Text Classification of Yelp's Category Based on Text Reviews

Zizhun Guo
Computer Science
Rochester Institute of Technology
Henrietta NY, USA
zg2808@cs.rit.edu

1 Introduction

1.1 Background

Machine Learning and Deep Learning has become two of the most popular Artificial Intelligence topics across the globe in recent times, especially for being proved perform well targeting Natural Language Processing (NLP) Tasks. One foundational application of task is **text mining**. For the purpose of acquiring the solid ML and DL techniques, we choose to conduct the text classification experiment using Yelp Dataset. Not only we can apply our text mining theories to the actual business scenario, but to understand the model's principles in-depth to enhance our skills. In our experiments, we have both trained many classic textual models using W2V and GloVe and feed the document representation into the ML and DL models like RL, SVM, CNN, and RNN. We have also tried implementing handcraft features and using validation accuracy to compare the experimental results.

1.2 Challenges

Unlike data mining, text mining is resolving inherently unstructured fuzzy data. It is involving more complex and variant strategies to handle it. Besides, text information covers multiple topics as information retrieving, information extraction, text analysis, and so forth. As it gets diverse frameworks to consider, it gets complex to choose an appropriate way to form the solution. In terms of the natural language, human language is grammarly diverse with vague semantics and lexicons, it adds the extra difficulty for researchers and engineers to tack. The language also relates to cultures, where even the translation is fetched correctly in wordings level but done wrong in domain knowledge. Tools for collecting, analyzing, researching, extracting for text mining are all in the developing stages.

1.3 Existing Approaches

Text mining focus on extracting information from the text. Several techniques are used to solving text mining problems.

Information Extraction is one famous texting mining technique that mainly seeks to extract the information between entities, attributes, and their relationship. The information is usually stored in a semi-structured or unstructured way.

Information Retrieval is a technique mainly solving to recognizing a specific text pattern composed of words or phrases.

Categorization is another essential technique that follows the supervised way to learn a set of categories from the mined text. This is also what this paper is studied.

Clustering is another ML related term that is also one of the text mining techniques. It seeks to find intrinsic structures in textual information.

Summarization works similarly in its literal means that it automatically generates the summary for a phrase of text in which the semantic meaning is similar to the original documents.

1.4 Motivation of this study

This paper mainly focuses on Categorization in text mining topic, also as known text classification, which is one of the popular NLP tasks that is employed by most commercial entities. As the NLP tasks are applied in multiple individuals' daily scenarios (i.e. user portrait classification, recommendation system, intelligent voice virtual assistant, etc.) This study is first set up in an Independent Study course, hence the goals not only seeking the solutions to the **text classification** task but also to practice the current text mining techniques combing with ML and NN techniques to resolve the problem. The study process includes collecting dataset, preprocessing dataset, researching, and practicing current ML and NN models, tunning the models, retrieving the results, and producing summaries. Meanwhile, the motivation also covers toolkit manipulation, coding, and engineering implementation. Hence, it is both endued the expectation in researching and engineering.

2 Related Works

Some previous works have proved that the neural network can be successfully applied in NLP tasks. These include, but are not limited to, sentence classification [2], sequence representation learning [3], phrase representation learning, and statical machine translation [1], and text classification [4]. Specifically, topic model LDA [5] and word embeddings [6] have been used widely to model the textual features from the source documents.

[2] modifies the original Convolutional Neural Network to enable applying on the text classification task. It borrows the CNN model

architecture used for image detecting and modifies the 3-dimensional convolutional layer into a 2-dimensional version. It also involves the concept of static and non-static word embeddings to configure whether the embeddings participate in the training process or not.

[1] introduced a new memory unit that extends the Recurrent Neural Network approach to solve the sequence text mining issues. Its encoder-decoder model help improves the performance of the statistic machine translation and meanwhile proved also helpful in the text classification task.

[4] research directly targeting text classification tasks. It compares with classic baseline NN models that are used for text mine task and introduced a novel model that take both advantages from CNN and RNN. The results proved it works as it has a better performance on text classification than other models.

