HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Report

Course: Natural Language Processing
Topic: Sentiment Analysis

Instructors: Prof.Le Thanh Huong

Prof.Nguyen Kiem Hieu

Group 9: Nguyen Ngoc Khanh 20204915

Bui Hong Nhat 20204890

Nguyen Van Hung 20204913

Nguyen Xuan Nam 20200422

Hà Nội - 2023

Contents

Source code: https://github.com/nhatbui97/nlp20222.git

1. Introduction

Sentiment analysis is the automated method of examining text to determine the sentiment expressed. By tackling sentiment analysis problems, enterprises can build a system to recognize the customer satisfaction level during text messages or calls with their consultants. This information can be further used to help companies adjust their product, marketing, business and especially evaluate the quality of consultants or agents.

In this project, we decided to use amazon product reviews dataset to train our systems. The system will take the customer review as an input and output rating stars(1, 2, 3, 4, 5) with the following meaning:

+ 1 star: very unsatisfied
+ 2 stars: unsatisfied
+ 3 stars: neural
+ 4 stars: satisfied
+ 5 stars: very satisfied

Therefore, we model our problem as a multi-class classification problem which takes a data point and classifies it into 5 classes: 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

2. Related works

To address the amazon sentiment analysis problem, we make use of two approaches: machine learning approaches and deep learning approaches.

2.1. Machine learning approach

2.1.1 N-grams

N-grams is a very popular technique used in natural language processing (NLP) to generate features for training Machine Learning models for Sentiment Analysis problems. N-grams generate features by dividing sentences or text into sequences of consecutive words of length N. These sequences are considered a feature of the text and are used to train models. Machine Learning like K-Nearest Neighbors, Naive Bayes, and Random Forest.

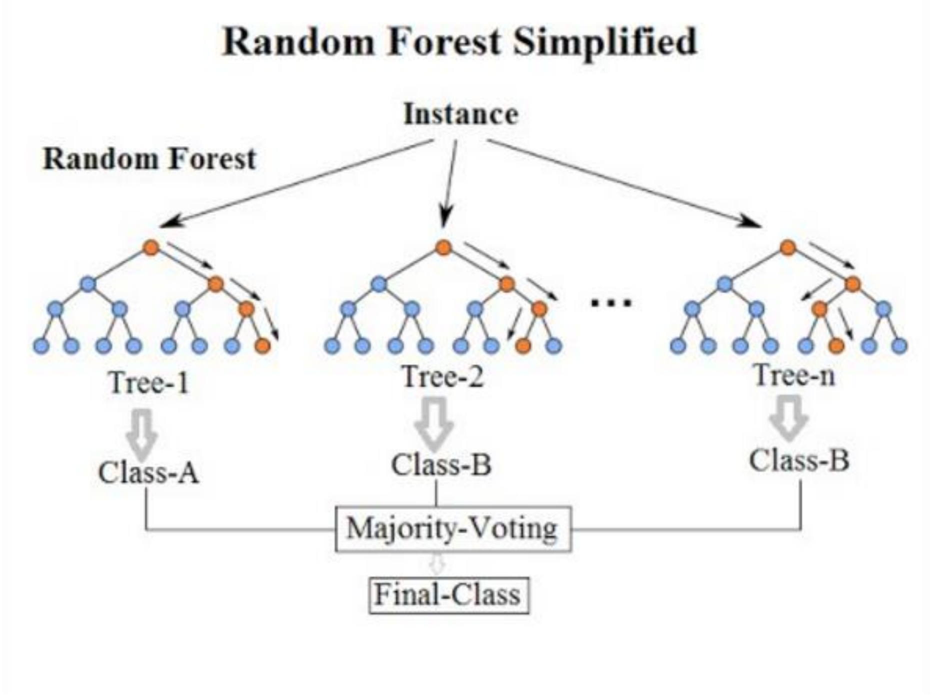
The types of grams in N-grams used depend on the problem and the complexity of the data.
1. Unigram: Unigram is the simplest technique of N-grams, using only single words in natural language processing. Because only single words are used, unigrams are not related to sentence structure.
2. Bigram: The next step of N-grams is to use bigram. Bigram uses the word pair in the next row until it parses the sentence. With this technique, a bigram word association can give the probability of a more complex sentence.
3. Trigram: Trigram uses 3 consecutive words in a sentence to create features. This can help the model understand the context of the passage and increase the accuracy of the model.

The results of data analysis based on different grams will yield different results depending on the intended use and input data. Usually, however, using bigrams and trigrams is the most popular because they take less computational resources than larger N-grams.

The result is a bag-of-n-grams model for a classifier to train the linguistic algorithm.

