Ensemble learning with different word embedding architectures for sentiment analysis

Manasa Sandhya Gunda
20360532
msgunda@uwaterloo.ca

Sai Teja Ponugoti
20841154
stponugo@uwaterloo.ca

Valliammai Periakaruppan
20831094
vperika@uwaterloo.ca

***Abstract*—In the recent times, due to proliferation in the usage of technology and online shopping, it has become essential to know, the number of reviews or comments generated, we need to know which are positive and negative reviews otherwise known as the sentiment of the text. It is helpful for both the buyers and sellers to know the quality of the product that they are buying or selling. This is sentiment analysis on text and also belongs to a broad category of text classification. To construct text classification models, we use machine-learning algorithms. We used the amazon reviews data set which has customer reviews, and the developed classifier is used to predict whether a review is positive or negative.In general Word embeddings are used to represent text to be able to input text for the classification model. Word embeddings can be obtained using any of the following methods:**

- **Learn word embeddings from the data available or**
- **Use available pretrained word embeddings or language models**

In the first case, the embeddings learned will be more specialized or customized to the context of the training data used. In the second case, the pre-trained embeddings available are learned on huge corpora like wikitext and twitter, so they possess huge vocabulary and are more generalized. In this paper we propose a way to use both these advantages of context-specific and pre-trained word embeddings or models by ensembling the results obtained from these models. Here we are classifying the reviews as either positive or negative in-order for the companies to retrospect further based on the reviews from the customers to improve their service or build their brand. This step will be performed by the process called sentiment analysis, and the sentiment analysis on reviews allows companies to identify customer's sentiment towards their products, brands, or services in online feedback.

***Index Terms*—Word2Vec, CBOW, Skip-gram, GloVe, ensembling, ULMFiT.**

I. Introduction

Sentiment analysis can be defined as a machine learning technique where the context of the text is detected in terms of polarity like positive or negative opinion. This opinion can be within the text or whole document or a single paragraph or phrase. Analysing people's opinion is essential for business since the customers can openly state their feelings and thoughts on online platforms. When these feedback from the customers from social network services is analysed, the services can be met with their needs.

Sentiment analysis is a process in which we create or automate a classifier which can differentiate whether a text or sentence is positive or negative or neutral. In recent times, it is considered to be very much important topic because it helps online shopping platforms to analyze and check their brand reputation, their social media review and get customer feedback. They will use the sentiment analysis model to build their brand further. By using sentiment analysis, online platform companies do not have to spend endless hours tracking and keeping a record of customer data such as survey responses, reviews, support tickets, and social media comments.

On the base level, sentiment analysis can be defined as classification problem in particular a text classification, i.e. it interprets and analyzes the emotions in the sentence. There will be predefined categories, and we need to fix the texts in the predefined categories. Understanding peoples emotion is essential for business companies since expressing the opinions on the product is more easier than earlier. For example, by utilizing the sentiment analysis model, the business companies can automatically analyze the reviews on the online platform about their product and identify or understand the customer's emotion or mentality after using their product whether are they satisfied with their service, the quality of the product or the pricing of the product.

Text classification is a supervised learning task. Text Categorization is merely assigning the texts or sentence its outcome based on its meaning since it is written in natural language. In machine learning, it is matching the features to the labels where the features can be words-level or character-level or sentence level and the labels being the sentiment or the opinion like positive or negative or neutral.

In the world's total data, it is estimated that 80% of it is unorganized or unstructured [1]. Doing a sentiment analysis makes it possible to analyze such vast amounts of data. The sentiment analysis does emotion detection. Generally, on online review platforms, the 5 primary types of emotions or the polarity category are, Very Positive, Positive, Neutral, Negative, Very Negative. But in our dataset Amazon review-2, we have 2 main categories of emotions only, Very Positive and Very Negative.

In this project, our main objective was to explore machine learning algorithms with NLP technique to build classification models by experimenting with different models and noting their accuracies. Our primary goal is to create a model that takes a review from our amazon dataset and produces either a 1 (indicating the sentence carries a positive sentiment) or a 0 (indicating the review carries a negative sentiment). Finally, we choose the model with the best accuracy based on average accuracy and average weighted accuracy.

II. Literature review

Here, the existing body of knowledge on online shopping platform and similar text classification models are discussed.

Text classification is a contemporary topic in natural language processing, and presently, there has been much research going on this topic. In recent years, the rate of research in NLP has been drastic. Since NLP with deep learning techniques is the emerging research area, there are many recent works related to Text classification or categorization

The paper [3] proposed convolutional-recurrent neural networks architecture which achieved 94.50% in the polarity sentiment analysis task on the Yelp dataset, and this classification algorithm works on a character level. But in recent years, researchers are proposing that classifying based on word levels.

In work by Vojtech Myska and Malay Kishore Dutta [4], they developed a Linguistically independent sentiment analysis model using convolutional-recurrent neural networks model. This classifier also works on character level, not on word or sentence level, their classifier achieved an accuracy of 93.64%, and their proposed model also works on various languages. In [5], they worked with a word-level classification algorithm. They show that shallow word-level CNN's works more accurately than deep nets such as character-level CNN.

