Abstractive Summarization

A Project Report

Submitted by:

Daivat Bhatt (201501023)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION AND COMMUNICATION TECHNOLOGY (ICT)

School of Engineering and Applied Sciences (SEAS)

Ahmedabad, Gujarat

May, 2019

DECLARATION

I hereby declare that the project entitled “Abstractive Summarization” submitted for the B. Tech. (ICT) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Signature of the Student

Place: School of Engineering and Applied Science, Ahmedabad University

Date: 07th May 2019

CERTIFICATE

This is to certify that the project titled “Abstractive Summarization” is the bona fide work carried out by Daivat Bhatt, a student of B Tech (ICT) of School of Engineering and Applied Sciences at Ahmedabad University during the academic year 2018-19, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Information and Communication Technology) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Signature of the Guide

Place: School of Engineering and Applied Science, Ahmedabad University

Date: 07th May 2019

ABSTRACT

Abstractive Summarization is one of the most important problems in Natural Language Processing currently. It essentially aims to read through a document, find the most relevant passages to the topic that the document talks about, and then generate an abstract. The abstract must be novel, and cannot be simply generated using text from the document verbatim. There have been attempts to use neural nets to generate abstracts, but none have been satisfactory. The current project aims to tackle this problem of Abstractive Summarization using a Recurrent Neural Network with the help a Sequence-to-Sequence model, an Encoder-Decoder architecture and a global attention model. Finally, the Beam Search decoder is used to find the best possible title of the description. The project uses the Keras library provided by TensorFlow. The model was able to generate short-length headlines after being run and trained on a set of newspaper articles and corresponding headlines. The typical training loss of the system was seen to reduce as the number of epochs increased.

ACKNOWLEDGEMENT

The student would like to thank his mentor, Dr. Mitaxi Mehta for her guidance and support throughout the project. The student would also like to thank Dr. Sanjay Chaudhary for his timely help at various stages of the project. The student is grateful to School of Engineering and Applied Science, Ahmedabad University for providing adequate resources in the form of a computer system, internet connectivity and other such for the project.

The student is also grateful to Dr. Ratnik Gandhi for his guidance and constant support in the previous semester.

LIST OF FIGURES

GANTT CHART

TASK NAME	START DATE	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	PERCENT COMPLETE
Abstractive Summarization						
Literature Review	1/9	1/21	13	13	0	100%
Introduction to necessary libraries and programming	1/15	1/29	15	15	0	100%
Working on possible approach to the problem	1/29	2/20	23	23	0	100%
Gathering corpus, Coding	2/16	3/15	28	28	0	100%
Prototype for Architecture and Algorithm and Replication of Results	3/16	4/20	36	36	0	100%
Review of design and improvements	4/21	4/26	6	3	3	50%

Table of Contents

1. INTRODUCTION

Abstractive Summarization is one of the most important and relevant problems in the current world. In this era of excessive data in the form of text, it has become increasingly important to find methods of representing, understanding and contextualizing text that has been created by humans in the first place. Natural Language Processing largely functions in the domain of understanding and representing textual data. It as become the need of the hour to find methods through which text can be easily understood by a machine, and then various tasks could be automated and carried out by the machine itself. By far, machines dealing with text has been a difficult problem to tackle simply because of the various syntactic and semantic meanings and rules that a language carries with itself, notwithstanding the fact that a language is not discrete like numbers are and therefore first has to be represented some or the other way so that a machine is able to access that information.

This project deals with the part of representing and contextualizing textual information in a machine and then generating relevant and semantically correct text from this corpus of data. Abstractive Summarization is precisely that. It reads human-generated data and finds important and relevant information from the data to generate an abstract-sized summary of this data. The corpus that stores the human-generated data may range from newspaper articles to opinion pieces to product or movie reviews to most any other text. The fact of the matter is that, like a child, a machine must be first taught the ways in which language functions, how grammar forms an underlying and important basis for it. The following sections will talk about the problem in detail, and how this project aims to tackle it.

1.1 PROBLEM DEFINITION

According to Radev et al. [1] a summary is defined as “a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that”. Automatic text summarization is very challenging, because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task. [2]

In general, there are two different approaches for automatic summarization:

a. Abstractive Summarization
b. Extractive Summarization

Extractive Summarization, as the name suggests, uses the text that is already in the document in question, and then *extracts* relevant data from the text verbatim. Although, as part of this project – no type of implementation regarding Extractive Summarization has been carried out, multiple papers regarding the same have been read.

Abstractive Summarization, on the other hand is different in the sense that it goes a step further and *generates* sentences that are similar in meaning to the original, bigger corpus and yet shorter in length. This, as you can sense, is closer to when a human would write a short summary about any document they have just read. This project has aimed to implement an Abstractive Summarization procedure using Neural networks and a Sequence-to-Sequence model.

Most of the research carried out has focused on extractive summarization. This is because abstractive summarization methods have to cope with problems such as semantic representation, inference and natural language generation which are relatively harder than data-driven approaches.

1.2 HARDWARE SPECIFICATION

The project was run on a desktop computer in Lab 211: Big Data and Computing Lab at School of Engineering and Applied Science, Ahmedabad University. The PC had an 8GB RAM, an Intel i7 octa-core processor with a processing power at 3.40GHz. The total external memory was 1TB.

