Reproduction of Sentiment Analysis of Short Texts Paper

Junha Park, Brendan Furtado, Jonathan Ng
COMP-551 Applied Machine Learning, McGill University
Instructor: Will Hamilton

1 ABSTRACT

This work presents reproduction of techniques for sentiment analysis of short texts proposed in (dos Santos and Gatti, 2014). Such analysis tasks are challenging because of the limited contextual information available in its original text length. In an attempt to solve such problems effectively, we propose few strategies which explore combination of knowledge gained from word-level and char-level contents to perform sentiment analysis of short texts. Our approach was tested with two corpora proposed in the original paper (dos Santos and Gatti, 2014) : the Stanford Sentiment Treebank (SSTb), which contains movie reviews from Rotten Tomato; and the Stanford Twitter Sentiment (STS), which contains Twitter posts. For SSTb corpus, our approach achieves an accuracy of 62.1% in binary positive/negative sentiment classification. For the STS corpus, our approach achieves a binary classification accuracy of 81.3%.

2 Introduction

With advances in deep learning algorithms on textual information (Go et al., 2009; Barbosa and Feng, 2010; Nakov et al., 2013), the ability to perform the task of sentiment analysis has improved considerably in recent years. However, sentiment analysis of short texts, such as Twitter posts or movie reviews, still remain challenging because of the limited contextual information available by nature in these texts. Pre-training unsupervised word embeddings approach (Mikolov et al., 2013) is a powerful application to capture syntactic and semantic information of words exploiting prior knowledge from large sets of unlabeled texts. However, more effective approaches were proposed in our work by exploiting the character-level features' ability to capture morphological and shape information of texts.

Given a set of textual Twitter posts and reviews, our goal was to predict whether the underlying sentiment is positive or negative. Ultimately, we aim to reproduce a model capable of accurately gauging the underlying sentiment of new, unlabeled sentences. This can potentially serve two main functions: (1) helping organizations understand the social sentiment of their brand, product or service (2) analyzing underlying intentions and reactions toward the topic it concerns.

We apply our model for two different corpora from two different domains : movie reviews and Twitter posts. The movie review dataset used is the Stanford Sentiment Treebank (SSTb), which contains 215,154 sentiment labels of distinct phrases in the dataset dictionary. Given such sentiment labels, we averaged the sentiments of the phrases in a complete sentences to define a 5 class (Very Negative, Negative, Neutral, Positive, Very Positive) label of an article (i.e.movie review).

Sentiment	Movie Review
Positive	Watching these eccentrics is both inspiring and pure joy
Neutral	Paul Bettany playing Malcolm McDowell?
Negative	It's like watching a nightmare made flesh.

Figure 1: **Stanford Sentiment Treebank Data Sample**

As for our Twitter post dataset, we used Stanford Twitter Sentiment (STS) which contains 1.6 million tweets (i.e.Twitter posts) that were automatically labeled based on the sentiment value of an emoticon they contain. As an example, tweets with positive emoticons [e.g. :)] are labeled positive and tweets with negative emoticons [e.g. :'(] are labeled positive.

Sentiment	Tweet
Positive	scostalis: Jquery is my new best friend.
Neutral	schuyler: just landed at San Francisco
Negative	jvici0us: History exam studying ugh.

Figure 2: **Stanford Twitter Sentiment Corpus Data Sample**

Based on this criteria and previous work on this task (dos Santos and Gatti, 2014), we experimented with the following classifiers, which we discuss in subsequent sections: CNNs, RNNs, and LSTM.

3 Related Work

Machine text classification has a rich history in the information sciences (Salton, 1968; Swanson, 1960), emerging as a solution to the problem of organizing and accessing large quantities of information. Today, electronic text documents are among the most ubiquitous types of information repositories that exist (Harish, Guru, & Manjunath, 2010; Khan, Baharudin, Lee, & Khan, 2010). As such, the flexible and efficient retrieval of meaningful information from these sources poses an important problem.

