QUIZ GENERATION

A Project Report

Submitted by

Vivek R. Bhave	**111508015**
Shreyansh R. Gopawar	**111508073**
Aditya D. Neralkar	**111508076**

in partial fulfillment for the award of the degree

of

Information Technology

Under the guidance of

Dr. Y. V. Haribhakta

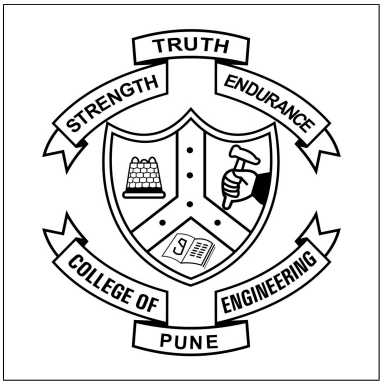
College of Engineering, Pune

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE-5



April, 2019

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

Certified that this project, titled "QUIZ GENERATION" has been successfully completed by

Vivek R. Bhave	**111508015**
Shreyansh R. Gopawar	**111508073**
Aditya D. Neralkar	**111508076**

and is approved for the partial fulfillment of the requirements for the degree of "B.Tech. Information Technology".

SIGNATURE
Dr. Y. V. Haribhakta
Project Guide
Department of Computer Engineering and Information Technology, College of Engineering Pune, Shivajinagar, Pune - 5.

SIGNATURE
Dr. V. Z. Attar
Head
Department of Computer Engineering and Information Technology, College of Engineering Pune, Shivajinagar, Pune - 5.

Abstract

Asking questions to students has always been regarded as the best method to gauge the students learning capabilities. However, with the amount of content available, generating questions from domain experts, is a time consuming and expensive process.

The task of Automatic Question Generation(AQG) aims to generate questions from a given piece of text which shall appropriately test the students mastery over the material. Plenty of research in the area of AQG has been done over the years. The current state of the art models use a bidirectional Long Short Term Memory encoder decoder system to solve the problem. The work presented here carries out an important task in one of the stages of the entire AQG system.

The model we have designed takes the sentence to generate questions on, as input, and tags a few words of the sentence as answer words on which questions can be created. The model has been trained using the SQUAD and Google datasets. The model has been created using LSTM networks and other linear layer neural networks.

As a part of the system, we also built a custom web crawler. On input of a specific topic to the crawler, it can find a list of relevant web pages and is also able to filter in the important content from those webpages. Thus, this is an end to end system, which can be used in many areas like education, quizzing, entertainment, etc.

Contents

List of Tables

List of Figures

List of Symbols

Chapter 1

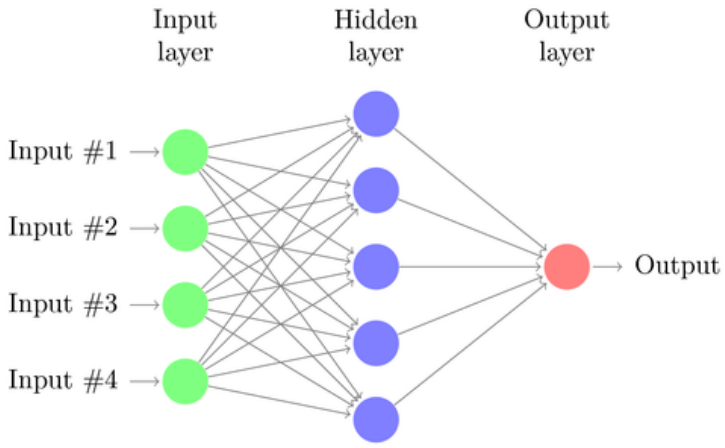
Introduction

1.1 Importance of Automatic Question Generation (AGS) Systems

Gauging a students knowledge over a certain topic is one of the fundamental problems in education systems. Quizzing students has long been accepted as one of the best techniques to solve this problem. Studies conducted over the last several decades have found that providing students with frequent and good number of quiz questions leads to better understanding of the topics, than spending an equal amount of time studying notes or textbooks. Quizzes can be used at several places in the education domain. They can be used in Massive Open Online Courses (MOOC) to test if a student has grasped the concept properly, they can be used as standalone tests in universities, or they can also be used as practice by students to discern their command over the subject.

