MiniProject 4: Implement and improve the baselines

for the paper: Hierarchical Attention Networks for Document Classification [1]
COMP 551 - Group 36

Jiangning Luo
260851506
McGill University
jiangning.luo@mail.mcgill.ca

Muhammad Hasnain Mamdani
260850043
McGill University
hasnain.mamdani@mail.mcgill.ca

Bide Xu
260711367
McGill University
bide.xu@mail.mcgill.ca

April 18, 2019

Abstract

Document classification is a popular task in natural language processing that aims to extract opinion or polarity of a text document. In this project, we survey and evaluate various supervised machine learning algorithms with different feature extraction pipelines for processing text data to analyze the sentiment of IMDB and Yelp reviews. In particular, we extensively implement multiple baseline methods suggested in [1], which proposes a hierarchical attention network for document classification, and were able to improve the baseline models to obtain even better results. LSTM model with 3 hidden layers, 512 hidden states and 400 dimensions in embedding layer achieved 64.29% accuracy on the Yelp data set, which is 6.09% higher than the corresponding baseline of [2]. Moreover, our CNN model with word embedding, a new baseline method, achieved 49.32% accuracy on the IMDB data set, which is marginally higher than all the baselines for IMDB data set provided in [1].

1 Introduction

Sentiment analysis involves predicting the attitude of a subjective text. Two fundamental challenges in text classification are the extraction of useful features that provide intuition about the sentiment of text and generating fixed-length features from variable length text that can be used to train learning algorithms. Traditional approaches of sentiment classification use sparse lexical features to represent documents, such as bag of words (BoW), n-grams and Term frequency times inverse document frequency (TF-IDF) [3], and then use a linear model or kernel methods on this representation [4] [5]. More recent approaches use deep neural networks such as convolutional neural networks [6] and recurrent neural networks based on long short-term memory (LSTM) [7] to learn text representations. Neural-network-based approaches have been quite effective as they incorporate some contextual information of texts that help understand the sentiment better [8] [2] [9] [10]. In this project, we apply both the traditional and modern approaches to analyse the sentiment of Yelp and IMDB movies reviews and analyse the results. The best LSTM model with 3 hidden 512 hidden states LSTM layers and 400 dimensions in embedding layer achieved 64.29% accuracy on yelp data set. The best CNN model with word embedding achieved 49.32% accuracy on IMDB data set. These results are better than the corresponding baseline models defined in the paper [1] and competitive with the Hierarchical Attention Networks (HAN) model especially for IMDB dataset.

2 Related Work

The case of directly using a high-dimensional one hot vector as input, where each word in the corpus corresponds to a unique dimension of the vector, is explored by [9]. [8] use neural network for text