[8] introduces three statistical feature extraction approaches to extract the quality aspect of QoS features from users' reviews. The approaches also classify reviews into either positive or negative sentiment. The workflow is first to model the data using either supervised LDA or LDA to extract verbal features as the input for l1-RLR.

[9] introduced an augmented web service clustering process by incorporating the semantic features extracted by the Doc2vec model and the features mined from the service relationship network model SiNE to perform the revised K-Means clustering algorithm.

[10] proposed an improved theory of the Web services clustering. It proposed a revised LDA approach adopting the latent topic information with help of W2V tool.

There is some other research in which the main target or the part of goals are aligned with the text mining task. They may implement different textual models or using different types of source datasets, but overall, their works help the practitioner know more information about the study's applicational scenarios.

3 Data

The dataset that we used was downloaded from Yelp. The original dataset has been firstly used for Yelp Dataset Challenge and is kept updating throughout the years. It contains information about reviews, business, users, and business check-ins. The main two datasets we use is business and reviews. We specially focus on **6000 businesses** evenly divided in **3 categories** with their whole reviews concatenating together that the documents being features. The vocabulary size is around 180,000 which makes our documents source complex and rich in information enough to represent each business. The type of business we select is the restaurant specifically are marked in Sushi Bars, American New, and Fast Food. We believe it is firstly rich in information of the reviews due the Yelp Dataset are mainly focus on Restaurant service collection, and these three categories are not exactly similar whether in the business model or food service it provides.

For each document, we only use 6000 characters (white space inclusive) for training due to the memory limitation while conduct deep learning training hence maintains the same dataset to use for the fair models' comparison. We conduct 10-fold cross-validation, and we assume each review for each restaurant extracted from JSON dataset file is randomly ordered, but we have ranked the restaurant based on the number of reviews they have so that when conduct concatenation, it guarantees the most reviewed restaurants to be selected as the source restaurants.

3.1 Textual Features

The documents we extract from the reviews are preprocessed through the removal of the stopping words. We used Natural Language Toolkit (NLTK) to process it since it provides a fully readily function to solve it in a relevantly quick time. After the removed the stopping words, we either process Stemming, or Lemmatization based on what textual model to use in the later step.

index	Stemmed Text	Lemmatized Text
0	husband live nearbi stop	husband live nearby stop
1	great attent servic great	great attentive service great
2	locat great fresh brew	location great fresh brew
3	nice restaur park vega	nice restaurant park vegas
4	star sushi joint serv size	make star sushi joint serve

Table 1: Stemmed or Lemmatized Texts

If we use pre-trained word embeddings, we only conduct Lemmatization since a complete form of a word is more often to be spotted. Whereas we stem the words only if we used our self-trained word embedding models for better efficiency.

review1: [em$_1$1, em12, em$_1$3, ..., em$_1$300]
review2: [em21, em22, em23, ..., em2300]
review3: [em31, em32, em33, ..., em3300]
review4: [em41, em42, em43, ..., em4300]
...
review6000 [em$_{6000}$1, em$_{6000}$2, ..., em$_{6000}$300]

Figure 1: Sequence Embeddings

The documents would be tokenized into the Sequence Embeddings (shown in Figure 1) as the extracted textual features and sent to the following classifiers or neural layers.

3.2 Target Labels

As the original labels are categorical labels: **Sushi Bars**, **American New**, and **Fast Food**. It has to transform the categorical labels into numerical labels for the models to learn. We have transformed two types of labels (**multiclass** vs **multilabel**) for our learning models to output.

index	Multiclass	Multilabel		
0	-1	1	0	0
1	1	0	0	1
2	1	0	0	1
3	0	0	1	0
4	1	0	0	1

Table 2: Two types of the labels

Multiclass labels are used for regular ML models as it contains only one value. Multilabel are used for NN models since the last layer is the Softmax layer with 3 neurons. However, both two types of labels can be used for both models if necessary, but with a different model, setting to fulfill the requirement.