Recent advances in machine learning (ML) and natural language processing (NLP) techniques have made this problem increasingly tractable, with ML classifiers often achieving comparable accuracy to human experts (Fung, Yu, Lu, & Yu, 2006; Swanson, 1960). Text documents are usually preprocessed into a more parsable representation (Harish et al., 2010; Porter, 1980). For example, bag-of-words methods represent each document as a vector containing the frequency counts of each term in the document (Salton, Wang, & Yang, 1975), and term weighting is applied to weigh each term to improve performance (Altınçay & Erenel, 2010; Lan, Tan, Su, & Lu, 2009). Some limitations of this approach can be addressed with regularization techniques (Forman, 2003; Yogatama and parts-of-speech tagging (Derczynski et al., 2013).

A unique challenge of sentiment analysis is classifying the polarity of a given text of a document whether it expresses positive, negative or neutral opinion in the document. Effectively solving this task requires extraction of information from input sentences in a disciplined way. The early attempts proposed to tackle such problem was based on Naive Bayes, Maximum Entropy Classification, and support vector machine (Pang et al., 2002). The method learns vector space representation in terms of word presence or word frequency. In (Bengio et al., 2003), with an attempt to learn joint probability function of words in a language using statistical modeling, the term 'word embedding' was first invented for distributed representation of words. The vector efficiently captures not only the semantic of the word itself but also that of the neighboring words in each training sentences. However, it was not until (Milkolov et al., 2013) that the true value of pre-trained word embedding was brought to the fore with the creation of Word2Vec toolkit.

Sentence level representation, before fully connected layers, is another key element in Sentiment Analysis. (dos Santos et al.,2014) proposes the use of convolutional neural networks (CNNs) for its sentence level representation model. Accordingly, we experimented with sentence level representation models and vector representation techniques of varying complexity, relying upon domain knowledge when it produced better performance.

4 Dataset and setup

Two different corpora proposed in the original paper (dos Santos and Gatti, 2014) : the Stanford Sentiment Treebank (SSTb), which contains movie reviews from Rotten Tomato; and the Stanford Twitter Sentiment (STS) are used for our task of sentiment analysis.

Stanford Sentiment Treebank (SSTb) (Socher et al., 2013b) composes of 215,154 phrases. Each phrase is labeled with its sentiment score (i.e. on a scale of 0-1 from Very Negative to Very Positive) in a phrase dictionary extracted using Stanford parser (Klein et al., 2003) from 11,855 movie review sentences. Keeping the phrase structure gives a considerable advantage on scoring the sentiment of sentences while maintaining the order of words.

Given such sentiment labels, we averaged the sentiment labels of phrases that compose each

complete sentences. This averaged labels (i.e Positive/Negative sentiment for sentiment score ≥ 0.5/ score < 0.5 respectively) were used for binary classification task on our SSTb dataset with training/validation/test split of 7,126/936/1,914.

Dataset	**set**	**#sentences / tweets**	**#classes**
SSTb	Training	7,126	2
	Dev	936	2
	Test	1,914	2

Table 1: **SSTb Data splits**

For Twitter posts, we used the Stanford Sentiment Corpus (STS) as our training set. The dataset originally contains 1.6 million tweets marked with positive or negative labels. For that reason, due the heavy computation cost of training models on huge dataset, we could not run 5-fold cross validation on all our experiments and also had to recreate our training/validation set with varying amount of samples for each fine-tuning experiment but with always the same proportion of 0.8/0.2. In addition, 3-class given testset of size 498 samples were recreated by ignoring samples with neutral sentiment labels.

Data sanitization. Basic data cleaning procedures included the removal of *html* tags, links, and non alphanumeric symbols. Specifically, we strippd comments from the html with use of BeautifulSoup, unencoded html tags, and used the following regular expression [^a-zA-Z0-9] to remove irrelevant symbols.

In addition, data samples with character lengths above 140 and less than 20 were ignored as outliers in the dataset.

The resulting strings were tokenized by each word and absolute significance of the occurrence in the string with the use of *Natural Language Toolkit (NLTK)*'s Regular Expression WordPunctTokenizer on a per word basis, which tokenizes a text into a sequence of alphabetic and non-alphabetic characters. Using this kit, we assigned the string "NaN" to replace the characters after '@' to remove usernames, and tweets with only usernames tags are ignored afterwards..