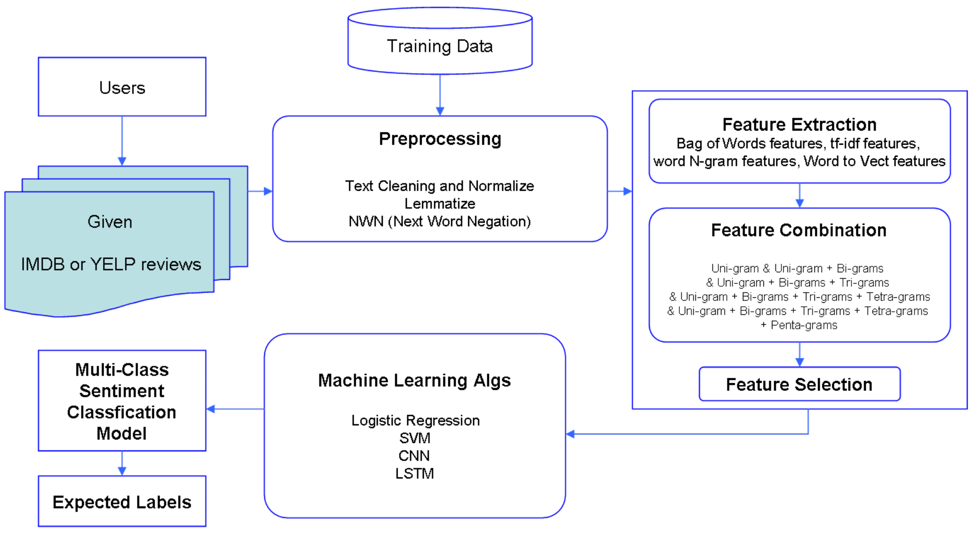

Training Data
Users
Preprocessing
Text Cleaning and Normalize
Lemmatize
NWN (Next Word Negation)
Feature Extraction
Bag of Words features, tf-idf features, word N-gram features, Word to Vect features
Feature Combination
Uni-gram & Uni-gram + Bi-grams
& Uni-gram + Bi-grams + Tri-grams
& Uni-gram + Bi-grams + Tri-grams + Tetra-grams
& Uni-gram + Bi-grams + Tri-grams + Tetra-grams + Penta-grams
Feature Selection
Given
IMDB or YELP reviews
Multi-Class Sentiment Classfication Model
Expected Labels
Machine Learning Algs
Logistic Regression
SVM
CNN
LSTM

Figure 1: Flow chart of our proposed approach

classification - a direct application of CNNs [11] as used in computer vision but with NLP interpretations. [2] use a character-level CNN for text classification and achieve competitive results. Recurrent Neural Networks, having the ability to remember important information across long stretches of time, have proven to be very effective in tackling this problem. Tree-structures LSTMs are used for classification in [12]. There are also some works that combine LSTM and CNN structure for sentence classification [13] [14]. [10] use hierarchical structure in sentiment classification. They first use a CNN or LSTM to get a sentence vector and then a bi-directional gated recurrent neural network to compose the sentence vectors to get a document vectors. The attention mechanism was proposed by [15] in machine translation. They use encoder decoder framework and an attention mechanism is introduced to select the reference words in original language for words in foreign language before translation. Further uses of the attention mechanism include parsing [16] and natural language question answering [17] [18] [19]. [1] for the first time introduces a *hierarchical* attention mechanism.

3 Dataset and Baselines Setup

The approach we considered is summarized in the flow chart (Figure 1), which will be explained in the following sub-sections.

3.1 Dataset

We used IMDB movie review data set[1] and Yelp review data set[2] in our project. IMDB movie review data set consists of 135,669 movie reviews with one paragraph per review. Each review has ten levels of ratings from 1 to 10 (1 is the worst, 10 is the best)[3]. Yelp review data set is obtained from Yelp Data set Challenge in 2015, consists of 649,999 reviews. Each review has five level of ratings from 1 to 5 (1 is the worst, 5 is the best)[4]. During training process, we randomly chose 80% of the data set as training set and 10% of the data set as validation set. During testing, we chose the rest 10% of the data set as test set. For SVM and Logistic Regression models, the experiments are carried out based on 5 fold cross-validation. The predicted results are evaluated on the held-out validation dataset to estimate and compare the performance among these different models.

[1]http://www.imdb.com/
[2]http://www.yelp.com/dataset_challenge
[3]The IMDB dataset with 10 labels were suggested in [20], can be downloaded from the following link: https://www.dropbox.com/s/0oea49j7j30y671/data.json?dl=0 See https://github.com/nihalb/JMARS for details.
[4]The Yelp 2015 dataset with 5 labels were suggested in [2], can be downloaded from the following link: http://goo.gl/JyCnZq See https://github.com/zhangxiangxiao/Crepe for details.

3.2 Baselines Setup

We implemented several baselines models of [1], such as Logistic Regression, SVM, CNN and LSTM models.

3.2.1 Logistic Regression and SVM

The Logistic Regression model and SVM [21] model are implemented using sklearn package [22]. For these models, we lower cased all letters of the reviews and lemmatize them.

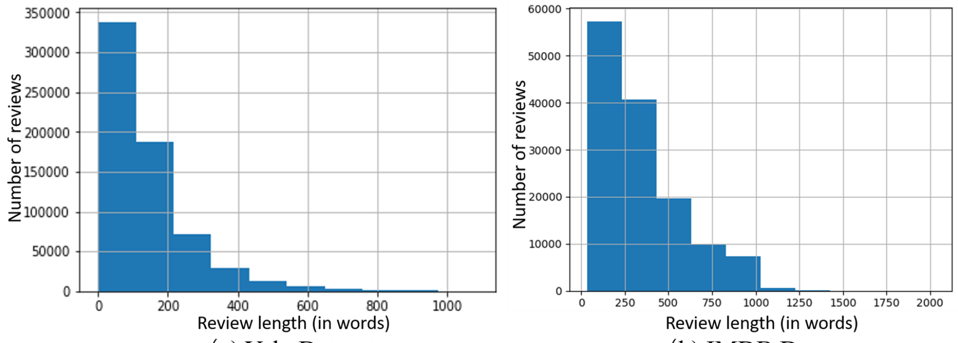
3.2.2 CNN and LSTM

Also, we have tested on both CNN and LSTM models based on PyTorch [23]. For these models, we first lower cased all letters and removed punctuation from the reviews. Secondly, we created an index mapping dictionary by assigning lower indexes to the words that occur more frequently. Then, we encoded the reviews using the index mapping dictionary we created. An example of encoding a sentence into a list of indexes is shown in Figure 2:

8	6	3	51	48	21
it	is	a	very	good	movie

Figure 2: An example of encoding a sentence into a list of indexes

Both CNN model and LSTM model need inputs with the same size. Therefore, we need to pad shorter reviews with zeros to specific length or truncate the longer reviews to specific length. In order to unify the sizes of inputs, we need to analyze the length of the reviews first. Figure 3 (a) shows how the length of Yelp reviews distributed. Figure 3 (b) shows how the length of IMDB reviews distributed. From Figure 3, we can tell that most of the IMDB reviews are less than 1000 words, while most of the Yelp reviews are less than 600 words. Therefore, we set the length of IMDB inputs to 1000 and set the length of Yelp inputs to 600.



(a) Yelp Dataset (b) IMDB Dataset

Figure 3: Distribution of the length (in words) for datasets of Yelp (a) and IMDB (b) reviews.

3.2.3 CNN-word

The CNN-word model, proposed by [8] is based on embedding the words into vectors using word2vec [24] vectors. Each of the vectors has a dimension of 300.

4 Experiments

4.1 Logistic Regression

For both datasets, we used TF-IDF [3] and bag of words (BoW) in our pipelines for feature selection. For BoW feature, we set the value of min-df to 0.0002, which means ignore terms that appear in less than 0.02 % of the documents. For both BoW and TF-IDF, we tuned the n-gram parameter. In Logistic Regression model, we used L2 regularization and set the value of C to 20. During hyperparameter selecting stage, we used 5-fold cross validation to choose our hyperparameters. To be consistent to the experiment baseline settings of [1], we tested out many different values for possible n-gram size, from 1-5. The max feature count is limited to 50000 for 1gram while limited to 500000 for 2grams to 5grams.

4.2 SVM

Similar to Logistic Regression model, we also used TF-IDF and BOW in our pipelines for feature selection for the SVM model [21]. For BoW feature, we ignore terms that appear in less than 2 documents. For both BoW and TF-IDF, we did 1-gram and 2-gram, follows the experiment settings of [1], without limiting the feature numbers. In SVM model, we set the values of penalty parameter C of the error term to 1. During hyperparameter selecting stage, we also used 5-fold cross validation to choose our hyperparameters. Also, 5-folder cross validation is used for both SVM and Logistic Regression models to validate the accuracy results. Experiments for SVM and Logistic Regression are carried out in local machine with 32G memory, since the sklearn implement for word feature retrieve requires large amount of memory, especially for large dataset like Yelp 2015 (650k reviews).

4.3 CNN

The model architecture of CNN was inspired by [8]. While our CNN model used integers to encode words and used fixed size matrices (1000 $\times$ 1) as inputs. Firstly, we constructed CNN model A with five convolution layers. In each convolution layer, the padding is 1 on height and 0 on width, the size of filter is 3 $\times$ 1 and the stride of filter is 1. All convolution layers are equipped with rectification (ReLU). Each convolution layer is followed by a max-pooling layer. First two max-pooling layers are performed over 2 $\times$ 1 window with stride 2. The other max-pooling layers are performed over 5 $\times$ 1 window with stride 5. The size of outputs at The end of convolution layers are 2 $\times$ 1. The stack of convolution layers is followed by two Fully-Connected (FC) layers. The first one has 512 channels and the second one has 10 channels. For regularization, we added a dropout layer with probability 0.5 before each FC layer and added batch normalization function after each convolution layer. After we constructed model A, we added convolution layers into model A and obtained model B, C and D. A summary of the layouts of CNN models are shown in Figure 4. We used Adam optimizer with learning rate 0.001 to optimize all CNN parameters. The batch size is 64 and 5 epochs are used during the training process.