4 Textual Model

4.1 Bag-of-Words Model: TF-IDF and LDA

Term Frequency–Inverse Document Frequency, as it is mostly known for TF-IDF, is a bag-of-words method to statistically represent the corpus. It computes the occurring frequency of words in each document to rank words and use inverse document frequency to compensate for their weights. Through this way, a final TF-IDF score would be calculated to reflect the importance of each word in each document.

In our experiments, we use the Scikit-learn libraries to employ TF-IDF on our Yelp dataset.

Unlike TF-IDF, Latent Dirichlet allocation (LDA) is a topic model designed to represent each document as a vector. This topic model is a generative model that each word in each document is chosen from random topics through probability distributions. A document could be consisted of the distribution of many topics, whereas a topic is formed in the distribution of many words. In the perspective of computer programs, the implementation format is in matrices multiplication. A vector can represent a distribution; a matrix can also represent a distribution because each row or column can be a distribution.

In our experiment, we employ LDA from the Scikit-learn library to convert our documents for each user in a vector in the topic's number of dimensions.

4.2 Word Embeddings

Word2Vector (W2V) is a word embedding technique. It uses a shallow neural network with one hidden layer to learn a large corpus of text. After tokenized the documents, it transformed the one-hot encoding words into the embedded words vectors through the hidden layer weight matrix. The output vector for this neural network ends up being the probability distribution for all word's vocabulary.

This technique considers the words pair through the window sampling, whereas the neighboring words would be taken as input and output for the neural network hence tuning the weights matrix through model training. There are two types of neural network structures: continuous bag-of-words (CBOW) and Skip-gram (SG). Both models assume the neighboring words are closely relevant. CBOW is motivated that the words are decided by its neighboring words. The Skip-gram is different in that each word determines the neighboring ones. In our experiments, we use an open-source unsupervised topic modeling library Gensim to model our documents.

Similar to W2V, Global Vectors (GloVe) considers the context information of a document like the order of words as to model semantic similarities between review documents. The global vector would be trained through aggregate the word-word co-occurrence to represent each document. In our experiments, we use the Stanford GloVe training program to generate the word embeddings of the entire documents' vocabulary. We also use pre-trained GloVe word embeddings (glove.42B.300d) to model our documents.

The word embeddings can be gained from an embedding layer of a deep learning model of which is a part of weights trained throughout the deep learning model training process. Flexibly, the embedding layer can be initialized with variable embeddings like SG, CBOW, GloVe, and so forth. We consider it as a non-static model if the embeddings participate in the entire training process or static if it is only initialized once and is unchanged.

In our experiments, we use the TensorFlow Keras library to implement the embedding layer. As it is designed a high-level Python class where the input of the documents data is default the words indices rather than the one-hot encoding matrices, therefore the documents need an additional preprocessing step to convert the documents into sequences of the vocabulary of indices. Through this way, the document representation that is used to be sparse and large-sized turns into a new dense form that speeds up the training process and becomes more space-efficient.

5 Machine Learning Model

Once we retrieve the document representation of the entire dataset, we are successfully extracted their textual features. To fulfill the complete data mining classification task, continuously, we have fed the preprocessed documents (i.e. word embeddings) into the learning model. There are multiple choices of classifiers to choose from. We use the Scikit-learn Machine Learning library to implement Logistic Regression (LR), Support vector machine (SVM), Random Forest (RF), K-Nearest Neighbors (k-NN), and Multinominal Naïve Bayes (MNB).

5.1 Logistic Regression

Logistic Regression (LR) is a traditional ML binary model that uses the logistic function to fit the text features. It is a binary classifier widely used in most data mining works. In our experiments, we use the Scikit-learn ML library to implement LR for the text classification task.

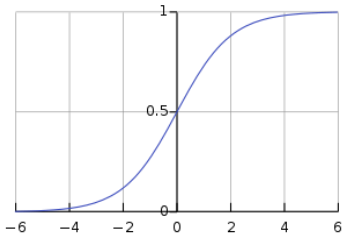

Figure 2: Sigmoid function Image

The classifier would take the vectors as the sequence embeddings into the Sigmoid function. If the output value is greater than 0.5, it the record is classified into one category. Same if the output is less than 0.5, it is classified as another category.

