KATHMANDU UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

ON

TEXT SUMMARIZATION

A Final year mini project report submitted in partial fulfillment of the requirements for COMP 473

Submitted By:

Rajshree Rai (40)

Prerana Khatiwada (21)

Submitted To:

Mr. Bal Krishna BalSubmitted By:

Date of Submission : 16th June 2019

ABSTRACT

In this new era, where tremendous information is available on the internet,it is most important to provide the improved mechanism to extract the information quickly and most efficiently . It is very difficult for human beings to manually extract the summary of a large documents of text. There are plenty of text material available on the internet. So there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it .In order to solve the above two problems, the automatic text summarization is very much necessary.Text summarization is the process of identifying the most important meaningful information in a document or set of related documents and compressing them into a shorter version preserving its overall meanings

INTRODUCTION

Text Summarization methods can be classified into extractive and abstractive summarization. An Abstractive summarization is an understanding of the main concepts in a document and then express those concepts in clear natural language.
This project performs abstractive summarization. In this project, we have used fine food reviews from Amazon to build a model that can summarize text. Specifically, we used the description of a review as our input data, and the title of a review as our target data. Our dataset was downloaded from kaggle. We built a seq2seq model is quite a bit different than with previous versions. To help generate some great summaries, we used a bi-directional RNN in our encoding layer, and attention in our decoding layer.

OBJECTIVES

The main objective of a text summarization system is:-
i)To identify the most important information from the given text and present it to the end users.
ii)To present the source text into a shorter
iii)To reduce the reading time.
iv)To build a model that can create relevant summaries for reviews written about fine foods sold on Amazon.

SYSTEM DESIGN

Programming Language and Libraries

Python: The project has been implemented in the object oriented programming language python.

Tensorflow:It is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

Pickle:It is used for serializing and de-serializing Python object structures, also called marshalling or flattening.

NumPy:It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas:It is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Re:Python has a built-in package called re, which can be used to work with Regular Expressions.

METHODOLOGY

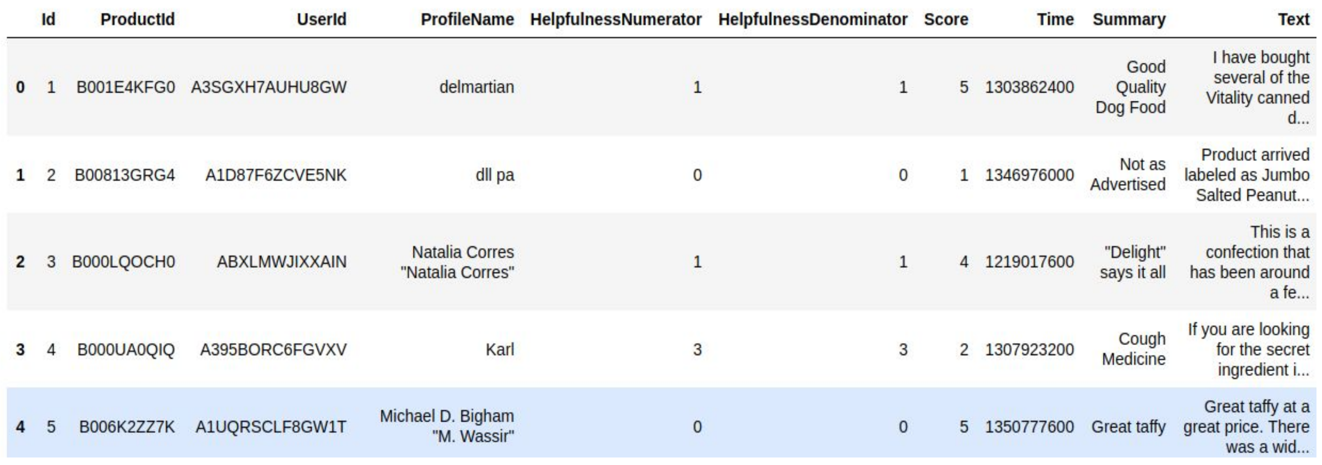
The sections of this project are:

1.Inspecting the Data

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all 500,000 reviews.Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.There are altogether 10 columns in its csv file including Id,ProductID,UserID,Profile Name,Helpfulness numerator,Helpfulness Denominator.

1. Id: Row Id
2. ProductId :Unique identifier for the product
3. UserId: Unique identifier for the user
4. ProfileName: Profile name of the user
5. HelpfulnessNumerator: Number of users who found the review helpful
6. Helpfulness Denominator: Number of users who indicated whether they found the review helpful or not
7. Score: Rating between 1 and 5
8. Time: Timestamp for the review
9. Summary: Brief summary of the review
10. Text: Text of the review

    We manipulate the dataset to only achieve the “Summary” and “Text” columns.



