An Approach Towards Enhancement Of Review Summarization Using Sequence Model And Word Embedding

Alka Londhe, Yash Deshpande, Gaurav Ghongde , Pratik Deshmukh, Hrushikesh Gate

Department of Computer Science, PCCOE, Pune, India

e-mail: alka.londhe.pccoe@gmail.com, yashd299792.458@gmail.com, 7gaurav.ghongde@gmail.com, pratik.desh.2015@gmail.com, hrushigate7@gmail.com

***Abstract*—** Traditionally, extraction based text summarization techniques have simply relied upon statistical measures for extraction of major points in a document. With popularization of deep learning in solving problems in the field of Natural Language Processing, there is a need to incorporate deep learning to improve the effectiveness of summarization methods. In this paper, we explore the results and findings of our approach to improve review summarization as well as to delineate the capabilities of these language models in improving Natural Language Understanding. The aim of our summarization approach is to generate a gist of customers' feedback of a product which would provide vendors and other customers with the necessary understanding in order to make informed purchases. Our proposed summarization approach is split into four major steps: Keyword Extraction, Sentiment Analysis, Adjective Mining and Sentence Extraction. Afterwards, all the outputs of these steps are combined to provide the user with a summary of all reviews about a product.

***Keywords*—** Review Summarization, Sequence Model, Word Embedding, Natural Language Processing, Sentiment Analysis

I. INTRODUCTION

With increase in the number of products and services provided by various sites on the internet, the amount of feedback or reviews about those products and services has also increased manifold. The textual data generated by these reviews contains valuable information which we aim through summarization extract out the text in order to gain insights on products. Earlier, since the textual data was limited analyses of these reviews were done either manually or by applying statistical analysis but with the popularization of deep learning in Natural Language Processing we can intelligently go through the textual data to make highly accurate hypotheses. In this project our aim is to build upon the currently existing solutions in review summarization to improve the accuracy of predictions using deep learning techniques. We have divided our task of summarization into four broad subproblems: Keyword Extraction, Sentiment Analysis, Adjective Mining and Sentence Extraction. We have used deep learning to improve each of the subproblems with positive results to show.

II. METHODOLOGY

1. Keyword Extraction:

Keyword Extraction is an extensively explored field of research in Natural Language Processing since keywords provide highly dense summary of a document leading to improvement in information retrieval. The most conventional and potent method for extracting keywords in a document is TF-IDF. Term Frequency - Inverse Document as the name suggests generates a numeric value for each and every word in all documents based on its frequency in the documents through the logarithm of inverse fraction of the documents that contain the word. [2]

Keywords in our case consist of noun or combination of nouns so we need to filter out words that are nouns from the text and apply TF-IDF on nouns only. In order to do so, we need to assign parts of speech to each word in the corpus. Part-of-speech tagging involves assigning every word in a corpus with a part of speech label such as Noun, Verb Adjective, Adverb, Preposition, Conjunction, Pronoun and Interjection. We make use of SpaCy's Part-of-speech tagger which uses a statistical model to make rule-based prediction of the tag corresponding to the word in a particular context.

1.1. Approach I:

TF-IDF is a relatively simple and efficient approach towards keyword extraction but it has its limitation as correctly stated in [1]. It fails to take into account the synonyms, plurals and relationship amongst words. For example, the keywords: "

earphones" and "earpods" basically refer to the same thing. But TF-IDF as a measure fails to take that into account.

Also, TF-IDF works on individual words or unigrams but our keywords may or may not be restricted to a single word. Our approach aims to resolve both of the above mentioned limitations by putting forward two separate solutions:

First, synonyms and semantic relationship between words are important to identify since considering these words as separate entity would lead to generation of redundant keywords. Such words should be identified and each and every iteration of these words in the document has to be replaced by any one of the words. In order to identify semantically similar words we use Word2vec to generate word embeddings. [3]

Word2Vec is a group of shallow neural network models used to generate word embeddings. Word2vec is trained on a huge corpus of textual data to generate vector space. Wherein, each word is represented by a vector having multiple dimensions typically in hundreds in the vector space. These word vectors are aligned in such a way that words which are semantically or syntactically similar lie in proximity to each other.

This property of word vectors is of immense importance to us since it allows to group similar words using measures like cosine similarity which use distance between vectors to compute similarity.

After extracting top keywords using TF-IDF we compute similarity of of each word with every other word using cosine similarity in percentages. After setting various thresholds for similarity, we found that setting a threshold of 50% produced optimum results. Hence, we eliminated redundant keywords from our original set of keywords obtained by TF-IDF.

