**Abstractive Text Summarization using Machine learning**

**A report submitted in partial fulfillment of the requirements**

**Of**

**Mini-Project (ISL64)**

**In**

**Sixth Semester**

**By**

|  |  |
| --- | --- |
| **1MS17IS123** | **Tarun R** |
| **1MS17IS120** | **Sumukh M Lohit** |
| **1MS17IS154** | **D Sai Viswa Sumanth** |
| **1MS17IS076** | **Nikhil Thadeshwar** |

**Under the guidance of**

**Rajeshwari S B**

Assistant Professor

Dept. of ISE, RIT



**RAMAIAH**

**Institute of Technology**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)

M. S. RAMAIAH NAGAR, M. S. R. I. T. POST, BANGALORE – 560054

**2019-2020**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute Affiliated to VTU)**

**M. S. Ramaiah Nagar, M. S. R. I. T. Post, Bangalore – 560054**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**



**RAMAIAH**

**Institute of Technology**

**CERTIFICATE**

This is to certify that the project work entitled **“Abstractive Text Summarization using Machine Learning”** is a bonafide work carried out by **Tarun** bearing **USN: 1MS17IS123, Sumukh M Lohit** bearing **USN: 1MS17IS120, D Sai Viswa Sumanth** bearing **USN: 1MS17IS154, Nikhil Thadeshwar** bearing **USN: 1MS17IS076,** in partial fulfillment of requirements of Mini-Project (ISL64) of Sixth Semester B.E. It is certified that all corrections/suggestions indicated for internal assessment has been incorporated in the report. The project has been approved as it satisfies the academic requirements in respect of project work prescribed by the above said course.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Signature of the Guide Signature of the HOD

**Rajeshwari S.B Dr. Vijaya Kumar B P**

Assistant Professor Professor and Head, Dept. of ISE, RIT, Dept. of ISE, RIT Bangalore-54 Bangalore-54

Other Examiners

**Name of the Examiners: Signature**

1.

2.

**ACKNOWLEDGMENT**

We would like to express our sincere gratitude to all those who helped us in making this project a success. We are grateful for their cooperation during the period of our project.

We are grateful to our Institution, **Ramaiah Institute of Technology** with its ideals and inspiration for having provided us with the facilities, which has made this project a success.

We would like to extend our hearty gratitude **to Dr. N V R Naidu**, The Principal, for the kind support and permission to use the facilities available in the Institute.

We take this opportunity to express our profound gratitude and deep regards to the Head of the Department and Professor **Dr. Vijaya Kumar B.P**. for his valuable information, exemplary guidance, monitoring and constant encouragement throughout the course of this thesis.

We acknowledge the guidance and support extended by our guide, **Rajeshwari S B**, Assistant Professor, Department of ISE. Her incessant encouragement and invaluable technical support have been of immense help. Her guidance gave us the environment to enhance our knowledge and skills.

The blessing, help and guidance given by them from time to time shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to all the teaching and non-teaching staff members of the ISE department, for the valuable information provided by them in their respective fields. Lastly, we thank everyone for their constant encouragement with this project.

**ABSTRACT**

Abstractive summarization is the technique used for the project with the end result generating a summary of a text from its main idea, not by copying verbatim most salient sentences from text. Almost 44 zettabytes of data is created each day at the current pace, but that pace is only accelerating with the growth of the Internet of Things. Hence, there is a need for automatic text summarization to be performed on such huge amounts of data (including the news articles).So, abstractive text summarization came into demand that condenses the document into a shorter version by preserving the meaning and the content. Abstractive summarization has more applications in the digital world, the limited implementations of it due to its complexities and the future problems with data has fuelled the desire to work under this topic, by making its use more mainstream.

Sequence to sequence modelling has been used for abstractive text summarization with the help of LSTM and tensorflow as the environment set for training, NLTK for processing the words in the dataset. This model takes in a sequence of words as input and generates a sequence of words (the summary) as output. Since its a training model the results are confined to a specific type of data (the news articles in this case). Implementing the mentioned mechanism into a model builds up on the architecture and requires the use of unique tech libraries and implementations.

This technique tries to understand the context of the sentences. It has the ability to develop new sentences to tell the important information from text documents. This generates short, concise and grammatically correct summary of news articles. A reliable scoring system was used to compare the summary generated for the article with the existing summary. For testing conducted on 100 random news articles we have obtained that within the summaries predicted 94 % of them still retain the actual meaning of the original summary. In this scenario the rouge scores obtained on the testing done by the team have a constantly varying value (15 to 80 %)which indicates that the summary which is predicted by the model does not fully replicate the actual summary in terms of words used ,which is one of the main  objectives of abstractive summarization.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL NO** | **Topic** | **Page No** |
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Scope | 1 |
| 1.3 | Objectives | 2 |
| 1.4 | Proposed Model | 2 |
| 1.5 | Organization of Report | 3 |
| 2 | Literature Review | 4 |
| 3 | System Analysis and Design | 19 |
| 4 | Modelling and Implementation | 26 |
| 4.1 | Use Case Diagram | 26 |
| 4.2 | Class Diagram | 27 |
| 4.3 | Sequence Diagram | 28 |
| 4.4 | Collaboration Diagram | 30 |
| 4.5 | Pseudo Code (Algorithm) | 31 |
| 5 | Testing, Results and Discussion | 36 |
| 5.1 | Testing | 36 |
| 5.2 | Results | 41 |
| 5.3 | Discussion | 42 |
| 6 | Conclusion and Future Work | 43 |
| 7 | References | 44 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No** |
| Figure 2.1 | Semantic graph reduction approach architecture | 5 |
| Figure 2.2 | Sequence to Sequence Model | 9 |
| Figure 2.3 | Clustered semantic graph-based approach | 12 |
| Figure 2.4 | Framework for DNN based encoder and Decoder | 14 |
| Figure 3.1 | Workflow diagram | 20 |
| Figure 3.2 | Types of Sequence to Sequence | 22 |
| Figure 3.3 | Encoder to Decoder Flow | 23 |
| Figure 3.4 | LSTM Model Representation | 24 |
| Figure 3.5 | Encoder-Decoder flow with Word Embeddings | 25 |
| Figure 4.1 | Use Case diagram | 26 |
| Figure 4.2 | Class diagram | 27 |
| Figure 4.3 | Sequence diagram of the project overview | 28 |
| Figure 4.4 | Sequence diagram of the project components | 29 |
| Figure 4.5 | Collaboration diagram | 30 |
| Figure 5.1 | Test Score analysis figure | 37 |
| Figure 5.2 | ROUGE\_1 Vs BLEU Score Graph | 38 |
| Figure 5.3 | Loss Value Vs Epoch No Graph | 40 |
| Figure 5.4 | Epoch analysis figure | 41 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Name** | **Page No** |
| Table 2.1 | Tools used for the project | 18 |
| Table 5.1 | Test Score table | 36 |

**Chapter 1**

**INTRODUCTION**

**Motivation**

The increasing growth of the Internet has made a huge amount of information available. It is difficult for humans to summarize large amounts of text. Thus, there is an immense need for automatic summarization tools in this age of information overload. A good summarization tool generates brief summaries of an article so that the user need not go through the entire article.

The main motive behind the project is to create a text summarization tool which summarizes a set of sentences by understanding the context of the text since the existing tools did not try to understand the context behind the text.

The extractive summarization tools pick up sentences from the original document based on their importance. The extractive tools do not generate new sentences and do not focus on the semantics of the text.

**Scope**

The overall scope of the project is to have a deeper knowledge of the techniques in Machine Learning and Deep Learning in order to generate concise summaries of news articles, which lets a user see a summary given a news article or of the latest news. The scope also involves understanding of why recurrent neural networks are successful at phrasing sentences and how they treat some input words more important than the others by assigning the appropriate weights, and have a better overview internally of how recurrent neural nets work.

**Objectives**

* To Implement Recurrent Neural Networks (LSTM), so that they can generate concise summaries of news articles.
* The summary generated should contain significant information along with the topic coverage which quickly enables the user to quickly comprehend vast amounts of information.
* To provide an easier way to maintain data in a compressed format, avoiding the need to require huge memory sizes.
* The summary generated should be satisfactory to the user.
* To generate grammatically correct summaries and generate new sentences in the summary.
* Use a reliable scoring system for evaluation of summary generated in terms of performance metrics precision, recall.
* To stress test it on different cases and conduct testability analysis to make the model as flexible and reliable as possible.

**Proposed Model**

Sequence to sequence modelling was used for abstractive text summarization. In this, input is a long sequence of words and output is a short summary. A typical sequence to sequence model has two parts – an **Encoder** and a **Decoder.** Both the parts are practically two different neural network models combined into one giant network.An LSTM (Long Short-Term Memory) which is a type of recurrent neural network is used for encoder and decoder. Broadly, the task of an encoder network is to understand the input sequence, and create a smaller dimensional representation of it. This representation is then forwarded to a decoder network which generates a sequence of its own that represents the output. This architecture is used in two phases- the training phase and the inference phase.

In the training phase, we will first set up the encoder and decoder. We will then train the model to predict the target sequence offset by one timestep. After training, the model is tested on new source sequences for which the target sequence is unknown in the inference phase.

Instead of looking at all the words in the source sequence, we can increase the importance of specific parts of the source sequence that result in the target sequence with the help of an attention mechanism**.** Hence, attention mechanism is used.

**Organization of Report**

Chapter 1: **Introduction** focuses on the limitations of existing extractive summarization tools and the need to come up with an abstractive summarization tool. It looks at the scope of the project and its motivations and objectives.

Chapter 2: **Literature review** focuses on the research going on within this field and the study of existing systems for text summarization. It focuses on the research that has been carried out in the areas of text summarization and abstractive text summarization.

Chapter 3: **System Analysis and design** provides a detailed walk through of the software engineering methodology adopted to implement the model, an overview of the system and the modules incorporated into the system.

Chapter 4: **Modelling and Implementation** provides a deeper insight into the working of the model. The various modules and their interactions are depicted using relevant descriptive diagrams.