Since LR is a binary classifier originally, Scikit-learn adopts a strategy called One-vs-The-Rest to enable it multiclass workable. The One-vs-The-Rest fits the data by turns on each class against all other classes. This is one of most common strategy to solve multiclass issue.

5.2 Support Vector Machine

Another main classifier we employ is SVM, which is a binary supervised machine learning model that seeks the best hyper-plane in the high dimensioned space. The sample is treated as a large dimensional vector that is classifier on either side of the hyper-plane. We still use the Scikit-learn ML library to implement it. The two hyper-parameters we have tuned is **C** (regularization parameter suggests the strength of the regularization) and **gamma** (the kernel coefficient).

The Scikit-learn takes the strategy called One-vs-One to wrap the SVM classifiers for it is also a binary classifier. This strategy would split all classes into pairs; therefore, the number of the pairs is n_classes * (n_classes - 1) / 2. Then, each pair of classes will be assigned a classifier. The classifiers would be trained and vote for the final prediction.

6 Convolutional Neural Network

As one of famous deep learning models, CNN has been widely used in computer vision tasks. Since the year of 2014, the CNN has been discovered to be proved successfully workable not only the computer vision jobs but also the text classification.

6.1 Model Architecture

In our experiment, we implement a CNN model with its architecture designed based on the YoonKim's architecture [2].

The Embedding Layer works for training the word embeddings as it was been discussed previously. The Convolutional Layer has hundreds of feature maps as filters to convolute the document representation. The Global Max Layer is charged for keeping the most unique value to represent an independent document among all. The Dropout layer works as a regularization role. At last, the Softmax layer computes the probability distribution for three categories. We then classify the sample into the category which has the greatest value. In our experiments, we use the TensorFlow Keras deep learning library to implement this CovNet.

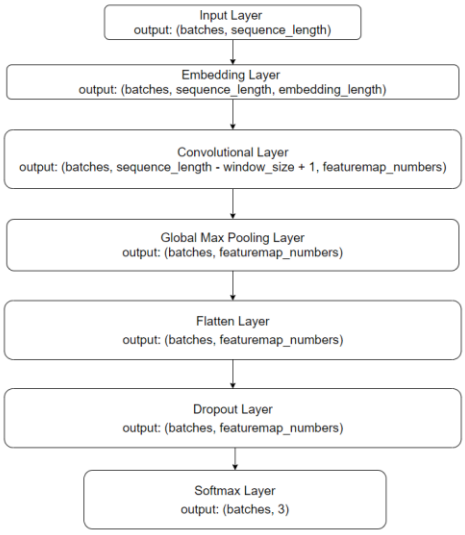

Figure 3: CNN baseline model architecture

6.2 Embedding Layer

The embedding layer is a special layer that is normally placed at the beginning of a neural network. Similar to W2V, it is an individual layer trained to work as a lookup table for the one-hot encoding input documents. The weights matrix represents the word embeddings. The documents which are made of the list of sequence would be converted into a stack of word vectors. We employ the average pooling layer to produce the mean word embeddings to represent a single document for normal W2V cases, but in our CNN model or RNN models, we would use more strategies to use the embedding layer's output. Besides, the weights matrix of the embedding layer would be trained along with the other parameters of the neural network using the back-propagation method.