5 Proposed approach

We first established our own baseline using a standard sequential convolutional neural network. The first CNN (CNN 1/model 1) consists of embedding, Conv1D, max pooling, and dense layers. Word embedding plays a major role in this classification task. For the task, we used Stanford University's pre trained word embedding model called GloVe. This is one of many other effective models that helps convert text into numeric form. With the above parameters, we ran two different sets of experiments. One set consisted of three subset experiments that ran with the same parameters but different sample sizes of the total data we cleaned (total = 1,596,041). These samples were random, however, we provided a seed for reproducibility purposes. From these experiments, we wanted to determine if we'd see an increase in accuracy as the number of datapoints we trained and validated increased. The first sample we took consisted of 100,551 data points (0.063 of total dataset), the second was half the dataset (798,020 points), and the last was the full dataset. All of these samples were then trained on .8 of the total and validated on the remaining fraction.

With the same sample sizes in the first set of experiments, the second set consisted of running 3-fold cross validation over them. The validation part in these runs was one third of the sample size. 3-fold cross validation is only used on our first CNN model. These experiments together provided a baseline for our research.

For our second and third neural networks, we implemented and trained two Word2Vec models and combined them. One model was Continuous Bag of Words and the other Skip Gram. Both models are trained under Word2Vec for 30 epochs and are later combined when creating an embedding matrix to feed into the neural networks.

The second neural network resembles the first one closely, with a few layers modified and the third one is an RNN (LSTM). We wanted to determine if using a modified Word2Vec model would help increase accuracy.

Vector Representation of Input Words

Word embedding. Word embedding is the technique of representing word representations in contexts by "learning a distributed representation of words" (Bengio et al., 2003). There are two different styles of word embeddings, one in which words are expressed as vectors of co-occurring words, and another in which words are expressed as vectors of linguistic contexts in which the word occurs. The difference of style is compared in (Lavelli et al.,2004).

Word2Vec. Created and published by Google researchers (Mikolov et al., 2013), Word2Vec model proposed an unsupervised learning of word embeddings. An improvement from previous Skip-gram methods were implemented by countering imbalance between rare and frequent words by subsampling approach and training models above noises in words by negative sampling. We've experimented with two main Word2Vec models : Wikipedia trained model and pre-trained model on google news with continuous bag of words (CBOW) technique.

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Figure 3: **Negative sampling**

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

Figure 4: **Subsampling Approach for Frequent Words**

GloVe. Another unsupervised model to obtain vector representations for words were devised by Stanford researchers (Pennington et al., 2014). The main intuition of the model is that ratios of co-occurrence of word probabilities have the potential to have similar vector weights. Accordingly, the model performs well word analogy tasks.

Character level embedding. Given a word w composed of m characters $\{c_1, c_2, ..., c_m\}$, we first transform each character c_m into a character embedding r_m at the embedding layer. Then, the character embeddings $\{r_1, r_2, ..., r_m\}$ are fed into convolutional layer to re-define vectors with k windows of adjacent embeddings. Performing a max operation over produced char window vectors of the word $\{z_1, z_2, ..., z_m\}$, we can extract the feature vector of the input word (dos Santos et al., 2013).

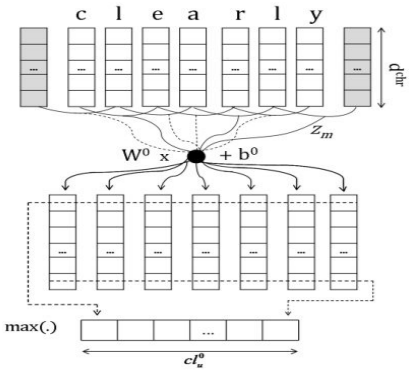


Figure 5: **Character level embedding**

(Hidden Dimension)Sentence Level Representation

CNN. CNNs are strong deep learning algorithms that learn weights and bias for specific predictions tasks. They are multilayers perceptrons that are usually composed of fully connected layers. For that reason, CNNs are prone to overfitting data. We've experimented with different regularization techniques such as drop out or max-pooling layers in order to reduce dimensionality but still learn key information from abstract form of our data.

LSTM. Unlike standard feed-forward neural networks, as a type of recurrent neural network (RNN) architecture, LSTM supports feedback connections (memory cell) in layers. This unique feature allows it to process well not only on single

data but also on sequences of data (e.g. English sentences in our case).