However, with the volume of information available, generating questions from domain experts is an expensive and time consuming process. The problem of Automatic Question Generation (AQG), in the field of Natural Language Processing, aims to generate questions from a given text, such that students are appropriately tested on their mastery over the subject.

Figure 1.1: Basic Neural Network Diagram



1.2 Neural Networks

A neural network is an interconnected collections of neurons. The connections between the neurons are modelled as weights and the final value that the neuron stores is the weighted summation of all the inputs. A neural network consists of an input layer, an output layer and multiple hidden layers.

1.3 Recurrent Neural Networks (RNN)

Recurrent neural network is a type of artificial neural network. Neural networks had a shortcoming that that they could not remember previous inputs given to the network. This shortcoming was solved by recurrent neural networks with the help of hidden layers. Because of their internal memory, RNNs are able to remember important things about the input, which allows them to make precise predictions of the input that is yet to be processed. Sequential data is an ordered data where related things follow each other. RNN's have proved extremely useful while working with sequential data.

Unlike the normal feedforward neural networks, recurrent neural networks feed the output of previous step to the current step. This helps the network to memorise the previous inputs. Thus RNN cycles through the information

Figure 1.2: Recurrent vs Feed-forward Neural Network

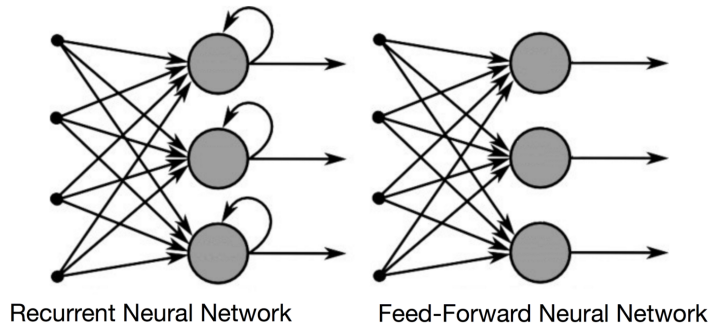
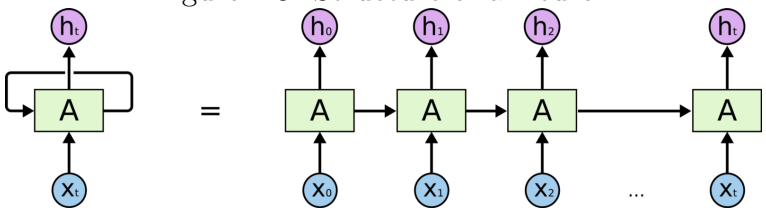


Figure 1.3: Structure of a Neuron



and when taking a decision it takes into consideration the current input as well as whatever it learned from the previous output. As an example, in the word 'Teacher', by the time a feed forward neural network reaches 'c', it has forgotten about the previous inputs 't', 'e' and 'a'. A RNN has the power to remember that. Recurrent Neural Network adds immediate past to the present. This is important since past data contains information about what is coming next.

RNNs apply weight to current and previous input and tweaks their weights using either the Gradient Descent or Backpropagation Through Time algorithms. Gradient Descent is an algorithm that is used to iteratively minimize a given function. Backpropagation Through Time (BPTT) is performing backpropagation on an unrolled RNN. Unrolling is a visualization and conceptual tool, which helps you to understand whats going on within the network. RNN can be viewed as a sequence of Neural Networks that is trained one after another with backpropagation.

The diagram above shows the RNN being unrolled after the equal sign.

The different timesteps are visualized and information gets passed from one timestep to the next. Within BPTT the error is back-propagated from the last to the first timestep, while unrolling all the timesteps. This allows calculating the error for each timestep, which allows updating the weights.

1.3.1 Problems with RNN

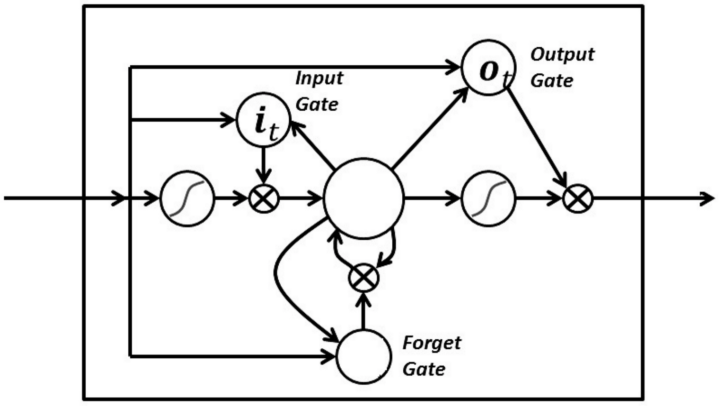
Although RNN's are quite powerful because of their inherent ability to remember previous inputs while processing each new one, they have to be used with care as a few of the following problems might occur.