2.1.2 Random Forest

Random forest algorithm takes the decision tree concept further by producing many decision trees. The approach first takes a random sample of the data and identifies a key set of features to grow each decision tree.
These decision trees then have their out of bag error determined (error rate of the model) and then the collection of decision trees are compared to find the joint set of variables that produce the strongest classification method.



The strength of Random Forest is its ability to handle big data, reduce overfitting and provide accurate point prediction. Combined with feature extraction through Bag-of-Words, TF-IDF or Word2Vec, the Random Forest model can help increase the efficiency of Sentiment Analysis by classifying sentences into sentiment scores.

2.2. Deep learning approaches

2.2.1 Word2vec

Word2vec is a group of simple and well-known models that are used to produce word embeddings. Word2vec takes as its input a large corpus of text then produces a vector space which has much lower dimensions than the number of words in the dictionary, with each unique word in the corpus being assigned as a corresponding vector in the space.

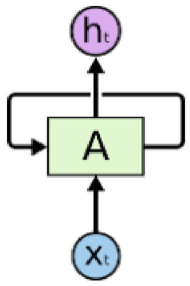
Word2Vec is based on the idea that the meaning of a word can be inferred from its neighboring words. It uses a shallow neural network with a single hidden layer to learn word representations from large amounts of text data. The two main architectures employed by Word2Vec are:

+ Continuous Bag-of-Words (CBOW): In this architecture, the model predicts the target word given a context of surrounding words. The input to the network is a sequence of words, and the output is the probability distribution over the vocabulary for the target word.

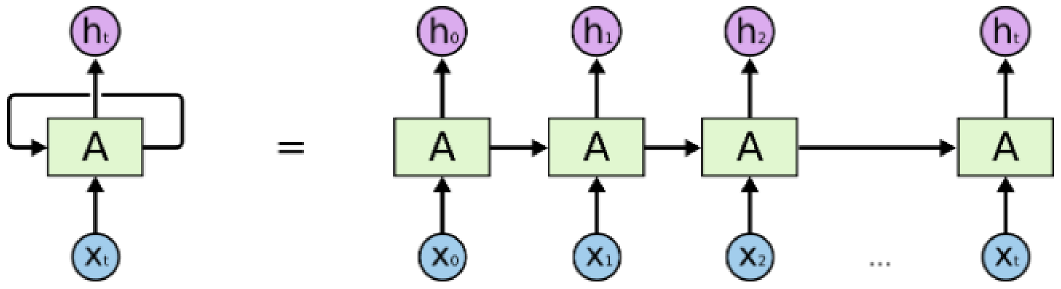
+ Skip-gram: This architecture is the reverse of CBOW. It takes a single word as input and predicts the context words around it. The objective is to maximize the probability of predicting the context words correctly. Skip-gram is more widely used because it tends to perform better on larger datasets.

2.2.2 Bidirectional Long Short Term Memory(BiLSTM)

Recurrent Neural Network (RNN) is a type of neural network that is capable of processing sequential input, including time-series data or text written in natural language. RNNs, in contrast to conventional feedforward neural networks, contain loops that let data from earlier inputs be stored and used to affect the current output.

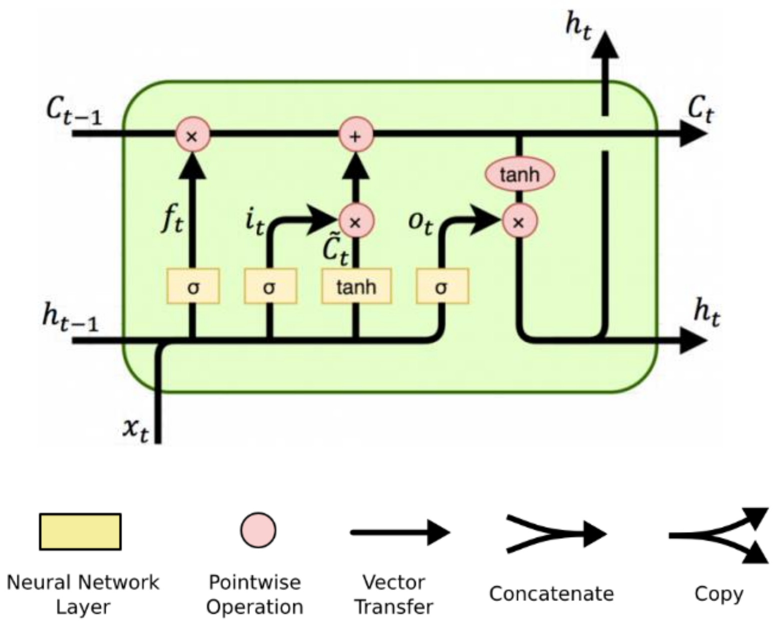


Recurrent Neural Networks have loops.