Second, as a consequence of applying TF-IDF on individual words we miss out on keywords that contain multiple words such as "fingerprint scanner". In order to resolve this problem we generate n-grams in the text corpus. Language models assign probabilities to words in sequences, higher the co-occurence of words, higher probability is assigned to them. In ngrams, such a set of n words with high co-occurrence are selected to generate new word tokens. We have limited n-grams to unigrams, bigrams and trigrams since keywords rarely exceed three words.

Probability of words: w1w2,....wn occurring together:

$$P(w_1, w_1, ...w_n) = P(w_n \mid w_{n-1}, w_{n-2}, ...w_1)\, P(w_{n-1} \mid w_{n-2}, w_{n-3}, ...w_1) \ldots P(w_2 \mid w_1)\, P(w_1) \quad (1)$$

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k^{k-1}{}_{N \mid 1}) \quad (2)$$

1.2. Approach II (Supervised Approach) :

In supervised approach, we manually generate a labelled dataset containing four fields as shown in figure 1. The tag field is the outcome variable containing two classes: 'O' and 'F'. Here, 'O' denotes that the word isn't a keyword whereas 'F' denotes that it is a feature. We use sequential model in order to predict the tag of words in unseen sentences from reviews. We use a category of attention based sequential models known as bidirectional LSTM to make predictions.

LSTM is a special type of gated recurrent neural network which has the special property of remembering inputs over a long period of time since LSTMs can read, write and delete inputs from its memory. It does so with the help of three gates: input, forget and output gate. Input gate determines whether to let new input in, forget gate determines whether to delete inputs which aren't important, and output gate is responsible for producing the output of the gate. This property comes in handy in this problem since sentences in review can range from 10 words to all the way up 60 words, so remembering each and every word would reduce the efficacy of the model. These models have proved to be extremely effective in similar types of problems such as Named-Entity-Recognition.

Firstly, words cannot be fed in text format directly as inputs to the neural network. We enumerate all words in the vocabulary in order to assign a numeric value to each word and replace words in the sentences with their numeric representations. The limitation of RNNs is that all inputs should be of the same size, that is, all sentences should be of the same size. In order to make all sentences of the same size we pad sentences with tag 'O' such that all sentences are of the same length. Finally, we train a bidirectional LSTM model on the dataset which predicts keywords in sentences.

Table 1. Dataset #1

Row #	Sentence #	Word	POS	Tag
0	Sentence: 1	After	IN	O
1	Sentence: 1	losing	VBG	O
2	Sentence: 1	my	PRP$	O
3	Sentence: 1	iPhone7	NN	F
4	Sentence: 1	EarPods,	NNP	O
5	Sentence: 1	I	PRP	O

6	Sentence: 1	bought	VBD	O
7	Sentence: 1	these	DT	O
8	Sentence: 1	replacements	JJ	O
9	Sentence: 1	my	PRP$	O
10	Sentence: 1	fingers crossed	NNS	O
11	Sentence: 1	that	IN	O
12	Sentence: 1	these	DT	O
13	Sentence: 1	would	MD	O
14	Sentence: 1	be	VB	O
15	Sentence: 1	legit	JJ	O

2. Sentiment Analysis:

Sentiment analysis is the process of determining the opinion or attitude of the writer(s) towards an entity, topic or product, these sentiments are classified into one of three categories: Positive, Negative or Neutral. In our work we aim to determine:

i) The sentiment of reviewers towards keywords extracted in the previous step

ii) Overall summarized sentiment of all reviews towards the product.

2.1. The sentiment of reviewers towards keywords extracted in the previous step:

To determine the sentiment of a particular keyword we employ a technique known as Aspect Based Sentiment Analysis (ABSA). ABSA differs from sentiment analysis since sentiment analysis focuses on computing the sentiment of an entire text whereas ABSA focuses on specific aspects of a product or a service. ABSA provides an insightful and precise account of a product compared to sentiment analysis.

Our ABSA model aims to accomplish two major objectives:

- To design and implement a deep learning model i.e. LSTM framework that extracts aspects as well as the sentiments pertaining to the aspect. For example, “The phone is good, but the camera is bad”. Here “phone” and “camera” are aspects, “phone” has a positive sentiment whereas “camera” has a negative sentiment.

