Social Media Text Analysis using Multi-kernel Convolutional Neural Network

Anna Philips

Social Media Text Analysis using Multi-kernel Convolutional Neural Network

by

ANNA PHILIPS

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2019

ABSTRACT

Social Media Text Analysis using Multi-kernel Convolutional Neural Network

Anna Philips, M.S.

The University of Texas at Arlington, 2019

Supervising Professor: Dr. Won Hwa Kim

Transportation planners and ride hailing platforms such as Uber and Lyft use their riders feedback to assess their services and monitor customer satisfaction. Social media websites such as Facebook, Instagram, LinkedIn and in particular Twitter provides a large dataset of micro-texts by users who regularly post to their social media accounts about their grievances with their ride experience. This data is often unorganized and intractable to process because of its extremely large size which is continuously increasing daily. In this project, we collected ride hailing service relevant text data from Twitter around New York and developed a novel Convolutional Neural Network (CNN) model that classifies and categorizes sentences automatically into a transit performance category. Our model uses multiple kernels for convolution to capture local context among neighboring words in texts; summarizing the parameters in a kernel. Its performance is comparable to state-of-the-art NLP methods but our model converges much faster during training which means it trains much more efficiently.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Sentiment analysis and text classification are some of the most popular and well-researched Natural Language Processing (NLP) techniques. They are used in conjunction with Twitter Data Analysis due to the large availability of tweets available online today due to the popularity of social media. With Sentiment analysis one can automatically extract the polarity (measurement of opinion) of a tweet and classify the sentiment on a scale of positive and negative. There are various levels of granularity for text classification which can take place at document, sentence or character level.

Tweet analysis in addition to traditional metrics helps businesses to assess their customers, monitor trends and also find out what the public thinks about their competitors. This is especially useful for transportation planners and stakeholders as it provides valuable market research into the public's opinion on transit services.

The New York Metropolitan Transportation Authority has 1 million followers on twitter which provides real time information about delays and service changes which is staffed 24/7. In October they received 52,000 messages via social media from customers. In our project we will be examining and classifying tweets collected about the NYC MTA and also ride-hailing services such as Uber and Lyft in NY.

A Neural Network based approach to sentiment analysis have taken precedence over statistical models or linear models since the latter can automatically learn complex patterns from a large quantity of data. The process of converting raw data into a dataset is called feature engineering which is a very labor intensive project if done manually. A single example can have multiple features and hand-crafting features which have high predictive powers is not easy.

Convolutional Neural Network (CNN) which is a commonly used neural networks architecture in computer vision, also has applications for text analysis. Text like images share a compositional property, where words can be combined to form sentences likewise images are made of individual edges and lines. CNNs also have fewer parameters to train than when compared to a Multi-layer perceptron (MLP). But CNNs can be computationally expensive and can take longer to converge with increasing number of layers. In this project we introduce Multi-kernel Convolutional Neural Network (MKCNN) which introduces a gaussian kernel that ”summarizes” the weights in a kernel hence it has a smaller number of parameters to train and converges faster than a traditional CNN.

1.1 Thesis Organization

Rest of the thesis is organized as follows -

- Chapter 2 discusses the work done related to this field of thesis
- Chapter 3 elaborates the preliminary work done in this field
- Chapter 4 describes the detailed design of the CNN model and MKCNN model.
- Chapter 5 elaborates the detailed implementation of MKCNN architecture
- Chapter 6 contains all the experiments that are performed as part of this thesis and results obtained for each
- Chapter 7 includes Conclusion and future work for this thesis

CHAPTER 2

RELATED WORK

Machine Learning algorithms are classified into three types: Supervised, Semi-Supervised and Unsupervised learning. We will be looking at supervised learning algorithms where labelled data is available (for each example in the dataset there is label or class associated with it).

Support Vector Machines [1] combined with a low level representation of the text (Bag of Words) was used before the advent of Neural Networks where the latter was effective enough for one-class classification problems but cannot naturally be extended for multi-class classification. Naive Bayes Classifier [2] was another popular and simple statistical based learning model used for text categorization which tends to do a little worse than SVM.

More recently neural network architectures such as CNN [3], Recurrent Neural Networks(RNN), and Transformers [4] are used for text categorization since they do not require any feature engineering, on the downside they do require a lot of data usually to train them. While linear models like SVM rely on sparse and high dimensional features, neural networks are non-linear and are trained on dense feature vectors. And so for the scope of this project we will be looking at Neural Network architectures of two types: Feed-forward neural networks and Recurrent Networks. There are usually two stages in the text classification process: First is the extraction of the features and then the classification.

Raw data cannot be understood by machines and so it needs to be converted into a machine-friendly format. One way is converting text data into feature vectors.

Some popular techniques before word embeddings were introduced were bag of words or n-grams and their TD-IDF. The problem with these techniques is that context or the sequence of words is not preserved. They can still be useful for document level classification, but for smaller texts word embeddings are more appropriate. Word embeddings are treated as parameters for the model that need to be trained along with other the other components. The introduction of pre-trained word embeddings like GloVe and Word2Vec improved the accuracy of models drastically. Especially for shorter texts like SMS messages or tweets where capturing the context is more important to decipher the correct meaning. GlOve [5] and Word2Vec [6] are unsupervised learning algorithms trained on large datasets, that can generalize betteron unseen events. It's better than learning the word embeddings from scratch(one-hot encoding). The biggest difference between a one-hot encoding vector and word embeddings is that while latter's vectors are completely independent from each other, word embeddings which are similar in meaning will have similar vectors.

RNN is a popular algorithm still used in a lot of NLP tasks since they can capture long range dependencies in sequential data but the downside is that RNNs need to be trained sequentially and cannot make full use of modern day GPUs. The most popular kind of RNN is a gated RNN called the long short-term memory (LSTM) [7] which has a memory cell to store or clear information depending on which gate is triggered. This prevents the gradient from vanishing down the line which was problem with a plain RNN. Layers of LSTMs can be stacked to form more complex structures.

Transformer is a type of encoder-decoder structure which makes use of attention mechanism. It allows more parallelization than RNNs and can process longer input sequences simultaneously which has given rise to state of the art results in a lot of NLP tasks. It consists of two different neural networks: encoder and decoder which are trained simultaneously. The BERT [8] model is an example of transformer architecture which is non directional in nature.