An unrolled recurrent neural network.

RNN in theory can handle long-term dependencies that are inherent in natural language text. However, in practice, that is not the case. Therefore, LSTM [1] – a special type of RNN is developed to solve the problem.



Feed-Forward:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

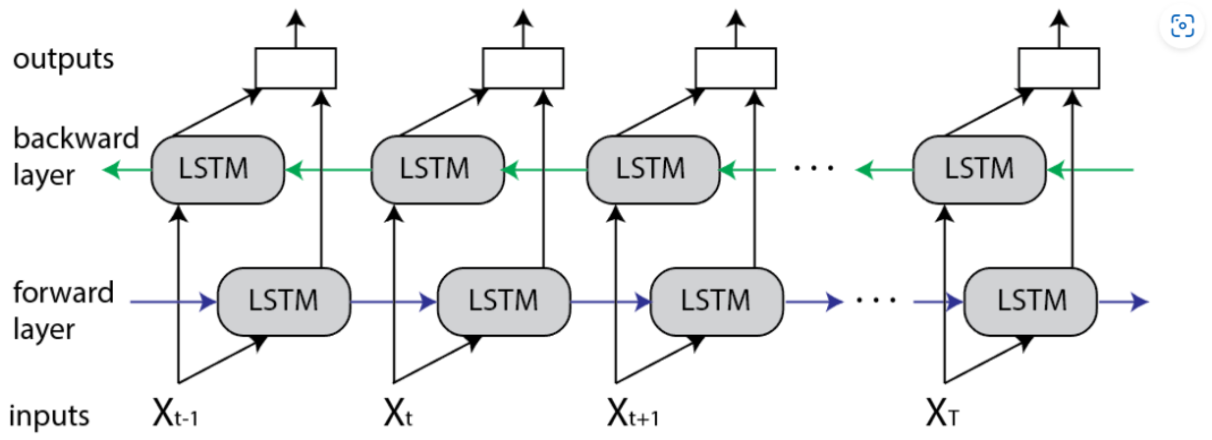
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

The core idea behind the success of LSTM over RNN is its cell state(C_t) which plays a role of the global or aggregate memory over all time-steps. While hidden state h_t cares more about the most recent time-step. Cell state corresponds to long-term memory and hidden state corresponds to short-term memory.

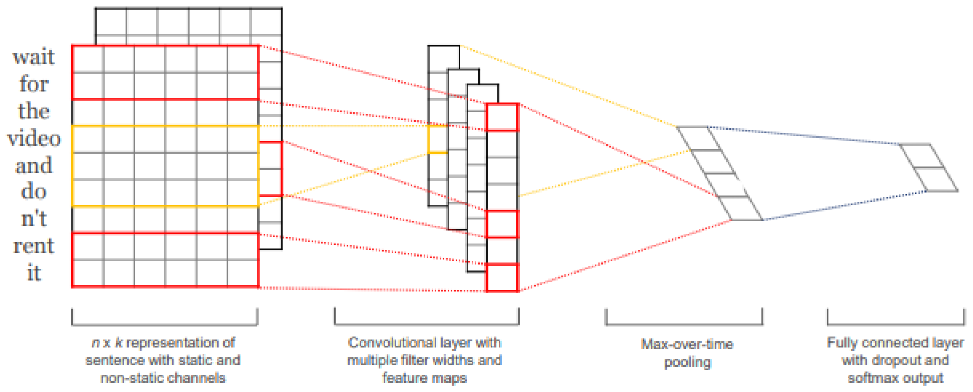
A LSTM cell consists of three gates:
+ Forget gate f_t decides which information to carry on
+ Input gate i_t decides which information to be updated in the cell state
+ Output gate decides which information to go out of a cell

To make the LSTM model more efficient, we simply add one more LSTM layer (backward layer) in the opposite direction of the forward layer. The output of the backward layer encodes the information of the sequence in the reversed order and will be concatenated to the output of the forward layer to make the final output.



2.2.3 Convolutional Neural Network(CNN)

CNN has a long history of solving computer vision related tasks. CNN has done a great job in understanding the content of a picture and extract features from it. In 2014, Yoon Kim has published a paper[2] showing that it turns out CNN can be very efficient in solving tasks in natural language processing(NLP) also.