- Improve accuracy compared to the baseline model [4]

Since ABSA was first proposed in 2010, there have been many approaches to improve to tackle ABSA using deep learning. We particularly focus on [2] and take it as a reference for our model. The major difference being that they use CNN for extraction of aspects and associated sentiment whereas we use sequential model for the same purpose and we limit our scope to In-domain ABSA. Also, we skip the part of aspect prediction since we have already derived keywords a.k.a aspects in previous section.

Dataset:

We take a dataset of 2107 reviews of laptops, containing four fields: “aspect_term” which refers to the aspect, “polarity” which may be positive, negative or neutral, “text” which contain sentence level reviews, and “category” which categorizes “aspect_term” into categories of laptops components.

Table 2. Dataset #2

Row #	aspect_term	polarity	text	category
1	computer	positive	This computer is absolutely AMAZING!!!	laptop
2	battery	positive	10 plus hours of battery...	battery
3	processor graphics card	positive	super fast processor and really nice graphics ...	processor graphics
4	storage ram	positive	and plenty of storage with 250 gb(though I wil...	hard disk memory
5	PORTABILITY PROCESSING	positive	GET THIS COMPUTER FOR PORTABILITY AND FAST PRO...	weight processor
6	portability longevity use	positive	As a computer science student in college, I fi...	weight laptop laptop
7	spotlight search	positive	A great feature is the spotlight search: one c...	os
8	HP	negative	My HP is very heavy.	laptop

2.1.1. Implementation:

1. Used SpaCy nlp pipe to extract sentiment words from each text review and generate a separate field named “‘sentiment_words” out of it.
2. Using keras in-built tokenizer to convert words from review into tokens understandable to our sequential model.
3. The vocabulary size is limited to 6000, and added two layered neural network with activation

functions ‘relu’ and ‘softmax’ respectively over 10 epochs.

4. Predictions are made based on sentiment words extracted from the review_text with better model accuracy as as shown in Figure 1

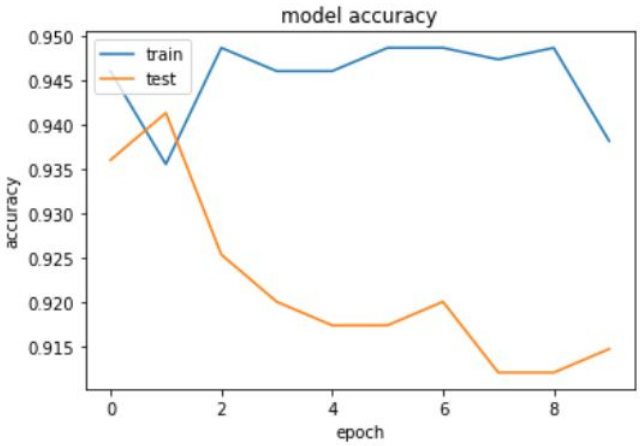

Figure1: Aspect Based Sentiment Model Accuracy

2.B. Overall summarized sentiment of all reviews towards the product:

In order to summarize the overall sentiment about the product we ought to find the sentiment of each and every review and find the meaning of it to determine the overall sentiment. On some review sites, customers are already provided with a rating system that indicates their sentiment towards reviews, but this facility might not be extended to other feedback systems. To train a model to classify reviews we use Amazon product data created by Julian McAuley, UCSD. We train our model on reviews to predict sentiment of the review which can be positive or negative. Firstly, we replace review text with tf-idf values using tf-idf vectorizer. Afterwards, we apply logistic regression for the purpose of classification, and we have achieved very good results.

Table 3. Metrics of Dataset #2

Metric	Score
Accuracy	0.921076322896869
Precision	0.9143582476563517
Recall	0.921076322896869
F-measure	0.9168427955119971

3. Adjective Mining

The drawback of sentiment analysis is that it gives us the opinion of an author in terms of positive, negative or neutral opinion on an aspect or a product, but it fails to take into account opinions which are descriptive of an aspect. For example, “the screen size is small”. Here, for the aspect “screen size” the author’s opinion is that the screen size is “small”. Our earlier sentiment analysis model would not be able to take it into account, and such descriptive adjectives are essential since they provide additional insight to us.