1.2 SOFTWARE SPECIFICATION

The system used had Ubuntu 18.04 LTS installed. It ran on a Jupyter Notebook. The entire code was written in Python v3.6.7. The model was trained with the following python3 libraries installed:

a. Numpy
b. Keras
c. scikit-learn
d. tensor-flow (v1. 0)

The model also had to make use of the GloVe (Global Vectors) [3] embeddings available for text representation.

2. LITERATURE SURVEY

The field of Natural Language Processing is bursting with research and ideas. Be that using Generative Adversarial Networks (GANs) to generate natural language [4], Deep Learning to generate dialogues [5], neural machines to translate one natural language into another [6] or different methods of Abstractive Summarization itself. As a result, the topic provided me ample scope to learn about natural language, and what could be a good problem to work upon. My choice of Abstractive Summarization was more to do with the fact as to how language and text can be manipulated such that it can become understandable. I was more interested in preserving the semantic meaning of the language rather than working on the representation of language in general. This is because there are multiple types of embeddings available [3], [7] that are extensively used. I had worked with GANs [8] and was uncertain whether they could be used for text generation. GANs have been used to work extensively on images and a lot of variants of the same exist. As a result, the next problem I landed upon was Abstractive Summarization. Abstractive Summarization is an interesting problem for even a lot of humans would disagree upon what is an ideal "summary". Thus, there is no real metric of deciding whether the generated summary – especially when new data with no corresponding value to make qualitative measurements is offered.

At the beginning of the literature survey for the project, I had come across two methods that existed to tackle this problem. One of them used neural nets (which I have implemented), and the other used graphs and represented text through nodes and edges to embed meaning and then generate sentences. Both of these will be discussed in the sections below.

2.1 EXISTING SYSTEMS

As I mentioned above, Abstractive Summarization has been tackled broadly through two methods: Neural Networks and Graphs.

The Neural Network system usually trains on a set of documents and their summary linked together. I was able to obtain the annotated, tokenized Gigaword English dataset that consisted of a newspaper article-pair that was split into training-test set. [21] The graph system worked on the semantic representation concept called “Abstract Meaning Representation” [10]. These structures were quite simply put, semantic structures in the form of rooted, directed, edge-labeled and leaf-labeled graphs. These graphs were a machine-readable form of the relationships between different parts-of-speech of the textual data.

2.1.1 Neural Networks for Abstractive Summarization

The Neural Model for summarization [11] is a fully data-driven approach that uses a local attention-based model to generate each word of the summary conditioned on the input sentence. This project is slightly different from attention-based model. The Neural Model has combined the neural language model with a contextual input encoder. The encoder is modeled off of the attention-based encoder of Bahdanau et al. [12] The encoder as well as the generation model are trained jointly on the sentence summarization task. The model has also incorporated a beam-search decoder. According to Rush et al. (2015), the model has incorporated less linguistic structure than comparable abstractive summarization approaches. However, it can scale to large amounts of data. The flexibility of the model as to how it can be trained on almost any document-summary pair is an advantage. This means that as there are no assumptions about the vocabulary, almost any type of summary can be generated provided there is enough data available. However, the other side of the coin is that this means that once a model is trained on a particular type of document, it will always produce summaries of the same style and type regardless of the document. That is, if a model is trained on the Gigaword article-headline data, providing an input of Amazon reviews, for example, will generate a title of the same formal type as seen in the Gigaword headline.

The sentence summarization task has been defined as follows:

Let the input consist of sequence of M words $\mathbf{x_1}, \ldots, \mathbf{x}_M$. These words come from a fixed vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = V$. Each word is represented as an indicator vector $x_i \in \{0, 1\}^{\mathcal{V}}$ for $i \in \{1, \ldots, M\}$, sentences as a sequence of indicators and χ as the set of possible inputs.

A summarizer takes $\mathbf{x}$ as input and outputs a shortened sentence $\mathbf{y}$ of length $N < M$. All the words in $\mathbf{y}$ come from the same vocabulary $\mathcal{V}$. The output is a sequence of words $\mathbf{y_1}, \ldots, \mathbf{y}_N$. In the paper, they have assumed the output length N to be fixed, and the system knows the length of the summary before generation.

Now, the problem of generating summaries has been dealt with by finding an optimal sequence of words from the set of all possible sequences of length *N*. This optimal sequence has been identified as one that optimizes the scoring function:

$$\arg\max_{\mathbf{y}\in\mathcal{Y}} s(\mathbf{x}, \mathbf{y}).$$

Figure 1: scoring function [11]

In general, this scoring function is defined as the log-likelihood *s*(**x**, **y**) = log *p*(**y** | **x**; θ). This log-likelihood is defined as:

$$\log p(\mathbf{y}|\mathbf{x};\theta) \approx \sum_{i=0}^{N-1} \log p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c;\theta).$$

Figure 2: log-likelihood [11]

The important assumption considered here while calculating the log-likelihood is the fact that the context would go back only as far as *C* characters. That is, the current word in the sequence **y**, is dependent only on previous *C* words in the sequence.