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

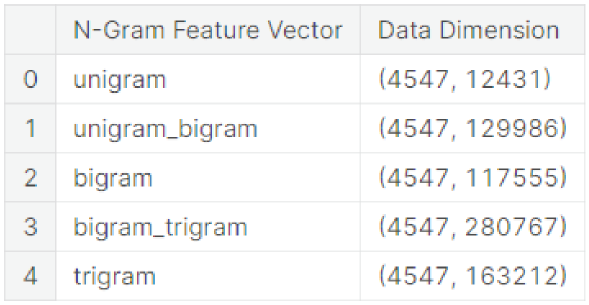
The text is first going through the embedding layer which has the output dimension of k to get a matrix of size n x k where n is sequence length. Similar to convolution operation in processing picture, n corresponds to spatial dimensions (width and height) of the image and k corresponds to channel dimension. Because the matrix only has one spatial dimension, we call it Conv1D. A Conv1D filter of the predetermined kernel size and stride will slide through the matrix from the top to the bottom in the sequence length dimension with the depth of full embedding size k.

3. Proposed methods

3.1 Machine learning method

We use 5 types of features to create the bag-of-word input: unigram, unigram_bigram, bigram, bigram_trigram, trigram. In 5 types of features, unigram_bigram and bigram_trigram are the combination of 2 types of grams to form a feature to help your model or algorithm understand the association between single words (unigram) and consecutive word pairs (bigram) in sentences, and is able to better predict or classify data based on this feature.

We treat unigram, bigram, unigram-bigram, trigram, bigram-trigram in N-grams as separate tuples and build a Random Forest for each of these datasets.
The data dimension for each feature as below:



	N-Gram Feature Vector	Data Dimension
0	unigram	(4547, 12431)
1	unigram_bigram	(4547, 129986)
2	bigram	(4547, 117555)
3	bigram_trigram	(4547, 280767)
4	trigram	(4547, 163212)

Because the data dimension of each feature vector is too high so we use a method to reduce the dimension that is SelectFromModel.

When using SelectFromModel on each of these 5 separate datasets, the model will perform an evaluation and determine the importance of each feature, then will remove unnecessary features to reduce the number of remaining features. The final result of SelectFromModel will be a set of the most important features that retain the necessary values, which improves the accuracy and processing speed of the model.

Random Forest evaluates the importance scores of features based on two methods: orthogonality importance and mean decrease impurity.

1. Permutation importance

Permutation importance is a way of assessing the importance of a feature by removing it from a data set and considering the impact of that removal on the accuracy of the Random Forest. In particular, a feature that is removed will significantly reduce the predictive power of the model.

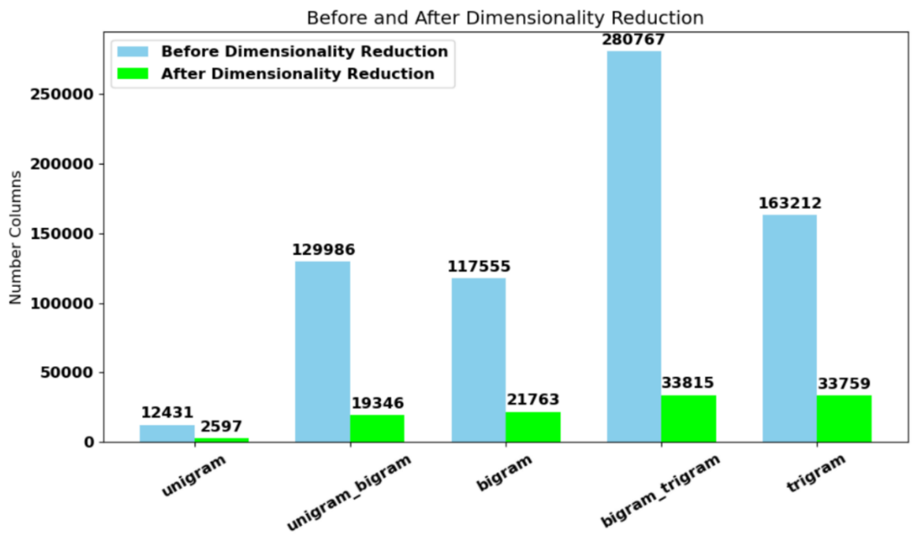
The process of calculating the permutation importance consists of the following steps:
- First, we train Random Forest with all the features.
- Then a feature is selected and removed from the data set.
- The model used to predict on the dataset has removed this feature.
- Score is calculated by comparing the new prediction result with the original prediction result. If the accuracy drops a lot, the feature is of high importance.

2. Mean decrease impurity

The mean decrease impurity is calculated by measuring the decrease in the error value of the Random Forest when the feature is used to build a decision tree in some depth. In particular, suppose a feature allows to separate samples into two groups of different purity, then that feature is of high importance.