	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

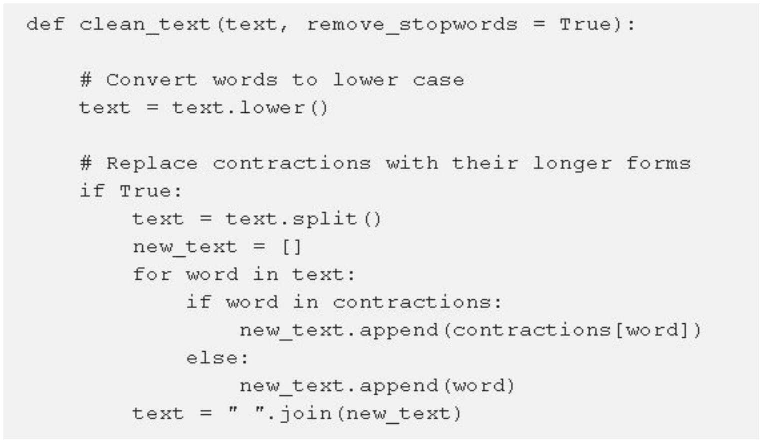
Figure1: Dataset

2.Preparing the Data

It comprises of:-

i)Convert to lowercase.

ii)Replace contractions with their longer forms.



```
def clean_text(text, remove_stopwords = True):

    # Convert words to lower case
    text = text.lower()

    # Replace contractions with their longer forms
    if True:
        text = text.split()
        new_text = []
        for word in text:
            if word in contractions:
                new_text.append(contractions[word])
            else:
                new_text.append(word)
        text = " ".join(new_text)
```

iii)Remove any unwanted characters

This is done after replacing the contractions because apostrophes will be removed. We used regular expressions to find and substitute characters such as links, html tags that were present in the dataset.

iv)Removal of stop words.

Removal of stop words helps the model in two ways:

i)Irrelevant words such as functional words are removed which helps make the dataset relevant.

ii) The dataset size is reduced and thus the model trains faster

v) Embeddings

We have used pre-trained word vectors to help improve the performance of our model. For the set of word embeddings, ConceptNet Numberbatch (CN) is used. Embeddings are required as our model does not except categorical values.

```
embeddings_index = {}
with open('/Users/Dave/Desktop/Programming/numberbatch-en-
17.02.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split(' ')
        word = values[0]
        embedding = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = embedding
```

vi)Building Word Embedding Matrix

We are going to limit our vocabulary to words that are either in CN or occur more than 20 times in our dataset. This will allow us to have very good embeddings for every word because the model can better understand how words are related when they see them more times.

```
embedding_dim = 300
nb_words = len(vocab_to_int)

word_embedding_matrix = np.zeros((nb_words, embedding_dim),
                                 dtype=np.float32)
for word, i in vocab_to_int.items():
    if word in embeddings_index:
        word_embedding_matrix[i] = embeddings_index[word]
    else:
        # If word not in CN, create a random embedding for it
        new_embedding = np.array(np.random.uniform(-1.0, 1.0,
embedding_dim))
        embeddings_index[word] = new_embedding
        word_embedding_matrix[i] = new_embedding
```

vii)Sorting the reviews

To help train the model faster, we are going to sort the reviews by the length of the descriptions from shortest to longest. This will help each batch to have descriptions of similar lengths, which will result in less padding, thus less computing.

viii)Filtering the reviews

Some reviews will not be included because of the number of UNK tokens that are in the description or summary. If there is more than 1 UNK in the description or any UNKs in the summary, the review will not be used. This is done to ensure that we are building the model with meaningful data. Fewer than 0.7% of words are UNKs, so not many reviews will be removed.

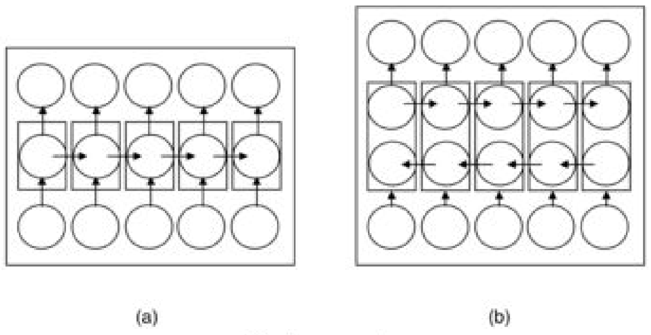
3.Building the Model

Our model is the sequence to sequence model with the following parts:

- Encoder:

  The encoder layer is a stack of recurrent units. It contains 2 LSTM layers with one bidirectional RNN. It accepts the input sequence (the reviews) and computes the hidden state. We use dropout layer to avoid overfitting the model. Recurrent Neural Network handles sequential data and uses reasoning about previous event to inform later ones. It memorizes parts of the input to make accurate predictions.