In our project we will be using CNNs since they are simple to implement and hence easier to deploy, also they have faster training times when compared to more complex models like the RNN or Transformer. CNNs were originally used for computer vision tasks but were modified in [9] for sentence modelling. By using many convolution and pooling layers we extract a hierarchical representation of the sentence. CNNs have also been used in many NLP tasks like Machine Translation, Sentiment Analysis [10], Question Answering etc.

In [11] used a standard one layer neural network architecture and 1D convolutions. While CNNs are easier to train compared to other neural network architectures they require many layers to capture long dependencies and the input for the CNN is fixed unlike RNNs which can take varying sentence lengths. Also the more layers a CNN needs it runs into two problems called Vanishing or Exploding gradients, where during back-propagation the gradient passed back can be exceedingly small or large which can cause the model to train very slowly.

CNN works better for longer sentences than short sentences like tweets since they do not have a lot of context and as a consequence they require additional knowledge. DCNN [9] is one type of convolutional neural network which deals with semantic modelling of sentences using dynamic k max pooling which tries to overcome this problem by selecting the k most active features, independent of their position in the sentence.

In [12] they discuss a CNN architecture called with deeper layers(29 layers) called a VDNN which operates at the character level. The model shows that performance increases with depth.

We tackle two important tasks, sentiment classification to classify tweets as negative and positive and sentence classification to classify tweets into one of the five transit performance measure (cost, human interaction, reliability, technology, safety) which is also applicable to ride-hailing services to capture the users experiences and perception about the service. Following [13] which explored how hyper-parameters affected the performance of a CNN on sentence classification, we will follow a similair tactic to find the optimal hyperparameters for our model. Since the space of possible model architectures for CNN is huge we will stick to a single convolution and max pooling layer for our experiments.

CHAPTER 3

PRELIMINARY WORK

Using the official Twitter API a total of 1,925,952 Uber/Lyft relevant tweets were collected between January 23 and February 1 in 2019 which contained a relevant keyword or the geotag was located in NY. After further data cleaning which removed non-English tweets there was a total of 3123 tweets. Each tweet is annotated with a sentiment and a performance measure. If a single tweet is annotated with two or more performance categories, the tweet was duplicated and annotated each performance measure. The dataset includes 579 duplicated tweets, most of these tweets were related to human interaction or reliability as theres a disproportionate number of tweets labelled as human interaction, which makes sense as Uber and Lyft follow a peer to peer ridesharing model. There was 471 positive tweets and 2650 negative tweets. The average length of the positive tweets is 118 and the average length of the negative tweets is 134. The skewed dataset makes sense, since users take to social media their complaints more often than praise.

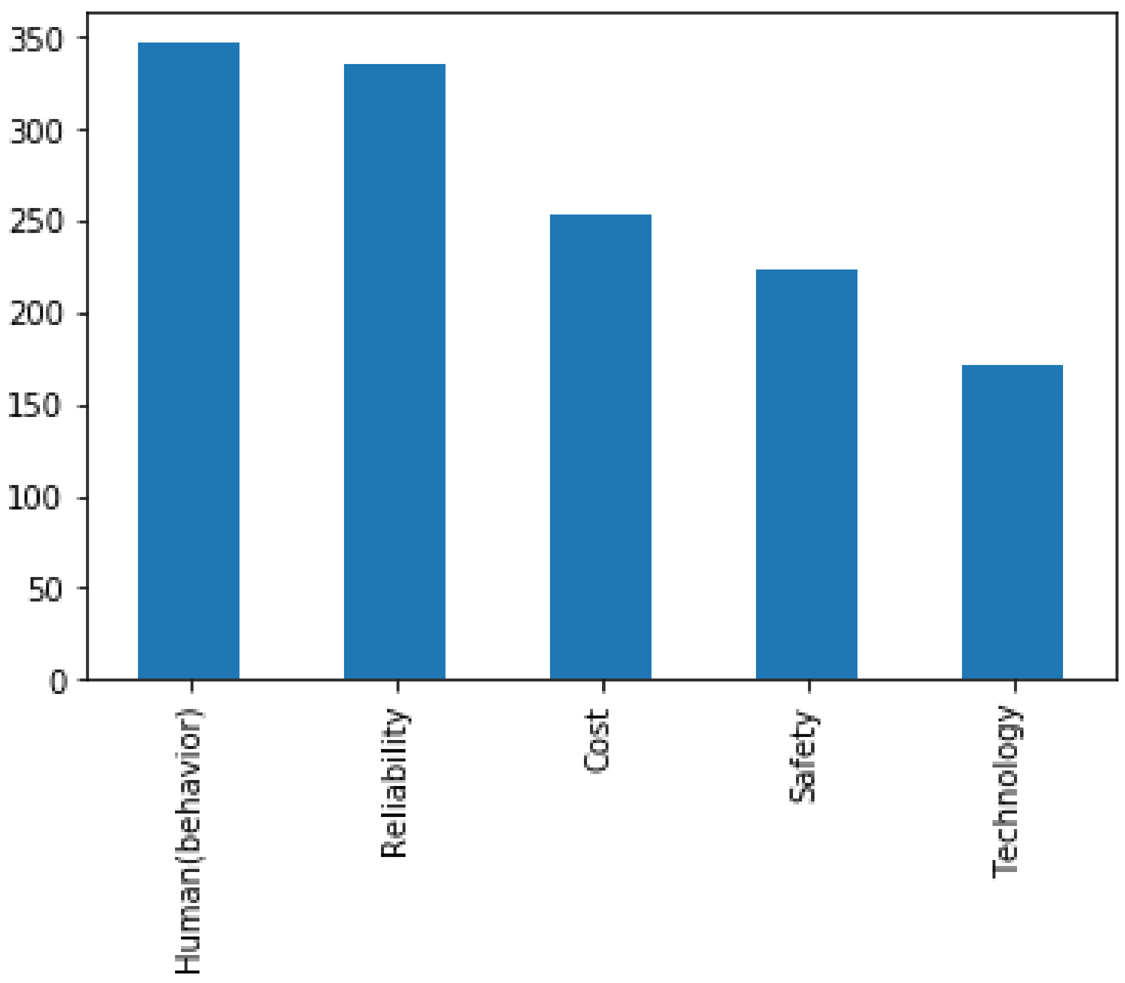

Figure 3.1: Distribution of the 5 performance measure categories (cost, human interaction, reliability, technology and safety) of our dataset.