Now, the neural language model has been based on the neural network language model (NNLM) described by Bengio et al. (2003) [13]. The full model is:

$$\begin{aligned} p(\mathbf{y}_{i+1}|\mathbf{y}_c, \mathbf{x};\theta) &\propto \exp(\mathbf{V}\mathbf{h} + \mathbf{W}\mathrm{enc}(\mathbf{x}, \mathbf{y}_c)), \\ \tilde{\mathbf{y}}_c &= [\mathbf{E}\mathbf{y}_{i-C+1}, \ldots, \mathbf{E}\mathbf{y}_i], \\ \mathbf{h} &= \tanh(\mathbf{U}\tilde{\mathbf{y}}_c). \end{aligned}$$

Figure 3: NNLM Model [11]

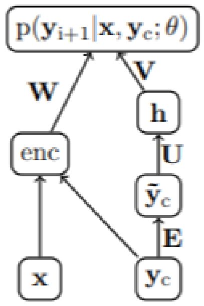


Figure 4 (a): Network Diagram for the NNLM model [11]

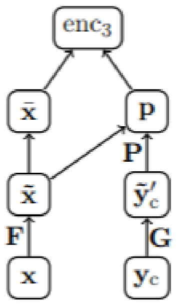


Figure 4 (b): Network Diagram for the encoder [11]

x here is the input sequence, **y** is the output sequence, $\mathbf{y_c}$ are the *C* previous letters. The rest are temporary, intermediate variables. The parameters are θ = (**E, U, V, W**) where E is a word-embedding matrix, U, V, and W are weight matrices. D is the size of the word embeddings and **h** is a hidden layer of size *H*. The **enc** function that returns a vector of size *H*, representing the input as well as the current context. The paper has suggested three different types of encoders. Two encoders are talked about below, for the mere reason that while the first one is a fairly simple one, the third (and the more complex one) utilizes the concept of context in representing the input.

Bag-of-Words Encoder

This is the most basic model, where the input sentence is embedded down to the size *H*. The proper relationships or order of the words is ignored. The model can be described as:

$$\begin{aligned} \mathrm{enc}_1(\mathbf{x}, \mathbf{y}_c) &= \mathbf{p}^\top \tilde{\mathbf{x}}, \\ \mathbf{p} &= [1/M, \ldots, 1/M], \\ \tilde{\mathbf{x}} &= [\mathbf{F}\mathbf{x}_1, \ldots, \mathbf{F}\mathbf{x}_M]. \end{aligned}$$

Figure 5: Bag of Words encoder [11]

The input-side embedding matrix **F** is the only parameter. **p** is a uniform distribution over the input words.

Attention-Based Encoder

This encoder utilizes an attention-based contextual encoder that constructs a representation based on the generation context. The model is similar to the bag-of-words encoder. The encoder is represented as:

$$\begin{aligned} \mathrm{enc}_3(\mathbf{x}, \mathbf{y}_c) &= \mathbf{p}^\top \bar{\mathbf{x}}, \\ \mathbf{p} &\propto \exp(\tilde{\mathbf{x}} \mathbf{P} \tilde{\mathbf{y}}'_c), \\ \tilde{\mathbf{x}} &= [\mathbf{F}\mathbf{x}_1, \ldots, \mathbf{F}\mathbf{x}_M], \\ \tilde{\mathbf{y}}'_c &= [\mathbf{G}\mathbf{y}_{i-C+1}, \ldots, \mathbf{G}\mathbf{y}_i], \\ \forall i \quad \bar{\mathbf{x}}_i &= \sum_{q=i-Q}^{i+Q} \tilde{\mathbf{x}}_i / Q. \end{aligned}$$

Figure 6: Attention-Based encoder [11]

Here, **G** is an embedding matrix for the context, **P** contains the parameters and Q is a smoothing window.

The aim while training the model is to minimize the negative log-likelihood of the probability value defined above (Fig. 2). This means that the training of the model is not affected by the generation constraints. The generative model works independent of the training model and therefore the two do not have an impact on each other. The details of the training are mentioned in the paper, and have been selected based on the multiple iterations with different values. They are therefore, independent of the process or the document constraints and are not relevant to this project as of now.

The task of generating summaries is, as specified above, is to find a sequence of words that maximize the scoring function. We also defined a relationship between the current generated word and the previous C words in that that the choice of current would only depend on the previous C words. This was the Markov assumption. In the research paper by Rush et al. (2015), they have made use of the beam-search decoder. The decoder, while maintaining the full vocabulary V while limiting itself to potential K hypotheses. The beam-search model has been a standard approach for neural Machine Translation models [14].

2.1.2 Abstract Meaning Representation and Graphs

Although this project is not based on the treebanks or semantic graphs, this type of approach at Abstractive Summarization is worth mentioning. The project was at one time, on crossroads as to what kind of approach to follow and work upon. The “road not taken” was this approach.

According to Banarescu et al. (2013), there idea behind designing such an approach was based on trying to represent semantic relationships between words while preserving their semantic as well as syntactic meaning of the corpus. They have defined their graphs (AMRs) as “rooted, labeled graphs that are easy for people to read, and easy for programs to traverse”. One of the most important barriers between making a machine understand a language is how different sentences can mean the same thing to a human, but can be difficult to make a machine understand. With AMRs, they have aimed to abstract away from syntactic idiosyncrasies by assigning same AMRs to sentences similar in meaning. The AMR is also strictly restricted to English.

The tree has leaves and links. The links are labelled with concepts that are usually the subject or the object of the sentence. Relationship links are entities that are either verbs, other nouns or adjectives. More on the relations and how they are specified is based on semantic relations, quantitative entities as well as date-time and other words. These entities are extensively derived from the PropBank framesets [15].

Now, the task of Abstractive Summarization using these Semantic Representations had been taken up by Liu et. al (2018). In this framework, the source text is parsed to a set of AMR graphs, the graphs are the transformed into a summary graphs and then a text is generated from these summary graphs. The framework is data-driven, meaning it is not specifically designed for a particular domain. The dataset used is a certain “proxy section” of the AMR bank [18].