1. Training RNNs is a difficult task.
2. **Exploding gradient problem :** Exploding gradient is when the algorithm assigns an extremely high importance to the weights, without much reason. But fortunately, this problem can be easily solved by truncating or squashing the gradients.
3. **Vanishing gradient problem :** Vanishing gradient is when the values of a gradient are too small and the model stops learning or takes way too long. This problem is solved by the concept of LSTM.

1.4 Long Short Term Memory

Long Short Term Memory (LSTM) is an artificial recurrent neural network. LSTM can be considered to be an extension to the recurrent neural network, as LSTM has extended memory as compared to RNN. Thus an LSTM is capable of learning things that have very large time lags between them. The memory used by an LSTM can be attributed to a computer memory as it has functions like read, write or delete information. This is quite different from that of a RNN which uses a vector.

Figure 1.4: Structure of a Basic LSTM node



The memory in LSTM are gated cells. Gated cell means that the cell decides whether or not to store or delete information based on the importance it assigns to the information. The assigning of importance takes place through weights which are learned through the algorithm.

A basic LSTM node consists of input gate, output gate and forget gate. The input gate decides whether or not to let new input in. The forget gate deletes the information because it is not important. The output gate impacts the output at the current time step. The gates in a LSTM are analog, in the form of sigmoids, meaning that they range from 0 to 1. The fact that they are analog, enables them to do backpropagation with it.

The problem of vanishing gradients which is very common in recurrent neural networks is solved by LSTM as it keeps the gradient steep enough and therefore the training short and the accuracy high.

Chapter 2

Literature Survey

1. **Computational Intelligence Framework for Automatic Quiz Question Generation**

   The model described in this paper, uses a rule based approach and is able to generate three types of questions, namely, Fill in the Blank, True/False, and Wh Questions. The training phase of this algorithm involves, detecting topically important sentences, NER and POS Tagging, sanitizing the input sentence and then constructing a tree for further processing of each type of question. For Fill in the Blank, it gives weight to each word involved and marks important words for setting up blanks. Thr true or false sentence involves identifying modal verbs, and also factual type of sentences and further rule based processing. For Wh questions, patterns identified during the training phase are matched, and used to generate questions.

2. **Automatic Question Generation System**

   This is a traditional supervised learning approach and the model can be broken down into a sequence of processes. The text input is stemmed to get the meaning of each word and then passed to the Phrase Mapper. Then the key phrases are extracted using pre trained documents and the

document is summarized. In the end, Nouns are filtered in, on which questions can be generated.

3. **Automatic Question Generation for Intelligent Tutoring Systems**

   This system generates only Multiple Choice Questions, and has been trained on Wikipedia articles. The wikipedia articles are processed and unigrams, bigrams are extracted. The keywords from them are then extracted and their weights using the TF IDF weighting scheme are calculated. Using the above distractors can be generated. Whenever a query is fired to generate questions on, the wikipedia articles of the keyword is taken up, and rule based methods are applied to the sentences to create questions and presented to the users including the distractors as other viable options.

4. **Automatic Question Generation from Childrens Stories for Companion Chatbot**

   This paper was specifically targeted for generating questions from childrens stories, and was trained and tested on much lesser data than others. The model is said to work in two parts, that is one for question generation and the other for ranking of the questions. The model uses the part of speech tags and dependency parsing algorithms along with pre trained language rules for the question generation phase. For ranking of the questions, the model uses logistic regression to determine the acceptability probability of the question.

5. **Deep Guessing**

   Generating Meaningful Personalized Quizzes on Historical Topics by Introducing Wikicategories in Doc2Vec: The aim of this model is to load

all of the wikipedia articles, classify them into categories, and use the knowledge from the above, for generation of distractors in multiple choice questions. The model works on a novel idea of creation of paragraph vectors, whose advantage is that they are trained from unlabeled data and thus can work well for tasks which do not have enough labelled datasets. After the score of wiki categories is obtained, thresholds are decided to classify the records into categories and thus, are eventually used for the creation of meaningful options in the questions generated.