In [6], they use a novel transfer learning approach BERT. BERT is an NLP pretraining technique for word embeddings. It is a Google model to help the search for word/text prediction. There have many research papers that used this novel approach. In [6], they used bert pretrained model to identify the hate speech in online platforms. [7] used Inductive transfer learning in natural language processing. They proposed the Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP [7]. Using the above-proposed method, they minimized the error by 18-25%. Conroy, Rubin, and Chen [10] developed models to classify the misleading articles in the journal.

There is been a extensive research on hate comments on social platforms like Twitter [11], Reddit [12], [13], and YouTube [10]. The key aspects that differentiate various methods are the features used in conventional machine learning approaches, and surface-level features such as word bags, word-level and character-level n-grams, etc., have proved to be the most predictive features [17], [18], [19].

This paper [20] has researched in finding an efficient baseline for text classification. They have put forward that the classifier they proposed performs on the same level as other deep learning classifiers and it can train more than one billion texts.

For tasks where a relationship is to be identified between two sentence pairs, such as Stance detection, a general approach entails encoding the two texts in some form, and then try to classify their relationship as discussed by Bowman et al. [23]. In this task, a new corpus of labelled sentences was shared and then the authors explored methods to classify them into "entailment" and "contradiction" based on semantic relationships..

Two other prominent papers include Bahdanauet al. [8] and Luong et al. [9] which have expanded the work of the Google team by adding new techniques such as "attention", which enables the model's decoder to concentrate on certain portions of the encoded input sequence at each stage of the output to make the best predictions possible.

III. Methodology

In this section we will discuss overview of typical NLP tasks that are used in sentiment analysis.

A. Text pre-processing

In any machine learning techniques, data preparation is an important part which not done properly can affect accuracy. Same is the case with the text input data, before applying any machine learning algorithm on text data, it requires data pre-processing. Text pre-processing if done correctly, will increase the accuracy for NLP tasks. Text pre-processing depends on the type of the input text, but it consists of normalization, tokenization and noise removal.

1) Normalization: Normalization is a process of converting the text into a standard format. Normalization is useful in the preparation of the text to pre-process later. Normalization is to handle abbreviations, slang and converting text to lower case. Handling abbreviations is when short forms like "NLP" converted to their corresponding equivalents like "Natural language processing". While words like "they've" is converted to "they have" while handling slangs. Even though converting text to lower case might seem to be an easy one, it is an effective and straightforward method of normalization.

2) Tokenization: In tokenization, the text converted into tokens. Tokens are usually words, phrases, symbols and other meaningful elements [32], [33]. In general, the main goal is to separate the words in a sentence.

3) Noise removal: Most of the input text data contain special characters and extra spaces which are not needed. Noise removal is where the data cleaned to remove the additional spaces and special characters.

There are more techniques like removing stop words, lemmatization, spelling correction, stemming [2] but not used in this project.

B. Text Representation

Extraction of features is an important step towards any problem with machine learning. No matter how well we use a modelling algorithm, we'll get bad results if you feed in bad features into the model. This is referred to as "garbage in, garbage out". [22]. How do we transform a given text into numerical form so that it can be fed into NLP and ML algorithms.In sNLP, this conversion of raw text into an appropriate numerical form is called text representation. Text representation has been an active area of research in the past decades, especially the last one.

For ML algorithms to work with text data, some mathematical form must be used to transform the text data.In general units of text (characters, phonemes, words, phrases, sentences,

paragraphs, and documents) are expressed in number vectors. Let's discuss the basic idea for text representation,discuss their drawbacks and then we move on to discussing word embeddings and their advantages over traditional methods.

1) One-Hot Encoding: In this type of text representation, each word in the text is represented using a unique number between 1 and $|V|$(vocab size), where V is the set of corpus vocabulary. Each word in the corpus is then represented using a V-dimensional vector, in which all indexes are filled with '0' except at the index which is equal to the unique number assigned to that word, at that index we simply put a '1'.The representation for single words is then combined to form the representation of a sentence [24].

The one-hot Encoded text has various drawbacks. The size of the one-hot vector is directly proportional to the vocab size(very high) and is also sparse, which poses computational issues. This type of representation also lead to inconsistent sentence length and also treats words as atomic units and has no notion of similarity or dissimilarity between words [24]. These days, the one-hot encoding scheme is seldom used. Some of the above disadvantages shortcomings can be addressed by the bag-of-words approach discussed next.

2) Bag of Words: Bag of words (BoW) is a classic text representation technique commonly used in NLP, particularly when it comes to text classification problems. The main concept is to portray the text being viewed as a bag of words while ignoring the order and meaning. The basic idea behind this is that it means that a single set of words characterises the text belonging to a given class in the dataset. If two pieces of text have almost the same words, then they belong to the same (class) bag. Therefore one can classify the class (bag) to which it belongs by examining the words present in a piece of text [24].