The paper has focused on the second task, more specifically the transformation of source AMR graphs as a structured prediction problem. The source text to AMR graphs conversion is carried through by a statistical semantic parser known as JAMR [17]. Therefore, the process of graph summarization has been divided into two stages: source graph construction, and subgraph prediction.

The “source graph” is a single graph constructed using the individual sentences’ AMR graphs by merging identical concepts. This merging is quite simply, collapsing certain graph fragments into a single concept, and then merging all these concepts under a label – thereby reducing redundancy. An example of a graph concatenated and then given a new label is shown below:

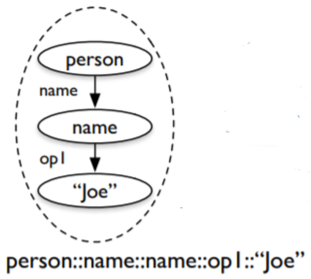


Figure 7: AMR Graph [16]

To ensure that the entire source graph for the text stays connected, a new “ROOT” node is added, and every concept is connected to it.

The task for subgraph prediction is, as mentioned above, a structured prediction problem that trades off among including important information without altering its meaning, maintain brevity and producing fluent language.

The model is defined as:

Let $G = (V, E)$ denote the merged source graph, where each node $v \in V$ represents a unique concept and each directed edge $e \in E$ connects two concepts. G is a connected, directed, node-labeled graph. Edges in this graph are unlabeled, and edge labels are not predicted using the subgraph selection. Therefore, the paper seeks to maximizes a score that factorizes over graph nodes and edges that are included in the summary graph. Therefore, for a subgraph (V', E'):

$$score(V', E'; \boldsymbol{\theta}, \boldsymbol{\psi}) = \sum_{v \in V'} \boldsymbol{\theta}^\top \mathbf{f}(v) + \sum_{e \in E'} \boldsymbol{\psi}^\top \mathbf{g}(e)$$

Figure 8: Scoring Function

[16]

Here, $\mathbf{f}(v)$ and $\mathbf{g}(e)$ are feature representations of node v and edge e. The θ and ψ are vectors of empirically estimated coefficients in a linear model. These parameters are learnt through supervised learning, using the structured perceptron. Although, the paper has considered other alternatives, they have come to the conclusion that the perceptron is the most well-suited to learn these vectors.

2.2 PROPOSED SYSTEM

In the recently proposed systems, LSTM cells haven’t performed as expected with scaling in input size, and this attention mechanisms were introduced to selectively store information and has since been adopted in most NLP sequence to sequence models. Luong et al. suggested global attention schemes can complement LSTM models well in NLP tasks. The attentional decoder has a dot attention mechanism from that paper. [19] For the decoding of words and choosing the best sequence, a Beam Search Decoder is used.

There was also an opportunity to use more sophisticated models such as Bidirectional LSTM cells where one layer of LSTM scans front to back and another LSTM layer scans back to front. However, Bi-LSTMs are computationally intensive and increasing the number of units in a layer will increase the computational requirements and training time.

The model takes inspiration from both the systems proposed by Rush et al. (2015) and Luong et al. (2015).

First, the preprocessed article (details in Ch. 3) is converted to GloVe word vector embedding models and is fed as an input the LSTM encoder. The LSTM encoder generates a vector, while the attention decoder learns the attention weights with the help of the training set. This will help to generate the summaries at the end. The Beam Search decoder then allows the generation of sample summaries generated by the network. The final headline is generated at the end.

Details and specifications about the model are described in Ch. 3

3. SYSTEM ANALYSIS AND DESIGN

3.1 Requirement Specifications

All the necessary requirements for hardware as well as software are listed in Ch. 1, Sec 1.3 and Ch 1, Sec 1.4.

Ideally, it would have been preferred if a separate GPU was available because the training time on an Intel inbuilt GPU was considerably higher as compared to what it might have been had there been a dedicated GPU.

3.2 Flowcharts

The flowchart of the model is as below:

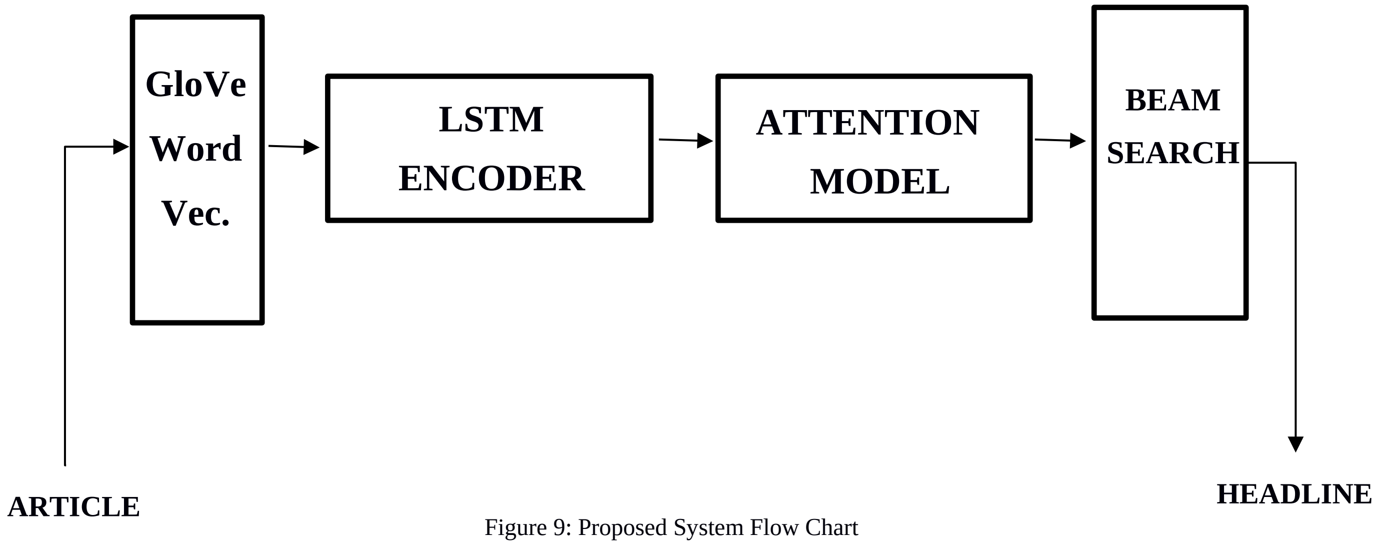


Figure 9: Proposed System Flow Chart

3.3 Design

An encoder-decoder architecture is used. Both the encoder as well as the decoders are recurrent neural networks – or more precisely Long Short-Term Memory (LSTMs) networks

The attention mechanism is inspired by the one proposed by Luong et al. (2015).

The encoder is fed the text (description) of the news article as input one word at a time. The words are passed through an embedding layer (GloVe vectors) individually. These words are therefore transformed into a distributed representation. This representation is combined using a 3-layer LSTM network.

The decoder then takes as input the hidden layers generated after the last word of the input text is fed. The decoder generates the words of the headline again a softmax layer and an attention mechanism.

The loss function used is the log loss function:

$$-\log p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = -\sum_{t=1}^{T'} \log p(y_t | y_1, \ldots, y_{t-1}, x_1, \ldots, x_T)$$

Figure 10: Log loss function [11]

Here, *T* is the length of the input text, and *T'* is the length of the summary.

I have used 3 hidden LSTM units with 512 cells in each. The parameters of the model lie uniformly in the range of [-0.1; 0.1]. The biases for each word in the softmax layer are initialized with respect to the probability of its occurrence in the training data, as suggested by Karpathy et. al. (2014) [22].

I have used a learning rate of 0.001 with batches of 640 samples and a total of three LSTM layers. I have also set the maximum description length at 25 words, and the maximum headline (title) length at 10 words.

Details about the attention model are specified in Sec 3.5

3.4 Dataset

3.4.1 Overview

The dataset used was the English Gigaword dataset as available from the Stanford Linguistics dept. The dataset consists of years of news articles from half a dozen news agencies. The articles have separated headlines and text, and the text is broken up into several paragraphs.

3.4.2 Preprocessing

A total of 500000 description-summary pairs from English Gigaword (maximum 25 words and minimum 10 words) are filtered out, and 40000 most common words are taken out and assigned GloVe word vectors. GloVe embeddings are by far the most ubiquitous and most-used embeddings available for tasks regarding NLP. The input to the GloVe embeddings is slightly preprocessed in the sense that unknown words (not present in the vocabulary) are assigned the <UNK> tag. The <EOS> tag is also assigned to the end of sentence vectors to signify that the sentence has ended. The digits are replaced with a '#'. If there are certain words that are in the text corpus but **not** in the GloVe embedding matrices, find the closest matching vector using cosine distance, and assign that value.

3.4.3 Dataset Issues

There were certain issues in the dataset itself. There are cases where the headline is not an accurate summary of the text. There was also a formatting issue, where the headlines had certain subtexts entirely unrelated to the topic which the article was about in the first place. There were also many headlines which were in certain coded forms and seemed entirely incoherent.

3.5 Algorithm

3.5.1 Learning weights

The model first learns the weights which will help it predict the titles later. Once the GloVe vectors have been found for each article-headline pair, it is then the task of passing it over to the model.

The model is an LSTM network as mentioned above. Moreover, the attention model implemented is a global attention mechanism meaning that the attention weights are learnt from *all* source words. This can help contextualize relationships between words. Simply put, the attention mechanism learns the similarities between the input sequence and the word which could be a potential target candidate. An example of a global attention model as proposed by Luong et. al (2015) is shown below:

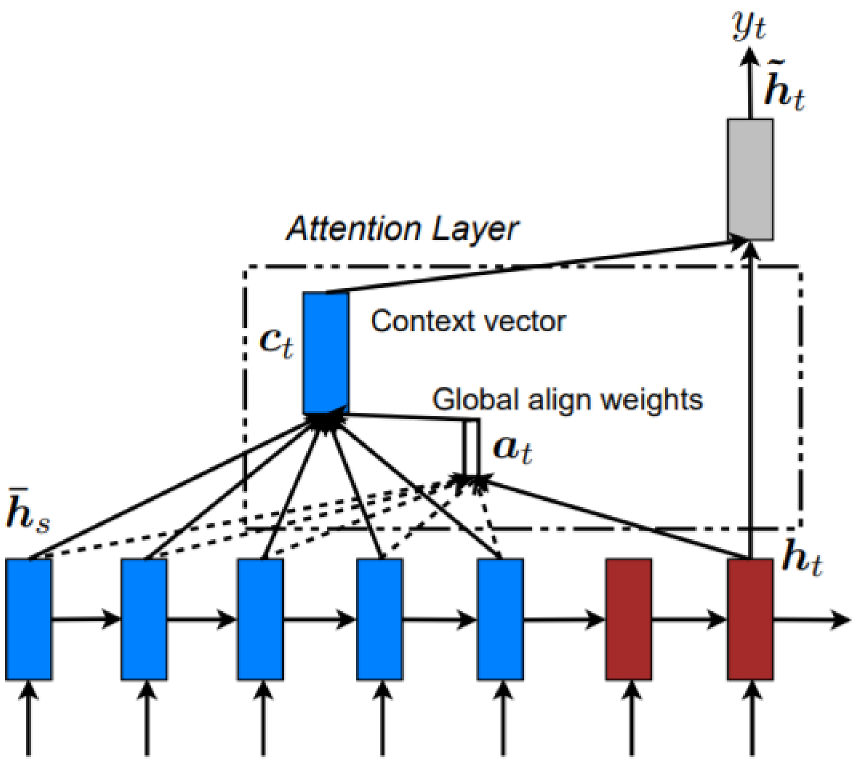


Figure 11: Global Attention Model [14]

The different variables and symbol as shown in the image have the following meanings:

h_t = current target hidden state of the LSTM cell. Here, the hidden state of the final cell is input to the headline generating beam search decoder.

$\bar{h}_s$= source hidden states (previous cells).

a_t = alignment vector, calculated using hidden state of target LSTM cell (h_t) and the attentional hidden state. The alignment vector is calculated using a *score* function, which is essentially a dot product of the hidden state and the attentional hidden state.

c_t = context vector, computed as the weighted average over all source hidden states h_t, and the weights as a_t.

$\bar{h}_t$= Attentional target hidden states; calculated as: tanh ($\boldsymbol{W_c}[c_t ;h_t]$). This attentional vector is then fed through a softmax layer to produce a predictive distribution.

The attention mechanism works in the sense that for each output word, the mechanism computes weights over each of the input words and determines the amount of attention that can be linked to the output word in question. These weights are then used to compute a weighted average of the final hidden layer – which essentially is the context vector.

The attention vector and the context vector are the generated from the topmost layer of LSTM cells. The generation of the attention vector and correspondingly context vector is slightly tweaked, as part of experimentation, in that that only the output activation of first 40 cells of the topmost LSTM layer is used to generate the attention vector. The alignment weights however are still calculated using the *dot* mechanism as proposed by Luong et. al (2015). The hidden states values from rest of the cells are used to generate the context vector.

The context vector after being transformed into a distribution over probability is what the Beam Search model will decode to find the best-fit-sequence.

3.5.2 Beam Search

The Beam Search decoder is, as the name suggests, keeps a certain *k* number of sequences in mind from all the possible likely next steps. This *k* is usually user-defined, but for the project, it has been set at 10.

The Beam Search model returns the *k* samples as well as their log likelihood scores to identify the most plausible sequence. We use the logarithm function so as to deal with underflowing floating point numbers.

Therefore, in the results and the output section, you will see a number besides the headline generated. The Beam Search decoder is a slight variation on the work done by Kiros et al. (2015).

3.6 Testing Process

The total data has been divided into a training and testing set. Once the attention weights and the context vectors were learnt, some randomly selected part of the test data were input into the model to generate outputs.

The entire data was divided into training-testing set by the ratio of 9:1.

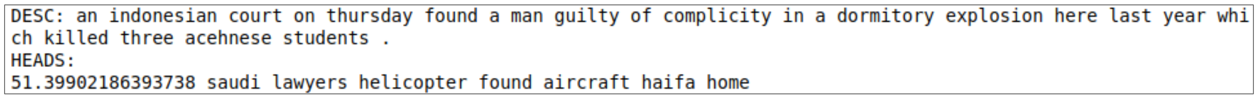
The outputs are currently only qualitatively measurable. The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [23] scores are yet to be generated. We also have the possibility of using BLEU (BiLingual Evaluation Understudy) [24] scores. This will be left to the future work section.

4. RESULTS/OUTPUT

The model was run for a different number of epochs. The maximum was 30. That took up about 14 hours to finish on computer with the hardware specifications as mentioned in Ch. 1, Sec 1.3.

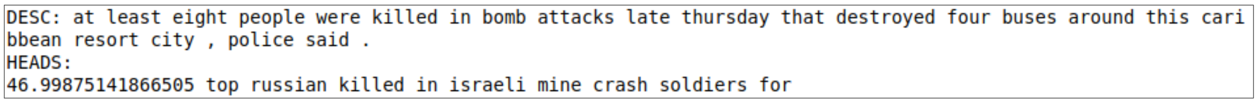
It was later also run with epochs = 8 and epochs = 25.

The proposed system, as mentioned was trained on the Gigaword headline-article pair. Below, are certain headlines that it generated:



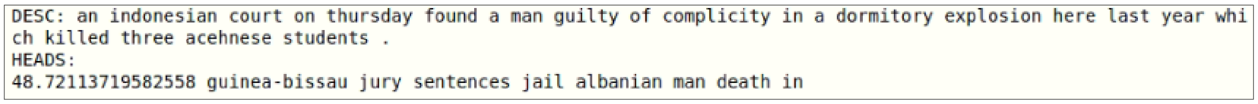
```
DESC: an indonesian court on thursday found a man guilty of complicity in a dormitory explosion here last year whi
ch killed three acehnese students .
HEADS:
51.39902186393738 saudi lawyers helicopter found aircraft haifa home
```