Chapter 5: **Testing** the model to ensure bug/error free model along with the **Results** obtained. **Discussion** then provides detailed analysis on quality assurance measures.

Chapter 6: **Conclusion** about the Results obtained after successfully running the model and Future Scope of the model is highlighted.

**Chapter 2**

**LITERATURE REVIEW**

Literature survey provides the material and publications that have been referred and has been helpful for the development of the model , they provide an idea on the pre existing work done on the topic along with the progress obtained in different aspects with its limitations and successes ,each of these may only be a part of the technique used in the project but all of them as a whole help build the foundation which has been improvised on and has helped in getting the whole project together. All those have been taken into consideration and the pros and cons have been weighed to get the best possible method to proceed.

Some of the existing techniques studied before proceeding with our model are:

## 2.1 Semantic Graph Reduction Approach for Abstractive Text Summarization

A novel approach is presented to create an abstractive summary for a single document using a rich semantic graph reducing technique. The approach summaries the input document by creating a rich semantic graph for the original document, reducing the generated graph, and then generating the abstractive summary from the reduced graph.

This approach exploits a new semantic graph called Rich Semantic Graph (RSG) [1, 2]. RSG is an ontology-based representation developed to be used as an intermediate representation for Natural Language Processing (NLP) applications. The new approach consists of three phases: creating a rich semantic graph for the source document, reducing the generated rich semantic graph to a more abstracted graph, and finally generating the abstractive summary from the abstracted rich semantic graph.

The main objective of the Rich Semantic Graph Creation Phase is to represent the input document semantically using Rich Semantic Graph (RSG). In RSG, the verbs and nouns of the input document are represented as graph nodes along with edges corresponding to semantic and topological relations between them. The graph nodes are instances of the corresponding verb and noun classes in the domain ontology.

The Rich Semantic Graph Reduction Phase aims to reduce the generated rich semantic graph of the source document to a more reduced graph. A model of heuristic rules is applied to reduce the graph by replacing, deleting, or consolidating the graph nodes using the WordNet relations [3, 4].

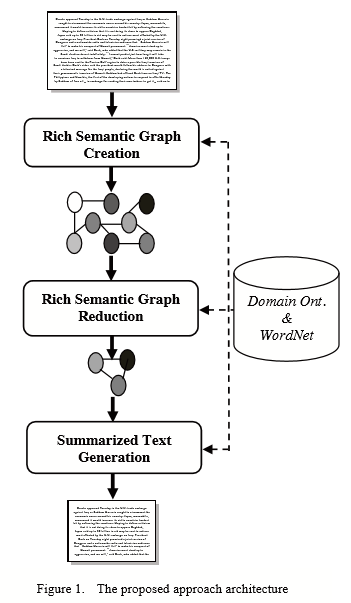


Figure 2.1: Semantic graph reduction architecture

The above fig 2.1 depicts the architectural flow of the semantic graph reduction approach.

## 2.2 Comparative Study of CNN and RNN for Natural Language Processing

There are two main DNN architectures: convolutional neural network (CNN) and recur-rent neural network (RNN). Gating mechanisms have been developed to alleviate some limitations of the basic RNN, resulting in two prevailing RNN types: long short-term memory (LSTM) and gated recurrent unit (GRU).

Based on the characterization “hierarchical (CNN) vs. sequential (RNN)”, it is tempting to choose a CNN for classification tasks like sentiment classification since sentiment is usually determined by some key phrases; and to choose RNNs for a sequence modelling task like language modeling as it requires flexible modeling of context dependencies.

RNNs perform well on document-level sentiment classification .It was recently shown in [5] that gated CNNs outperform LSTMs on language modeling tasks, even though LSTMs had long been seen as better suited.

The above tasks are organized in four categories.

1. TextC. Text classification, including SentiC and RC.
2. SemMatch including Textual Entailment (TE), Answer Selection (AS) andQuestion Relation Match (QRM).
3. SeqOrder. Sequence order, i.e., PQA.
4. ContextDep. Context dependency, i.e., POS tagging.

By investigating these four categories, the aim is to discover some basic principles involved in utilizing CNNs / RNNs.

In [6], word2vec, CNN, GRU and LSTM approaches were compared in sentiment analysis of Russian tweets, and it was found that GRU outperforms LSTM and CNN. In empirical evaluations, it was found there is no clear winner between GRU and LSTM. In many tasks, they yield comparable performance and tuning hyperparameters like layer size is often more important than picking the ideal architecture.

For TextC, GRU performs best on SentiC and comparably with CNN in RC. For SemMatch, CNN performs best on AS and QRM while GRU (and also LSTM) outperforms CNN on TE. For SeqOrder (PQA), both GRU and LSTM outperform CNN. For ContextDep (POS tagging), CNN outperforms one-directional RNNs, but lags behind bi-directional RNNs.[8]

RNNs are well suited to encode order information (for PQA) and long-range context dependency (for POS tagging) [8]. But for the other two categories, TextC and SemMatch, some unexpected observations appear. CNNs are considered good at extracting local and position-invariant features and therefore should perform well on TextC; but in the experiments they are outpeformed by RNNs, especially in SentiC because RNNs can encode the structure-dependent semantics of the whole input. [5] and [8]

GRU is better when sentiment is determined by the entire sentence or a long-range semantic dependency – rather than some local key-phrases. GRU and CNN are comparable when lengths are small.[5] Hence, which DNN type performs better in a given text classification task depends on how often the com-prehension of global/long-range semantics is required.

This can also explain the phenomenon in Sem-Match – GRU/LSTM surpass CNN in TE while CNN dominates in AS, as textual entailment relies on the comprehension of the whole sentence [7], question-answer in AS instead can be effectively identified by key-phrase matching.

All models are relatively smooth with respect to learning rate changes. In contrast, variation in hidden size and batch size cause large oscillations. Nevertheless, it can be observed that CNN curve is mostly below the curves of GRU and LSTM in SentiC task [6], contrarily located at the higher place in AS task.

**Conclusions**

For sequence modelling tasks (text summarization in our case), while CNN performs better than one-directional RNN, bi-directional RNN performs better than CNN. LSTM or GRU could be used to capture long term dependencies. GRUs train faster while LSTMs can remember longer sequences than GRUs and are more accurate. Tuning the hyperparameters is also important.

## 2.3 Abstractive method of text summarization with sequence to sequence RNNs

The model consists of a conditional recurrent neural network, which acts as a decoder to generate the summary of an input sentence, much like a standard recurrent language model. In addition, at every time step the decoder also takes a conditioning input which is the output of an encoder module. Depending on the current state of the RNN, the encoder computes scores over the words in the input sentence. These scores can be interpreted as a soft alignment over the input text, informing the decoder which part of the input sentence it should focus on to generate the next word. Both the decoder and encoder are jointly trained on a data set consisting of sentence-summary pairs. Furthermore, the encoder uses a convolutional network to encode input words.

**2.3.1 Attentive Recurrent Architecture**

Let x denote the input sentence consisting of a sequence of M words x = [x1,...,xM], where each word xi is part of vocabulary V, of size |V| = V .The task is to generate a target sequence y = [y1,...,yN], of N words, where N < M, such that the meaning of x is preserved: y = argmaxy P(y|x), where y is a random variable denoting a sequence of N words.

Training involves ﬁnding the θ which maximizes the conditional probability of sentence-summary pairs in the training corpus. If the model is trained to generate the next word of the summary, given the previous words, then the above conditional can be factorized into a product of individual conditional probabilities:

P(y|x;θ) = N Y t=1

p(yt|{y1,...,yt−1},x;θ)

This conditional probability using an RNN Encoder-Decoder Architecture is modelled after the implementation.

The author tries to present the applicability of sequence to sequence RNNs in abstractive text summarization. A pre-trained word to vector file was used for word embedding to check for word frequency as well as similarity. The summarizer uses an encoder and decoder architecture. [10]. An end-to-end approach to Multilayer LSTM has been described. Encoder uses a fixed length of text as input and Decider represents the output. An approach towards applying self-encoder, decoder RNN attention model on machine translation to text summarization was employed. In the text summarize method, a word embedding file was used. Then the vocabulary size of those files was counted which was used by the model.

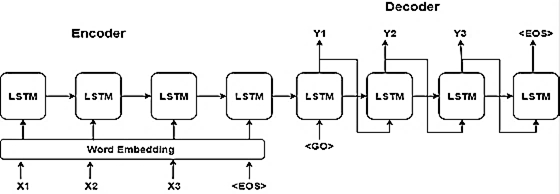


Figure 2.2: Sequence to Sequence Model

Figure 2.2 is illustrating the various components of a seq2seq model-encoder, decoding and word embeddings.

## At the first 2 layers, RNN encoder-decoder architecture [11] is introduced. Later this has been extended [9] where the encoder and decoder model used only machine translation.

## This neural network contains 2 layers of RNN’s. Encoder contains a fixed length of a sentence and decoder contains the sequence of output. The 2 layers RNN’s networks are trained unitedly and keep the maximum conditional probability of target text sequence. The hidden unit was used to improve memory capacity and training. Then train the model to learn the probability of an English sentence to the corresponding English sentence.

## 2.4 Review on Abstractive Text Summarization for single and multi-documents

Single document text summarization is to build a summary from a single source document. This type of text summarization technique accepts only one document as input, then uses different techniques to extract important sentences from the source document and then from extracted sentences summary to be generated. Generation of summary is in more understandable, syntactically or semantically correct and most important in reduced form. Various techniques of single document text summarization are discussed in the following section.

**2.4.1 Semantic Graph Reduction Approach**

In this approach generation of semantic graph is identified as a rich semantic graph (RSG). Then that semantic graph can be further reduced and a final abstractive summary can be generated from the reduced semantic graph. This approach consists of three phases. The first phase is RSG creation. The main aim of the RSG creation is to represent the input document semantically. The second phase is RSG reduction. In this phase generated semantic graph is reduced by applying a certain set of rules like merging and deleting the graph nodes. Third phase is abstractive summary generation from reduced RSG. This approach succeeds to reduce the source document up to half of the original document. Limitation of this approach is that it will not take multiple documents as input to generate an abstractive summary.

