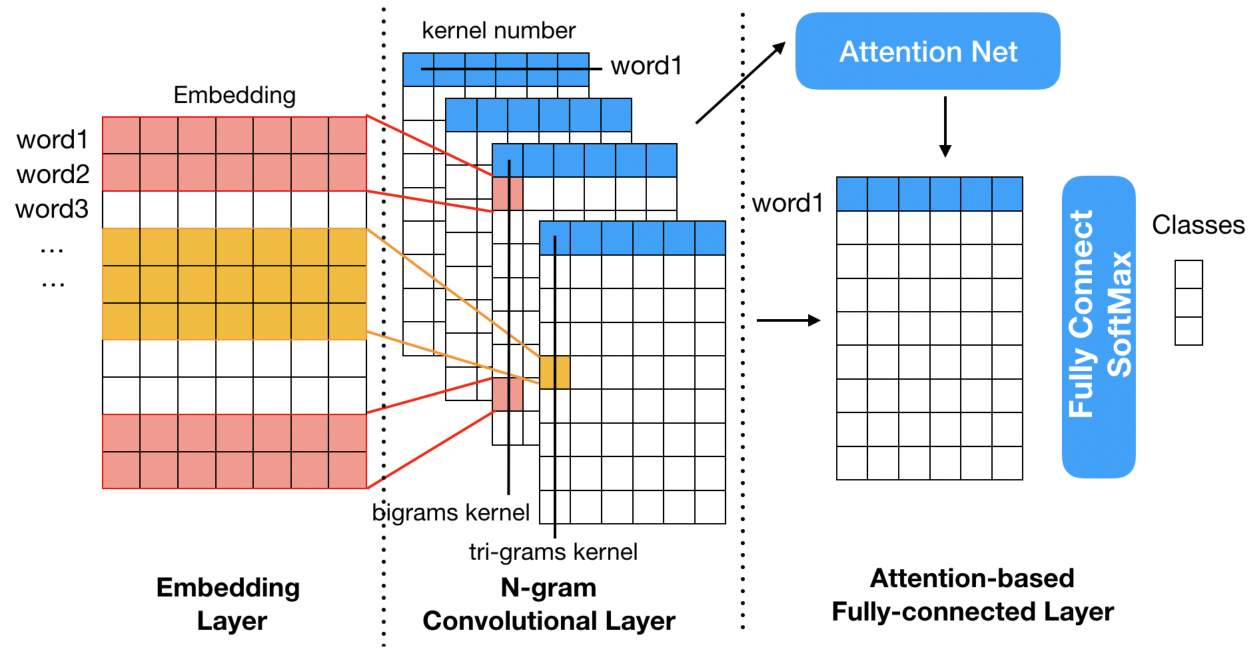


Figure 4: Attentive CNN Architecture

4.3 Attentive Fully-connected Layers

As mentioned in the Related Work section, attention mechanisms have become widely used in many domains such as computer vision (CV) and natural language processing (NLP), and have achieved impressive results in various tasks. The main idea is to allow network to learn how much attention to pay to word contexts and compress context information into a single word embedding.

In the original model (Kim, 2014), a max pooling layer is applied on the feature vector (result of previous N-gram model) in order to get the most important feature as well as to deal with variable sentence lengths caused by the non-padding kernel operation. With sufficient training data and zero padding operations, we can choose a more complicated fully-connected layer followed by a ReLU activation layer instead to learn feature fusion automatically. After projecting different level features into scalars, we propose to employ the attention mechanism by using gram-level re-weighted sum. In general, the attention score can be achieved by passing through a Softmax function rescaling into sum 1 as shown in Equation 4 (Xiao et al., 2017). Also, after feature generation and selection process, each word becomes mapped into an N-gram vector $\boldsymbol{a}$.