Figure 12: Generated Headline. #Epochs = 8



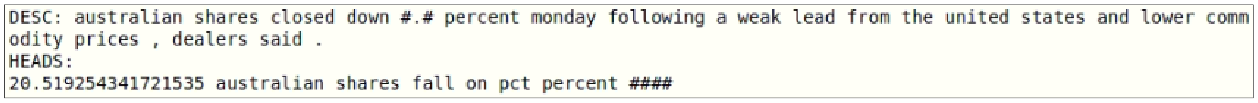
```
DESC: at least eight people were killed in bomb attacks late thursday that destroyed four buses around this cari
bbean resort city , police said .
HEADS:
46.99875141866505 top russian killed in israeli mine crash soldiers for
```

Figure 13: Generated Headline. #Epochs = 8



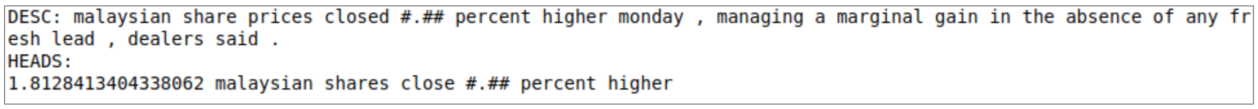
```
DESC: an indonesian court on thursday found a man guilty of complicity in a dormitory explosion here last year whi
ch killed three acehnese students .
HEADS:
48.72113719582558 guinea-bissau jury sentences jail albanian man death in
```

Figure 14: Generated Headline. #Epochs = 25



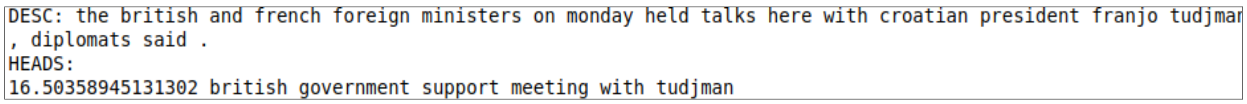
```
DESC: australian shares closed down #.# percent monday following a weak lead from the united states and lower comm
odity prices , dealers said .
HEADS:
20.519254341721535 australian shares fall on pct percent ####
```

Figure 15: Generated Headline. #Epochs = 25



```
DESC: malaysian share prices closed #.## percent higher monday , managing a marginal gain in the absence of any fr
esh lead , dealers said .
HEADS:
1.8128413404338062 malaysian shares close #.## percent higher
```

Figure 16: Generated Headline. #Epochs = 30



```
DESC: the british and french foreign ministers on monday held talks here with croatian president franjo tudjman
, diplomats said .
HEADS:
16.50358945131302 british government support meeting with tudjman
```