4.4 LSTM

The architecture of our LSTM [7] model is a basic RNN model with LSTM (Figure 5). Firstly, we started from LSTM model A with two hidden LSTM layers. In each LSTM layer, the dimension of hidden state is 256. Before LSTM layers, there is an embedding layer which convert our encoded words into embedding with specified size. The dimension of embedding layer was set to 200 in model A. The stack of hidden LSTM layers is followed by a FC layer, which has 10 channels for IMDB and 5 channels for YELP, respectively. We consider the output from the last time step as the final output of our LSTM network. For regularization, we added a dropout function with probability 0.3 before FC layer. In addition, we used gradient clipping function with threshold 0.5 in order to prevent exploding gradient. After we constructed model A, we added the number of hidden layers, tuned the dimension of hidden state and the size of embedding, then we obtained another six models. A summary of layouts of

A	B	C	D
5 weight layers	10 weight layers	15 weight layers	20 weight layers
Input (1000x1 matrix, 1 channel)			
1×Conv3-128	2×Conv3-128	3×Conv3-128	4×Conv3-128
Maxpool2			
1×Conv3-256	2×Conv3-256	3×Conv3-256	4×Conv3-256
Maxpool2			
1×Conv3-512	2×Conv3-512	3×Conv3-512	4×Conv3-512
Maxpool5			
1×Conv3-1024	2×Conv3-1024	3×Conv3-1024	4×Conv3-1024
Maxpool5			
1×Conv3-1024	2×Conv3-1024	3×Conv3-1024	4×Conv3-1024
Maxpool5			
FC-512			
FC-10			

Figure 4: The layouts of CNN models. Note that the convolution layers are denoted as 'conv(filter size)-(number of channels)' and the ReLU, dropout and batch normalization functions are not shown.

LSTM models are shown in Figure 6. We used Adam optimizer with learning rate 0.001 to optimize the parameters. The batch size is 64 and 4 epochs are used during the training process. The LSTM as well as CNN models were tested on kaggle kernel[5], with GPU support (11G GPU memory) and 14G memory, for without GPU support, these deep learning models would run slowly.

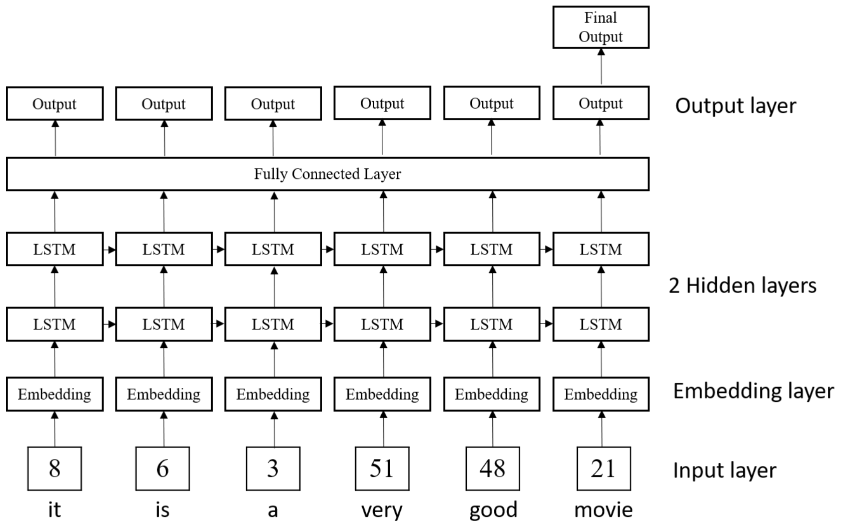
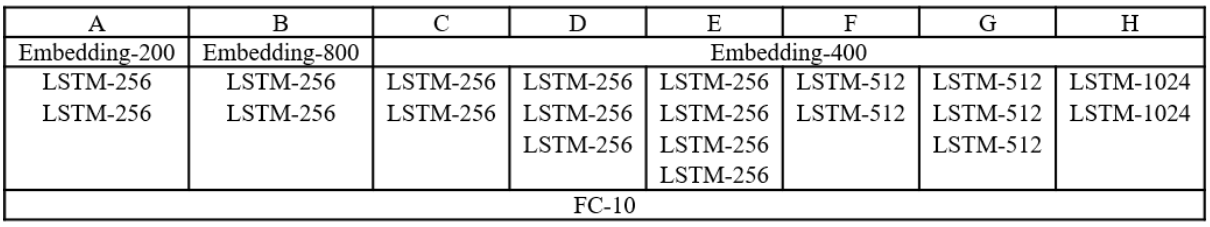