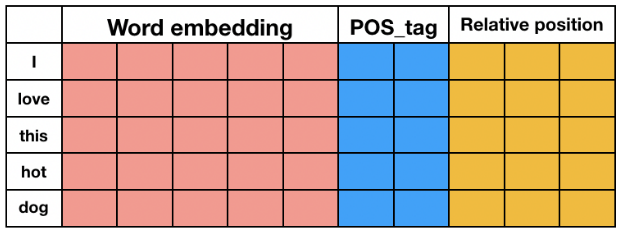


Figure 5: Embedding Layer

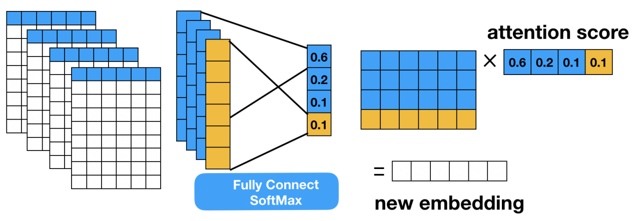


Figure 6: Attention Net

Method	Keyword	Related Words						
Yelp trained	*good*	decent	great	tasty	solid	yummy	bad	nice
Google trained	*good*	great	bad	terrific	decent	nice	excellent	fantastic
Yelp trained	*bad*	terrible	horrible	good	awful	poor	horrid	lousy
Google trained	*bad*	terrible	horrible	Bad	lousy	crummy	horrid	awful
Yelp trained	*best*	BEST	worst	greatest	tastest	WORST	Best	finest
Google trained	*best*	finest	worst	greatest	strongest	smartest	easiest	good
Yelp trained	*worst*	WORST	best	BEST	Worst	grestest	slowest	blandest
Google trained	*worst*	Worst	weakest	scariest	ugliest	best	bleakest	strongest

Table 1: Comparison of trained vector between Yelp review and Google news

$$\boldsymbol{A} = [\boldsymbol{a_1}^T, \boldsymbol{a_2}^T, \boldsymbol{a_3}^T, ..., \boldsymbol{a_N}^T]^T \tag{3}$$

with $\boldsymbol{A} \in \mathbb{R}^{N\times M}$ and each entity $\boldsymbol{a_i} \in \mathbb{R}^M$ as one N-gram level feature vector of one word.

$$\begin{aligned} \boldsymbol{\alpha'} &= ReLU(\boldsymbol{w}^T \cdot \boldsymbol{A} + b) \\ \boldsymbol{\alpha} &= Softmax(\boldsymbol{\alpha'}) \end{aligned} \tag{4}$$

where $\boldsymbol{w} \in \mathbb{R}^M$ and $\boldsymbol{\alpha} \in \mathbb{R}^N$. $\boldsymbol{w}$ represents the weights and $\boldsymbol{\alpha}$ is the attention score on each N-gram level for a single word. The output of the attention net is a vector of the re-weighting of N-grams for a single word defined as:

$$\boldsymbol{a'} = \boldsymbol{A}\cdot\boldsymbol{\alpha} = \textstyle\sum_{i\in \boldsymbol{N}} \boldsymbol{\alpha_i} \cdot \boldsymbol{a_i}^T \tag{5}$$

where $\boldsymbol{a'} \in \mathbb{R}^M$, a sentence of length n has been converted into one sentence embedding $s \in \mathbb{R}^{n\times M}$ following by a fully-connected layer with Softmax activation layer.

In the classification task, we define the loss function as the cross-entropy of target and prediction as shown below:

$$\mathscr{L} = \sum_{c\in \boldsymbol{C}} (\hat{y}_{ACNN} - y)^2 \tag{6}$$

5 Experiments and Results

5.1 Word2Vec

Compared to the pretrained vectors trained on part of Google News dataset (about 100 billion words), which contains 300-dimensional vectors for 3 million words and phrases, the trained vectors on Yelp review by using word2vec fit our model better.

To be specific, the words in the Google model are not limited to one specific domain. It contains domains such as business, sports and politics, while the Yelp review dataset just contains customers' review for restaurants. Thus, certain related words in the Yelp model have stronger sentiments and are more valuable for our model to learn.

5.2 Experimental Setup

In this project, the model is built based on `Tensorflow-1.20.0` and trained on the `1080-Titan`. The data described in the Data section was separated into training data and testing data at a ratio of 7:3. To evaluate our Attentive Convolutional Neural Network, two experiments were conducted to answer the following questions:

1: Does our Attentive Convolutional Neural Network have a better performance than traditional Machine Learning methods?

2: Can the performance of our Attentive Convolutional Neural Network's performance be improved by tuning the hyperparameters?

5.3 Results

The Attentive CNN model was initialized by applying 3 different language models (unigram, bigram, and trigram) with 100 kernels used for each language model. For each review, the final result was generated by majority voting of its sub-samples. The initialized model was trained for 60 epochs with a learning rate of $1e - 4$. Each epoch takes about 25 minutes, and the entire training progress for 3 language models takes almost 25 hours.

The training results are shown in Figure 7. The plot shows that the sentence level accuracy of the test dataset can reach 66.8%.

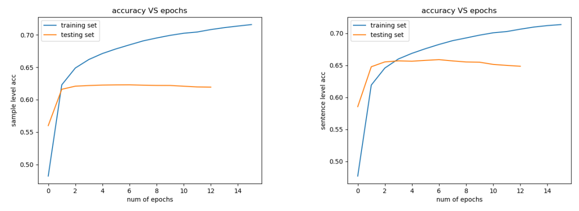


Figure 7: Sample and sentence training accuracy vs. epochs

5.3.1 Comparing with baseline

In Table 2, it demonstrates that our Attentive CNN model has a better performance than traditional machine learning methods for sentiment analysis. Unlike the linear and decision tree models, the Attentive CNN model includes a convolutional and attentive component for better computing performance and accuracy, respectively. Even if the training for the attentive model is complex and time consuming, it is acceptable to make that sacrifice for higher precision, and it is balanced out to some extent by the better parallelization that the convolutional layers provide.