  Bidirectional RNN connects 2 hidden layers of opposite direction to the same output i.e. information from the past and the future.



Structure overview
(a) unidirectional RNN
(b) bidirectional RNN

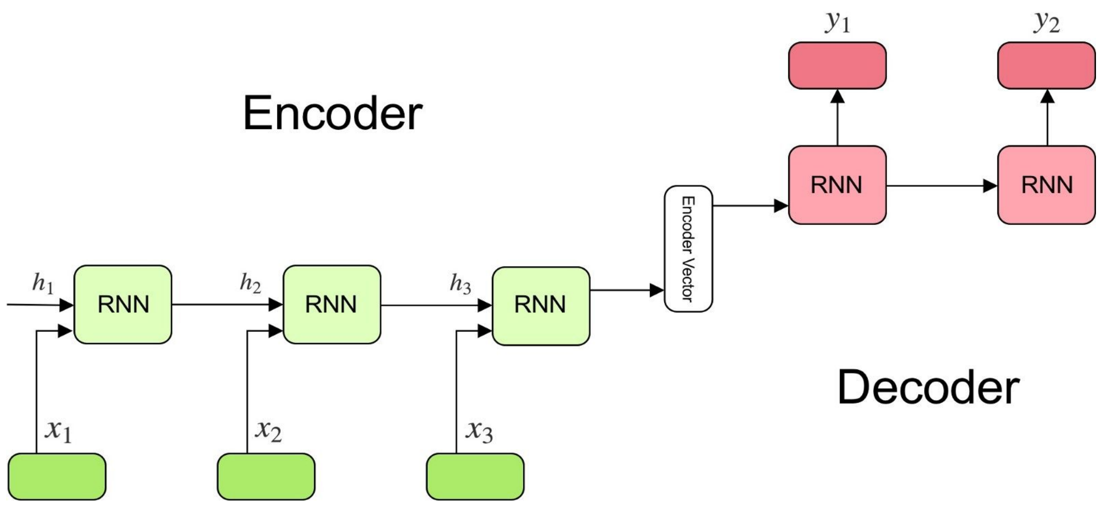
  LSTM is a special kind of RNN that remembers long term dependency.

- Decoder

  The decoder layer is also a stack of recurrent units which predicts the output (the summary).

- Attention

  A limitation of encoder decoder is that it encodes the input sequence to a fixed internal representation. This imposes a limit on the length of input sequences that can be learnt from the model. Attention helps overcome this limitation. Attention helps the neural network learn the positions of the source input to focus its attention on. Each time the model generates a word, attention searches for positions in the source sentence where the most relevant information is concentrated. It relates each word in the output summary to specific words in the input.



4.Training the Model

The model was trained only with a subset of 50,000 reviews.To help train the model faster, we sorted the reviews by the length of the descriptions from shortest to longest such that each batch will have descriptions of similar lengths, which will result in less padding, thus less computing.

Some of the Hyperparameters used to train the model are

```
epochs = 100
batch_size = 64
rnn_size = 256
num_layers = 2
learning_rate = 0.01
keep_probability = 0.75
```

The following figure shows that the loss function are decreasing.

```
Epoch  41/100 Batch  660/781 - Loss:  0.449, Seconds: 26.12
Epoch  41/100 Batch  680/781 - Loss:  0.377, Seconds: 26.69
Epoch  41/100 Batch  700/781 - Loss:  0.410, Seconds: 27.97
Epoch  41/100 Batch  720/781 - Loss:  0.438, Seconds: 28.15
Epoch  41/100 Batch  740/781 - Loss:  0.482, Seconds: 26.32
Epoch  41/100 Batch  760/781 - Loss:  0.443, Seconds: 28.68
Average loss for this update: 0.436
New Record!
Epoch  41/100 Batch  780/781 - Loss:  0.480, Seconds: 27.91
Epoch  42/100 Batch   20/781 - Loss:  0.532, Seconds: 27.54
Epoch  42/100 Batch   40/781 - Loss:  0.442, Seconds: 27.98
Epoch  42/100 Batch   60/781 - Loss:  0.435, Seconds: 25.91
Epoch  42/100 Batch   80/781 - Loss:  0.454, Seconds: 26.79
Epoch  42/100 Batch  100/781 - Loss:  0.440, Seconds: 26.90
```