**2.4.2 Word Graph based Approach**

Word graph [12] was used to represent a source document. This approach includes two phases. First phase is sentence reduction and second is sentence combination. Word graph is used for sentence combinations and to represent word relations between texts. New sentences are generated from the word graph. In word graph nodes are used to represent the information about words and their part of speech tagger and the adjacency relations between word pairs are represented on edges. This approach generates a syntactically correct sentence but does not care about word meaning.

**2.4.3 Sentiment Infusion Approach**

This approach works on a graph-based technique[13] that makes summaries of redundant opinions and utilizes sentiment analysis to join the statements. This approach uses word graphs for compressing and merging information and then summaries are generated from resultant sentences. The graph captures the excess in the document using words that happen more than once in the texts are mapped to the similar node. For getting an abstractive summary, a score is given to every one of the ways as well as the sentences have been combined and then the sentences are ranked.

**2.4.4 Multi document text summarization**

**Genetic Semantic Graph Based Approach**

Semantic graph [14] is to be created for each document in such a way that the Predicate Argument Structure (PASs) is represented as graph vertices and the semantic similarity weight is to be represented on edges. For constructing PASs semantic role labeling was used. First this phase will extract predicates from each sentence of the document and then split the predicate structure into meaningful tokens. After that semantic similarity matrix operation was performed. Once the semantic similarity matrix is built then an undirected weighted semantic graph is constructed. Maximal marginal relevance (MMR) algorithm was used. This approach automatically merges similar information across the documents to reduce the overlapping information in summary.

**Clustered Genetic Semantic Graph Based Approach**

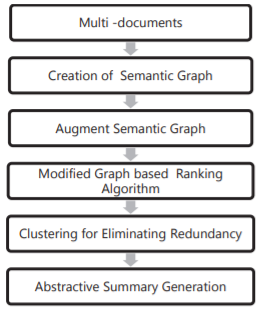


Figure 2.3: Proposed Clustered Semantic Graph Based Approach

Fig 2.3 depicted above is the Clustered Semantic Graph Based [15] approach similar to genetic semantic graph-based approach but here they used clustering algorithms to eliminate redundancy. In the clustering algorithm PASs with the highest similarity weight score from each cluster is chosen and applied to language generation. Then language generation rules are used to generate abstractive summary sentences. For making clusters they use the Hierarchical Agglomerative Clustering (HAC) algorithm. HAC algorithm accepts the semantic similarity matrix as input. Algorithm merges most similar clusters and updates the semantic similarity matrix to represent similarity between the nearest cluster and the original cluster. End user will decide the compression rate of the summary document. This process will be repeated until the user defined compression rate of summary is reached.

**2.5 Building Encoder and Decoder with Deep Neural Networks**

The basic principle of deep learning is based on mimicking the operation of neurons in the human brain such that the multiple layers of neuron nodes are stacked up with non-linear activation function between each layer, so called a deep neural network.

As deep learning is known to have very powerful classification capability, authors in [16] utilized CNN to classify modulation level. Also, using the universal approximation capability of DNN, the transmit power control scheme for communications systems was proposed in [17] and [18]. One of the most notable applications of deep learning in communications systems is a DNN-based codec i.e., encoder and decoder.

There have been several attempts to apply DNN in the design of an encoder and a decoder. In particular, autoencoder structure, which is originally devised to reconstruct the original data from the corrupted data, was adopted in the development of an encoder and a decoder. In [19], autoencoder structure was proposed to encode and decode Hamming code without prior knowledge by adopting fully-connected networks.

**Learning framework for end-to-end communications systems**

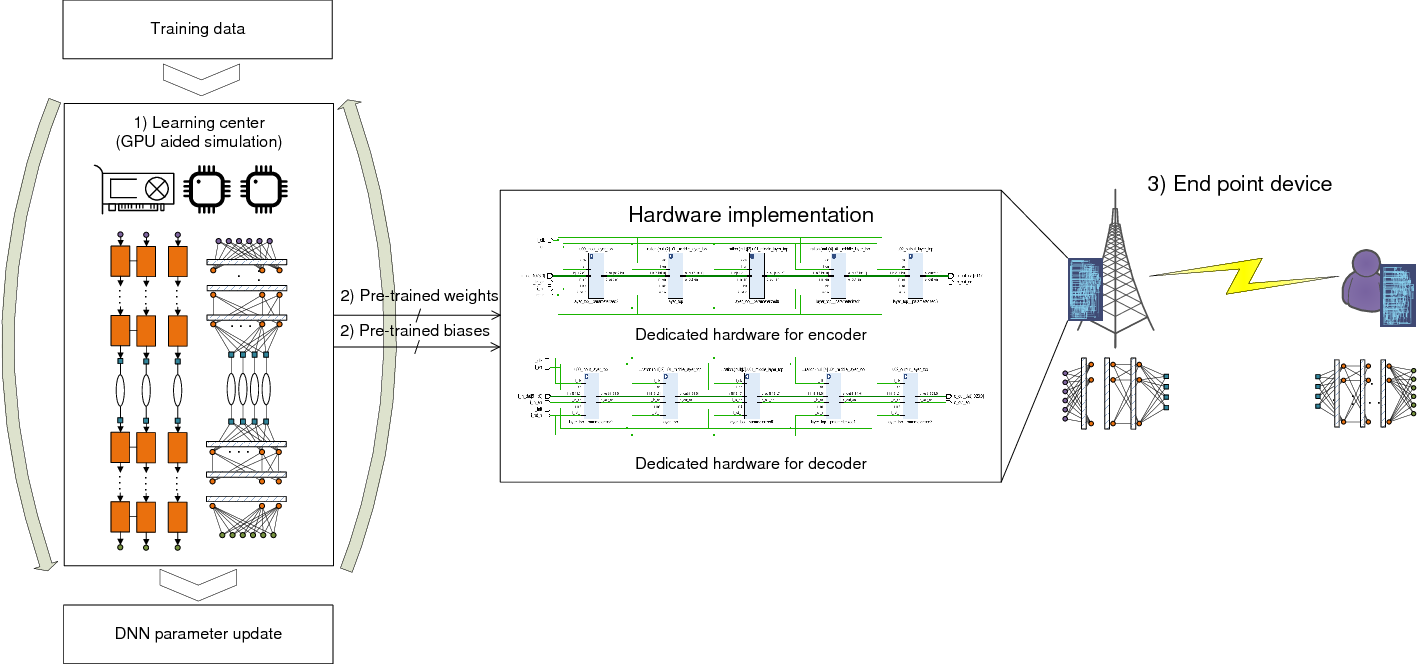
****

Figure 2.4: Proposed learning framework for training and implementing DNN-based encoder and decoder.

Fig 2.4 depicts the learning framework. The training process and 6 inference processes are separated at the learning center with large computation power and at the endpoint device with high-speed and real-time operational digital circuits, respectively.

**The major components of the learning framework are described as follows:**

**A. Components of Learning Framework**

1. **Learning center:** In the learning center, the DNN parameters of the DNN-based encoder and decoder are learned with the actual channel data or synthesized channel data if actual channel data is insufficient. This learning center is necessarily equipped with parallel computing devices.
2. **Parameter delivery:** After sufficient training, the DNN parameters for an encoder and a decoder are supposed to be delivered to endpoint communications devices, i.e., base station (BS)and user equipment. The DNN parameters, such as weights and biases, which are pre-determined by the learning center, are put into the encoder and decoder of each device.
3. **Endpoint device:** For endpoint devices, the encoder and decoder modules are built with dedicated hardware for boosting the signal processing where the weights and the biases are fed in by the learning center. The DNN-based encoder and decoder have been already informed with the pre-trained DNN parameters, thereby they only require one-shot inference process for one transmission.

**B. Overhead Reduction in the Proposed Framework**

The proposed framework and the training process are beneficial because the training process does not take place at the endpoint device. Therefore, the overhead originated from the high hardware cost and huge power consumption of the training process can be effectively removed.

**Conclusions and further works**

In the investigations on the HDL-based DNN communications systems, it has been confirmed the feasibility of the DNN-based communication technologies for enhancing the flexibility and productivity of the systems through actual HDL-implementation. Especially, the observed measurement has highlighted the applicability and potential of the proposed DNN algorithms for communications systems in practice.

## 

## 2.6 Limitations faced by neural network models for abstractive text summarization

Neural Sequence to Sequence attention models have shown promising results in Abstractive Text Summarization. But they are plagued by various problems. The summaries are often repetitive and absurd. Exploring and reviewing different techniques that can help overcome these issues would help. Also using a Reinforcement Learning based Training procedure using intra-Attention that significantly improves the model’s performance. Analyzing the problems that plague the area in detail, reasons and possible ways to improve upon will lead to better understanding. A novel architecture has been proposed to solve the problem of long summary generation, uncaptured by the current models.

**The results are plagued by many problems:**

They sometimes tend to reproduce factually incorrect details; they struggle with Out of Vocabulary (OOV) words. So, many UNK tokens are observed in the summary, they are also a bit repetitive and focus on a word/phrase multiple time. Focus is mainly on single sentence summary tasks like food reviews or headline generation.

**Possible improvements:**

**Large Vocabulary Trick and Feature-rich Encoder:**

In order to capture more meaning into the inputs fed into the encoder [20], going beyond the word-embeddings based representation of the input like word2vec or GloVe and also incorporate more linguistic features like POS (parts of speech) tags, named-entity tags, and TF-IDF statistics helps.

**Hierarchical Attention:**

Based on the idea that for the summary some sentences are more important than others. So, they use two Bi-Direction RNN to scan the source text, one at word level and another at the sentence level [20]. Then calculate word-level attention using the first encoder and sentence level attention using the second encoder.

**Pointer Generator Network:**

It helps to solve the challenge of OOV words and factual errors. It works better for multi-sentence summaries [20]. The basic idea is to choose between generating a word from the fixed vocabulary or copying one from the source document at each step of the generation. It brings in the power of extractive methods by pointing [21]. So, for OOV words, simple generation would result in UNK, but here the network will copy the OOV from source text.