Methods	Attentive CNN	Linear Model	Decision Tree
Accuracy	66.8%	56.01%	44.42%
Time	25 hours	9 minutes	32 minutes

Table 2: A-CNN model compared to baseline

5.3.2 Tuning hyperparameters

Neural network hyperparameters usually play important roles in the performance of the neural network. In our Attentive CNN model, the key parameters are the number of kernels and the kernel size.

Comparing the effect of different kernel size, we see that using a larger kernel of size([1, 5, 9]) leads to worse performance on both sample level and sentence level. The reason is that a larger kernel includes more noise, which distracts the model from learning key relationships in the sentence. On the other hand, the smaller kernel of size([1, 3, 5]) contains less irrelevant information and can make the model focus on keywords that it is expected to learn.

Kernel Size	[1,3,5]	[1,5,9]
Accuracy	66.8%	64.9%
Time	25 hours	27 hours

Table 3: Kernel size vs. accuracy & time

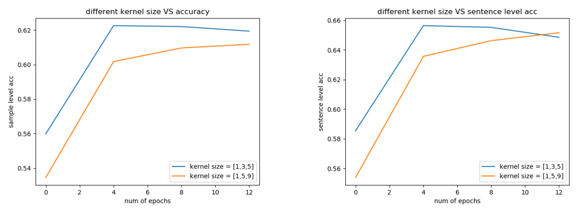


Figure 8: Kernel size vs. sample & sentence accuracy

Next, we would like to compare the effect of varying the number of kernels, since different kernels can learn different features from the sentence embedding. Increasing the number of kernels also means potentially increasing the networks ability to capture all the latent features. However, using more kernels may cause latent problems such as overfitting and an increase in computational complexity. Results are shown in Table 4. From Figure 9, we can conclude that the model with more kernels has a higher accuracy than the one using fewer kernels. Due to the very long training time, we only tuned the parameters two times. One direction for future work would be to perform additional tuning work in order to better characterize the effect that the number of filters has on our model's performance.

Number of Kernels	100	200
Accuracy	66.8%	69.85%
Time	25 hours	40 hours

Table 4: Number of filters vs. accuracy & time

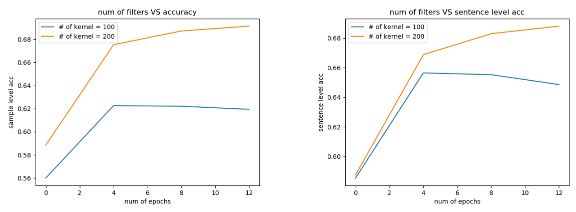


Figure 9: Number of filters vs. sample & sentence accuracy

6 Conclusion and Future Work

We proposed the first neural network model that combines the convolutional neural network architecture with attention mechanisms with the intention of achieving good results in both performance and accuracy in sentiment analysis. Based on our results, our Attentive Convolutional Neural Network achieves better performance than traditional machine learning methods like the linear and decision tree models. Also, it can be trained in parallel on a GPU, which is much more effective performance-wise than recurrent neural networks.

However, there are many improvements and future directions that we can explore for our network design. In particular, the word embedding that we used in this project is a static method, which is not robust for representing words with multiple meanings. Google recently proposed a method to dynamically embed words in order to solve this problem. In the future, perhaps we can replace our original embedding layer with this advanced one. Additionally, we are interested in continuing to explore the hyper-parameter space, especially the effects of kernel size and number of kernels on our model's performance.

7 Team Member Contributions

Equal work was done by each team member.

References

Auli M. Grangier D. Yarats D. Gehring, J. and Y. N. Dauphin. 2017. Convolutional sequence to sequence learning. *International Conference on Machine Learning*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Tetsuya N. Hiroshi, K. and W. Hideo. 2004. Deeper sentiment analysis using machine translation technology. *Proc. of COLING '04*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Empirical Methods in Natural Language Processing(EMNLP)*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Cao S. Zhang R. Li Z. Chen Y. Song Y. Ming, Y. and H. Qu. 2017. Understanding hidden memories of recurrent neural networks. *Proc. of the IEEE Conference on Visual Analytics Science and Technology*.

Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. N. Kaiser L. Vaswani, A. and I. Polosukhin. 2017. Attention is all you need. *Annual Conference on Neural Information Processing Systems (NIPS)*.

Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *CoRR*.

Wang S. Zhang, L. and B. Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.