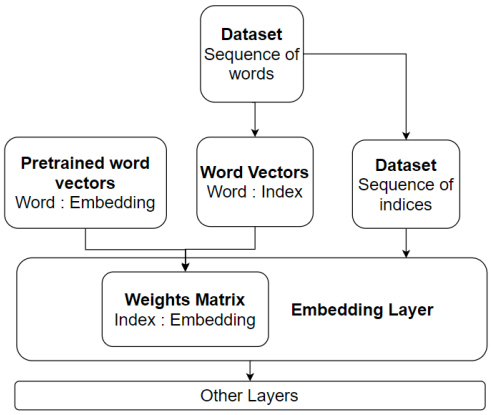

Figure 4: Textual Model Structure using TensorFlow Keras Embedding Layer

We use the TensorFlow Keras Deep Learning Library to implement the embedding layer to train the word embeddings. The figure 3 shows the structure of the embedding layer and how it is implemented in the Keras library. For more details we thought which is essential for practitioners to acquire before the implementation, the shape of the representation of the documents would be changed from 2D to 3D due to the addition of the word embeddings. Another essential concept is that the input dataset does not need to fit the one-hot encoding requirement, since the embedding layer class in the package allows to receive the form of matrices in a denser way which saves the spaces.

7 Recurrent Neural Network

7.1 Encoder Model

Recurrent Neural Network (RNN) has long been used for learning the sequence of documents employing a directed node graph along temporal steps. We adopt the Encoder part of the RNN-Encoder-Decoder model [1] as our RNN architecture since the architecture fits the requirement of the text classification task that documents can be input along with the temporal steps and in the end, the memory unit will produce a document representation. As seen from the figure 5, the encoder takes in a variable-length sequence $(X_1, X_2, \ldots, X_T)$ and converts it into a fixed-length vector representation C.

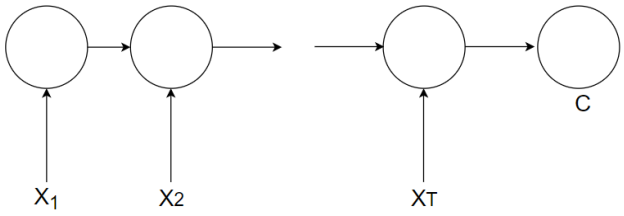

Figure 5: RNN Encoder Model

As for memory unit, the Encoder model can use either Gated Recurrent Unit (GRU) or Long-short term Memory (LSTM) to memorize the historical dependencies as in the form of the hidden state.

7.2 GRU and LSTM

LSTM has a longer history than GRU. It was firstly introduced in 1997 and was modified throughout the time. New features like forget gate and peephole would add during the evolution. The GRU is a newer generation of RNN work similar to LSTM that enables memorizing the short-term and long-term dependencies by implementing a reset gate and an update gate during training. There is no memory cell as it is called in LSTM but has a hidden state works the same function. The figure 6 shows the inner structure of two memory units.

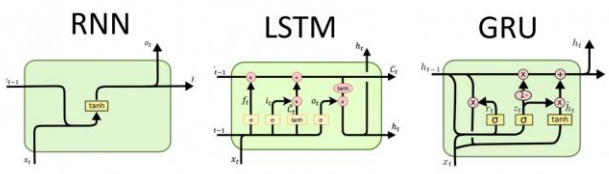

Figure 6: Unit Comparison between naïve RNN, LSTM and GRU

In our experiments, we use the TensorFlow Keras Deep Learning library to implement GRU and LSTM in our RNN Encoder model. It is similar to the convolutional layer in CovNet that receives the output from the last layer as the input and output of the results to the nest layer.

7.3 Model Architecture

In our experiment, we implement a RNN decoder model with its architecture designed based on the Cho's architecture [].

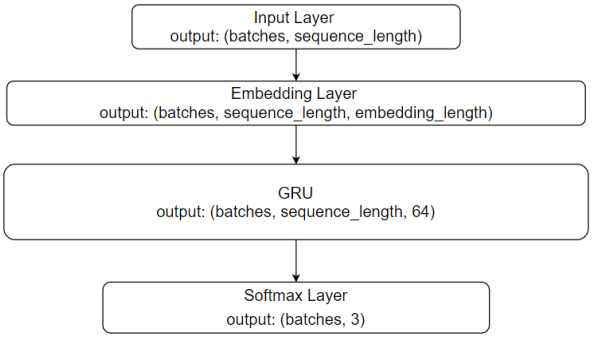

Figure 7: RNN-GRU model architecture

It works the same way as it was described in the CNN model's section. The document-representation sends the batches of samples in the Input Layer at the beginning. After the embedding process by the Embedding Layer, the embeddings are continuously transferred to GRU layer. Through a recurrent time-based learning process using the Backpropagation method, the weights of the model would be updated. The final layer is the Softmax Layer which works to classify the samples into three categories.