A very big problem with the baseline summarization models is during the sampling stage. During training, we always feed in the correct inputs to the decoder, no matter what the output was at the previous step. So the problem that this gives rise is that the model doesn’t learn to recover from its mistakes and assumes that it will be given the golden token at each step in the decoding. This works fine during training but during test time, we sample the next word from the output of the previous step, so if the model produces even one wrong word then the recovery is hard. A naive way to rectify this problem is to toss a coin with P[heads] = p, during decoding at training time and choose the token produced at the previous step with probability p and the golden output token with probability 1 − p. Now the network can’t always assume that it’ll be given the correct summary and hence learns to generalize better. In practice, this method gives only slight improvement but impacts convergence. But very recently modifications have come up with a Reinforcement Learning based training [22] method that gave huge improvements.

**Conclusion**

This review shows that Deep Learning based approaches are promising and give some hope in solving Abstractive text summarization which had been largely unsolved till now. But the problems with the metric and lack of dataset are a challenge to scalability and generalizing to multi-sentence summarization.

Table 2.1: Overview of Tools Used

|  |  |
| --- | --- |
| **Tool** | **Description** |
| Dataset | Harvard NLP/sent-summary which consists of news articles with summaries |
| Python | Programming Language Used |
| Tensorflow | An end-to-end open source machine learning platform for everyone, artificial intelligence library, using data flow graphs to build models like neural networks |
| Keras | Deep-learning library |
| NLTK | Natural language Toolkit for preprocessing of data |
| Gensim module | It is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. |
| Glove | Used for obtaining vector representations for words. |

The above table gives the set of tools used for the project - dataset, programming language and the libraries.

**Chapter 3**

**SYSTEM ANALYSIS AND DESIGN**

System Analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. Analysis specifies what the system should do.

System Design is a method of planning a new business system or replacing an present system by defining its components or modules to satisfy the specific requirements.

System analysis and design mainly focuses on −

* Systems
* Processes
* Technology

Under System analysis, study of the best possible approach is performed and to find any complications within the model under work. This will next lead to the design which lays out the blueprint of the implementation based on the results of the analysis.

Discussion about the method of flow based on one’s requirement along with the techniques that have been implemented are mentioned below.

**3.1 Workflow**

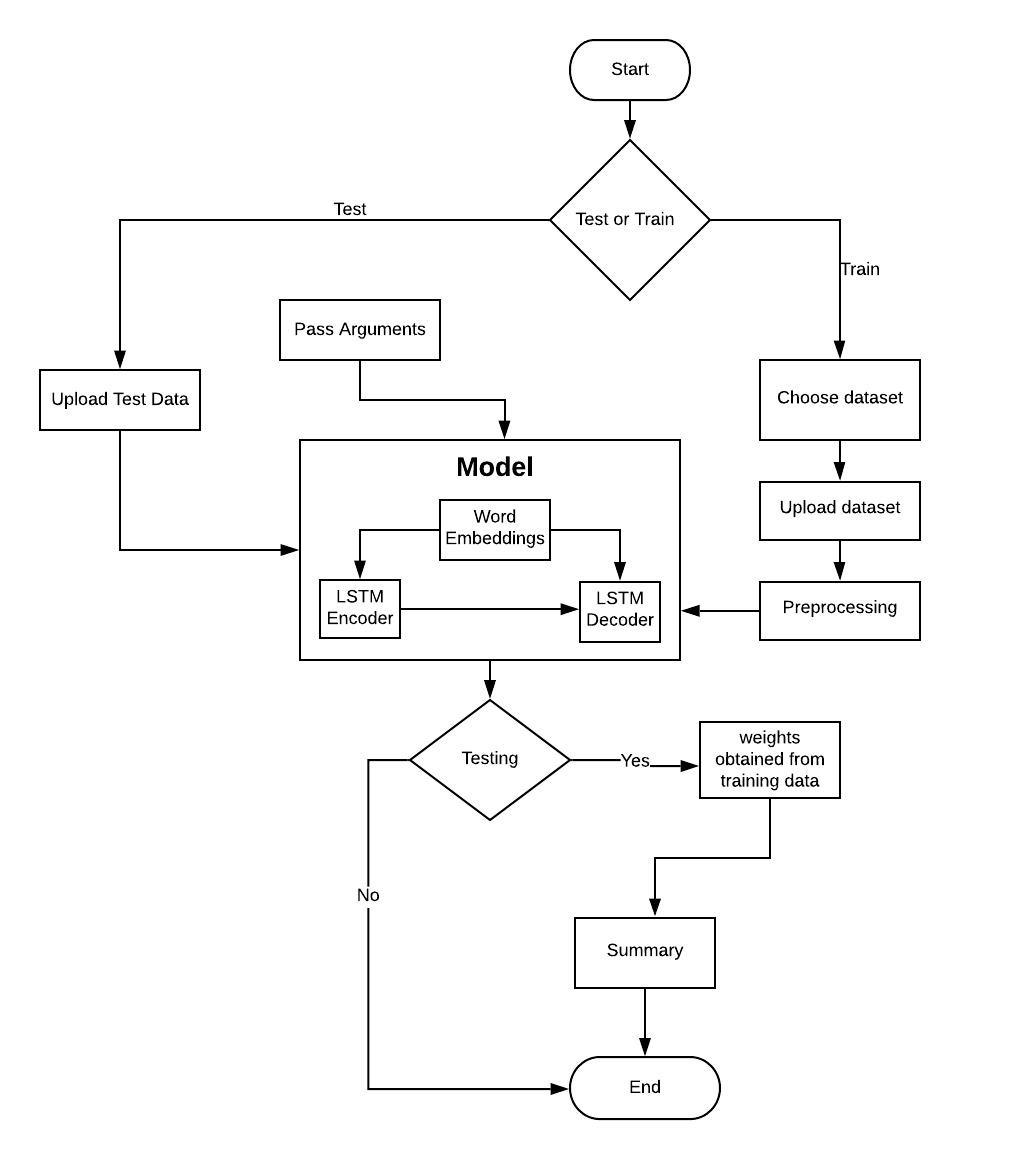
****

Figure 3.1: Workflow diagram

The Fig. 3.1 gives an outline of the sequential flow of the model from start to finish to finish and its decision variables are the ones that alter this flow based on the given input. The flow starts with choosing whether to train or test data, the process of training includes steps like choosing dataset to train on, upload the dataset, run a few preprocessing functions on it and then this dataset is fed to the model, now this model trained will provide the weights to the corpus values in the form of a datafile, this datafile in turn is used during the testing process ,before the prediction on test data we need to pass the test data to the model for the iteration study of the data provided.

The trained weights are used for the prediction on the test data to obtain the summary and it ends based on the choice of whether we need to proceed with testing or not. This ends the workflow.

**3.2 Sequence to Sequence Technique**

For the purpose of abstractive text summarization, **sequence to sequence modelling technique has been used.**

Sequence to sequence models rely on what is called an encoder-decoder architecture – a combination of layered RNNs that are arranged in a way that allows them to perform the tasks of encoding a word sequence and then passing that encoded sequence to a decoder network to produce an output.

The input sequence is first tokenized (transformed from a collection of words into a collection of integers that represent each word) and then fed word-for-word into the encoder. The encoder transforms the sequence into a new, abstracted state, which then, after being passed to the decoder, becomes the basis of producing an output sequence.

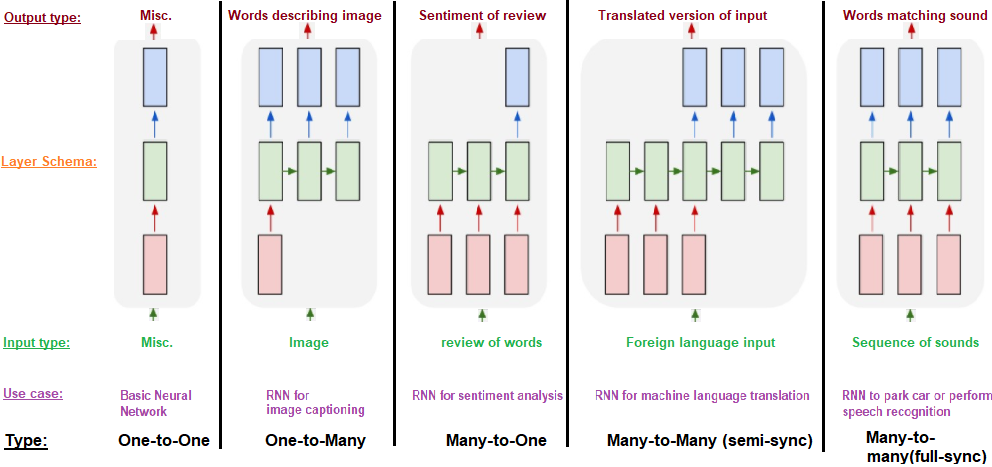


Figure 3.2: Types of Sequence to sequence models

Fig 3.2 shows the various types of sequence to sequence models - one to one, one to many, many to one and many to many. Many to many seq2seq techniques have been used.

Various types of models are used for different purposes. One-to-one is used for a basic neural network, one-to-many is used for image captioning, many-to-one is used for sentiment analysis, many-to-many is used for machine language translation and speech recognition.

Long short-term memory (LSTM) is an artificial [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN) architecture[[1]](https://en.wikipedia.org/wiki/Long_short-term_memory#cite_note-lstm1997-1) used in the field of [deep learning](https://en.wikipedia.org/wiki/Deep_learning). Unlike standard [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network), LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

The following are the modules incorporated in the system: -

1. **Encoder** - The encoder reads the entire input sequence word by word, producing a sequence of encoder hidden states. At each time step, a new token is read and the hidden state is updated with the new information. Upon reaching the end of the input sequence, the encoder puts out a fixed length representation of the input, regardless of input length, which is called the encoder vector. The encoder vector is the final hidden state that is used to initialize the decoder. LSTM has been used for the encoder.
2. **Decoder** - The decoder is trained to output a new, fixed-length sequence (word-for-word) given the previous word for each time step. It is initialized by receiving the encoder vector as its first hidden state, as well as a “start”-token, indicating the start point of the output sequence. The true output sequence is unknown to decoder while decoding the input sequence, it only knows the last encoder hidden state and the previous input (the “start” token or the next token from the input sequence), which it receives at each time step. The decoder has the ability to freely generate words from the vocabulary. LSTM has been used for decoder.