The mean decrease impurity calculation process includes the following steps:
- Each decision tree in the Random Forest is used to generate a set of regles containing the feature under consideration.

- Each feature is scored based on the improvement in the purity of the decision tree when using them to separate samples.



Finally, we train the model again with the reduced dimensional dataset

We use N-grams to represent the text as a vector of features then pass that vector of features to the random forest model.

3.2. Deep learning methods

In both the architecture, we use an embedding layer to learn the word representation, BiLSTM or CNN layer to capture the information of the review and dropout to regularize the model. Another point to consider is that the difference between the number of parameters of BiLSTM architecture and CNN architecture is insignificant (4 124 745 and 4 125 445). Therefore, in our design, the magnitude of two models are nearly the same.

3.2.1 The BiLSTM architecture

Layer	**Hyperparameters**
Embedding	embedding_dimension = 200
Dropout	dropout_rate = 0.2
BiLSTM	units = 42
BiLSTM	units = 42
Dropout	dropout_rate = 0.2
Fully-connected layer	activation = softmax

3.2.2 The CNN architecture

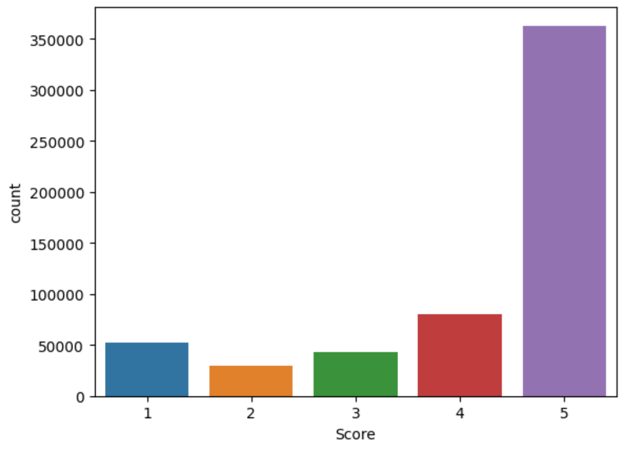
Layer	Hyperparameters
Embedding	embedding_dimension = 200
Dropout	dropout_rate = 0.2
Conv1D	filters = 64, kernel_size = 3, padding = same
Conv1D	filters = 64, kernel_size = 3, padding = same

MaxPooling1D	pool_size = 3
Conv1D	filters = 128, kernel_size = 3, padding = same
Conv1D	filters = 128, kernel_size = 3, padding = same
GlobalAveragePooling1D	
Dropout	dropout_rate = 0.2
Fully-connected layer	activation = softmax

4. Experiment

4.1. Dataset

- Name: Amazon Product Reviews
- Source: https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews?datasetId=1461623&sortBy=voteCount&searchQuery=senti
- Description: This dataset contains more than 568k consumer reviews on different amazon products.
- Size: 568454 records
- Usage: In the original dataset, there are 10 columns in total. We make use of 2 columns: Text(actual text of the review) as the predictors and Score(Product Rating) as the label
- Distribution:



Star	Account for
1	9%
2	5%
3	7%
4	14%
5	64%

The dataset is quite skewed with a large number of records rated 5 stars.
- Sample data:
+ Text: I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
+ Score: 5

4.2. Experimental scenarios

Computing configurations: Kaggle kernel with RAM 13GB and GPU P100

Due to our lack in computing resources, more specifically RAM. We stratified sample 10% of the dataset. The newly dataset is then stratified-splited into 3 set: training set, validation set and test set with the ratio of 80%, 10% and 10%

All the experiment is validated by the accuracy on the validation set.

4.2.1. Machine learning methods

Random forest will be trained with these hyperparameters:
- n-esimators = 100
- max_depth = 30
- min_samples_split = 4

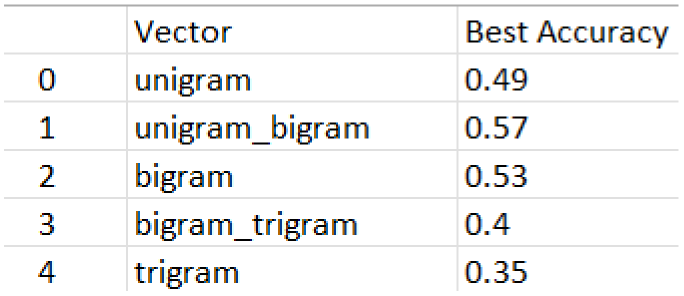
4.2.2. Deep learning methods

All deep learning models use Adam optimizer with learning rate = 1e-3, batch size = 128, epochs = 5,. For the pretrained versions, we use GoogleNews-vectors-negative300.bin.gz (learned by Word2Vec) as the weight initialization of the embedding layer and we let the models fine-tune the embedding weights in the training process. GoogleNews-vectors-negative300.bin.gz is produced by the model trained on part of Google News dataset (about 100 billion words) with the following details:
+ Architecture: Continuous bag-of-words (CBOW)
+ Dimensions: 300
+ Window size: 5
+ Training method: Negative sampling