Similar to one-hot encoding, BoW maps every word in the text to unique integer number(Id) between 1 and $|V|$. Then a vector of $|V|$ dimension is used to represent each document, in which the ith component of the vector, i = unique number, is simply the number of times that word occurs in the document, i.e., we score each word in V by their occurrence count in the document. A variation of this is also to ignore the frequency and just put '1' for every word that occurs, ignoring their frequency.

With BoW, we overcome the inconsistent length representations. But BoW still has lot of drawbacks. Even with BoW representation also, the vector increases with the vocab size and the sparsity continues to be a problem. The BoW is not efficient at capturing the similarity between different words that mean the same as "I run" and "I ran". There is not a way to deal with out of vocabulary words (i.e., new words that were not seen in the corpus) with this representation [24].

3) Bag of N-grams: The n-gram technique is a set of n-words that occurs in a text set, "in that order". This can be used to represent text. This representation captures some context and word-order information in the form of n-grams. Thus, resulting vector space can capture some semantic similarity. However, with the increase in vocab size, this representation is also computationally expensive.

4) Term Frequency-Inverse Document Frequency(TF-IDF): TF-IDF addresses this issue the issue of treating all words with equal importance. It aims at quantifying the value of a given the word in the text and in the corpus in relation to other words. TF (term frequency) measures the frequency in which a phrase or word appears in a given document and IDF (inverse document frequency) measures the significance of the term in a corpus. The TF-IDF score is a product of these two terms. Even today, TF-IDF continues to be a popular representation scheme for many NLP tasks [24].

Looking back at all the representation schemes we have discussed so far, and we find three basic drawbacks:

- They are discrete representations — that is, they treat units of language (words, n-grams, etc.) as atomic units. This discreteness hampers their ability to identify inter-word relationships.
- The function vectors are high-dimensional and sparse representations. With the scale of the vocabulary, the dimensionality increases. This hampers learning capability.
- They cannot handle out of vocab words.

5) Word Embeddings: Word embedding is a technique of learning features of the text in which every phrase or word or the vocab is mapped to a N dimensional real numbers. Various word embedding methods for machine learning algorithms have been proposed to convert unigrams into understandable inputs. This work focuses on Word2Vec and GloVe, two of the widely used methods for deep learning techniques successfully [25].

1) **Word2Vec**
"Word to Vector" representation model is presented by T. Mikolov et al. [26], [27] as an advanced word embedding learning architecture. Word2Vec has two hidden layers and can be learned with continuous bag-of-words (CBOW), and the Skip-gram models (as shown in Figure 1), the Word2Vec approach uses shallow neural networks to construct a high-dimension vector for each word. We will look at these two architectures in detail below.

   a) **Continuous Bag-of-Words Model**
   In CBOW, the primary task is to create a language model which correctly predicts the centre word provided the context words in which the word centre word appears. CBOW attempt is to learn a language model that tries to predict the "centre" word in its meaning from the context words. Using a toy corpus, as shown in the Figure 2, let us understand this. If we take the word "jumped" as the centre word, then its meaning is created by words in the vicinity of it. If we take the context size of 2, then the context is given by brown, fox, over, the for our example. As seen in Figure 2, CBOW uses the context terms to predict the target word — "jumped" [24].

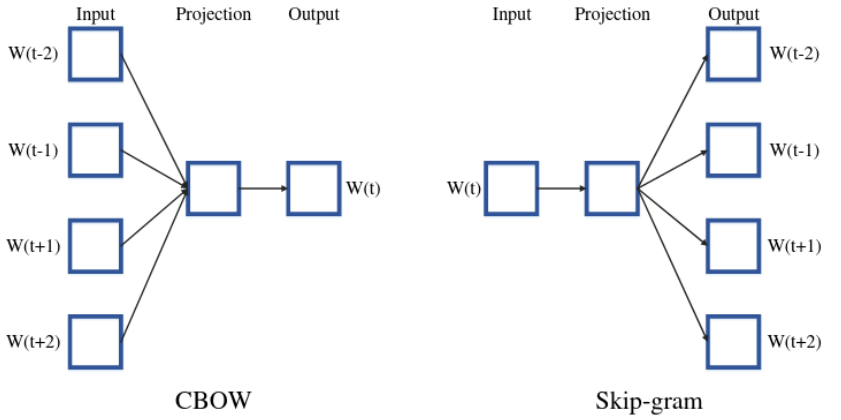

Fig. 1. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. [26]

The concept discussed in the above paragraph is then applied to the entire corpus to create the training set by running a 2k+1 sliding window over the text corpus, where k is the context size. The training examples obtained for above-mentioned example when k =2 is shown in the Figure 3 below.

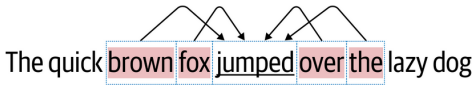

Fig. 2. Given the context words CBOW predicts the center word [24].