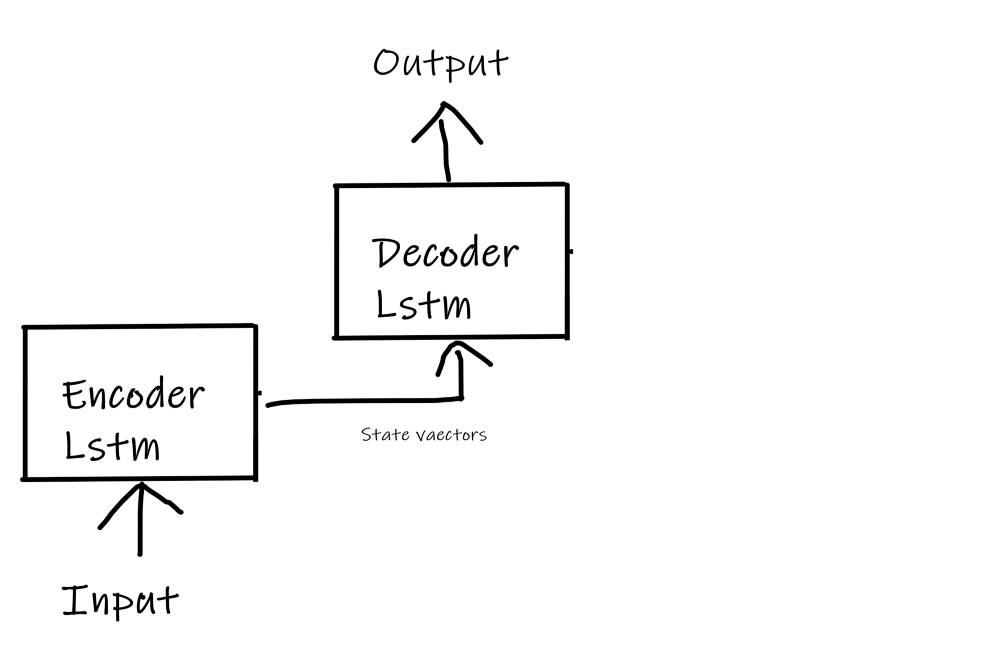


Figure 3.3: Encoder and Decoder Flow diagram

The Fig 3.3 is just a simple depiction on how Encoder-decoder process flow works. First the input is passed to the encoder system thereby getting state vectors. Then these are fed to the decoder to get the desired output.

1. **Attention** - Attention serves to assist the encoder-decoder model in specifically focusing on certain, relevant sections / words in the input text when predicting the next output token. This helps to mitigate the issue of lost context from earlier chunks of an input sequence. With Bahdanau attention (which has been used), instead of a one-shot context vector based on the last (hidden) state of the encoder, the context vector is constructed using **ALL** hidden states of the encoder. When combining the hidden states into the final encoder output (decoder input), each vector (state) gets its own random weight. These weights are re-calculated by the alignment model, another Neural Network trained parallel to the decoder which checks how well the last decoder output fits to the different states passed over from the encoder. Depending on the respective fit scores, the alignment model weights are optimized via back propagation. Through this dynamic weighting, the importance of the different hidden states varies across input-output instances, allowing the model to pay more attention (weight / importance) to different encoder states based on the input.

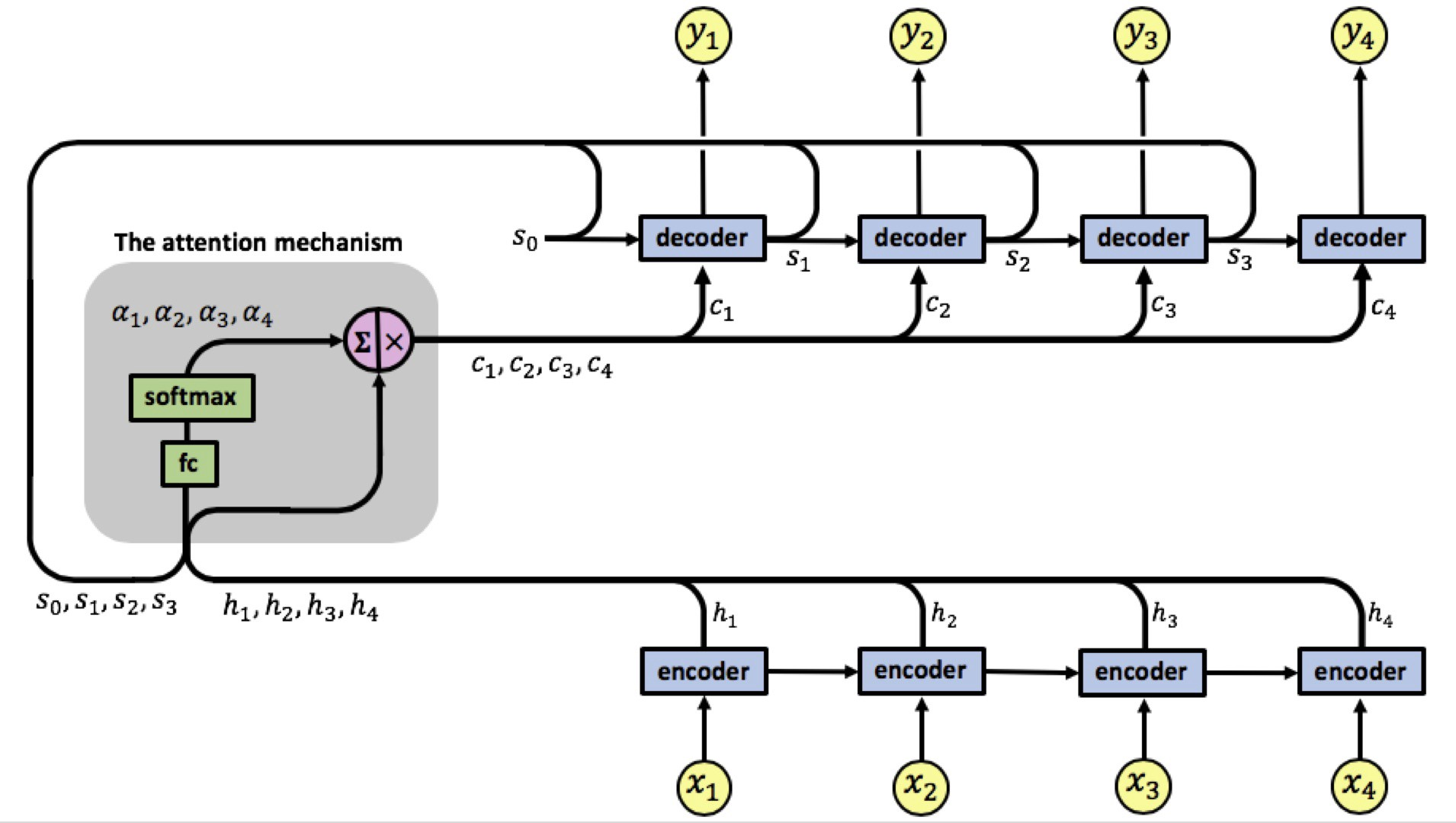


Figure 3.4: LSTM Model Representation

Fig 3.4 illustrates the components of LSTM model- encoder, decoder and the attention mechanism wrapping over it for an input with 4 tokens x1-x4.

1. **Word2vec -** It is the technique/model to produce **word embedding** for better **word** representation. It captures a large number of precise syntactic and semantic **word** relationships.

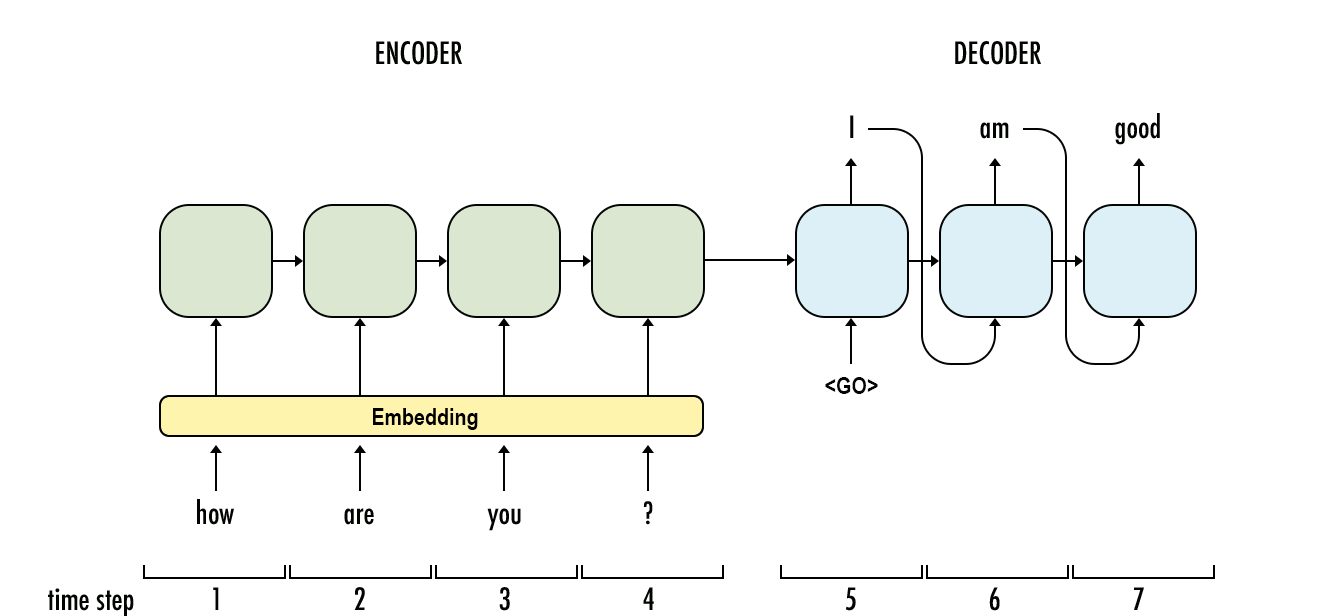


Figure 3.5: Encoder-Decoder Flow with Word Embedding

Fig 3.5 illustrates the mechanism of word embeddings being used in the system. Word embeddings map a sequence of words from vocabulary to vectors of real numbers in such a way that words with the same meaning have a similar representation.