4.3. Experimental results

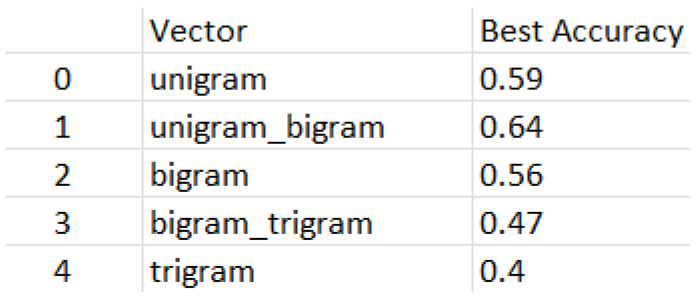
4.3.1. Machine learning model

Each vector will be evaluated by the accuracy value on validation data to choose the vector that best affects the model
Before reduction:



	Vector	Best Accuracy
0	unigram	0.49
1	unigram_bigram	0.57
2	bigram	0.53
3	bigram_trigram	0.4
4	trigram	0.35

After reduction:



	Vector	Best Accuracy
0	unigram	0.59
1	unigram_bigram	0.64
2	bigram	0.56
3	bigram_trigram	0.47
4	trigram	0.4

So we choose unigram_bigram as the main vector data to create the Random Forest model again and use that model to predict the labels.

We see how dimensionality reduction affects on our Machine learning model

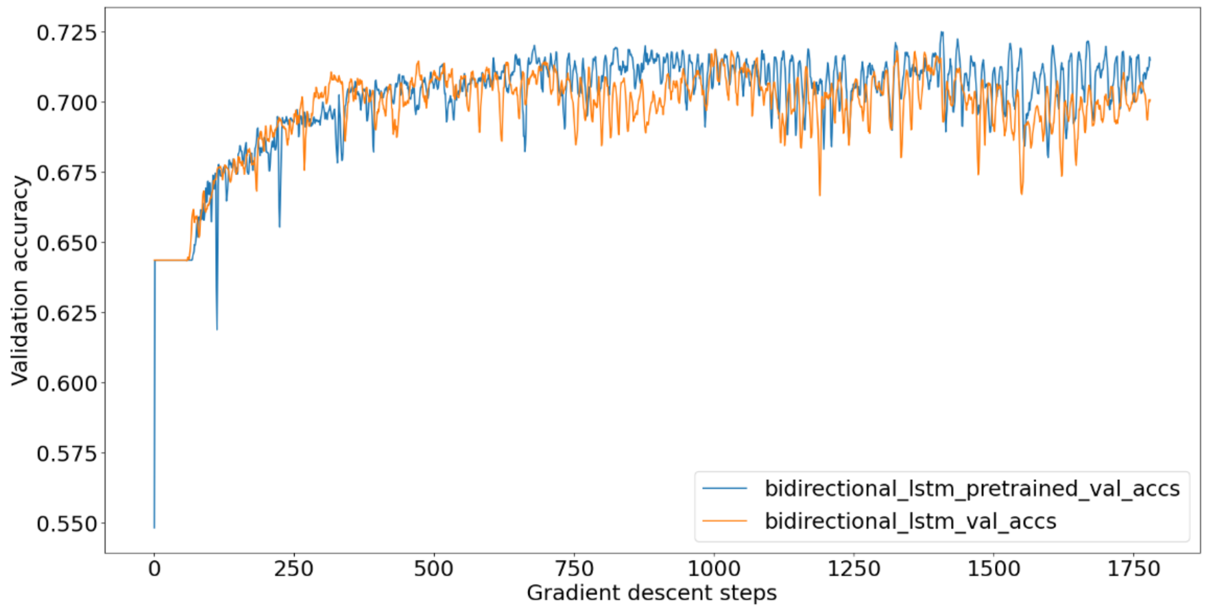
	Training time	**Accuracy**
Before reduction	10h 30min	57%
After reduction	2h 10min	64%