6 Results

Starting with our baseline CNN, we saw an increase in test accuracy as the number of sample points increased. The table below shows some of the results of the model with different sample sizes.

Performance of Baseline CNN

Dataset size	Cost	Accuracy
100,551 (frac = 0.063)	0.5610	0.7242
100,551 (W/ 3-fold)	0.6350	0.7410
789,020 (frac = 0.5)	0.4588	0.7632
789,020 (W/ 3-fold)	0.4568	0.7883
1,596,041 (frac = 1)	0.4389	0.8022
1,596,041 (W/ 3-fold)	0.4700	0.8050

Table 1: **Cost & accuracy of baseline CNN model**

From the above table, it can be seen that the accuracy increases as the sample size increases. All the k-folds for each sample size also improved a bit over just the one run through. The graphs for these experiments can be seen in the Appendix below. Because bigger sample sizes led to better results, we decided to use larger sample sizes for our subsequent neural networks.

The results for the first Word2Vec neural network is shown in table 2. In this network we tested out a few different hyperparameters to see if they would make a difference. One of them was the trainable and validation split. We ran roughly 5-6 epochs at different times and the best model was recorded and then loaded to run on the test set.

Performance of W2V-CNN

Dataset size	Val Split	Trainable	Avg Runtime	Cost	Accuracy
1,596,041	0.01	True	510 sec	0.5072	0.7883
1,596,041	0.01	False	105 sec	0.4377	0.7743
1,596,041	NA	True	~500 sec	0.4300	0.8106

Table 2: **Cost & accuracy of W2V-CNN model**

From these experiments we determined that setting trainable to true will yield better results despite each epoch taking approximately 510 seconds long (~5 times longer). Since accuracies often ended being higher than the baseline accuracies (even just slightly) we ran this neural network more than several times to get an accuracy of 81% (best case we ran into).

Based on the graphs (in the appendix) validation accuracies through training seemed to show gaps at times. The validation loss for each experiment showed some consistency by always having losses above the training line. This is probably due to overfitting issues on our network.

Performance of W2V-RNN

Dataset size	Val Split	Trainable	Avg Runtime	Cost	Accuracy
100,551	0.15	False	8 min	0.4785	0.7799
100,551	0.15	True	8 min	0.5273	0.7549
532,014	0.15	False	30 min	0.4149	0.8134

Table 3: **Cost & accuracy of W2V-RNN model**

Clearly from this experiment set-up, higher test accuracies were achieved when word embedding and LSTM layers' weights were not set trainable. Another noticeable observation is that, with bigger sample sizes, the test accuracy of RNN model improved significantly.

Performance Across Models

Models (best results)	Cost	Accuracy
CNN 1 (baseline)	0.4389	0.8022
CNN 1 (K-fold =3)	0.4699	0.8050
CNN 2 (W2V)	0.4301	0.8106
RNN (W2V)	0.4150	0.8134

Table 2: **Cost & accuracy across different models**

Out of all the models, the RNN yielded the best accuracy despite running training and validation on only a third of the cleaned data.

Table 4: **Cost & accuracy of different models implemented**

Discussion and Conclusion

In this project, although we did not fully replicated the accuracy of the original paper, we have demonstrated the effectiveness of feed-forward neural network being an effective tool for sentiment analysis. Hyperparameters should be explored more thoroughly and deeply in future attempts for the paper's replication. Overfitting seemed to be an issue across all models, something that was difficult to fix especially with handling this much data. We would also like to explore more detailed or lower level representations of inputs and their effect on the accuracy towards the predicted model. Exploring how to make use of other databases and developing mixed parameters is also a topic we would like to explore in the future.

Statement of Contribution

Brendan Furtado implemented the models and graphs. Wrote part of the paper.
Junha Park was responsible for implementing word-char embedding model, dataset preparation and write-up.
Jonathan Ng

References

NLTK's list of english stopwords (August 27, 2010). Github discussion thread. Accessed 2019 at https://gist.github.com/sebleier/554280.

