Analyzing Baselines for Hierarchical Attention Networks

Shashank Murugesh
Department of Electrical and Computer Engineering
McGill Universiy
Montreal, QC
shashank.murugesh@mail.mcgill.ca

Hamza Rizwan
Department of Electrical and Computer Engineering
McGill University
Montreal, QC
hamza.rizwan@mail.mcgill.ca

Ashray Malleshachari
Department of Computer Science
McGill University
Montreal, QC
ashray.malleshachari@mail.mcgill.ca

Abstract

In this mini-project, we analyzed the baseline performance of several algorithms as reported in the paper "Hierarchical Attention Networks for Document Classification" by Yang et al. (2016) [1]. We implemented baseline models from the paper (Logistic Regression, Support Vector Machines, and LSTMs) without using the author's code and explored another simple algorithm (Multinomial Naive Bayes) to try to beat the performance on the task. We worked on a subset of the datasets used in the paper (Yelp'13, Yelp'14 and the Yelp'15), and performed several experiments to try to fine-tune and improve performance of the baseline models including hyperparameter tuning and feature reduction. Our best performing model was SVMs with bigrams and feature reduction which achieved an accuracy of 60.8 % on the Yelp'15 dataset. Also, due to computational constraints and in order to make our implementations of the various baseline models more tractable, the size of each dataset was reduced and 60,000 random samples were chosen from each of the datasets(Yelp'13,Yelp'14, Yelp'15)

1 Introduction

Document classification and Sentiment Estimation are at the heart of many information management and retrieval tasks. In the context of development, document classification plays an important role for several applications, particularly for organizing, classifying, searching and concisely representing large volumes of information. Reproducibility is a critical issue in machine learning. In this mini project, we reproduced and extended on the results from the paper by Yang et al[1]. Our goal was to improve/analyse the different baselines in the said paper. In order to achieve this, we leveraged the following baseline models as mentioned in the paper such as Logistic Regression and Support Vector Machine in order to replicate the corresponding baseline performances from the paper. This was followed by fine tuning the baselines by extensive hyperparameter tuning. We also explored simple machine learning algorithms like Multinomial Naive Bayes (not given in the original paper) to analyse it's performance on our datasets. Modern Deep Learning models like LSTM were also explored by running controlled experiments to analyse the same. We found that our improved baseline model that incorporated SVM with feature reduction and bigrams demonstrated the best test set accuracy at 60.8 % on the Yelp'15 dataset.

2 Related Work

With the upsurge in the availability of electronic documents and lightening fast growth spurt of the Internet, the task of document categorization has become a pivotal means of organizing all the information at hand. Proper classification of e-documents, online news, blogs, e-mails and digital libraries requires an effective combination of text mining(intelligent text analysis), machine learning and natural language processing (NLP) techniques. Sentiment classification is one of the most widely used natural language processing techniques in many areas, such as E-commerce websites,stock forecast,political orientation analyses[2]. Document-level sentiment classification is a fundamental task in sentiment analysis[3].

Brzezinski et al [4] way back in 1999 in their paper focused on a Machine Learning approach to document classification for information retrieval by applying a simple logistic regression [5]based algorithm for binary, multiple and hierarchical classification tasks. The results revealed that it was indeed an effective machine learning algorithm- it wasn't only a robust classification algorithm but also proved to be an effective dimensionality reduction method.

Naïve Bayes text classification has been used in industry and academia for a long time (introduced by Thomas Bayes in the 1700s). However, this technique is being studied since the 1950s for text and document categorization. Kibriya et al[6] in their paper present empirical results for several versions of the multinomial naive Bayes classifier on four text categorization problems, along with demonstrating a way of improving it using locally weighted learning- all while upholding the assertion that SVMs are still the method of choice if the aim is to maximize accuracy, which brings us to Support vector machines or SVM. The original version of SVM was introduced by Vapnik and Chervonenkis in 1963. The early 1990s saw the advent of it's nonlinear version which was introduced by Boser et al. in 1992[7]. Original version of SVM was designed for binary classification problem, but many researchers have worked on multi-class problem using this technique.