6. **Automatic question generation on the basis of the discourse connectives**

   This problem of question generation has been divided into two modules in this paper. The first part is that of Content selection and the next that of Question formation. The Content selection phase consists of recognizing the part in text that is relevant and important to generate questions on. The second phase of Question formation includes several subtasks like Word Sense disambiguation of the discourse connectives then the Identification of the type of question to be created and eventually applying syntactic transformations on the context. The paper mainly takes into consideration the seven main discourse connectives - although, as a result, because, for example for instance and since. Using the output generated so far, the type of question gets decided and then the question can be formed.

7. **Semantic Based Automatic Question Generation**

   The paper explains a system that applies two fundamental Natural Language Processing concepts namely Semantic Role Labeling and Named Entity Recognizer technique. These tasks are used to convert the in-

putted sentence to semantic pattern. The model described in the paper has developed a system which has identified patterns for each type of question. The question types under consideration here are all of the Wh-questions - who, when, why, where, what. The system for classification uses learning, storage memory, feature extraction, and associative Retrieval.

The sentence given as input will be first parsed using Named Entity Recognition and SRL technique. The output of these two algorithms has a direct correlation with the exact question type to be created. Thus after the question type identification, the question pattern is known using the pretrained rules.

8. **Automatic Multiple Choice Question Generation System for Semantic Attributes Using String Similarity Measures**

The paper has described a model which first selects a factual sentence and an important word to generate questions on from the text given as input. This selection is done on the basis of the semantic labels and named entities in the sentence. Then for actually generating a question the SRL and NER tag is used, which directly helps in finding out the type of question.

Then the model focuses on finding distractors i.e. the incorrect options of a question. For this task different similarity measure between sentences of the data set is taken into account. Eventually, when a question is generated, the measures of similarity between the actual question and the sentences of the input text is considered and sorted. The top three sentences obtained from the above procedure, are considered for finding a relevant important word to the question generated which are eventually

used as distractors.

Table 2.1: Comparison Table

Sr No.	**Algorithm**	**Methodology**	**Type of Question**	**Evaluation of Result**
1.	Computationally Intelligent Techniques	Rule Based Approach - Training for patterns and then generate	Fill in the Blanks, True/False, Wh questions	Manual checking for finding difference between Automatic Questions and Domain Experts - users found difference
2.	Automatic Question Generation System	Series of pre processes like stemming, summarize, noun extract, and then rule based question generation	All Wh questions along with type like - Person, Definition, Procedure, Consequence, etc	Compression and Omission Ration considered - acceptable results

3.	AQG for Intelligent Tutoring Systems	Keywords from Wikipedia and then TF-IDF, then rule based	MCQs along-with distractors	Manual evaluation with domain expert
4.	AQG from Childrens Stories for Companion Chatbot	POS Tagger, Dependency Parsing and Logistic Regression	Only Wh questions	Manual evaluation with over 50% acceptance
5.	Deep Guessing: Quizzes on Historical Topics by Wikicategories in Doc2Vec:	Generating paragraph vectors from wikipedia articles, categorization for distractors	MCQ Questions with distractors	Manual evaluation of 45 MCQs
6.	AQG on the basis of the discourse connectives	Content selection and Question formation	Question generation like Why, when, where, in which	Manually evaluated for semantic and syntactic soundness of question by two evaluator

7.	Semantic based AQG	Feature Extraction using NER, SRL, Sentence pattern finding and get question type pattern	Only Wh questions	Manual evaluation with small dataset, About 87% accuracy
8.	AQG System for Semantic Attributes Using String Similarity Measures	Extracting sentence, check viability by similarity of sentences and select top three sentences	Only MCQ type questions	Manual evaluation with 75% success.

Chapter 3

Problem Statement

The problem of question generation, is currently solved by domain experts, who create questions over text or topics by their intuition or knowledge. Automatic Question Generation can solve this laborious, time consuming task quite efficiently and with much higher accuracy. Such an automatic question generation system can have applications in multiple domains.

3.1 Motivation

The main motivation of designing an Automatic Question Generation System, was to contribute in the education domain. Quiz questions can be of utmost importance for teachers to ensure that students are able to understand the material taught in class. Similarly quizzes can be used by students to find out their preparedness of the subject at hand.