Keras's Concatenate Word and Character Embeddings https://github.com/CyberZHG/keras-word-char-embd

Harish, B. S., Guru, D. S., & Manjunath, S. (2010). Representation and classification of text documents: A brief review. *IJCA, Special Issue on RTIPPR (2)*, 110-119.

Salton, G., & Lesk, M. E. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM (JACM)*, *15*(1), 8-36.

Swanson, D. R. (1960). Searching natural language text by computer. *Science*, *132*(3434), 1099-1104.

Khan, A., Baharudin, B., Lee, L. H., & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, *1*(1), 4-20.

Fung, G. P. C., Yu, J. X., Lu, H., & Yu, P. S. (2005). Text classification without negative examples revisit. *IEEE transactions on Knowledge and Data Engineering*, *18*(1), 6-20.

Fiallos, A., & Jimenes, K. (2019, April). Using Reddit Data for Multi-Label Text Classification of Twitter Users Interests. In *2019 Sixth International Conference on eDemocracy & eGovernment (ICEDEG)* (pp. 324-327). IEEE.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, *3*(Mar), 1289-1305.

Altınçay, H., & Erenel, Z. (2010). Analytical evaluation of term weighting schemes for text categorization. *Pattern Recognition Letters*, *31*(11), 1310-1323.

Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013, September). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013* (pp. 198-206).

dos Santos, C. & Gatti.M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *Proceedings of COLING 2014, the 25th International Convergence on Computational Linguistics: Technical Papers*, 69-78

Alec. G, Richa. B & Lei. H, (2009) Twitter Sentiment Classification using Distant Supervision, *Stanford University, Technical Paper*

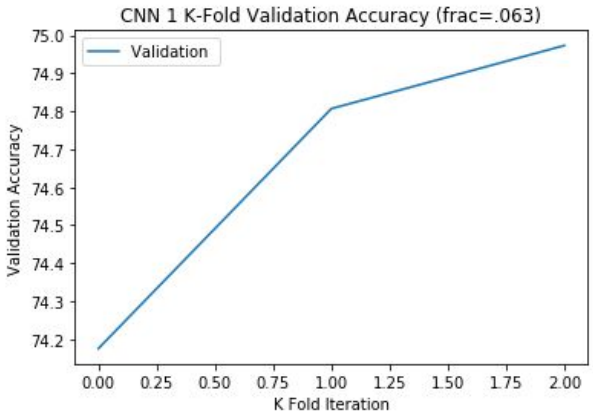
L. Barbosa & J. Feng. (2010). Robust Sentiment Detection on Twitter from Biased and Noisy Data. *COLLING 2010: Poster Volume*, 36-44.

Mikolov. T, Chen. K, Corrado. G & Dean. J (2013). Efficient Estimation of Word Representations in Vector Space. *https://arxiv.org/pdf/1301.3781.pdf*

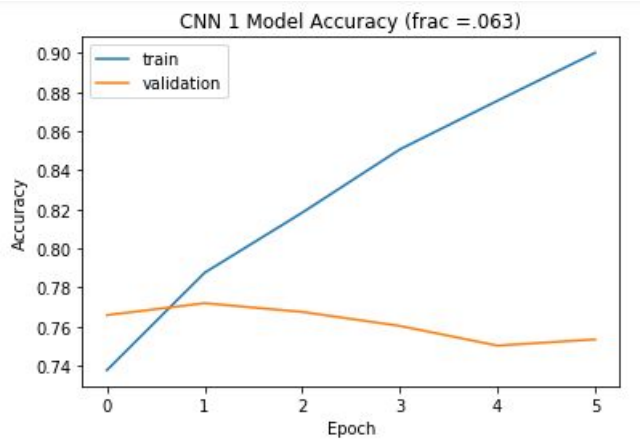
Pang. B, Lee. L & Vaithyanathan. S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*,, 79-86.