**Chapter 4**

**MODELLING AND IMPLEMENTATION**

The Modelling and Implementation section includes the UML diagrams and the Pseudocode. The modelling section includes the UML diagrams that have been made to describe the various components of the project and their relationships along with the actors involved. The implementation section gives the pseudocode of the project.

## 4.1 Use Case Diagram

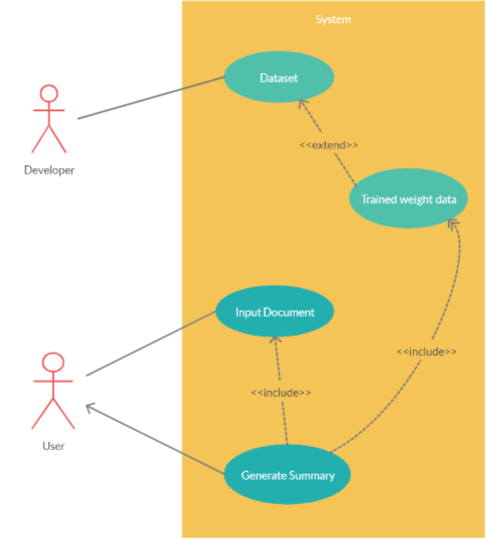
****

Figure 4.1: Use Case Diagram

Fig 4.1 provides a clear picture of the parties involved and their interaction with the system implicitly. The entities involved here are the developer and the user with the system, developer has the access to the system to make changes and upload the necessary dataset based on requirement and train this dataset to extend and obtain the trained weights. The user provides the testing data based on the domain of the dataset and this input testing data along with the trained weights will be included to generate the required summary from the model which will later be displayed to the user.

## 

## 4.2 Class Diagram

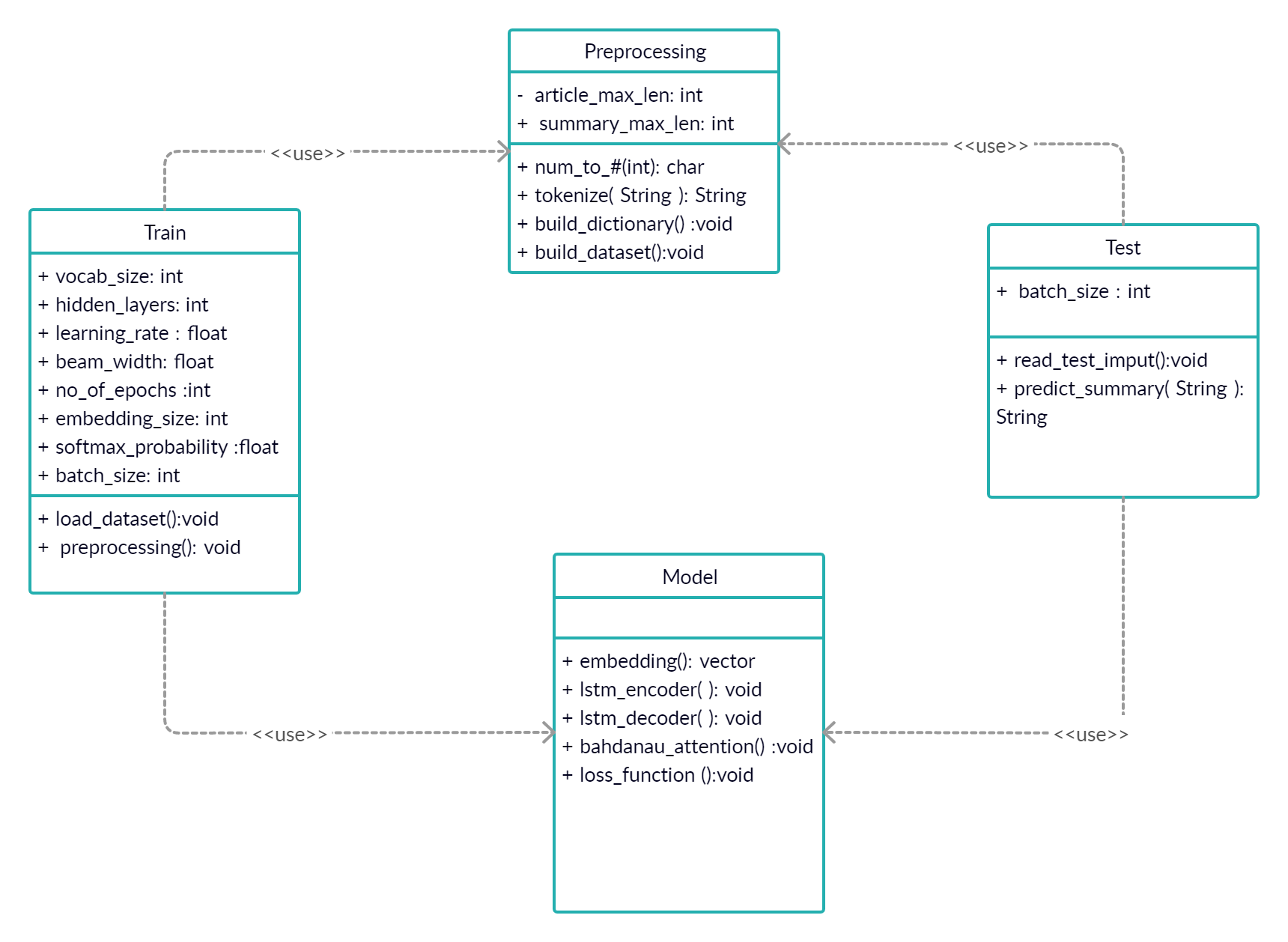
****

Figure 4.2: Class Diagram

Fig 4.2 indicates the various classes/parts of the project. The classes involved in the project are- preprocessing, model, train and test. During the training process, input is fed to the train class which in turn makes use of the methods of the preprocessing class. The train class is involved in the training of the model. The preprocessing class does the preprocessing steps like building the dictionary, building the dataset, word embeddings, paddings etc. During the testing phase, the input is fed to the testing class which in turn makes use of some of the methods of the preprocessing class. The test class generates the summary for the given input.

## 

## 4.3 Sequence Diagrams

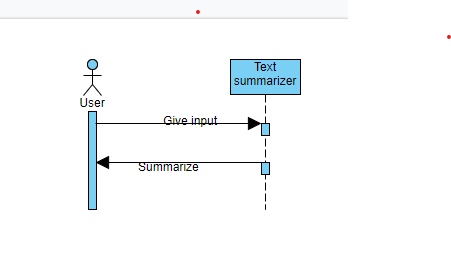
****

Figure 4.3: Sequence diagram of the project overview

Fig 4.3 shows the overall perspective of the project. Input article is provided by the user and the text summarizer generates the output summary. The flow is represented with the arrows and also their purpose of interactions.

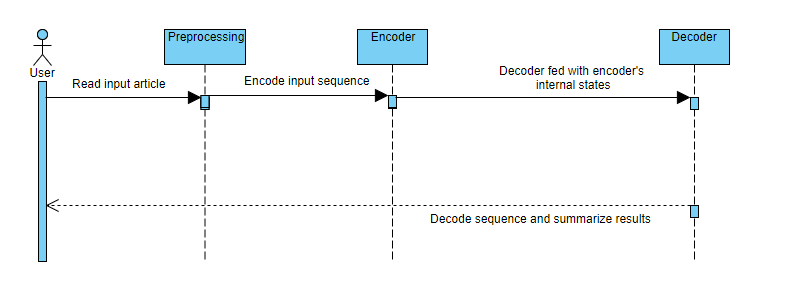
****

Figure 4.4: Sequence diagram of the project components

The fig 4.4 explains the interactions between the classes of the model from the model perspective. The input article is read by the model. Preprocessing is performed on it. Encoder encodes the input sequence and feeds to the decoder which decodes the sequence and generates the summary for the article.

## 4.4 Collaboration Diagram

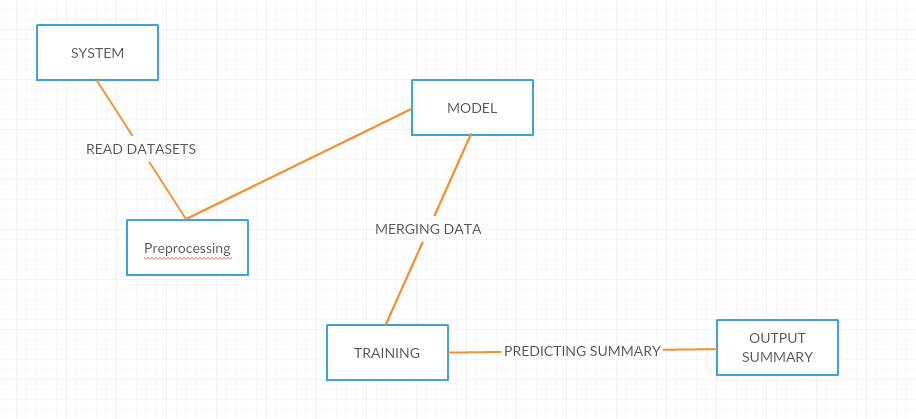


Figure 4.5: Collaboration Diagram of the project

Fig 4.5 illustrates the relationships and interactions among the objects in the project. The system reads the dataset and preprocesses it. Depending upon whether it is the training phase or testing phase, it either trains the model or generates the summary for the article. This diagram provides the entities that are interacting along with the outcome of their interaction for all the different entities involved.