Figure 5: The architecture of LSTM model

4.5 CNN-word

The model architecture of CNN-word is shown in Figure 7. The inputs of CNN-word model are the embedded reviews, the ith row of an input is the ith embedded word in the corresponding review. The model of CNN-word consists of one 50-Channel convolution layer followed by maxpool layer and FC layer. A convolution operation is performed in a filter which is applied over a window of 3 words, the stride of filter is 1. There are 50 channels obtained from convolution operation. After convolution operation, maxpool operation will be performed over each channel, obtaining the maximum value of each channel. Finally, an FC layer is performed over the 50 maxpooled outputs from 50 channels and obtaining an output of 10 classes for IMDB and 5 classed for Yelp. We used log softmax function a activation function and for regularization, we added a dropout layer with probability 0.5 before each FC layer. The CNN-word model was tested on Google colab[6], because it provides GPU (11G GPU memory) support and the Google Drive[7] could be easily mounted to Google Colab during the data preprocessing procedure to

[5]https://www.kaggle.com
[6]https://colab.research.google.com/
[7]https://www.google.com/drive/



A	B	C	D	E	F	G	H
Embedding-200	Embedding-800	Embedding-400					
LSTM-256	LSTM-256	LSTM-256	LSTM-256	LSTM-256	LSTM-512	LSTM-512	LSTM-1024
LSTM-256	LSTM-256	LSTM-256	LSTM-256	LSTM-256	LSTM-512	LSTM-512	LSTM-1024
			LSTM-256	LSTM-256		LSTM-512	
				LSTM-256			
FC-10							

Figure 6: The layouts of LSTM models. Note that the Embedding layers are denoted as ”Embedding-number of dimensions” and the LSTM layers are denoted as ”LSTM-number of channels”. Dropout layer is not shown.

store the pre-processed data.

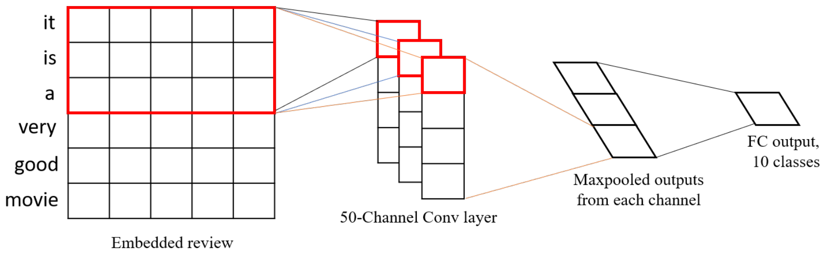


Figure 7: The architecture of CNN-word for IMDB review dataset (10 classes)

5 Results and Discussion

We evaluated the performance of our models using accuracy function. Which is calculated by number of right predictions divided by total number of predictions. Using the settings described above, we obtained the accuracy on validation set for each model. The results of logistic regression models, SVM models, CNN models and LSTM models are shown in Table 1. Among logistic regression models, 2-gram BoW model performed the best on Yelp data set while 4-gram TF-IDF model performed the best on IMDB data set. Among SVM models, 2-gram BoW model performed the best on Yelp data set and 2-gram TF-IDF model performed the best on IMDB data set. All CNN models, except the CNN-word model, performed poorly, therefore we did not test any of them on test set. Among LSTM models, model F performed the best on both data sets. We tested the best logistic regression model (2-gram BoW on Yelp, 4-gram TF-IDF on IMDB), SVM model (2-gram BoW on Yelp, 2-gram TF-IDF), CNN-word model and the best model among LSTM (model C) on test set. Table 2 shows a detailed comparison of results obtained in this paper against the baselines. For Yelp Data set, LSTM performed the best on test set. Comparing to the best performance of previous models (HN-ATT), our accuracy is only 6.71% lower. For IMDB data set, CNN-word performed the best, and it results in accuracy very similar to the Hierarchical Attention Networks (HAN) models. Table 3 shows the results taken directly from the HAN model in [1] for reference. HN stands for Hierarchical Network, AVE indicates averaging, MAX indicates max-pooling, and ATT indicates their proposed hierarchical attention mode.