Recently, neural networks approaches have gained big success on sentiment classification. Efforts have also been made to deal with long texts modeling process, in the tryst of which- LSTM has emerged extremely popular to deal with sentiment classification problems. Long short-term memory network (LSTM) was proposed by Hochreiter and Schmidhuber in 1997 [8], and is a typical recurrent neural network which alleviates the problem of gradient diffusion and explosion. It can capture the long dependencies in a sequence by introducing a memory unit and a gate mechanism which aims to decide how to utilize and update the information kept in the memory cell.

The paper we're working with has applied a hierarchical attention network(HAN), which is a classification method from Yang et al[1] as mentioned before. The reason they developed it, in spite of the existence of other well working neural networks for text classification is because they wanted to pay attention to certain characteristics of document structures which have not been considered previously- prior work used context to discover when a sequence of tokens was relevant. HAN made this simpler by filtering for sequences of tokens, taken out of context. The above mentioned models were all the different baseline models as mentioned in the paper[1] that were analysed and implemented by us.

3 Datasets and Setup

3.1 Selecting Datasets

The original paper[1] performed experiments on six large scale document classification datasets in two categories: sentiment estimation (Yelp 13, Yelp 14, Yelp 15, IMDb, Amazon Reviews) and document classification (Yahoo Answers). We chose to perform our experiments on 3 datasets in sentiment estimation - Yelp 2013, Yelp 2014, and Yelp 2015 instead of document classification due to computational constraints as well as the complexity involved in running experiments on a dataset with 10 classes.

3.2 Size and Split

Each dataset has 5 classes (Yelp ratings from 1-5). We reduced the size of each dataset (Yelp 13,14,15) by randomly selecting 60,000 examples each to allow us to perform a broader range of experiments, test several models, and optimize hyperparameters easily due to computing resource limitations.

We split each dataset into training (80%), validation (10%), and test (10%) sets so we were able to perform hyper-parameter tuning on the validation set and test our results on a held-out test set.

3.3 Setup

All experiments were run on a Google Colab instance running a Intel Xeon CPU (2 Ghz) with a Tesla K-80 GPU and 12.6 GB RAM

4 Proposed Approach

4.1 HAN Implementation on selected datasets

We ran an implementation of HAN by H. Sankesara[1] on the datasets.

4.2 Implementing Baselines from the paper

Preprocessing

Before running the data through any models, we cleaned raw Yelp data by removing special characters, punctuation, numbers, stopwords (such as "the", "a", "an", "in"). We also removed words occurring very frequently in every document (top 10%) as those words don't encode unique information about the class they belonged to. Some columns that don't contribute to learning representations ("user ID", "business(product)", "rating" and "reviews") were dropped.

4.2.1 Logistic Regression

We then trained a Logistic Regression classifier on the datasets using scikit-learn (sklearn.linear_model.LogisticRegression) without any changes to the default parameters using different feature extraction methods as reported in the paper:

- **BOW:** Bag of Words model using sklearn.feature_extraction.text.CountVectorizer
- **BOW TFIDF:** TF-IDF (term frequency-inverse document frequency) encoding using sklearn.feature_extraction.text.TfidfVectorizer.

[1]https://medium.com/analytics-vidhya/hierarchical-attention-networks-d220318cf87e

4.2.2 Support Vector Machines (SVM)

We used sklearn.svm.LinearSVC to train an SVM classifier on the datasets without many changes to the default parameters. We varied the number of n-grams for testing baseline SVM performance without any hyperparameter optimization as reported in the paper:

- **SVM + Unigrams:** Uses 1-gram features (each word is a separate feature)
- **SVM + Bigrams:** Uses 2-gram features (every 2 adjacent words are separate features)

4.2.3 Long-Short-Term Memory Networks (LSTM)

LSTMs are a type of artificial recurrent networks used in Deep Learning that can solve the long term dependency problem. It is basically considered to avoid the problem of vanishing gradient in Recurrent Neural Networks(RNN).