## 4.5 Pseudo Code

### **Training process**

|  |
| --- |
| *import necessary libraries FUNCTION add arguments(parser)  Add necessary arguments that can be overridden ENDFUNCTION   open file args.pickle  dump argument values to pickle file IF path saved\_model does not exist then  create directory ELSE   Set path to previously saved model ENDIF Print "Build Dictionary" CALL build\_dict() Print "Loading train dataset" CALL build\_dataset() WITH tensorflow session as sess CALL Model() INITIALISE tensorflow session variable IF old\_ckpt in global variables Print "Continuing from previous trained model" restore previously saved model  print "Iteration starts" print "Number of batches per epoch" FOR batch x and y in batches: CALCULATE batch decoder variables  run tensorflow session Print "Loss value" Print "Epoch No" Print "Elapsed Time" ENDLOOP ENDWITH* |

### **Utilities**

|  |
| --- |
| *import necessary libraries   SET path for training and validation   FUNCTION clean\_str(sentence)  Pre-processing a sentence  RETURN sentence END FUNCTION   FUNCTION text\_list(file\_path,toy)  WITH file open as f  IF not toy  RETURN CALL clean\_str()  ELSE  RETURN CALL clean\_str() for 50000 lines  ENDIF  ENDWITH END FUNCTION     FUNCTION build\_dict(step,toy as FALSE)  IF step is "train"  Clean sentences for the data in data path    FOR clean sentence  Append tokenization  ENDLOOP  Enter dictionary for unknown values  WITH file open as word\_dict.pickle  Enter dictionary to pickle file  ENDWITH  ELIF step is "valid"  Load dictionary from pickle    ENDIF END FUNCTION   FUNCTION build\_dataset()  IF step is "train"  GET train data given in the data path  ELIF step Is "valid"  GET test data given in the data path*  *ENDIF    Tokenize the data END FUNCTION FUNCTION batch\_iter()  Pass inputs into array FOR epoch in range of num\_epochs  GET inputs and outputs ENDLOOP   FUNCTION get\_init\_embedding()  GET word2vec and word\_embeddings  Print "Loading Glove vectors"  FOR word in reversed\_dict.items()  Try  word\_vectors is in word\_vec(word)  ENDtry  ENDLOOP*  *ENDFUNCTION   Assigning vectors to tokens  RETURN numpy array with word\_vec\_list END FUNCTION* |

### **Model**

|  |
| --- |
| *import necessary libraries   CLASS Model FUNCTION \_\_init\_\_()  INITIALISE variables  Assign cell to BasicLSTMCell  INITIALISE decoder length,input,target,step,batch\_size WITH tf.name\_scope "embedding" EXECUTE EMBEDDING FUNCTIONS INITIALIZE encoder and decoder embedding inputs ENDWITH WITH tf.name\_scope "encoder" EXECUTE LSTM ENCODER FUNCTIONS COMPUTE Sequence length and output ENDWITH WITH tf.name\_scope "loss" EXECUTE LOSS FUNCTIONS COMPUTE gradient and loss values ENDWITH ENDFUNCTION* |

**Testing**

|  |
| --- |
| *import necessary libraries WITH file open as args.pickle  LOAD pickle file to pass arguments ENDWITH Print "Loading dictionary..." CALL Build\_dict("valid",args.toy) print("Loading validation dataset...") CALL build\_dataset()   WITH tensorflow as sess  Print "Loading saved model..."  CALL Model()  GET tensorflow global variables  LOAD saved model checkpoint and restore    Print "Writing summaries to 'result.txt'..."  FOR batch in batches:  CALCULATE batch\_len    INITIALISE valid\_feed\_dict for batch variables    RUN prediction model with dictionary  SAVE Prediction\_output    WITH file open as result.txt  FOR line in prediction  Print word in summary  ENDLOOP  Print summary line by line within file    print('Summaries are saved to "result.txt"...')  ENDWITH  print('Summaries are saved to "result.txt"...') ENDWITH* |

**Chapter 5**

**TESTING, RESULTS AND DISCUSSION**

Testing is very important for the process of validation ,this process is important to check if the expectations meet with reality, this process is very crucial for the understanding of the models anomalies and outliers based on various inputs, for which we use many types of testing like stress testing, parameter testing, variable input etc. Specifically, for machine learning models the results get better with progression in terms of training and the arguments passed by the user based on which the model is being trained.

**5.1 Testing based on sentence type**:

**Test Case 1**:

Actual summary - “us business attacks tough immigration law.”

Predicted summary - “us business leaders lash out at illegal immigration.”

**Test Case 2**:

Actual summary - “laura bush and rice to attend sirleaf 's inauguration in liberia.”

Predicted summary - “us first lady to attend liberian presidential inauguration.”

Table 5.1: Test Score Table

|  |  |  |
| --- | --- | --- |
| Test Case | BLEU Score | Rouge\_1 Score |
| 1 | 2.055 | 0.400 |
| 2 | 1.1003 | 0.2857 |

The table 5.1 is to illustrate and compare the scores obtained to be clearer on the effectiveness of the algorithm and how close it is to the original text, in this case BLEU and Rouge\_1 Scores.

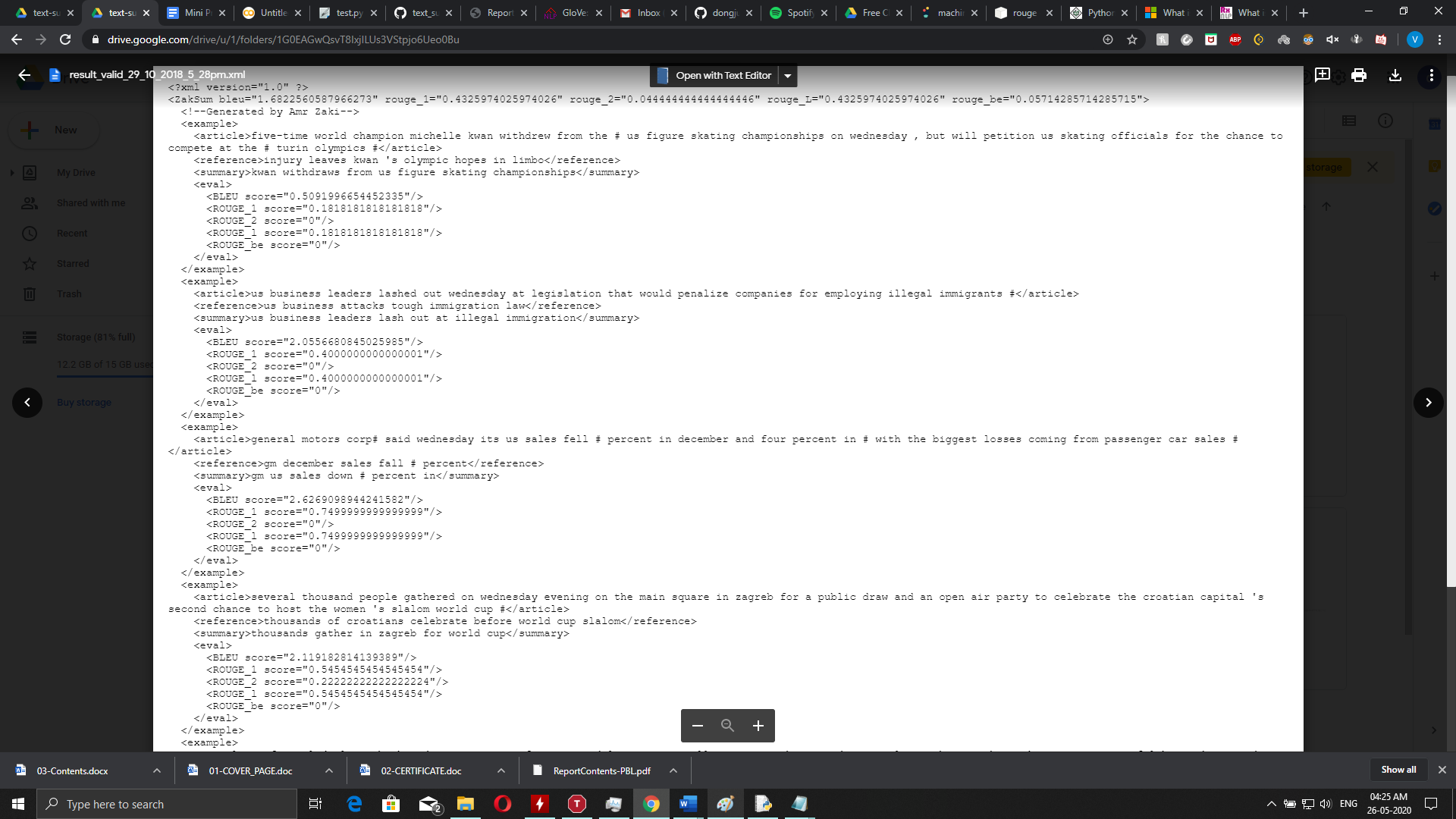


Figure 5.1: Test Score Analysis

The Fig 5.1 displays the raw output obtained while scoring the predicted data with the actual data here we have systematic data of different scores obtained for different predictions along with the display of actual headline, predicted headline along with the articles and score.

**Graph on Test Score Analysis**

Figure 5.2: ROUGE\_1 Vs BLEU Score Graph

Fig 5.2 is the graphical representation between ROUGE and BLEU scores which is useful to estimate how generated summaries are related with the reference summaries, the trend observed in this graph says that the one variable is progressively increasing with the other which states that they are proportional.

In conclusion the increase in similarity of the words between a generated and reference summary symbolizes the increase in the standard quality of the model predicted summary.

**5.2 Testing based on epoch:**

**Article:**

The white house vigorously rejected wednesday suggestions that us soldiers in iraq do not hesitate to fire on civilians, after a bombing killed eight iraqi non-combatants, including two children .

**Actual Summary:**

”us insists soldiers act with restraint to protect civilians”

**Results for epoch 2:**

white house refuses to fire troops in iraq.

**Result for epoch 31:**

white house denies killing of iraqi civilians.