3.1 Transit Performance Category

For this thesis we adopted the conventional transit performance measures which were applicable to ride hailing services and revised them to capture the user's experience. 5 main categories were created including cost, human interaction, reliability, technology and safety. Descriptions of each category are as follow.

- Cost refers to the cost of the ride to the user of the service. Tweets containing information about the user cost and cost comparisons were labeled under this category
- Human Interaction characterizes the drivers behavior and interaction with their passengers.
- Reliability indicates how much ride hailing service provided the promised experiences. For example, users evaluate if their driver shows up on-time or arrives at the destination on time.
- Safety refers to riders perceptions on safety on their trips which includes unsafe driving behavior, speeding, or traffic rule violations.
- Technology measures customer service and mobile app functionality if one is available. Tweets talking about the difficulty or ease of reaching customer service to resolve their issue falls under the category of customer service. It also captures any technological issues or suggestions on the mobile app platforms.

The Figure 3.1 shows the distribution of performance measure categories of our dataset.

CHAPTER 4

DESIGN

In this section we will go over a one layer CNN model since our model has been based it. Given a set of sentences of s_i and their labels y_i, the objective of a CNN model for classification is to learn a model that accurately predicts its label $\hat{y}_i$. The input to the model are tweets and the output is the performance measure or sentiment it is classified into.

The words in the tweet is split into tokens and each unique word or token is represented by a fixed size embedding or vector. That is each feature is embedded into a k dimensional space and is represented as a vector in that space. A dictionary mapping each unique token to an embedding is stored and learned during training. Formally, a word can be represented as a k-dimensional vector $x_i \in R^k$, a sentence s that consists of n words can be represented as a concatenation of these vectors as $s = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ where $\oplus$ is a concatenation operator. These word embeddings can be randomly initialized and learned while training a CNN model or a pre-trained word embedding like Word2Vec or GloVe can be used. The embedded matrix is constructed by concatenating the word embeddings vertically that result in a matrix of size $N \times k$ where N is the maximum number of words in a sentence and k is the length of the word embedding.

We use a convolutional filter over the embedding matrix to extract useful features. A typical convolutional layer will have filters of different sizes which learn n-grams of different sizes such as 3 gram 4 gram 5 gram etc. Convolving two matrices will give a higher value if they are more similar. There is also a bias parameter for each filter added to the result of the filter before passing it to the activation function. The weights and bias of the filters are randomly initialized from a normal distribution and learned using gradient descent with back-propagation. The stride on a convolution determines the step of the window where the output is smaller when the stride is larger and the padding of a convolution determines the output of the matrix which becomes larger when the padding is larger.

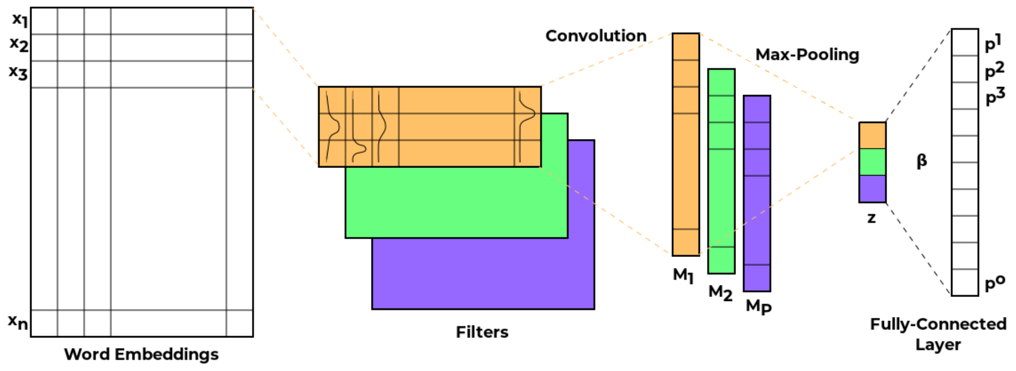

Figure 4.1: The architecture for the MKCNN model (multi-class classification) with a single Convolutional, Max pooling and Fully connected layers.

The filter is slid over the embedding matrix from the top to the bottom, whereas in computer vision tasks it slides from the left to right and top to bottom. Specifying a small receptive region of h words in a sentence, weights $w \in R^{h \times k}$ convolved with the word embedding in that specific field yielding an outcome $c(l)$ at index l as

$$c(l) = \sum_{j=1}^{k} \sum_{i=1}^{i+h} w(i,j) s_{l:l+h-1}(i,j) + b \tag{4.1}$$

where b is a bias term and $s_{l:l+h-1}$ are words from the l-th to $(l+h-1)$-th location in a sentence.

These filtered input c are fed into an activation function which is an element-wise non-linear function to yield a feature vector M of size $N-h+1$. The most commonly used activation function is the ReLU which is used in the hidden layers with a different activation function for the final layer depending on the task.

The pooling layer usually follows the convolution layer and the filter is applied on top of the output of the feature maps with a fixed operator such as max, min or average and reduces the number of parameters to learn. In our case we use max-pooling, which is very popular as it picks the most significant features from the feature vectors. Performing convolution with P different filters yields multiple feature vectors M_p

These pooled features are then inputted to the fully-connected (FC) layer, where each node is connected to all the other nodes in the next layer, to obtain a prediction y_i whose error between y_i will be backpropagated to train all the parameters in the FC layer, the filters and look-up table via optimization methods such as gradient descent. The necessary error is formulated as a loss function typically using cross-entropy.

In multi-class classification we need to classify the tweet into more than 2 classes. Where the y label can be one of the five performance measures. So for the final activation function we need to use a softmax function and Cross Entropy. The number of output nodes in the final layer is equal to the number of classes. Where each node gives the probability that label belongs to a particular class and we then use argmax to find the class index.