LSTM takes the whole document as a single sequence and the average of the hidden states of all words is used as features to perform classification.

4.3 Exploring Simple Algorithms

Most simple algorithms were already reported as baselines in the paper. However, one simple model that wasn't tested was Multinomial Naive Bayes. We used sklearn.naive_bayes.MultinomialNB to train a classifier on the datasets without any changes to the default parameters (MNB)

4.4 Fine-tuning Baselines

We ran several experiments to try to improve the baseline performance of the baseline models (Logistic Regression, SVM) and Multinomial NB.

4.4.1 Regularization

We used sklearn.grid_search.GridSearchCV to look for the most optimal value for the regularization strength (C) parameter in Logistic Regression and SVMs.

4.4.2 Extraction Methods

We explored the difference between two feature extraction methods: BoW (Bag of Words) encoding using sklearn.feature_extraction.text.CountVectorizer TF-IDF (term frequency-inverse document frequency) encoding using sklearn.feature_extraction.text.TfidfVectorizer.

4.4.3 n-grams

We tested the difference in performance between unigram and bigram features in every model.

4.4.4 Feature Reduction

We measured the effect of reducing features on runtime and accuracy.

4.4.5 Laplace Smoothing

We used sklearn.grid_search.GridSearchCV to look for the most optimal value for the smoothing parameter for Multinomial NB:

4.4.6 LSTM

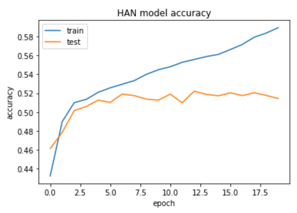
We tried to run some experiments to try and improve LSTM performance on the task, including:

- **Dense Layers:** Varying the number of dense layers: 100, 150, 200 layers.
- **Optimizer:** Testing the performance of Adam Optimizer vs Adadelta Optimizer
- **Dropout:** Varying dropout to regularize the model: 0.2 and 0.5

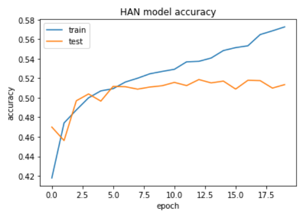
5 Results

5.1 HAN Implementation

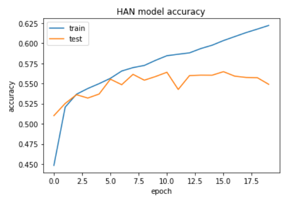
Fig 1: HAN accuracy per epoch



(a) Yelp 13



(b) Yelp 14



(c) Yelp 15

Figure 1: HAN performance on Yelp 13, Yelp 14, Yelp 15

5.2 Baselines from the paper

5.2.1 Logistic Regression

Table 1 shows the accuracy of Logistic Regression with different feature extraction methods. We observe that Logistic Regression with TDIDF has the highest accuracy of 59.4%.

Model	**Yelp 13**	**Yelp 14**	**Yelp 15**
BOW	0.461	0.483	0.531
BOW TFIDF	0.530	0.519	0.545

Table 1: Logistic Regression Baseline Performance

5.2.2 Support Vector Machines (SVM)

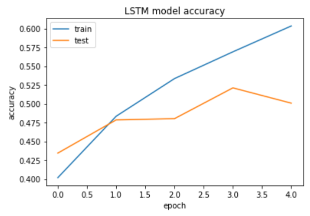
Table 2 shows the accuracy of SVM with different n-gram lengths. We observe that SVM with bigram has highest accuracy of 57.4%.