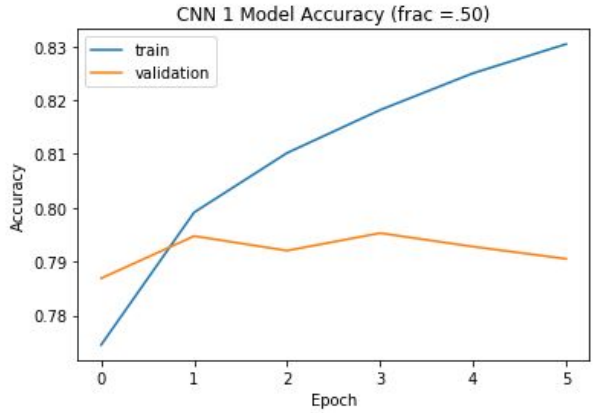
Bengio. Y, Ducharne. R, Vincent. P & Jauvin. C. (2003).A Neural Probabilistic Language Model. *Journal of machine learning research, 3*(Feb), 1137-1155.

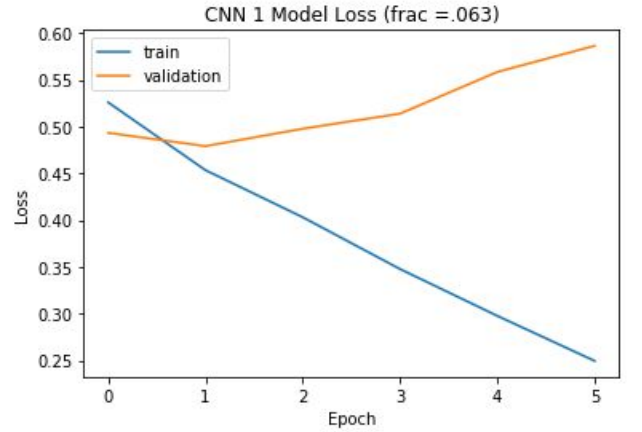
Pennington. J, Socher. R & Manning. C. (2014).Global Vectors fo Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532-1543. https://www.aclweb.org/anthology/D14-1162.pdf