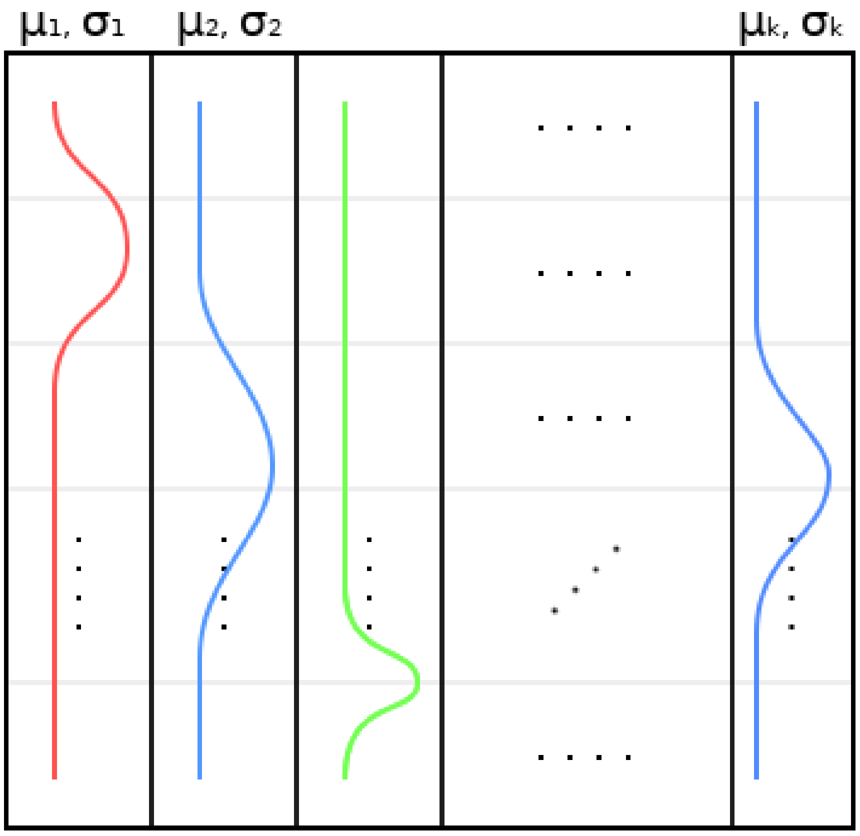

Figure 4.2: Modified Gaussian kernel used in MKCNN which has k (embedding size) number of means and standard deviations.

4.1 Multi-kernel Convolutional Neural Network

The convolution function is commonly used in digital image processing for smoothing, blurring or enhancing depending on the kernel used. To decipher the meaning of a sentence we need to take into consideration the context and position of words. The convolution operation can extract useful information from data computed from local context (i.e., neighboring) information. While traditional CNN models define a $w \in R^{h \times k}$ as a filter whose number of parameters are large and the size of the window h often remains small (e.g., 3 or 5) focusing only in a small region in a sentence. We therefore define a kernel function, e.g., Gaussian kernel $g_{\sigma,\mu}(\cdot)$ with mean and standard deviation , that are characterized by a small number of parameters

$$c(l) = \sum_{j=1}^{k} \sum_{i=1}^{i+h} g_{\sigma,\mu}(i,j) s_{l:l+h}(i,j) + b \tag{4.2}$$

whose outcome is a vector $c(l)$ that will go through an activation function (i.e., ReLU) to yield a feature map M. This concept of kernel convolution is visualized in Fig. 1 with a Gaussian kernel of two parameters. Here, the defines the localization of the kernel (i.e., the center of the filter) and controls its dilation (i.e., scale of the filter) that determine the width of local regions to cover with the filter. We define this kernel function for every dimension of the word embedding to extend our framework to a Multi-kernel Convolution Neural Network (MKCNN), where we learn different kernel functions adaptively for individual dimension of a word embedding. In short we are learning a probability density function.

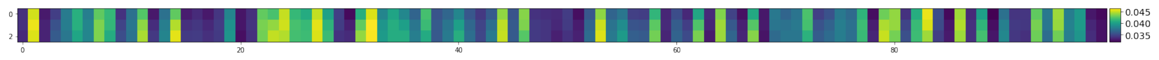

Figure 4.3: Initial weights for a MKCNN kernel of size 10 with and an embedding size of 100

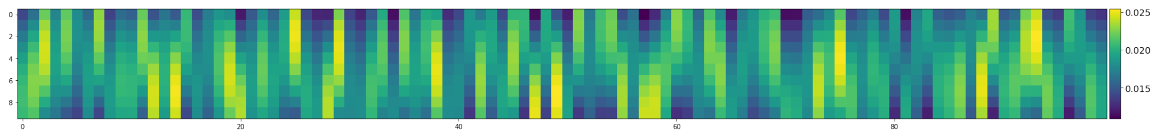

Figure 4.4: Initial weights for a MKCNN kernel of size 3 with and an embedding size of 100

CHAPTER 5

IMPLEMENTATION

The text from the dataset is tokenized so as to build the dictionary to map word to word embeddings in the Vocab object. Our model is similar to CNN model (Kim 2014) with one convolutional layer followed by a pooling layer and then the final activation layer which depends on the type of task.

Similar to other fundamental CNN architectures, our framework consists of an input layer, convolution layer, pooling layer and Fully-connected (FC) layer as shown in Figure 4.1 The input to the model is a text s_i (i.e., sentence) which has a label y_i assigned using a one-hot-vector encoding representing different classes (a vector in the size of the number of available labels with all 0s except for one 1 representing a specific class). First, an s_i is converted into an word embedding matrix which has a size of Nk. The dimension of the matrix is fixed and padded with 0s as necessary for differing lengths of sentences.

A layer of multi-kernel convolutions of s_i with P different filters sizes, which gives us P feature maps of M_p. Then, max pooling is performed over each feature map to extract the most important features and concatenated as a vector z. This z becomes the input to the FC layer whose parameters are denoted as with the final output layer with a softmax activation function. The output layer has O number of nodes that correspond to different labels to classify. To an input sentence s_i, the softmax function assigns probability distribution p_i^o for each class label that identifies the most likely label y_i (i.e., predicted label) as

$$p_i^o = \text{softmax}\left(\beta^T z_o\right) = \frac{e^{\beta^T z_o}}{\sum_o e^{\beta^T z_o}} \tag{5.1}$$

whose sum equals to 1. We select an output node that has the largest probability and assign a label that corresponds to that node, i.e., $\hat{y}_i = \arg\max_o p_i^o$.