Model	**Yelp 13**	**Yelp 14**	**Yelp 15**
SVM + Unigrams	0.503	0.503	0.537
SVM + Bigrams	0.536	0.543	0.574

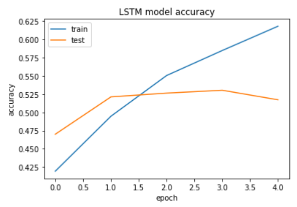
Table 2: SVM Baseline Performance

5.2.3 Long-Short-Term Memory Networks (LSTM)

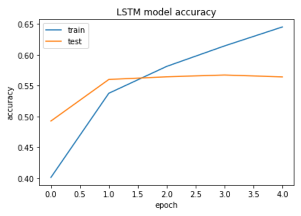
Fig 2: LSTM accuracy per epoch.



(a) Yelp 13



(b) Yelp 14



(c) Yelp 15

Figure 2: LSTM performance on Yelp 13, Yelp 14, Yelp 15

Model	Yelp 13	Yelp 14	Yelp 15
LSTM	0.516	0.501	0.563

Table 3: LSTM performance

5.3 Simple Algorithms (Multinomial Naive Bayes)

Table 4 illustrates the accuracy of Multinomial Naive Bayes on Yelp 13, Yelp 14 and Yelp 15 datasets. We observe that the model didn't perform as well as the Logistic Regression and SVM models.

Model	Yelp 13	Yelp 14	Yelp 15
MNB	0.443	0.407	0.497

Table 4: Multinomial Naive Bayes Performance without hyperparameter tuning

5.4 Fine-tuning Baselines

Two baseline models- Logistic Regression and SVM are fine tuned to their best parameters. Table 5 compares Logistic Regression and SVM model accuracy with hyperparameter tuning on Yelp 13, Yelp 14 and Yelp 15 datasets. We observe that Logistic Regression with bigram demonstrates highest accuracy of 60.3%.

5.4.1 Regularization

Model	Yelp 13	Yelp 14	Yelp 15
LR Before Tuning	0.530	0.519	0.545
LR After Tuning	0.535	0.528	0.564
SVM Before Tuning	0.503	0.503	0.537
SVM After Tuning	0.503	0.53	0.561

Table 5: Effect of regularization hyperparamter tuning on Logistic Regression and SVM Baseline Performance

5.4.2 Feature Extraction Methods

Model	Yelp 13	Yelp 14	Yelp 15
LR BOW	0.519	0.488	0.543
LR TFIDF	0.535	0.528	0.564
SVM BOW	0.511	0.453	0.533
SVM TDIDF	0.503	0.530	0.561
MNB BOW	0.498	0.500	0.524
MNB TDIDF	0.453	0.496	0.520

Table 6: Performance of different feature extraction methods on Logistic Regression, SVM, and Multinomial Naive Bayes

5.4.3 n-grams

Model	Yelp 13	Yelp 14	Yelp 15
LR-Unigrams	0.535	0.528	0.564
LR-Bigrams	0.571	0.554	0.603
SVM-Unigrams	0.503	0.530	0.561
SVM-Bigrams	0.536	0.553	0.574
MNB-Unigrams	0.453	0.496	0.520
MNB-Bigrams	0.541	0.551	0.577

Table 7: Effect of using different n-gram lengths on Logistic Regression, SVM, and Multinomial Naive Bayes

5.4.4 Feature Reduction

Model	Yelp 13	Yelp 14	Yelp 15
LR w/o Feature Reduction	0.535	0.528	0.564
LR with Feature Reduction	0.540	0.532	0.576
SVM w/o Feature Reduction	0.503	0.530	0.561
SVM with Feature Reduction	0.544	0.527	0.576
MNB w/o Feature Reduction	0.453	0.496	0.520
MNB with Feature Reduction	0.470	0.465	0.529

Table 8: Effect of reducing features on Logistic Regression, SVM, and Multinomial Naive Bayes

Model	Yelp 15
LR- without feature reduction	6.88 sec per loop
LR- with feature reduction	3.24 sec per loop

Table 9: Runtime of Logistic Regression model on Yelp 15 dataset

5.4.5 Laplace Smoothing

Model	Yelp 13	Yelp 14	Yelp 15
MNB without tuning	0.443	0.407	0.497
MNB with tuning	0.453	0.496	0.520

Table 10: Effect of Laplace smoothing on Multinomial Naive Bayes

5.4.6 LSTM

Table 11 shows accuracy of LSTM after fine-tuning the hyper parameter values. We observe that there is no significant change in accuracy when the parameter values are changed.