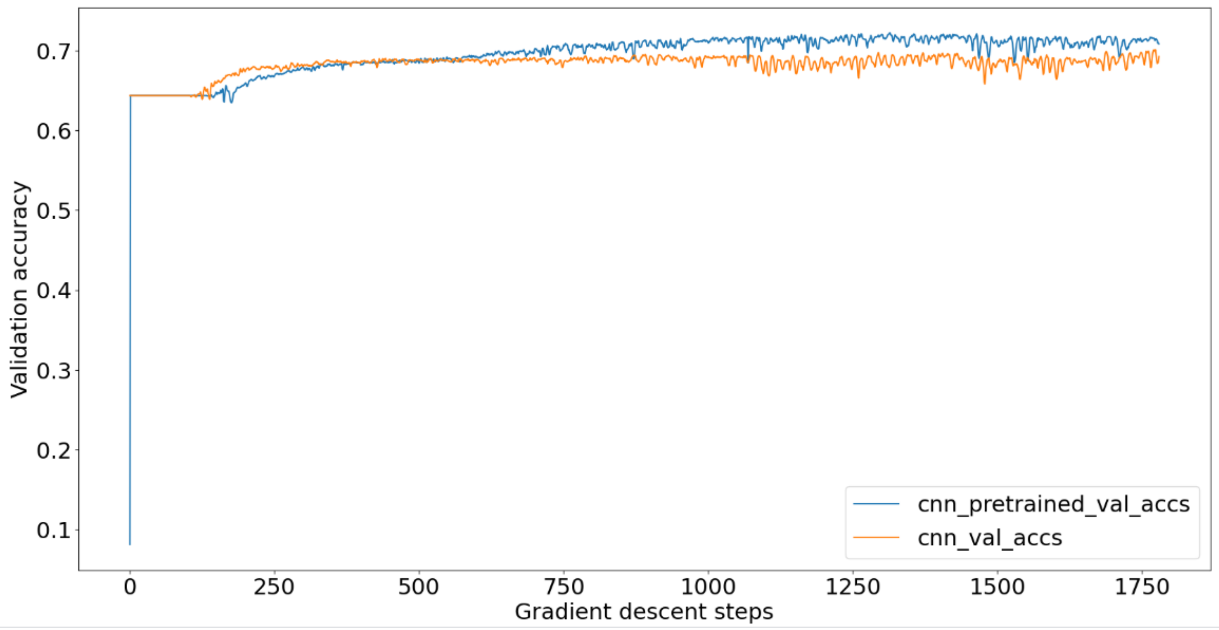
Obviously, dimensionality reduction reduces the training time. However, accuracy in the validation set is surprisingly improved. This may be due to the low frequency features that add noise to the data.

4.3.2. Deep learning models

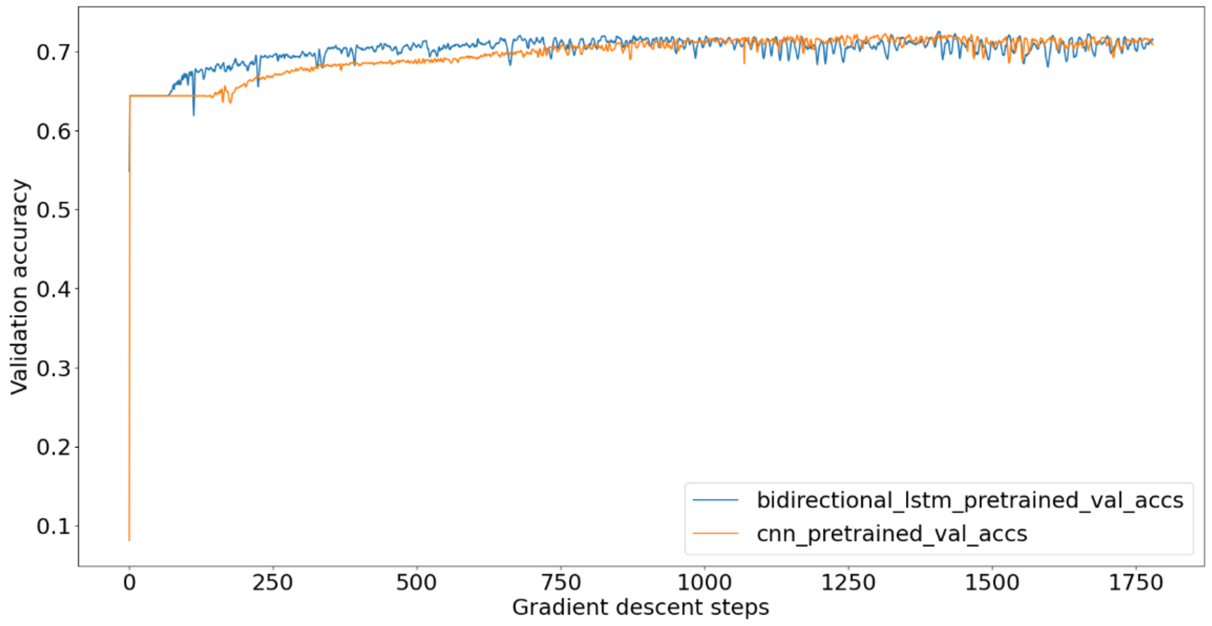
Model	**Training time**	**Storage capacity**
BiLSTM	30min 58s	46 092KB
CNN	2min 30s	45 488KB