To train the model, we formulate a loss function that measures the error between the prediction p_o and the true label y_i which we want to minimize. One of the loss functions commonly used for multi-class classification is the cross entropy which is defined as

$$L(p) = \sum_i \sum_o y_i^o \log\left(p_i^o\right) + \lambda \|\beta\|_2 \tag{5.2}$$

where y is the true class label, p is the models predicted probability, are the parameters in the FC layer and is a user parameter to control balance between cross-entropy error and l2-regularization to avoid over-fitting of the model. The output of the loss function is a positive scalar. The error is back propagated through the neural network and used to update the learnable of the model which are β, μ, σ and the look-up table (i.e., word embedding) in such a way to reduce the L_p that eventually reduces the error between prediction and groud truth. The gradient descent is performed using partial derivatives from each of these learn-able parameters, and we used the Adam algorithm in Tensorflow package in Python for back-propagation.

CHAPTER 6

EXPERIMENTS AND RESULTS

This study performed various experiments on the ride hailing text dataset that we collected, comparing MKCNN with three other baselines: 1) CNN introduced in [11], 2) traditional Recurrent Neural Network (RNN) and 3) Long Short Term Memory (LSTM) inside Recurrent Neural Network (RNN) architecture which is popular for text analysis [7]. For each of the task we will also be exploring and tuning the hyper-parameters (filter region size and number of feature maps) for the MKCNN model.

6.1 Sentiment Analysis

Sentiment analysis assess if a sentence yields positive or negative sentiment of the user and is one of the most popular and useful applications in text analyses. We performed sentiment analysis to identify ride-hailing user perceptions to their overall experiences using the tweets collected and annotated in this study.

The baseline configuration for all the models in the experiments is that the filter sizes are 7,9,11 with 10 feature maps each unless specified otherwise for the individual experiments. It has a dropout rate of 0.5 which is applied at the penultimate layer. The optimizer we used is Adam instead of the usual stochastic gradient algorithm. We train a model (MKCNN Random) where the word embeddings are randomly initialized with a fixed word embedding size of 32.

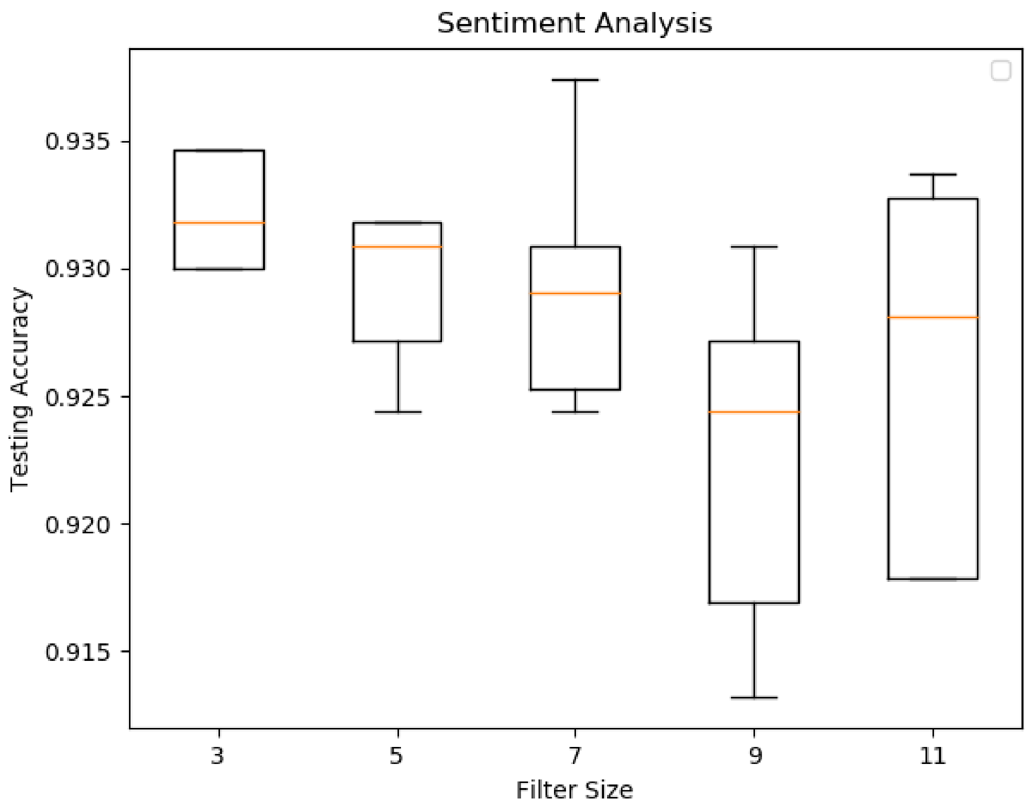

Figure 6.1: Effect of the filter size (one filter for each size) for Sentiment Analysis on Testing accuracy

The filter size is the first parameter we will explore using only one region size, we explore filter sizes from 3 to 11 because the length of tweets are relatively short. The mean testing accuracy is recorded over 5 replications.