The results obtained depend on the dataset used and the vastness of the dataset because the predicted data is much more accurate if the dataset is trained for longer periods of time, having skeptical examples in the dataset will help in also identifying such cases which are not generic in nature. Better results will be obtained if more tech was available on a working model which could also be executed on low resource modules. On a higher scale the dataset does not have to be restricted under one domain if one possesses the required processing power.

**Graph on Epoch Analysis**

Figure 5.3: Loss Value Vs Epoch No Graph

Fig 5.3 represents the relationship between Loss Value and Epoch No, while training it is seen that the dataset is processed in batches and the model goes through the whole dataset to complete one Epoch. During this process some information and weights that have been learned will undergo variations because of the different combinations possible with a similar set of words, this leads to a loss within the learning process which is displayed as loss value.

This can be changed by training the dataset and complete more cycles or Epochs, which in turn will reduce the loss value since more information is taken and studied by the model to come to a better conclusion than in the previous state which improves the accuracy of the predictions obtained.

Therefore, the above-mentioned content states that the Loss value reduces with the increase in Epoch No’s or training time.

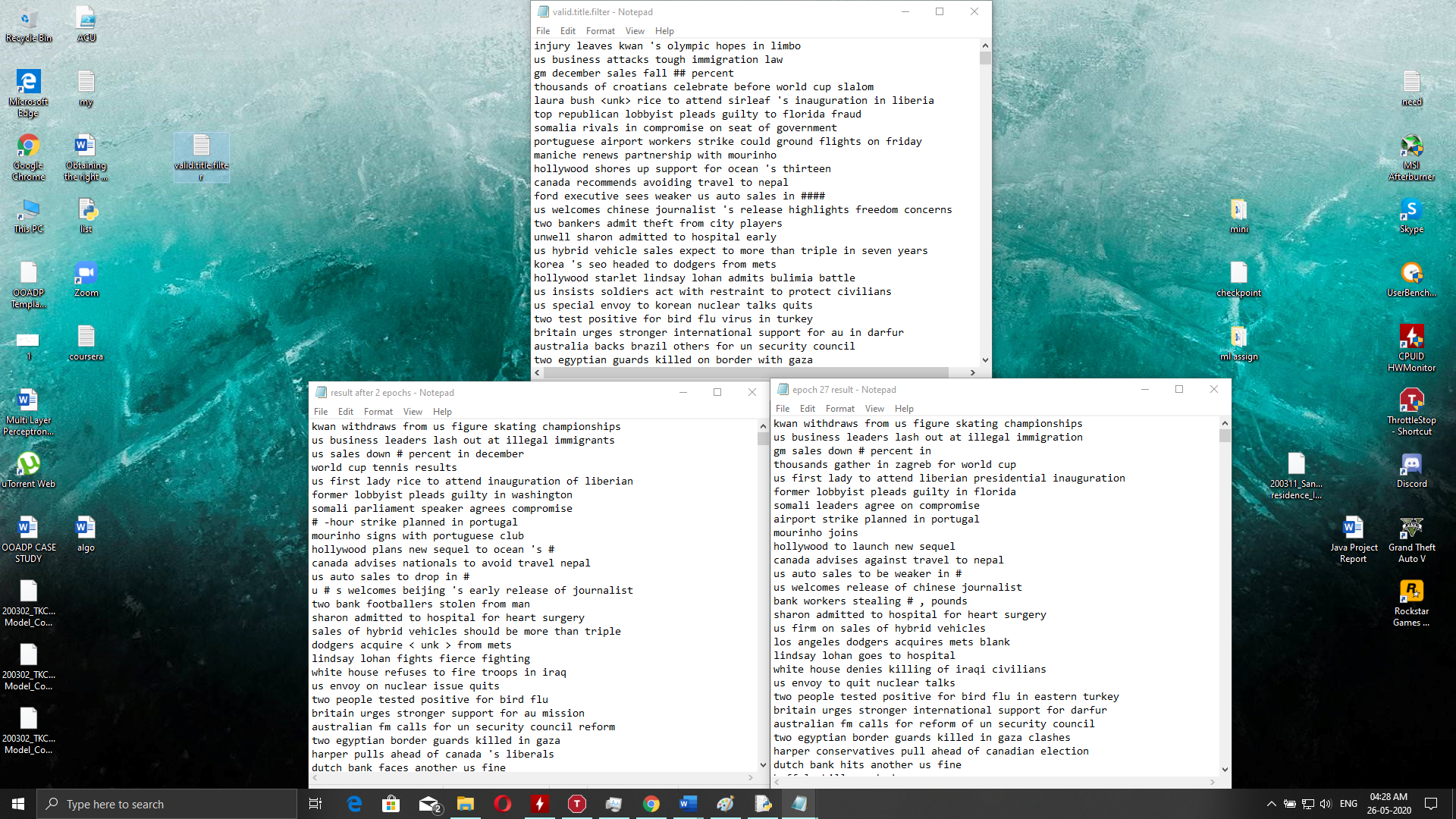


Figure 5.4: Epoch Analysis

The Fig 5.4 displays the predicted outputs for a set of data but are generated for different numbers of epochs or training cycles. It is observed that an increase in the number of epochs always helps in improvising the model and reduce the loss function progressively. So, if a dataset is given more time on training this leads to better results, the valid title which is the actual title is also displayed to compare the results obtained.

**5.3 Discussion about the results**

* The ROUGE and BLEU results show how close the predicted summary is to the actual summary and provides a metric to test their result and effectiveness of the model.
* The ROUGE-1 score indicates the unigram similarity between actual and generated summaries. BLEU score, is a metric for evaluating a generated sentence/summary to a reference sentence (actual summary).
* Since the summary is abstractive, it is important to notice that the difference between the two summaries compared is only in terms of order of words but not the meaning of the sentence.
* A decent BLEU score indicates that there is a certain degree of similarity. However, a not very high score shows that new sentences are being generated.
* It can be observed that the summaries are grammatically correct and coherent.
* We can decrease the loss function by increasing the number of epochs and increasing the training size.
* Some words which are present in test data will be shown as UNK because they won't be assigned weights as they are not present in the trained dataset.
* If there are more than 3 input sentences in a particular test dataset headline, then there will be a problem in reference resolution where pronouns etc are not identified.
* As numbers are not logically right for prediction sentences. So, we replaced them with # for each number as there is no specific pattern and keep varying and hence not useful during training.

**Chapter 6**

**CONCLUSION**

It has been observed that abstractive text summarization using the sequence modelling technique performs better in capturing the content of the article and generating grammatically correct sentences. It produces a coherent, less redundant and information rich summary.

It also generates new sentences which summarize the news articles well. Also, the technique performs well on relatively short articles.

It doesn’t produce very accurate results for large articles. It struggles for out of vocabulary words which are unknown to the text summarizer. It is used mainly for generating summaries of around one sentence like newspaper headlines.

We can also improve the model by increasing the training size. Pointer generator networks and coverage mechanisms could be used to improve the performance even further.

In order to capture more meaning into the inputs fed into the encoder, we can go beyond the word embedding representations like GloVe.

For future scope we can integrate Abstractive text Summarization in various applications for ease of use. A user-based application where summarization occurs on server and result is displayed to the user, use larger and different datasets to predict more accurately and any type of data given.

**REFERENCES**

[1] M. Aref, I. Moawad, S. Ibrahim., "Rich Semantic Graph Generation System Prototype", The tenth Conference on Language Engineering, Cairo, Egypt, 2010.

[2] I. Moawad, M. Aref, S. Ibrahim, "Ontology-based Model for Generating Text Semantic Representation", the International Journal of Intelligent Computing and Information Sciences “IJICIS”, Vol. 11, No. 1, pp. 117- 128, January 2011.

[3] C. Fellbaum, "WordNet: An Electronic Lexical Database", MIT Press, 1998.

[4] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller, Five Papers on WordNet. Cognitive Science Laboratory, Princeton University, Princeton, 1990.

[5] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks.

[6] K. Arkhipenko, I. Kozlov, J. Trofimovich, K. Skorni-akov, A. Gomzin, and D. Turdakov. 2016. Compar-ison of neural network architectures for sentiment analysis of russian tweets.

[7] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large anno-tated corpus for learning natural language inference.

[8] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures.

[9] Dzmitry Bahdanau, K.Cho, Y.Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. International Conference on Learning Representation (ICLR), 19 May 2014.

[10] Sutskever et al “Sequence to Sequence Learning with Neural Networks”. Conference on Neural Information Processing Systems (NIPS,2014).

[11] Cho, Kyunghyun. et al. (2014) “Learning Phrase Representations using RNN Encoder Decoder”. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[12] H.T Le, T.M Le –“An approach to abstractive text summarization“,Soft Computing and Pattern Recognition (SoCPaR), IEEE-2013.

[13] R Bhargava, Y Sharma, G Sharma – “ATSSI: Abstractive Text Summarization using Sentiment Infusion”, Procedia Computer Science, Elsevier-2016.

[14] A Khan, N Salim, YJ Kumar – “Genetic Semantic Graph Approach for Multi Document Abstractive Summarization“, Digital Information Processing and Communications (ICDIPC), IEEE-2015.

[15] A Khan, N Salim, H Farman – “Clustered Genetic Semantic Graph Approach for Multi-document Abstractive Summarization”, Intelligent Systems Engineering (ICISE), IEEE-2016.

[16] T. J. O’Shea, J. Corgan, and T. C. Clancy, “Convolutional radio modulation recognition networks,” in Proc. of EANN, Aberdeen, U.K., Sep. 2016.

[17] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, “Learning to optimize: Training deep neural networks for wireless resource management,” arXiv preprint arXiv:1705.09412, 2017.

[18] W. Lee, M. Kim, and D. H. Cho, “Deep power control: Transmit power control scheme based on convolutional neural network,” IEEE Commun. Lett., vol. 22, no. 6, pp. 1276–1279, 2018.

[19] T. O’Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563–575, 2017.

[20] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023, 2016.

[21] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer generator networks. arXiv preprint arXiv:1704.04368, 2017.

[22] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017.