5.1 Insights concerning Hardware and Software constraints

We found that when applying BOW or TF-IDF feature generation method for SVM and Logistic Regression models, the training time (as detailed in Table 2) increases significantly as the number of grams increase. For instance, the 1-gram BOW takes only 30 minutes to train for the Yelp 2015 dataset, and this time could be further reduced by parallelizing the task with multi-core CPUs, for carrying out the multi-fold cross validation simultaneously. While for 5-grams BOW or TF-IDF, the training time could reach 136 minutes (more than 2 hours). We observe that the bottleneck is the memory usage, especially for large dataset like Yelp 2015 (includes 650k reviews), the memory usage would exceeds

Model	Method	Yelp ‘15		IMDB	
		Accuracy (%)	Train time (mins)	Accuracy (%)	Train time (mins)
Logistic Regression	1-gram BoW	59.02	30	42.64	4
	2-gram BoW	61.32	43	45.15	7
	3-gram BoW	61.31	70	45.63	11
	4-gram BoW	61.26	100	45.76	17
	5-gram BoW	**61.33**	136	45.82	14
	1-gram TF-IDF	58.47	28	40.72	3
	2-gram TF-IDF	59.17	41	44.93	5
	3-gram TF-IDF	58.97	67	45.82	10
	4-gram TF-IDF	58.98	96	45.96	14
	5-gram TF-IDF	59.02	129	**46.09**	12
SVM	1-gram BoW	58.55	22	42.23	41
	2-gram BoW	**61.85**	41	45.20	8
	1-gram TF-IDF	56.97	21	40.71	2
	2-gram TF-IDF	60.19	32	**45.63**	7
CNN	model A	-	-	24.79	8
	model B	-	-	25.32	17
	model C	-	-	23.58	26
	model D	-	-	24.67	35
	CNN-word	**60.28**	345	**49.32**	115
LSTM	model A	61.15	23	48.24	22
	model B	60.77	50	47.65	23
	model C	62.70	23	49.15	26
	model D	63.24	46	49.16	35
	model E	59.85	63	45.46	46
	model F	**64.29**	60	**49.28**	56
	model G	63.21	214	48.04	70
	model H	61.49	205	49.12	113

Table 1: List of experiments performed in this paper along with the accuracy and training time.

Methods	Yelp ‘15		IMDB	
	Zhang et al., 2015 (baseline)	This paper	Zhang et al., 2015 (baseline)	This paper
BoW	58.0	59.02	-	42.64
BoW TF-IDF	59.9	58.47	-	40.72
N-grams	56.3	61.33	-	45.82
N-grams TF-IDF	54.8	59.17	-	46.09
	Tang et al., 2015 (baseline)	This paper	Tang et al., 2015 (baseline)	This paper
SVM + Unigrams	61.1	58.55	39.9	42.23
SVM + Bigrams	62.4	61.85	40.9	45.2
	Zhang et al., 2015 (baseline)	This paper	Zhang et al., 2015 (baseline)	This paper
LSTM	58.2	**64.29**	-	49.28
CNN-word	60.5	60.28	-	**49.32**

Table 2: Comparison of accuracies (in percentage) obtained in this paper against the baselines used in [1]

32G (the upbound of our local machine), and then the training process hangs due to page swapping. Therefore, to increase the physical memory would be a recommended pactice for text classfication tasks based on SVM and Logistic Regression Models.

In addition, the deep learning models such as CNN and LSTM(RNN) requires GPU support. So for this

Methods	Yelp '15	IMDB
HN-AVE	69.9	47.8
HN-MAX	70.1	48.2
HN-ATT	71.0	49.4

Table 3: Results taken directly from the Hierarchical Attention Networks (HAN) model in [1] for reference. HN stands for Hierarchical Network, AVE indicates averaging, MAX indicates max-pooling, and ATT indicates the proposed hierarchical attention model

project, we carry out most of these deep learning tasks on kaggle and google colab, which provide us with GPU resource (11G GPU memory) to run basic deep learning models.

Also, we found that the memory management of python is not quite fail-safe, so it is necessary to deal with memory release manually for scenario that requires large memory usage, like the case that we are working on large dataset.