Figure 17: Generated Headline. #Epochs = 30

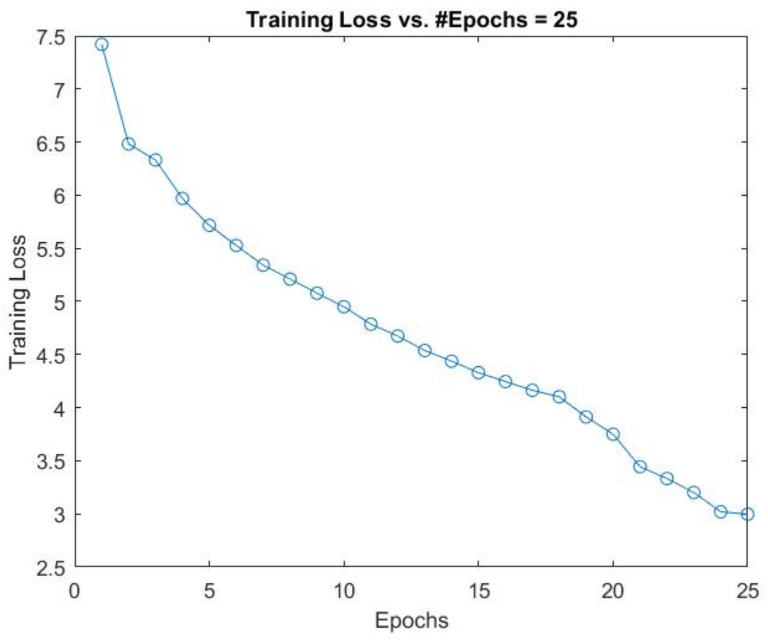


Figure 18: Training Loss vs #Epochs = 25

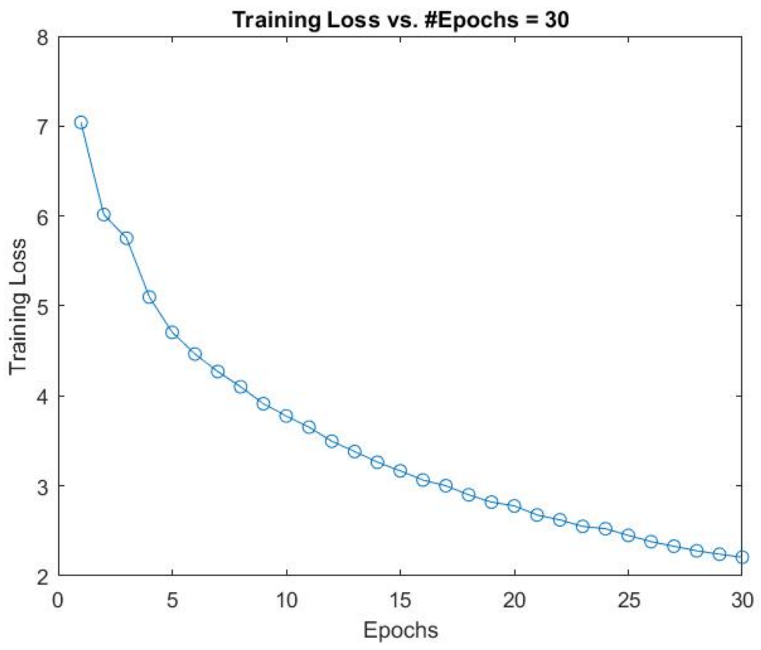


Figure 19: Training Loss vs #Epochs = 30

5. CONCLUSIONS AND FUTURE WORK

Although the field of Natural Language Processing is still unchartered territory in a lot of ways, there are exciting promises. Abstractive Summarization might just be a problem to pursue especially when the need to categorize, represent and contextualize data is the need of the hour. Generating adequate and meaningful summaries will not only help reduce data by making sure important information is not lost, it will also help improve information dissemination in the world of news and publications.

The project was not able to include quantitative measurements such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) or BLEU (BiLingual Evaluation Understudy) scores.

The neural network model still has a lot to improve upon, specifically regarding whether the current method of GloVe embeddings is feasible and adequate enough to represent information and help to train the network when dealing with languages. The attention model has been utilized by multiple authors trying to use neural networks whether in the form of LSTMs or other RNNs. Indeed, it may well be the case that this approach can only sustain up to a point.

Regarding the project, there are multiple ideas for future work including:

a. Exploring whether data can be represented in some other form and yet still be trained on a neural network.
b. Inculcating grammar and semantic representation in the text embeddings.
c. Reducing the time it takes to train the data. It can either mean the algorithm is too complex, or that the system cannot cope up.
d. Bi-directional RNNs, although computationally heavy, could be tweaked to run with attention mechanisms, as I believe attention models can be combined with Bi-LSTMs efficiently.
e. The model, if appropriately adequate, can be ‘transferred’ to be used on another syntactically similar language.
f. Instead of word-level embeddings and predictions, character-level approach can also be explored.

g. More complex attention models such as Pointer Generator networks is another avenue to improve decoding results.

6. REFERENCES

[1]. Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. ‘Introduction to the special issue on summarization’. Computational linguistics 28, 4 (2002), pp. 399–408.

[2]. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B. and Kochut, K. "Text Summarization Techniques: A Brief Survey." arXiv.org. N. p., 2017. Web. 18 Apr. 2019.

[3]. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. ‘GloVe: Global Vectors for Word Representation.’

[4]. Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D. and Carin, L. "Adversarial Feature Matching For Text Generation." Proceedings of the 34th International Conference on Machine Learning - Volume 70 (2017): 4006-4015. Web. 19 Apr. 2019.

[5]. Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J. and Jurafsky, D. "Deep Reinforcement Learning For Dialogue Generation." arXiv.org. N. p., 2016. Web. 19 Apr. 2019.

[6]. Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation By Jointly Learning To Align And Translate." arXiv.org. N. p., 2014. Web. 19 Apr. 2019.

[7]. Mikolov, T., Chen, K., Corrado, G. and Dean, J. "Efficient Estimation Of Word Representations In Vector Space." arXiv.org. N. p., 2013. Web. 19 Apr. 2019.

[8]. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. "Generative Adversarial Nets." Papers.nips.cc. N. p., 2014. Web. 19 Apr. 2019.

[9]. Lopyrev, Konstantin. "Generating News Headlines With Recurrent Neural Networks." arXiv.org. N. p., 2015. Web. 30 Apr. 2019.

[10]. Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. ‘Abstract meaning representation for sembanking’. In Proc. of the Linguistic Annotation Workshop and Interoperability with Discourse.

[11]. Rush, Alexander M., Sumit Chopra, and Jason Weston. "A Neural Attention Model For Abstractive Sentence Summarization." arXiv.org. N. p., 2015. Web. 19 Apr. 2019.

[12]. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. “Neural machine translation by jointly learning to align and translate.” CoRR, abs/1409.0473.

[13]. Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. 2003. ‘A neural probabilistic language model’. The Journal of Machine Learning Research, 3:1137–1155.

[14]. Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. ‘Addressing the rare word problem in neural machine translation.’ In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 11–19.

[15]. P. Kingsbury and M. Palmer. 2002. From TreeBank to PropBank. In Proc. LREC

[16]. Liu, F., Flanigan, J., Thomson, S., Sadeh, N. and Smith, N. A. "Toward Abstractive Summarization Using Semantic Representations." arXiv.org. N. p., 2018. Web. 21 Apr. 2019.

[17]. Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. ‘A discriminative graph-based parser for the abstract meaning representation’. In Proceedings of ACL.

[18]. Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. Abstract meaning representation (AMR) annotation release 1.0 LDC2014T12. Web Download. Philadelphia: Linguistic Data Consortium.

[19]. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective Approaches To Attention-Based Neural Machine Translation." arXiv.org. N. p., 2015. Web. 19 Apr. 2019.

[20]. Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. "Skip-Thought Vectors." arXiv preprint arXiv:1506.06726 (2015).

[21]. Graff, David, and Christopher Cieri. English Gigaword LDC2003T05. Web Download. Philadelphia: Linguistic Data Consortium, 2003.

[22] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[23] Lin, Chin-Yew. "ROUGE: A Package for Automatic Evaluation of Summaries". Aclweb.org. N. p., 2019. Web. 3 May 2019.

[24] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a Method for Automatic Evaluation of Machine Translation". Aclweb.org. N. p., 2019. Web. 3 May 2019.