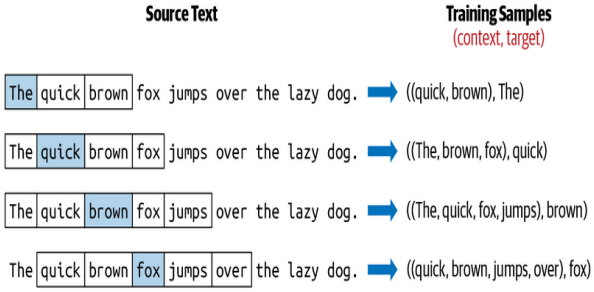

Fig. 3. Dataset preparation for CBOW [24].

b) **Skip-gram Model**
Skip-gram is very similar to CBOW, with a few slight modifications. In Skip-gram, the goal is to predict the context words from the centre word. Using the centre word "jumped", we try to predict every word in context — "brown", "fox", "over", "the" — as shown in Figure 4 for our toy corpus with context size 2. For every word in the corpus as centre word Skip-gram repeats this one step.
Similar to CBOW, Skip-gram training samples are created by running a sliding window of size 2k+1 on the text corpus. The centre word in the window is the X and k number of Y's on each side of the centre word. It does offer us 2k data points, unlike CBOW. Figure 5 demonstrates the way we slide the window over the whole corpus to build the training collection.
There are several implementations available to use both the CBOW and Skip-gram algorithms in practice which abstract the mathematical details for us. Gensim is among the most commonly used implementations.

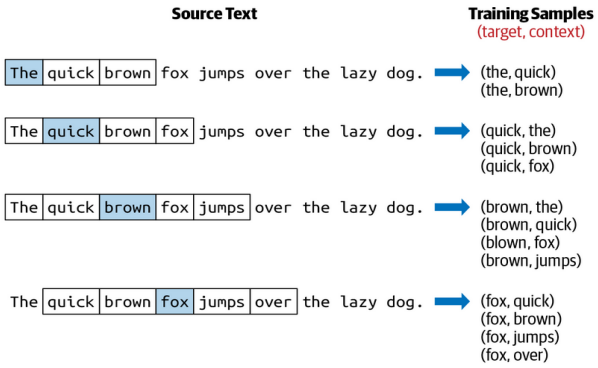

Fig. 4. given the center word Skip-gram predicts every other word in context [24].

2) **Global Vectors for Word Representation(GloVe)**
Another powerful word embedding technique used for text classification is for Word Representation (GloVe) [28]. The technique is somewhat same to the Word2Vec process, where all words are presented by using a vector of high dimensions and trained over a vast corpus based on the context words. The GloVe pre-trained word embedding is used in many NLP tasks is based on 4,00,000 vocabularies with 50 dimensions for word presentation and trained over Wikipedia 2014 and Gigaword 5 text corpus [25]. GloVe also offers many pre-trained word embedings with different dimesnions of 100, 200, 300

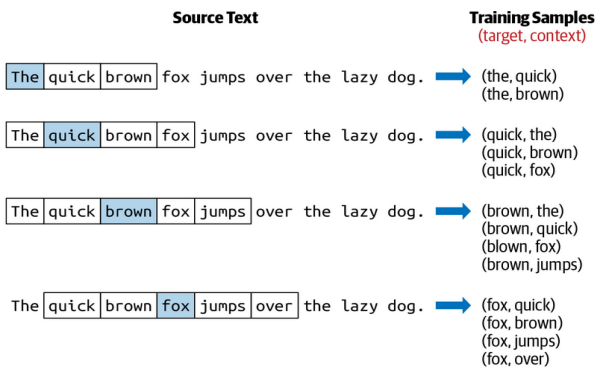

Fig. 5. Dataset preparation for Skip-gram [24].

that are trained in even larger corpus like Twitter content. t-SNE [29] technique is used to visualize word distances over a sample dataset is shown in Figure 6.

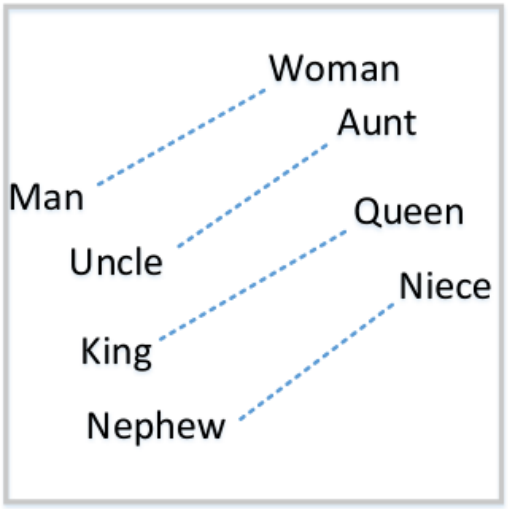

Fig. 6. Global Vectors for Word Representation [25].

C. Classification models

Any classification can be done with the statistical models like Naive Bayes, Logistic Regression, Support Vector Machines and Neural networks.