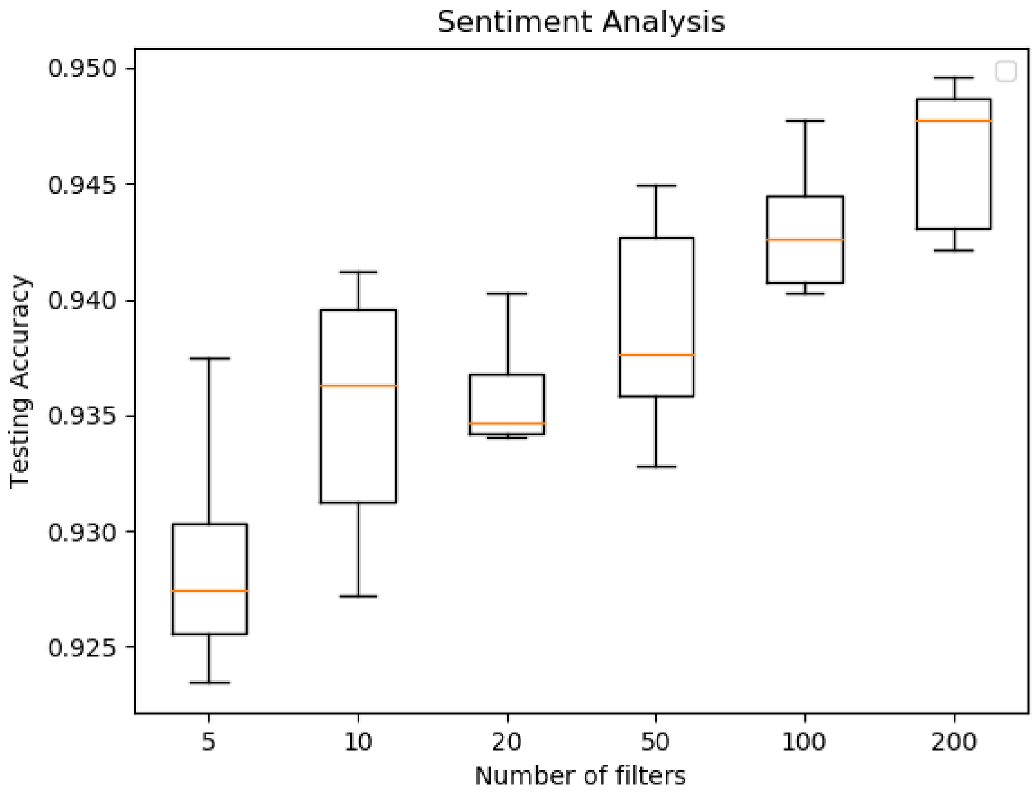

Figure 6.2: Effect of the Number of filters (filter size of 10) for Sentiment Analysis on Testing accuracy

Holding all variables constant we calculate the average performance over the number of filters for each filter size. From the Figure 6.2, increasing the number of filters helps with the accuracy. It's important to know that increasing the number of filters also increase the training time.

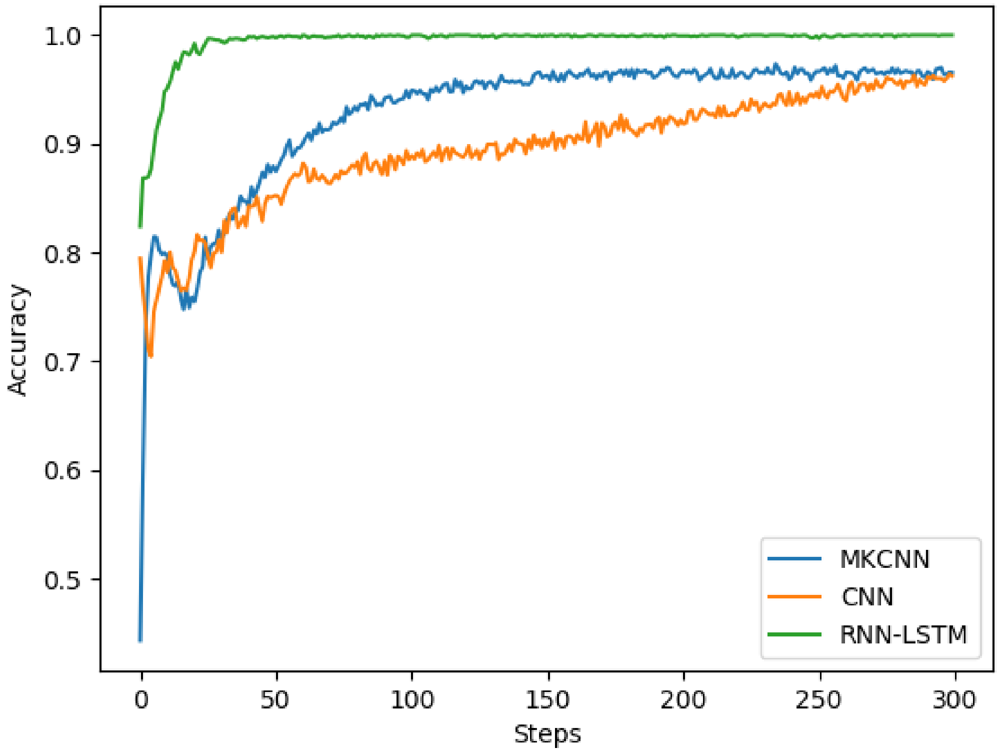

Figure 6.3: Testing accuracy vs steps for Sentiment Analysis on MKCNN, CNN and RNN-LSTM

Models	**Testing Accuracy**	**Precision**	**Recall**
MKCNN - Random	0.948	0.9415	0.9411
CNN - Random	0.949	0.9490	0.9458
RNN	0.8400	0.8923	0.8151
RNN - LSTM	0.8600	0.9123	0.9081

Table 6.1: **Performance of MKCNN and baseline models on Sentiment Classification**

Using our framework, the average accuracy for sentiment classification is high considering that there are many ambiguous and somewhat meaningless text on typical social media platforms. This sentiment analysis is a binary classification task which is the simplest form of a machine learning task, and it validates that our model is showing sound performance on a social media text classification task.

6.2 Sentence Categorization

In this task we will be performing a Multi-class classification to categorize the tweets into one of the five performance measure.

The label $y \in \{$cost, human interaction, reliability, safety, technology$\}$

From a total of 1,901 tweets, 1,331 (70%) were selected as the training set and the remaining 570 tweets were designated as the testing set for the machine learning experiments. The distribution of the 5 classes is as follows.

Performance Measure	**Number of tweets**
Human(behavior)	347
Reliability	335
Cost	254
Safety	223
Technology	172

Table 6.2: **Number of tweets for each performance measure on the entire dataset**

The same or similar hyper parameters were used to train the models. Where the embedding dimension was 32, number of filters was 10 and filter sizes were 7,9,11 and dropout was 0.5.