8 Experiment

8.1 Methods Comparison

We compare different baseline model used in Text Classification methods.

SG/CBOW + LR/SVM/MNB/RF/K-NN These methods mainly use machine learning algorithms with averaged word embeddings as features. Respectively, we use LR, SVM, Multinominal NB, RF and K-NN.

CNN We also select a convolutional nerural network [] for comparison. It uses either the pre-trained GloVe embeddings or

self-trained SG/CBOW/GloVe embeddings as the initializer. We also initialize the model with the weights in normal distribution of which the weights participate throughout the training progress.

RNN + GRU/BiLSTM We also implement RNN encoder model as our feature extractor. The select either GRU or LSTM as our memory unit. When use LSTM, we choose to take the bidirectional strategy.

8.2 Model Parameters Tuning

Once we implement our word embeddings, it is necessary to fine-tune the model parameters which makes the model as much accurate as possible to represent the documents so that to improve the textual models' performance. We use Gensim Library, an NLP python library to implement the textual model. Here is one example illustrating the Gensim tuning results on the self-trained SG and CBOW model. Given a range of word embedding dimensions from 10 to 500, and a range of iteration time from 1 to 50. The tuning process meanwhile conducts a grid search tuning method on Logistic Regression classifier to collect its best setting for LR.

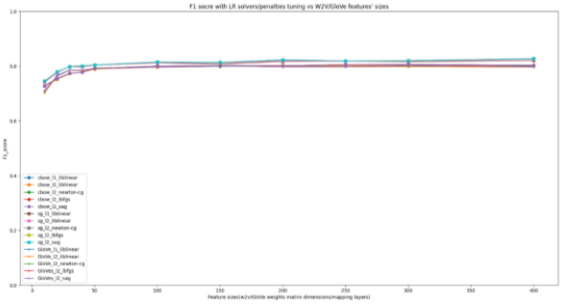

Figure 8: F1 scores of SG/CBOW/GloVe vs feature sizes

The size of the word vectors shows a general similar upwards trend comparing with SG and CBOW. However, the f1 scores soar at the very beginning when the word vectors size is below 100, the increasing rate is higher than SG and CBOW, but since the starting f1 score is lower than both, the GloVe model ends up being in the middle performance before CBOW eventually exceeds it. The overall performance shows that SG is still outperformed CBOW and GloVe, whereas the differences between CBOW and GloVe are not significant.

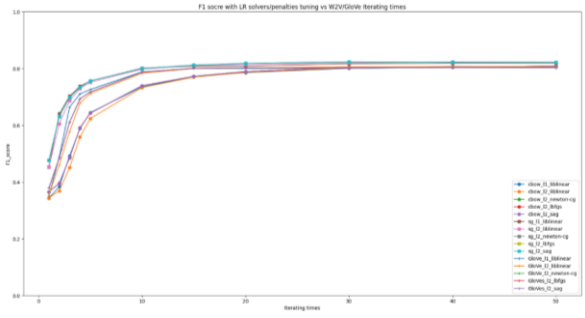

Figure 9: F1 scores of SG/CBOW/GloVe vs iteration times

Same as previous experiments illustrated, SG has better performance than CBOW and GloVe in all cases. GloVe also plays the middle role between SG and CBOW, but in this case (comparing iteration times), GloVe has almost the same score increasing rate. Interesting thing is, when the iteration is low, the GloVe performs close to SG, whereas when iteration time gets larger (greater or equal to 20), the GloVe is beaten by SG and close to the CBOW. In the end, CBOW and GloVe have emerged into similar performances.