- **Naive Bayes:** uses Bayes theorem and a Naive assumption that there would be no relationship among the different features to predict the outcomes. This is one of the probabilistic algorithms.
- **Logistic Regression:** is a traditional classification algorithm which is better used when the outcome should be in binary i.e, either 0 or 1.
- **Support Vector Machines:**, which is a non-probabilistic method that uses the text examples representation in multidimensional space as points.
- **Neural Networks:** have a different set of algorithms which try to attempt to mimic the human actions. This also can be used to classify the text data.

In our project the base model is Convolutional neural networks (CNN).

1) CNN for document classification: CNN is one of the popular Deep learning neural networks. A CNN, which is a feed-forward neural network with an ensemble of processing nodes arranged in a stack of layers, usually trained end-to-end in a supervised manner. The basic structure in a CNN consists of convolutional layers interleaved with pooling layers. A convolution layer is composed of multiple computing units; each of this unit takes the input of a region vector which is a part of the input and applies a non-linear function to it. An important feature of these convolution layers is weight sharing. The computation units share the weights and biases in a layer. The output of the convolution layer is then passed to the pooling layer. A pooling layer is used to decrease the input size. Most frequently used pooling layer types are max-pooling and average-pooling.

2) CNN for text: Now let us consider for the application to be text to which CNN should be applied. The input of a CNN should be represented in vector form. Thus there is a need to convert the text to vectors of real-valued numbers. For example, consider the vocabulary to be V = {“I”, “it”, “love”, “hate”, “don’t” } and if the text input I = “I love it”. One way to represent this is to use one-hot encoding for each word so the vector would be [34]:

$$x = [10000|00100|01000]^T$$

In the convolutional layer, the input text vectors are converted to feature vectors. In other words, in the convolutional layer trains so that the input text vector can be represented in a lower-dimensional vector space.

The output of the convolutional layer is then passed to pooling layers. In the conventional CNN pooling layer, it subsamples the input which it gets. Usually in text classification methods pooling is applied over the complete output of convolutional layer which will give a single value as output for every filter.pooling over the complete output, yielding just a single number for each filter. Pooling usually reduces the dimension of the input data while keeping the essential features of the input data.

There can be other layers like flatten where the pooled feature map is flattened and dense layers connected to the output layer in CNN.

Other neural network methods like RNN, LSTMs can also be used for sentiment analysis. However, in this project, a CNN network is chosen, as similar accuracy can be obtained with it.

D. Ensembling methods

Many researchers have studied the technique of combining multiple classification predictions to create a single classifier. Among many methods like voting, averaging, Bagging

and Boosting [31], stacking, blending we have chosen the first three types namely max voting, averaging and weighted averaging ensemble methods which are simple but powerful techniques, for this experiment.

1) Max Voting: In Max voting ensemble method, multiple models are used to build predictions for each data point. Every model's prediction is considered as a 'vote'.The predictions(class), which gets the highest number of votes, is considered as the final resultant prediction. The techniques are quite simple but work well for classification tasks.

2) Averaging: Similar to the technique of Max voting, predictions from several models are considered and are averaged for each data point. We take an average of predictions from all models in this approach and use it to make the final prediction. Averaging may be used when dealing with regression problems or can be used when probabilities are obtained for classification tasks.

As our task is a binary classification task, the probability given by any classifier is in the range of (0,1), the closer the probability to 1, the higher is the chance that the predicted class is positive. We averaged the probabilities given by all classifiers and decided the result as a positive review if the probability is higher than 0.4 or as a negative review if it is less than or equal to 0.4.

3) Weighted Average: This method is an extension of the averaging method. All models are given different weights specifying the importance of the predicted value of each model. We considered the probability given by each model as the value to be weighed on.

IV. Dataset

The dataset used for this project is an Amazon customer reviews dataset [35], which consists of a few million reviews and the corresponding star ratings. The format of the dataset is in the form of "__label__ <label number><review>" for each review in a single line. The dataset with this format is loaded, and the labels and reviews are separated for further processing.

There are two labels in the dataset. 1-star and 2-star reviews are considered negative ratings, 4 and 5-star reviews are considered positive ratings and are labelled 1 and 2, respectively in the dataset. The reviews with 3 star which are neutral are not considered in the dataset [36].

The dataset contains reviews with positive and negative polarities. In other words, the number of classes is binary. The dataset contains a total of 4 million amazon reviews. The train data contains 3.6 million reviews, and the test data contains 400k reviews.

V. Proposed Method

The proposed method is an ensemble learning of 5 models with 4 different word embeddings and a pre-trained ULMFiT model. Below are the steps we followed:

1) **Text pre-processing:** The input, as mentioned in the dataset section, is in the form of "__label__ <label number><review>" for each review. This is first separated to labels and reviews for both train and test data. Data is then processed with normalization, noise removal and tokenization. These tokens are then used to create the sequences.