5.2 CNN With and Without Word Embedding

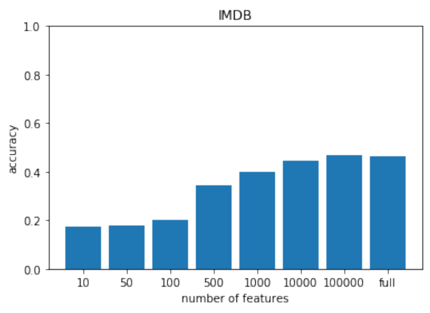
Comparing CNN models with and without word embedding, we can easily say that CNN with word embedding performs much better than CNN without word embedding. In CNN model without word embedding, the features and the meaning of a word was lost. Therefore, it is hard for CNN network to find distribution patterns in a sentence, which leads to the failing of CNN model without word embedding. While in CNN model with word embedding, there are relationships between words. Two similar words are close to each other, and two different words will be far away from each other on vector space. The relationships between words make CNN network easier to find distribution patterns in a sentence.

5.3 LSTM Hyper-parameters

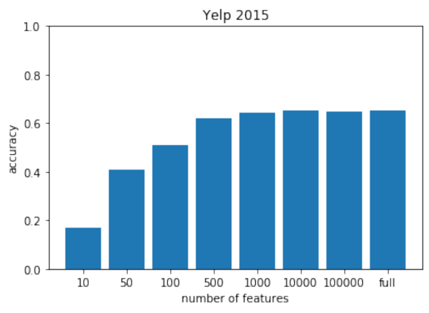
From our six LSTM models, we found several relationships between hyper-parameters and accuracy. Comparing model A, B and C, we found that larger dimension of embedding layer not always lead to better performance. The performance reaches it's peak when the dimension of embedding layer is set to 400. Comparing model C, F and H, we can get that larger dimension of hidden state in LSTM also not always lead to better performance. The performance reaches it's peak when the dimension of hidden state in LSTM is set to 512. Comparing model C, D, and E, we can get that when the dimension of hidden states is set to 256, the performance reaches it's peak when the number of hidden layers is set to 3. While comparing model F and G, we can get that when the dimension of hidden states is set to 512, the performance reaches it's peak when the number of hidden layers is set to 2. Generally, we can say that simpler LSTM models will lead to underfit while more complex models will lead to overfit.

5.4 LSTM number of features (dictionary size) with respect to Accuracy

For the LSTM basic model A, we have carried out extensive experiments to study the impact of dictionary size, in terms of how many words are selected to form the word embedding set. Our results are obtained by applying SGD (Stochastic gradient descent) for 20 Epochs. As summarized in Figure 8, our results show that we do not need to embed every unique word appeared in data set. Even with very limited size of dictionary, such as 1000 (the total vocabulary size are 146678 and 234945, for IMDB and Yelp datasets, respectively), the text classification accuracy could already reach 90% of the accuracy achieved by embedding every word in data set. Therefore, a very small set of words would have significant impact concerning the text classification tasks.



(a) IMDB Dataset



(b) YELP(2015) Dataset

Figure 8: Impact of number of features/vocabulary size on accuracy for the IMDB and YELP(2015) reviews data sets

5.5 LSTM vs. CNN

We compared some of the reviews from Yelp data set and IMDB data set, and found that Yelp reviews tend to be more straightforward than IMDB reviews. Yelp reviewers evaluate a restaurant based on the quality of food, the environment and the service. While IMDB reviewers evaluate a movie based on much more aspect. Combining the fact that LSTM models are better at capturing long term dependency while CNN models are better at capturing local dependency. We conclude that LSTM models work better on more straightforward reviews while CNN models work better on reviews with more evaluation criteria.

6 Conclusion

In this project, five types of models were trained to predict the rating of restaurants and movies from Yelp and IMDB data set. LSTM achieved the best accuracy 64.29% on Yelp data set while CNN-word achieved the best accuracy 49.32% on IMDB data set. Our best accuracy on both Yelp and IMDB data set are better than their corresponding baselines and competitive with the HAN models. We found that we can get impressive performances by tuning and modifying a simple baseline.

For future work, to catch up with the best performance of previous models on Yelp data set, we would like to add attention [1] into current LSTM model. To improve our performance on IMDB data set, we can modify the CNN network in CNN-word model by adding more layers and tuning the number of channels.

7 Statement of Contributions

All members contributed significantly to the project. Jiangning, Hasnain and Bide searched for related work and also implemented various baseline models. We all contributed to the development and training of the models. The report was also written together.

References

[1] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[2] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[3] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[4] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[5] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[6] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[9] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[10] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[13] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[14] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[16] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.

[17] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[18] Ask Me Anything. Dynamic memory networks for natural language processing. *Kumar et al. arXiv Pre-Print*, 2015.

[19] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

[20] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM, 2014.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

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

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.