Our tunning results suggest that a word embedding dimension between **300 to 500** and the iteration time greater than **30** would yield a higher F1 score when employing LR classifiers. Therefore, in the following experiments, we trained our textual models especially SG and CBOW setting the word embedding dimensions as 300 and the iteration time as 30.

8.3 Hyperparameter Tuning

In order to find the CNN model that produces the highest prediction accuracy, we have tuned the model regarding three hyperparameters that are hugely affected the model's performance: Sequence Size, Filter Region Size and Feature Maps numbers.

For other hyperparameter's setting:

Description	Values
Input word vectors	GloVe 300d
Static or non-static	static
Filter region size (window)	3
Stride	1
Feature maps	1000
Activation function	ReLU
Pooling	GlobalMaxPooling
Dropout rate	0.5
Optimizer	Adam
Batch size	128
Epochs	20
Cross Validation	10-folds

Table 3: CNN baseline model configuration

We attach a figure 10 illustrating the accuracy curves against different feature maps. The range of feature maps number is set from 1 to 1000. As it can be observed from the figure that with the increasing number of feature maps, the accuracy goes higher. From the result shown, when it gets 1000, the accuracy reached the greatest value. Therefore, the feature maps number is set at 1000. We also observe when sequence size equals 1000 and filter region size equals to 3, the CovNet performed the best.

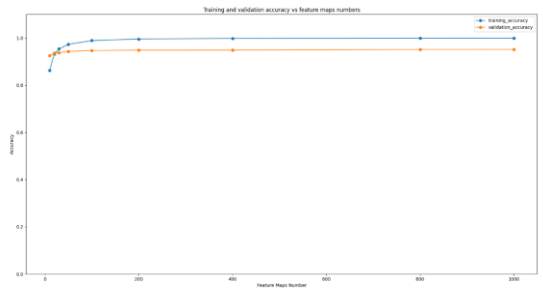

Figure 10: Accuracy vs feature map numbers

8.4 Results

The experiment results are shown in Table 4.

Embeddings	Model	Validation Accuracy
	LR	0.946410
	SVM	0.947334
	MNB	0.888441
	RF	0.942951
	K-NN	0.937315
CBOW	LR	0.944231
	SVM	0.947032
	MNB	0.894449
	RF	0.945386
	K-NN	0.937797
GloVe	LR	0.949056
	SVM	0.941156
	MNB	0.896581
	RF	0.939904
	K-NN	0.940003
Stanford GloVe42b300d	**CNN**	**0.951833**
	RNN-GRU	0.941400
	RNN-BiLSTM	0.917170

Table 4: Accuracy comparison vs different models

8.5 Summary

From the first three main Embedding models SG, CBOW, and GloVe, we use 5 different ML models to fit our dataset. For the LR, SVM and KNN classifiers, using GloVe model produces the greater accuracies over the usage of SG and CBOW, which has the opposite effects while using RF and MNB. Besides, using MNB classifier results in the lowest accuracies, among others. The results are shown that the final accuracies value in a small range. We believe it is due to the embedding models extracted the textual features well enough therefore it shows no significant difference in validation accuracies.

We use Stanford GloVe 42b 300d pre-trained word embeddings to initialize our Deep Learning Models. As the results show that the CNN has the highest validation accuracy which outperforms the RNN models. The RNN-GRU has less difference to CNN than what it has for RNN-BiLSTM. The RNN-BiLSTM produces the lowest accuracy.

Comparing the neural networks with the traditional machine learning models, we found our neural networks are not strong enough to outperform the ML models like it was usually proved in recent state-of-art theories. In our results, CNN works the best but with only a tiny improvement against ML models. RNN models completely failed to exceed the ML methods. We believe one of the big reasons is the RNN model does have the highest number of weights to train which makes it hard to converge the model in a small amount of dataset. At the same time, the traditional ML models may suffer from the overfitting issue.