Also this is a 5 class classification problem, therefore 20% accuracy is expected by a random guess. With shorts tweets it's much harder to capture fine-grained semantic information and including any external data enriches the feature representation.

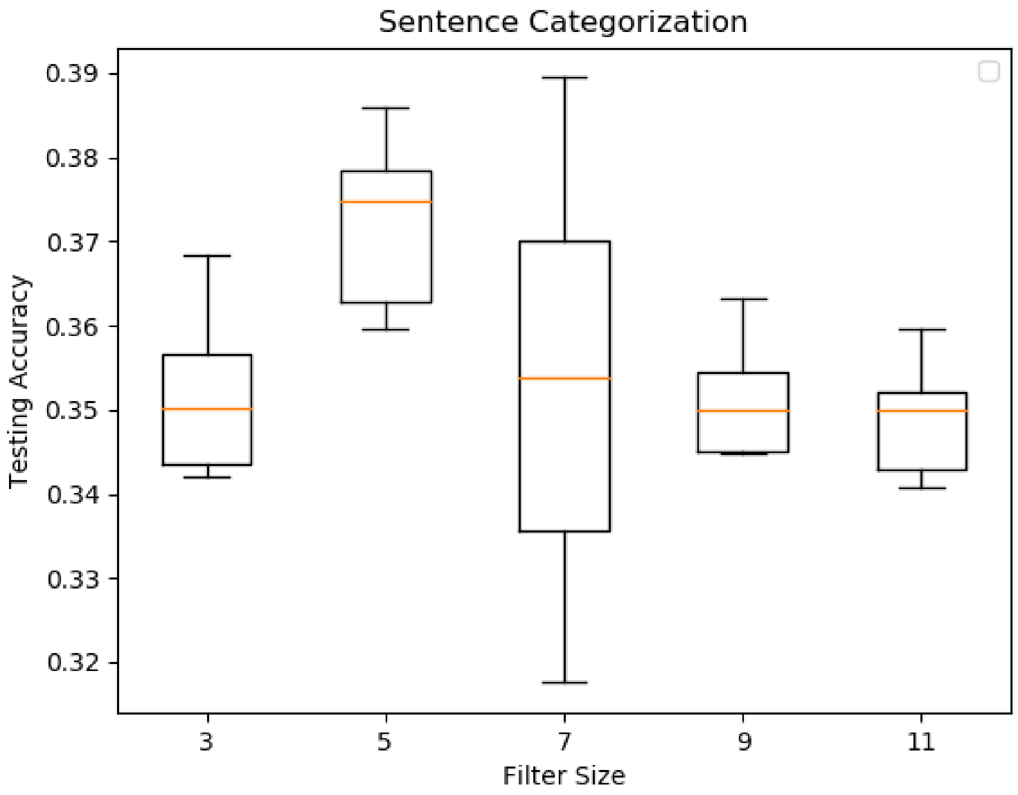

Figure 6.4: Effect of the filter size (one filter for each size) for Multi-class Classification on Testing accuracy

We perform a linear search over a single filter region size. From Figure 6.3 Again we see like with Sentiment prediction a larger window can capture dependencies across the sentence.

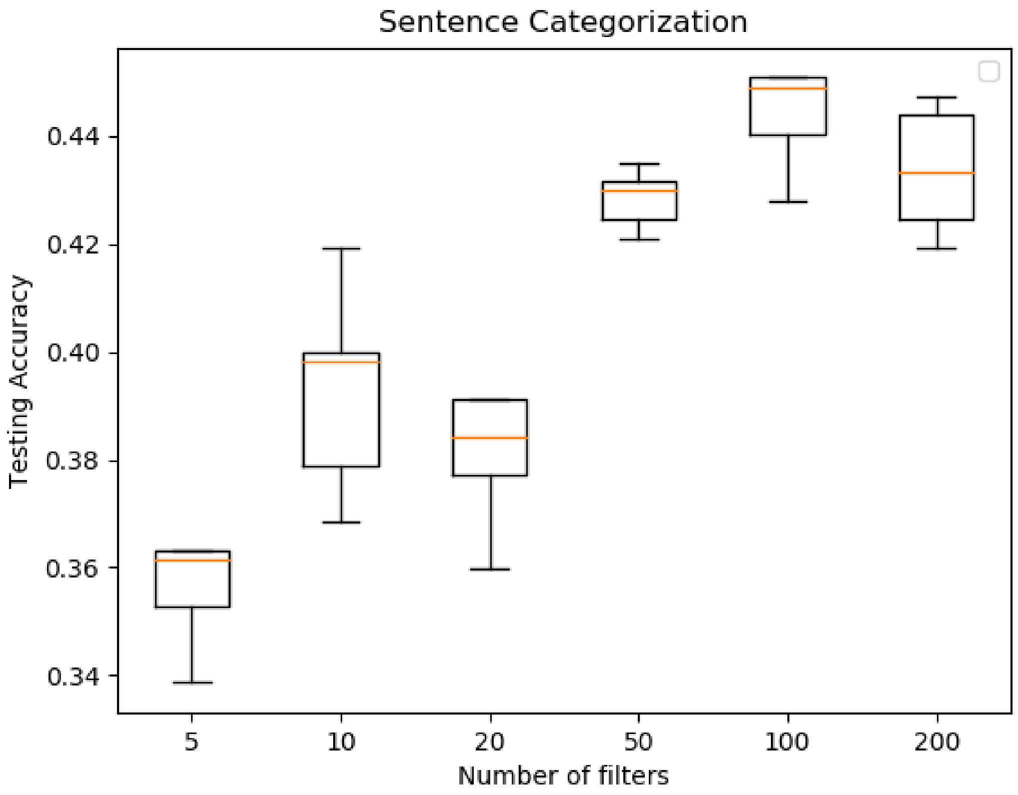

Figure 6.5: Effect of the Number of filters (filter size of 10) for Multi-class Classification on Testing accuracy

From Figure 6.4 we can see that increasing the number of filters can help the performance.

From the Figure 6.6 we can see that the model is overfitting as there is a significant difference between training accuracy and testing accuracy, due to significant amount of noise and desultory text which are natural in social media. Both models converged with training accuracy above 90% but MKCNN converged a little sooner.