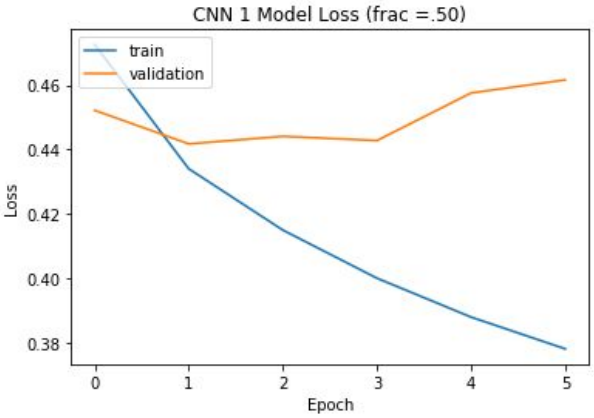


Appendix

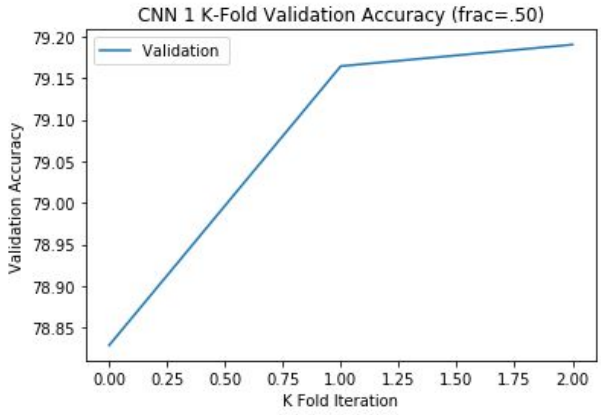


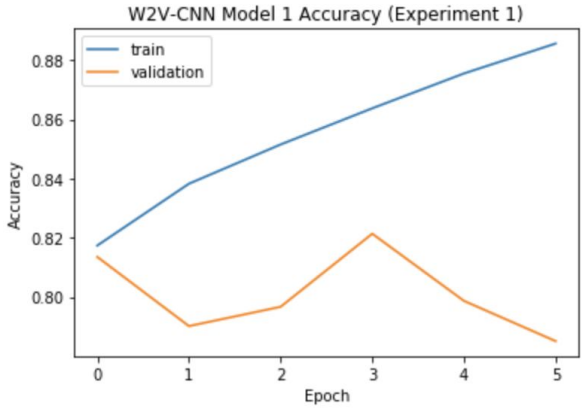


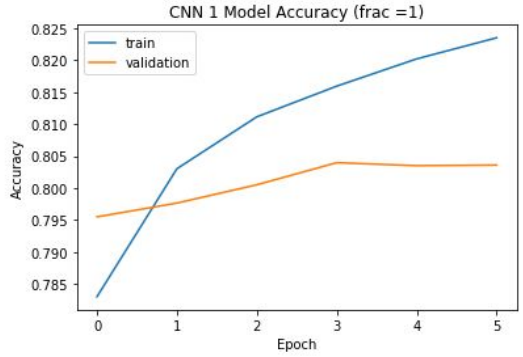


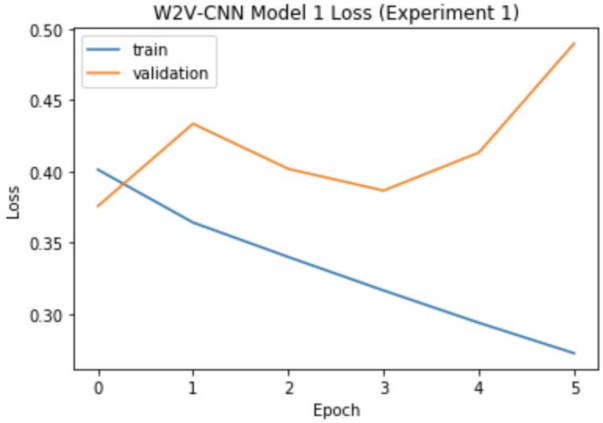


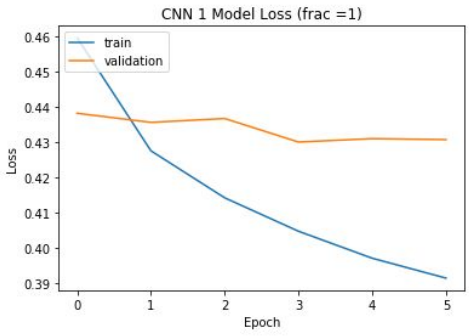
- K-Fold graphs
-

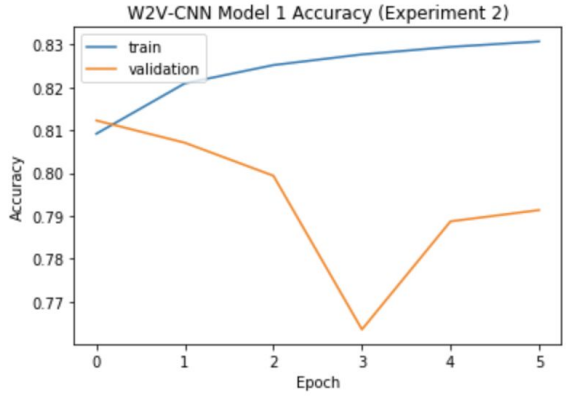

CNN 1 K-Fold Validation Accuracy (frac=.50)
Validation
Validation Accuracy
K Fold Iteration

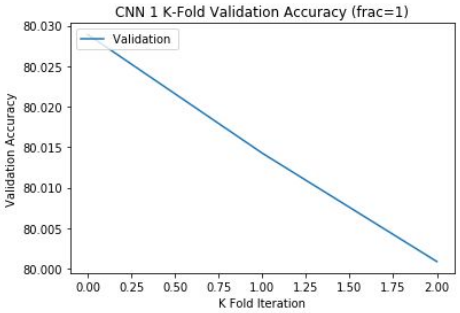

W2V-CNN Model 1 Accuracy (Experiment 1)
train
validation
Accuracy
Epoch


CNN 1 Model Accuracy (frac =1)
train
validation
Accuracy
Epoch


W2V-CNN Model 1 Loss (Experiment 1)
train
validation
Loss
Epoch


CNN 1 Model Loss (frac =1)
train
validation
Loss
Epoch


W2V-CNN Model 1 Accuracy (Experiment 2)
train
validation
Accuracy
Epoch


CNN 1 K-Fold Validation Accuracy (frac=1)
Validation
Validation Accuracy
K Fold Iteration

December, 2019, Quebec, Montreal Canada	J. Park, B.Furtado, J.Ng