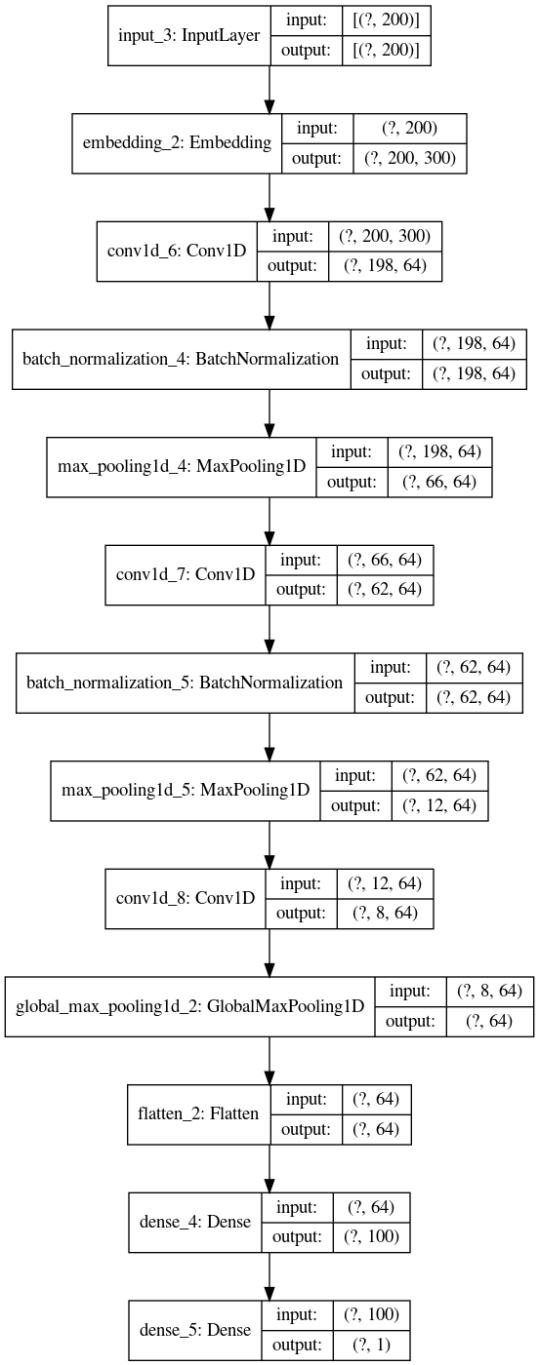

Fig. 7. Representation of the CNN model used.

2) **Word embeddings:** We used 4 different word embeddings to represent text and then use these embeddings as input to CNN model described later. We used both embeddings learned from scratch and which are also

pretrained:

a) **Continuous Bag-of-Words(CBOW) Model** The CBOW model is used to learn embeddings using the entire train corpus to get the word embeddings with embedding size as 300 and vocab size as 12,000 words. We used gensim [15] module to train these vectors with parameter sg=0 (to use CBOW).
b) **Skip-gram Model** Skip-gram, which is a similar model to CBOW, is applied on the entire train corpus to get the word embeddings with embedding size as 300 and vocab size as 12,000 words.To train these vectors we used sg=1(use skip gram) using gensim [15] module.
c) **Global Vectors for Word Representation (GloVe)** the First model uses GloVe pre-trained word embeddings [28] for training the network, with the embedding size as 200 and considering the top 20,000 words.
d) **Google's pre-trained Word2Vec model** The second is also a pre-trained Google's trained Word2Vec model. This includes a vocabulary of 3M words and is trained on a vast amount of 100 billion words from the Google News dataset [37]. The embedding size is 300, and the top 20,000 words are considered.

3) **CNN for classification:** The model shown in Figure 7 is used in this project as the classification algorithm for all 4 word embeddings mentioned above. The embedding layer in different model is designed according to the word embeddings used other than the model in ULMFiT. The model constitutes of one convolutional layers, two max-pooling layers, one global max-pooling layer, two batch normalisation layers, flatten and then a dense layer.
4) **Universal Language Model Fine-tuning ULMFiT** ULMFiT is a transfer learning technique used for NLP tasks. ULMFiT is a model training technique where the word embeddings are created in the process. Using ULMFiT, the training was done only with 1,00,000 reviews in the training data since ULMFiT on the whole training corpus would take huge amount of computational capability.
5) **Ensemble methods:** Three types of ensemble methods are used to do the learning.

a) **Max Voting:** When multiple models are present, we can consider each model's prediction as 'vote'. Max voting is when the majority predictions are considered as the final prediction. We can say that this as a calculation of mode of the model's predictions.
b) **Average:** In this the average of the model's predictions is taken as final predictions.
c) **Weighted average:** In this the weighted average of model's prediction is taken as final prediction. After multiple trials we assigned the weights for different models as shown in Table I for best accuracies.

TABLE I
WEIGHTS ASSIGNED FOR EACH MODEL FOR WEIGHTED AVERAGE

Model	Weight
CBOW Model	0.225
Skip-gram Model	0.225
GloVe Model	0.2
Google Model	0.2
ULMFiT	0.15

VI. EXPERIMENTAL RESULTS AND ANALYSIS

Based on the above proposed method, we used various models to classify the sentiment of Amazon-2 review dataset. We have listed classification accuracy, precision, recall and f1-score of different models, obtained with our experiments. The precision, recall and f1-score and reported are of macro-average.