A confusion matrix showing the category prediction result using our model is given in Figure 6.6. The rows in the confusion matrix correspond to the true label and the columns are our predictions. We can see that the diagonal elements, which show the correct prediction. This shows that our trained model is relatively correctly classifying unseen tweets in the testing dataset for categories which have significantly more tweets. The actual examples of the correctly classified tweets with both category and sentiment using MKCNN are given in Table 6.6. As mentioned earlier, the texts from social media contain significant noise and ambiguity from grammar error, typo, meme and special notations; nevertheless, our model was able to successfully categorize these tweets to an extent that make sense to human understanding.

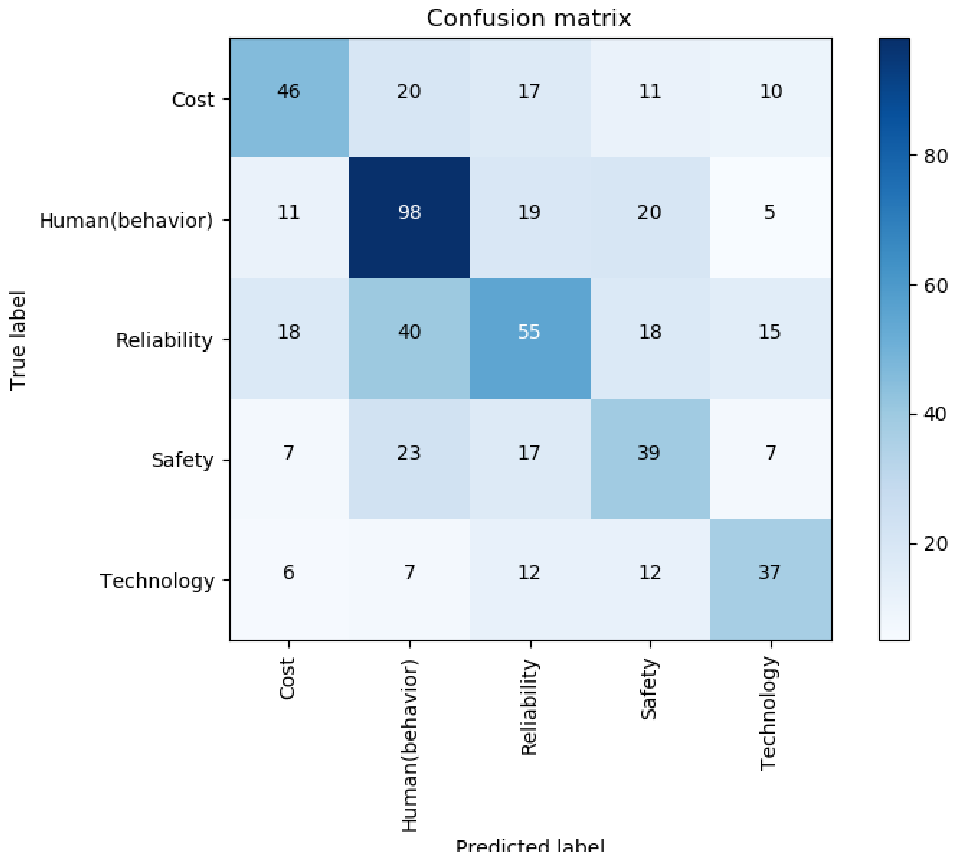

Figure 6.6: Confusion Matrix for Sentence Categorization with MKCNN on testing dataset.

The embedding size for RNN and RNN-LSTM is 100.

Models	**Testing Accuracy**	**Precision**	**Recall**
CNN - Random	0.475	0.4721	0.4701
MKCNN - Random	0.4824	0.4812	0.4824
RNN	0.2670	0.6430	0.2491
RNN - LSTM	0.3509	0.4210	0.3526

Table 6.3: **Performance of MKCNN and baseline models on Sentence Categorization**

In Table 6.4 we have some of the correctly classified tweets from the testing dataset.

Tweet	**Category**	**Sentiment**
@lastborn0805 @Taxifyng Well to me uber is cheaper	Cost	Positive
these lyft drivers need to shut up - no i dont wanna hear about your dog tammy	Human Interaction	Negative
Four layers: heat teach leggings, fleece lined tights, regular leggings and jeans. The cold went right through. Thank God for @uber. No way was I waiting for a bus and two trains in this cold.	Reliability	Positive
Yo this uber driver doing like 100 mph just going down Southern Avenue	Safety	Negative
uber man ran a red light im https://t.co/gO3A34lwOX	Technology	Negative

Table 6.4: **Examples of correctly classified tweets using MKCNN**

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis we introduce a novel algorithm, MKCNN, which trains and converges much faster than a CNN by reducing and summarizing the weights in a kernel using a Gaussian function. We also introduce a transportation performance measurement dataset which contains tweets labelled with sentiment and a new performance measure to understand user perception and experience on ride-hailing services. We show how hyper-parameters such as filter size and number of filters affects our model. MKCNN has shown significant improvement in the sentiment classification and sentence categorization task over our new dataset in both accuracy and faster convergence. The use case can be extended to other practical business applications to discern customer polarity and sentiment on large text based datasets. For the future, we will try and extend this problem to multi-label class classification.

REFERENCES

[1] B. Pang and L. Lee, “A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts,” in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, July 2004, pp. 271–278. [Online]. Available: https://www.aclweb.org/anthology/P04-1035

[2] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? sentiment classification using machine learning techniques,” in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, July 2002, pp. 79–86. [Online]. Available: https://www.aclweb.org/anthology/W02-1011

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[5] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[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 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, cite arxiv:1810.04805Comment: 13 pages. [Online]. Available: http://arxiv.org/abs/1810.04805

[9] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 655–665. [Online]. Available: https://www.aclweb.org/anthology/P14-1062

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D.

Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 649–657. [Online]. Available: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014, pp. 1746–1751. [Online]. Available: http://aclweb.org/anthology/D/D14/D14-1181.pdf

[12] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1107–1116. [Online]. Available: https://www.aclweb.org/anthology/E17-1104

[13] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 253–263. [Online]. Available: https://www.aclweb.org/anthology/I17-1026