3.2 Objectives

The objective of the project is to devise a system to generate a set of questions based on a topic provided by the user. The questions generated by this system, should be similar to questions generated by a domain expert. The difficulty level of the questions generated, should be such that they appropri-

ately test the knowledge of the user.

Chapter 4

System Requirement Specification

The system uses the following technologies and libraries relating to them -

Technology:

- Python 3.6

Libraries:

- Numpy
- Scikit
- Keras
- TensorFlow
- Torch
- Pytorch

The hardware specifications of the system used for running experiments are -

- Nvidia GEFORCE RTX 2080Ti
- Ram 16 GB

- 512 SSD
- 2 TB HDD

The dataset used in training, validation and testing of the model was the SQUAD 2.0 (Stanford Question Answering Dataset). This is the most recent and state of the art dataset used as a benchmark for testing Automatic Question Generation Systems.

Chapter 5

System Design

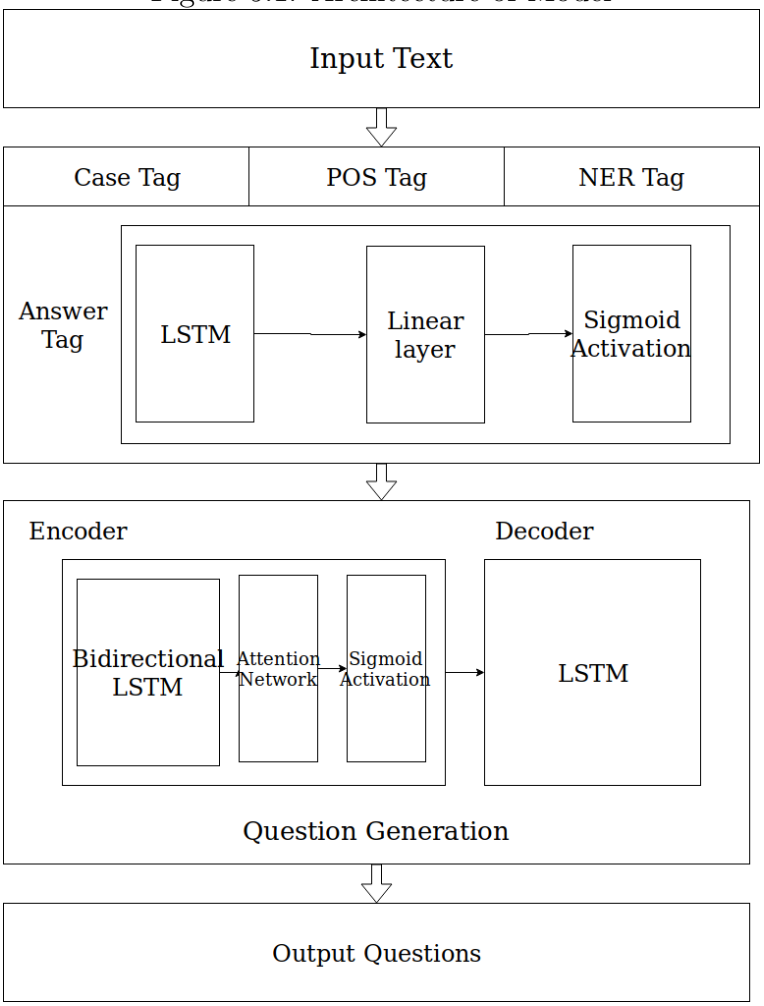
The system designed can be broken down into subsystems performing the tasks of receiving a topic as input, finding out relevant documents, tagging words as probable answer words, and eventually generating questions from it. Each of these subsystems have been described below.

5.1 Topic Input & Data gathering

Initially, the software shall accept any topic as input from the user. The first step of the process is to crawl the web and gather links to various web pages relevant to the topic given by the user. On obtaining the links, the second step involves scraping those links and parsing the HTML pages to get the information content in those web pages.

The content thus received can be classified into three parts. The first part consists of markup language tasks and syntax, the second part shall contain data unrelated to the topic of interest, and the third shall be the data which is both useful and relevant to the topic of interest. In this step, an algorithm has been written which searches the data related to markup language tags and eliminates the same from the scraped data. The algorithm then identifies unrelated data by taking into consideration the semantic meaning of the

Figure 5.1: Architecture of Model



words. After the elimination of such, the useful and relevant data obtained is fed as input to the next subsystem.