From the results (Table II) obtained on Amazon-2 dataset, we can compare the results before and after using ensembling methods.

TABLE II
EXPERIMENTAL RESULTS

Model	Accuracy	f1-score	precision	recall
CBOW	0.939	0.940	0.940	0.940
Skip-gram	0.943	0.942	0.943	0.943
GloVe	0.925	0.925	0.925	0.926
Google	0.932	0.932	0.932	0.933
ULMFiT	0.887	0.889	0.887	0.887
Max Voting	0.949	0.949	0.949	0.949
Average	0.949	0.949	0.950	0.949
Weighted Average	0.951	0.950	0.950	0.951

Unensembled models:

- We can observe that CBOW and Skip-gram models tend to outperform the models which used pre-trained embeddings. Based only on this text corpus, as we have learned CBOW and Skip-gram word embeddings using gensim [15] from scratch, the context of the reviews is better captured by the embeddings learned on the text.
- The time taken to train CBOW word embeddings was comparably less than learning Skip-gram based embeddings. As we have seen in the text representation section, training examples created for Skip-gram are 2k times (k is context size) that of CBOW training examples.
- With pre-trained word embeddings like GloVe or Google's Word2Vec, the context or the word embeddings is learned on more general text corpus like wiki and twitter. So these embeddings are more generalised to real word text than the specific context of reviews.
- We can notice that Skip-gram has given the highest accuracy of 94.3% followed by CBOW with 93.9% followed by other models which are pre-trained. We also noticed that the same performance is exhibited when f1-score, precision and recall are considered.

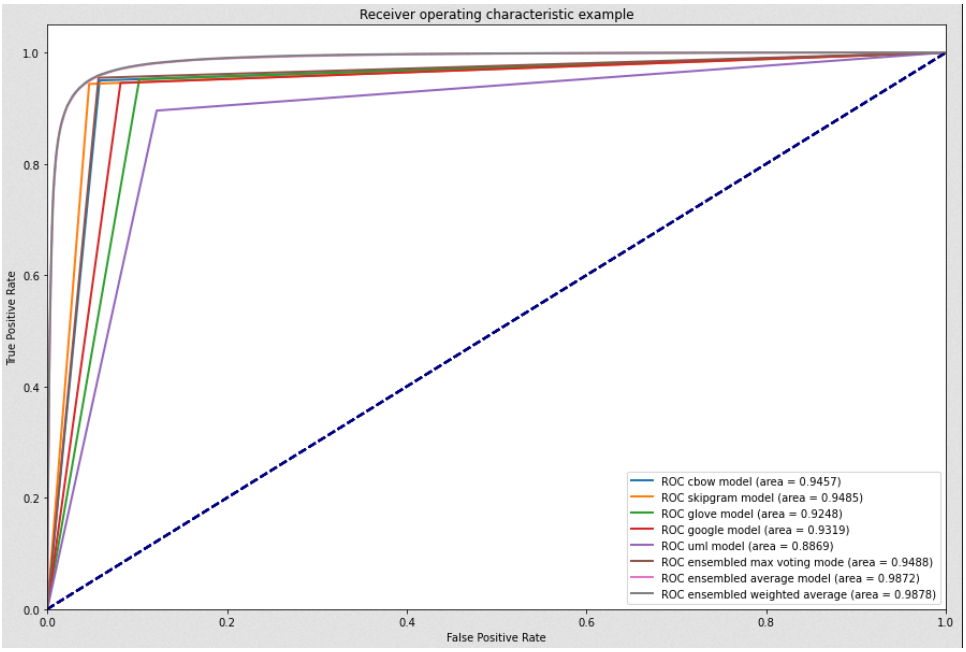

Fig. 8. ROC and AuC graphs for all models

- Computational complexity was experienced during fine-tuning pre-trained ULMFiT for the binary text classification task. In order to overcome this, we used only 1M training examples to fine-tune the ULMFiT model. Even with limited fine-tuning, the model achieved an accuracy of 88.7%. Better results can be obtained if the model can be fine-tuned with complete training examples, as ULMFiT is one of the state of the art model for text classification.

Ensembled models:

- We performed ensembling using the predictions of all five models trained for text classification. We have chosen simple but powerful ensembling methods like "Max Voting", "Averaging" and "Weighted average".
- All the ensembled methods used, outperformed each model when compared individually. We can notice that for the Weighted Average Ensembling method has achieved the highest accuracy of 95.1% and 95% f1-score and precision along with 95.1% of recall.
- Average and Max voting follow Weighted average in performance and achieve almost similar results in all metrics.