There are three tasks we have to accomplish in order to perform adjective mining:

- Extract the adjective which describes the keyword. To accomplish this, we take the help of SpaCy dependency parser. It provides POS tagging of each and every word along with its dependency.
- Only extracting adjective doesn’t suffice, since “good” and “extremely good” are two different opinions varying in intensity.
- Form a word cloud depicting frequencies of adjectives

3.1 Algorithm

```
set adjective = ""
set adjective2 = ""
sample_adjectives = []
for i in range(len(dependencies)):
    if (dependencies[i][0] equals search_noun):
        set j=i
        set k=i
        set dist1=0
        set dist2=0
        set flag1=0
        while(j greater than or equal to 0):
            if(dependencies[j][1] equals 'ADJ'
            and (dependencies[j][2] equals 'amod'
            or dependencies[j][2] equals 'conj')):
                set adjective += " "
                              + dependencies[j][0]
                set flag1=1
            if(dependencies[j-1][1] equals 'ADV'):
                print("adv:",dependencies[j-1][0])
                set adjective = dependencies[j-1][0]
                             + " " + adjective
                break;
            decrement j
            set dist1 = dist1+1
        if(flag1 equals 0):
            set dist1 = infinity

        while(k less than len(dependencies)):
            if(dependencies[k][1] equals 'ADV'):
                set adjective2 = adjective2
                               + dependencies[k][0]
                               + " "
            if(dependencies[k][1] equals 'ADJ'
            and (dependencies[k][2] equals 'acomp'
            or dependencies[k][2] equals 'attr'
            or dependencies[k][2] equals 'advmod' )):
                if(dist2 less than dist1):
                    set adjective2 += " "
                                   + dependencies[k][0]
                    set adjective = adjective2
                break;
            increment k
            increment dist2
        sample_adjectives.append(adjective)
        adjective = ""
        adjective1 = ""
```

```
print("adjectives=", sample_adjectives)
```

4. Summarization

Instead of directly performing summarization of all review, we instead try to perform summarization of all phrases which contain the keywords that were extracted in the first step since our interest is in what reviewers think about various aspects of the product.

Broadly speaking, summarization is of two types: extractive summarization and abstractive summarization. Abstractive summarization requires the use of sequential models for generating summaries. The amount reviews we take as our input runs into thousands, which in turn means that there would be a large amount of words i.e. large input to the neural network and it also has to take into consideration the sequence of the words. Since neural networks are not very effective in dealing with such constraints, the obvious choice is to go for extractive summarization.

Our aim with extractive summarization is to extract sentences which retain maximum opinion about features and at the same discards irrelevant or spam opinions.

For example, “battery is good”, “phone and battery are both excellent”, “battery is good”. Our model should pick the second choice since it retains maximum opinion.

We have implemented extractive summarization using the TextRank method. TextRank is an extractive, unsupervised summarization technique which is inspired by the page rank algorithm. [5]

TextRank implementation involves the following steps:

1. Extracting sentence containing keywords from all reviews
2. Generate vectors for each and every sentence
3. Generate similarity matrix based on similarity between sentences
4. Convert similarity matrix into a graph where vertices are sentences and weights are similarity score.
5. Apply page rank algorithm on the graph
6. Select top sentences

Conventionally, we generate vectors in step two using countvectorizer or tf-idf vectorizer, but these methods do not understand the semantic and syntactic similarity between words in sentences, ergo, making them less effective. Instead we use Google’s Universal Sentence Encoder which are models for encoding sentences into embedded vectors. [6] These embeddings have been proven to be highly effective in spotting semantic textual similarity between sentences.

III. Conclusion and Future Scope

In this work, all the four steps in our review summarization approach are implemented in the form of a review summarization tool which provides intelligent understanding of textual data i.e. customer reviews to provide comprehensive and multi-faceted understanding of products to the end user. Use of deep learning and unsupervised methods has allowed for a generalized model which works well for cross-domain reviews. In future, we are interested in making a generalized tool which would work for reviews not only confined to electronics or ecommerce but also for other domains such as Yelp reviews or TripAdvisor.

Acknowledgment

We would like to thank the Computer Engineering department of Pimpri-Chinchwad College of Engineering for their support and guidance which enabled us in conducting our research.

References

[1] TFIDF: Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. **Vol. 242. 2003**.

[2] Keyword extraction: Hulth, Anette. "Improved automatic keyword extraction given more linguistic knowledge." Proceedings of the 2003 conference on Empirical methods in natural language processing. Association for Computational Linguistics, **2003**.

[3] Word Embedding: Mikolov, Tomas, et al., "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. **2013**.

[4] ABSA: Wang, Bo, and Min Liu. "Deep learning for aspect-based sentiment analysis." Stanford University report (**2015**).

[5] Textrank: Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." Proceedings of the 2004 conference on empirical methods in natural language processing. **2004**.

[6] Sentence embedding: Cer, Daniel, et al. "Universal sentence encoder." arXiv preprint arXiv:**1803.11175 (2018).**