Pretrained dataset might be another issue that causes the DL models to perform poorly than the traditional ML models. We employ the GloVe 42b 300d word embeddings from Stanford NLP lab, it works great on our review document. But the embeddings did not participate in the training process which limits the entire neural network performance due to the fewer weights to update. Besides, the missing rate for the words fail showing in the pre-trained vocabulary dictionary is high. Those missing word embeddings were replaced by the <unknown token> embeddings which harms the overall embedding accuracy. The way to resolve it is to improve the documents preprocessing operations for better textual feature extraction.

9 Additional Works

We have also tried to implement handcraft features extracted from the regular attributes from the original Dataset. We believe that the business running hours can also be helpful therefore we engineered the running hour (opening and closing times from Monday to Sunday) feature into 14 separate individual numeric features. Besides, we engineered the 14 features into another 8 features that include the flag features (i.e. is24hoursRunning, isMondayOpening, etc) and statistical features (i.e. weeklyBusinessHours, weeklyAverageHours). We found these handcraft features work fine for the text classification task with a worse performance comparing with the models using textual features.

We have also tried to combine the extracted regular features with the textual features together to fit our model, the results show that the accuracy for such feature combination is higher than fit any of them separately. Unfortunately, such experiments were conducted based on the old dataset therefore it cannot be used to compare with other methods.

10 Conclusion and Future Works

On the current scale of the dataset, the traditional ML models proved to work well compared to some recent neural network baseline models such as CNN or RNN. Though they may have such competitive performance, the results may exist other factors that affect the comparison results such as the overfitting issue, etc. Meanwhile, the neural network model that our experiment

implemented is all baseline models that may not be designed suitable enough to handle the task. The RNN model even has the least of the parameters tuned during the experiments. Besides, the textual model fed to the neural network model is also a pre-trained version and does not participate in the training process (static) due to the lower training speed. These are also the elements that triggered the results.

In the future, we plan to set the textual model as non-static mode so that the word embeddings will be updated throughout the entire training process. We also plan to do more research on recent state-of-art neural network models and use them in our text classification task. Moreover, we would try to update our developing skills by taking advantage of the cloud computing platform to migrate the experiment code online and conduct the experiments with a more powerful machine which will facilitate enough on our working progress. Additional knowledge would be also acquired throughout the process such as libraries using, preprocessing improvement, new model theories. As it is continuously devout time on it, we believe the results would be improved.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to professor Xumin Liu, who has advised me throughout the entire independent study course in this special Covid-19 period of times. Her professional advice and suggestion have guided me conquer many obstacles of the course works.

REFERENCE

[1] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. arXiv preprint arXiv:1406.1078

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

[3] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. *Sequence to sequence learning with neural network*s. In Advances in neural information processing systems (pp. 3104-3112).

[4] Lai, S., Xu, L., Liu, K. and Zhao, J., 2015, February. *Recurrent convolutional neural networks for text classification*. In Twenty-ninth AAAI conference on artificial intelligence.

[5] Hingmire, S. and Chakraborti, S., 2014, July. *Topic labeled text classification: a weakly supervised approach*. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (pp. 385-394).

[6] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781.

[7] dprogrammer Lopez, 2019, April. *RNN, LSTM & GRU*. Available at: http://dprogrammer.org/rnn-lstm-gru (Accessed: 30 November 2020)

[8] Liu, X., Shi, W., Kale, A., Ding, C. and Yu, Q., 2017. *Statistical Learning of Domain-Specific Quality-of-Service Features from User Reviews*. ACM Transactions on Internet Technology (TOIT), 17(2), pp.1-24.

[9] Cao, Y., Liu, J., Shi, M., Cao, B., Zhang, X. and Wang, Y., 2019, July. *Relationship Network Augmented Web Services Clustering*. In 2019 IEEE International Conference on Web Services (ICWS) (pp. 247-254). IEEE.

[10] Shi, M., Liu, J., Zhou, D., Tang, M. and Cao, B., 2017, June. *WE-LDA: a word embedding augmented LDA model for web services clustering*. In 2017 ieee international conference on web services (icws) (pp. 9-16). IEEE.