The Area under the Curve (AuC) is an indicator of a classifier's ability to differentiate between classes and is used as a ROC curve description. From the ROC-AuC plot in Figure 8 for all the models, we can see that Average and weighted average ensembling models are about 98.7% sure about the difference between positive and negative reviews. CBOW, Skip-gram, ensembled Max voting models can distinguish classes around 95%.

VII. Conclusion and Future Works

With the proposed method, i.e. using the advantages of context-specific learned embedding (task-specific knowledge) and pre-trained word embeddings (generalized knowledge) like GloVe and Google's Word2Vec for text classification and preforming ensembling on the predictions increased the accuracy of classification around 1%. The ensembled models have an AuC of around 99%, which is an increase of 4% on the individual models which shows that the model can distinguish between classes much better.

For future work, advanced ensemble learning methods like bagging, boosting, stacking can be used, which might give better results than the traditional ensemble methods used. ULMFiT can be trained on the complete training corpus for better results or an alternate transfer learning method like BERT or XLNET can also be used.

VIII. Appendix

Source code for the project is available in the below github, *https://github.com/sai-teja-ponugoti/Amazon-Reviews-for-Sentiment-Analysis*

References

[1] https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/
[2] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. Information, 10(4), 150.
[3] Xiao, Y., & Cho, K. (2016). Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. ArXiv, abs/1602.00367.
[4] Myska, V., Burget, R., Povoda, L., & Dutta, M. K. (2019, July). Linguistically independent sentiment analysis using convolutional-recurrent neural networks model. In 2019 42nd International Conference on Telecommunications and Signal Processing (TSP) (pp. 212-215). IEEE.

[5] Johnson, R., & Zhang, T. (2017, July). Deep pyramid convolutional neural networks for text categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 562-570).
[6] Mozafari, M., Farahbakhsh, R., & Crespi, N. (2019, December). A BERT-based transfer learning approach for hate speech detection in online social media. In International Conference on Complex Networks and Their Applications (pp. 928-940). Springer, Cham.
[7] Howard, J., & Ruder, S. (2018). Fine-tuned language models for text classification. arXiv preprint arXiv:1801.06146, 194.
[8] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
[9] Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
[10] Conroy, N. K., Rubin, V. L., Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology, 52(1), 1-4.
[11] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009.
[12] Mittos, A., Zannettou, S., Blackburn, J., & De Cristofaro, E. (2020, May). "And We Will Fight for Our Race!" A Measurement Study of Genetic Testing Conversations on Reddit and 4chan. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 14, pp. 452-463).
[13] Olteanu, A., Castillo, C., Boy, J., & Varshney, K. R. (2018). The effect of extremist violence on hateful speech online. arXiv preprint arXiv:1804.05704.
[14] Ottoni, R., Cunha, E., Magno, G., Bernardina, P., Meira Jr, W., & Almeida, V. (2018, May). Analyzing right-wing youtube channels: Hate, violence and discrimination. In Proceedings of the 10th ACM Conference on Web Science (pp. 323-332).
[15] RaRe Technologies. gensim: Topic Modelling for Humans, (GitHub repo). Last accessed June 15, 2020.
[16] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". Last accessed June 15, 2020.
[17] Mehdad, Y., & Tetreault, J. (2016, September). Do characters abuse more than words. In Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (pp. 299-303).
[18] Waseem, Z., & Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).
[19] Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., & Chang, Y. (2016, April). Abusive language detection in online user content. In Proceedings of the 25th international conference on world wide web (pp. 145-153).
[20] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.
[21] http://www.fakenewschallenge.org/
[22] https://www.oreilly.com/library/view/practical-natural-language/9781492054047/ch03.html
[23] Bowman, Samuel R., et al. "A large annotated corpus for learning natural language inference." arXiv preprint arXiv:1508.05326, 2015
[24] Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. O'Reilly Media.
[25] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D. (2019). Text classification algorithms: A survey. Information, 10(4), 150.
[26] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[27] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrasesand their compositionality.Adv. Neural Inf. Process. Syst.2013,26, 3111–3119
[28] Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedingsof the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar,25–29 October 2014; Volume 14, pp. 1532–1543
[29] Maaten, L.V.D.; Hinton, G. Visualizing data using t-SNE.J. Mach. Learn. Res.2008,9, 2579–2605.
[30] Way, M. (2012). Advances in machine learning and data mining for astronomy. Chapman and Hall/CRC.
[31] Bühlmann, P. (2012). Bagging, boosting and ensemble methods. In Handbook of Computational Statistics (pp. 985-1022). Springer, Berlin, Heidelberg.
[32] Gupta, G., & Malhotra, S. (2015). Text document tokenization for word frequency count using Rapid Miner (taking resume as an example). Int. J. Comput. Appl, 975, 8887.
[33] Verma, T., Renu, R., & Gaur, D. (2014). Tokenization and filtering process in RapidMiner. International Journal of Applied Information Systems, 7(2), 16-18.
[34] Johnson, R., & Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058.
[35] https://www.kaggle.com/bittlingmayer/amazonreviews
[36] https://deeplanguageclass.github.io/lab/fasttext-amazon/
[37] https://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/