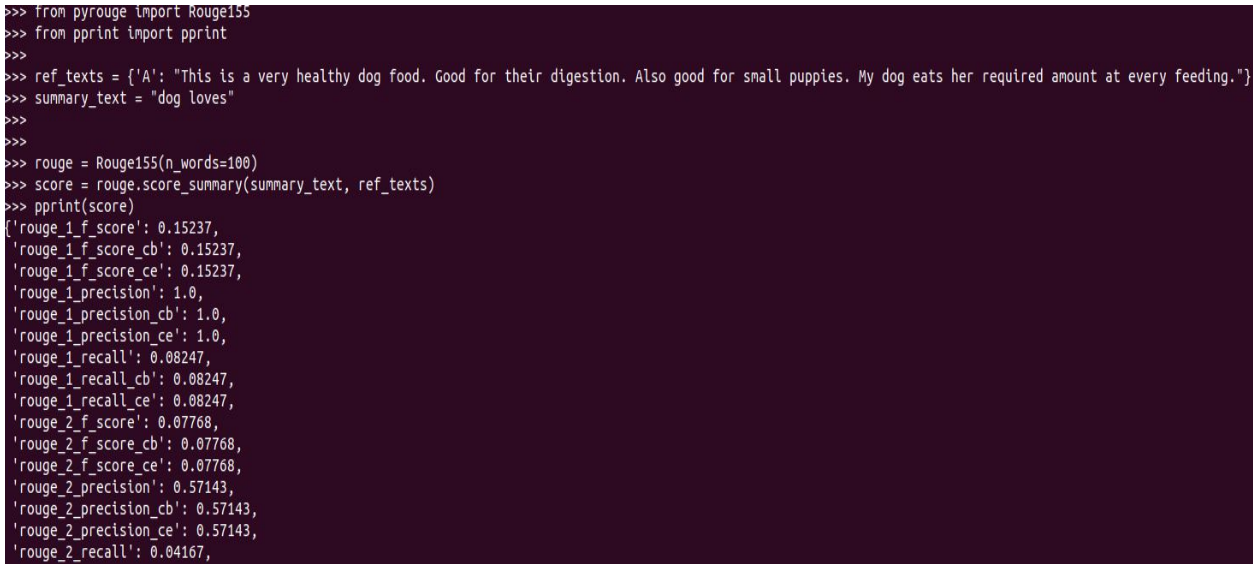
5.Making Our Own Summaries

We are able to either create our own descriptions or use one from the dataset as our input data.To generate new summaries, we need to load quite a few more tensors.

6. Evaluation of the results

To measure the quantitative assessment of the summary the ROUGE evaluator tool is used which consist of precision, recall and F-measure.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially of a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced).Let us say, we have the following system and reference summaries:



```
>>> from pyrouge import Rouge155
>>> from pprint import pprint
>>>
>>> ref_texts = {'A': "This is a very healthy dog food. Good for their digestion. Also good for small puppies. My dog eats her required amount at every feeding."}
>>> summary_text = "dog loves"
>>>
>>>
>>> rouge = Rouge155(n_words=100)
>>> score = rouge.score_summary(summary_text, ref_texts)
>>> pprint(score)
{'rouge_1_f_score': 0.15237,
 'rouge_1_f_score_cb': 0.15237,
 'rouge_1_f_score_ce': 0.15237,
 'rouge_1_precision': 1.0,
 'rouge_1_precision_cb': 1.0,
 'rouge_1_precision_ce': 1.0,
 'rouge_1_recall': 0.08247,
 'rouge_1_recall_cb': 0.08247,
 'rouge_1_recall_ce': 0.08247,
 'rouge_2_f_score': 0.07768,
 'rouge_2_f_score_cb': 0.07768,
 'rouge_2_f_score_ce': 0.07768,
 'rouge_2_precision': 0.57143,
 'rouge_2_precision_cb': 0.57143,
 'rouge_2_precision_ce': 0.57143,
 'rouge_2_recall': 0.04167,
```

For the review of:
"This is a very healthy dog food. Good for their digestion. Also good for small puppies. My dog eats her required amount at every feeding."
And the summary:
"dog loves"
We get a precision score of 1 and recalls score of 0.08 with an f score of 0.15 for unigram evaluation by rouge.

7. Limitations

The following are the limitations in our project:

- A low recall value in comparison to the high precision value in rouge evaluation.
- The short length of the generated summary.
- The summaries are only trained on food reviews and not extended to other diverse dataset.

8. Future Improvements

The following improvements can be made in the project in the coming future:

- Extending the project to summarize other diverse data that do not focus only on reviews.
- A better user interface for the project.
- A better recall value through model improvements.