Parameters		Accuracy (Yelp 15)
Number of dense layers	100	0.561
	150	0.564
	200	0.569
Optimizer	Adam	0.564
	Adadelta	0.557
Dropout	0.2	0.564
	0.5	0.562

Table 11: Results of attempted LSTM hyperparameter tuning

6 Our Improved Baseline

Table 12 compares accuracy of Logistic Regression and SVM with feature reduction - both with and without hyperparameter tuning. We observe that SVM with feature reduction and bigram performed the best when compared to other models with the accuracy of 60.8%

	Model	Yelp 13	Yelp 14	Yelp 15
Without tuning	LR-feature reduction+unigram	0.535	0.529	0.556
	LR-feature reduction+bigram	0.566	0.556	0.596
	SVM-feature reduction + unigram	0.529	0.527	0.557
	SVM-feature reduction + bigram	0.531	0.545	0.584
With tuning	LR-feature reduction+unigram	0.540	0.532	0.576
	LR-feature reduction+bigram	0.579	0.562	0.604
	SVM-feature reduction + unigram	0.543	0.527	0.576
	SVM-feature reduction + bigram	0.578	0.552	0.608

Table 12: Accuracy of Logistic Regression and SVM with feature reduction

7 Discussion and Conclusion

As can be seen in our results, simple machine learning models worked best. Logistic regression outperformed LSTM which is a deep neural network approach. This can be hypothesised due to the smaller size of the dataset and the length of the text in question. Unlike a task such as sentence classification where the number of words is low and the order of words greatly influences the meaning of the sentence, the document classification problem consists of a very large number of words that are used to use to classify each document. This causes the order of the words to contain less predictive power over class, reducing the effectiveness of techniques such as Convolutional Networks, LSTMs, and Hierarchical Attention Networks (HAN), all of which perform really well on massive datasets, as was seen in the original paper by Yang et al[1].

LSTM model didn't outperform both the Logistic Regression and SVM. Also, computational time of LSTM was significantly high. Fine tuning hyper parameters helped in improving the model performance. Accuracy of the Multinomial Naive Bayes model was slightly lower than Logistic Regression and SVM model. Using feature reduction technique we were able to achieve the best accuracy which also significantly reduced the computation time of the model.

Further scope of this project would be to try various other models including Ensemble methods, and other Deep Neural Network models. Also, we can explore various techniques such as word2vec, variable ranking and Best-Subset selection for feature extraction.

8 Statement of Contributions

All the team members contributed to the project equally.

References

[1] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. 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*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1480–1489. [Online]. Available: https://www.aclweb.org/anthology/N16-1174

[2] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity in phrase-level sentiment analysis,” in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT ’05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 347–354. [Online]. Available: https://doi.org/10.3115/1220575.1220619

[3] 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, Jul. 2002, pp. 79–86. [Online]. Available: https://www.aclweb.org/anthology/W02-1011

[4] J. R. Brzezinski and G. J. Knafl, “Logistic regression modeling for context-based classification,” in *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99*, Sep. 1999, pp. 755–759.

[5] J. Cramer, “The origins of logistic regression,” *Tinbergen Institute, Tinbergen Institute Discussion Papers*, 01 2002.

[6] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, “Multinomial naive bayes for text categorization revisited,” in *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence*, ser. AI’04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 488–499. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30549-1_43

[7] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT ’92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735