CNN makes a huge acceleration in training time in comparison to BiLSTM (approximately x15 times faster). This phenomenon is due to convolutional operation has a higher degree of parallel coefficient than the sequential structure encoded in the architecture of BiLSTM layer. CNN and BiLSTM take up nearly the same amount of memory, this is understandable because of the fact that CNN and BiLSTM has nearly the same amount of parameters.





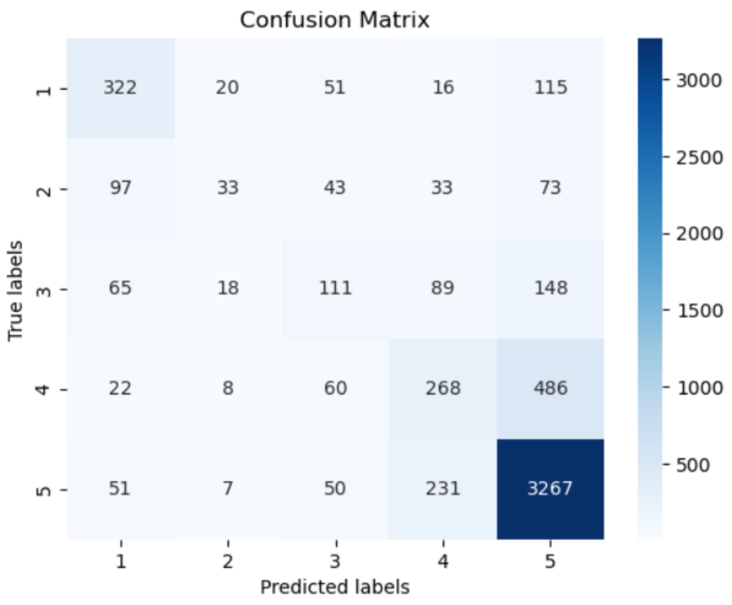
What we derive from two pictures is that CNN curves are smoother than BiLSTM curves. The use of pretrained embedding weights make the models performances better



BiLSTM seems to perform better in the early gradient descent steps. But in the following gradient steps, CNN model keeps up with the performance of the BiLSTM

4.4. Evaluate

From the experiment results, we can derive that pretrained CNN yields the highest performance in both accuracy and training time. Therefore, we choose this model to train on both train set and validation set, then evaluate the model on test set. The details of the final result is shown below:



Star	Precision	Recall	F1-score
1	0.58	0.61	0.60
2	0.38	0.12	0.18
3	0.35	0.26	0.30
4	0.42	0.32	0.36
5	0.80	0.91	0.85

- Accuracy: 0.70
- Macro-average f1-score: 0.46

The model does quite a good job of predicting the instances that fall in class 5. For the hard classes such as 2, 3, 4, the model still does not do well, we may improve this situation by training on more instances from these classes. The instances that belong to 4-star class tend to be misclassified into 5-star and vice versa.

5. Conclusions

In this project, we present our work of using three types of model to solve the amazon sentiment analysis problem CNN, BiLSTM and random forest . And we find that:

-Word2vec embedding matrix enhance the performances of CNN and BiLSTM

-CNN gives a more stable performance

-CNN is more computing efficient than BiLSTM using hardware such as GPU

- Deep learning models that can take advantage from the sequence property of the text produce a better result than machine learning model

-CNN with a pretrained word2vec embedding matrix is the best model not only by its accuracy but also by its saving in computing resources. However, the model still needs to improve its performance in predicting the classes that are inherently hard. The most promising solution is using more computing resources to exploit all possible data points in the dataset.

References

[1] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics
[2] Long Short-Term Memory (Sepp Hochreiter and Jürgen Schmidhuber), In Neural Computation, volume 9, 1997
[3] Efficient Estimation of Word Representations in Vector Space (Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean) , 2013
[4] Understanding LSTM Networks -- colah's blog
[5] Differences Between Bidirectional and Unidirectional LSTM | Baeldung on Computer Science
[6] https://developers.google.com/machine-learning/guides/text-classification