5.2 Pre processing block

The processing block consists of four independent tasks. The output of each of these four tasks works as features for the question generation algorithm. The four tasks and their work can be summarized as follows -

1. Case Tagging - This feature determines whether the first letter of the word is in lower or upper case. The output of this case thus determines whether the word is the starting word of the context. In quite a few cases, the starting word of the context dominates in meaning, and hence has been considered as a feature in the problem.

2. NER Tagging - The 7 set Stanford NER Tagging is used in the system. The NER Tagging helps in finding out the type of a noun to which the word belongs, i.e if it is a place, a date or a person. Depending on the tag different kinds of 'Wh' questions are created and hence the feature is of great significance in the problem.

3. POS Tagging - POS tags are important in generating patterns based on their frequency in the contexts. Frequently occuring series of consecutive POS tags that form answer words in the training sentences are most probable to be the best sources for generating questions in test contexts as well.

4. Answer Words - This feature is the most important of all the four features. This feature identifies whether the word in consideration could be a potential answer word, on which a question can be formed. Only

those words that are answer words are given paramount importance in the question generating process.

In order to achieve this goal, a model containing a LSTM and dense layers is developed. The output of the model is binary string which is 1 for the Answer Tag and 0 for non answer words. The output of this preprocessing subsystem is fed to the Question Generation Algorithm subsystem as input.

5.3 Question Generation - Algorithm

The Question Generation Algorithm performs three major tasks -

- **Context Reader and Encoder**

  Context reader is a bi-directional long short-term memory (bi-LSTM) network with 600 hidden layers. Bi-LSTM processes the input words in both the forward and backward direction. Encoder creates context word vectors from the preprocessed input provided.

- **Decoder and Softmax activation**

  The question-generator generates a question word one-by-one by following two internal calculations. The unidirectional LSTM in the decoder maps the current question word with fixed-sized vector, which is the hidden state of the network. Then the second calculation, i.e. the softmax function calculates the probability distribution over all words from a fixed question vocabulary.

- **Relevance of Question**

  To select the relevant question, pointer network is used. Pointer network calculates the output word probabilities as a mixture of two probabilities,

one over the question vocabulary and the other over the input context vocabulary

5.4 Output

This is the last subsystem of the process. Here, the system shall generate questions on either the topic of interest specified to the user or the text given as input to the system. The system can generate three types of questions. The first type is Fill in The Blank questions, where a factual or important word is left blank in a sentence from the input text. The second type is a one word answer, where an important sentence from the input text, is converted to a question and presented to the user. Finally, the system can also generate reading comprehension type of questions, wherein the input text on which questions are generated is made available to the user while he/she is answering the questions.

Chapter 6

Timeline Required for Overall Implementation

Project Stage I (Till December) - Finalisation of Problem Statement, Literature Summary, Dataset Discovery

January, February - Implementation and Analysis of existing systems

February, March - Proposal of our own Model to solve the Problem Statement

April - Running experiments, Generating results, Publication Work,

Chapter 7

References

During the literature summary an exhaustive survey of the research already conducted in Automatic Question Generation Systems was conducted. Following are a few of the research papers, we referred -

1. Deep Guessing: Generating Meaningful Personalized Quizzes on Historical Topics by Introducing Wikicategories in Doc2Vec `https://ieeexplore.ieee.org/document/8501891/`

2. Automatic Question Generation for Intelligent Tutoring Systems `https://ieeexplore.ieee.org/document/8066538/`

3. Context Aware Restricted Tourism Domain Question Answering System `https://ieeexplore.ieee.org/document/7877473/`

4. Automatic Question Generation System `https://ieeexplore.ieee.org/abstract/document/6996216/`

5. Automatic Question Generation from Childrens Stories for Companion Chatbot `https://ieeexplore.ieee.org/document/8424749/`

6. Computational Intelligence Framework for Automatic Quiz Question Generation `https://ieeexplore.ieee.org/document/8491624`

7. QG-net: a data-driven question generation model for educational content `https://dl.acm.org/citation.cfm?id=3231654`

Among other things, we would also like to add a few of the online resources we used to learn concepts of Deep Learning in Natural Language Processing, which were of foremost importance in the design and implementation of the model we have proposed.

1. https://www.coursera.org/learn/neural-networks-deep-learning?specialization=d learning
2. https://www.coursera.org/learn/nlp-sequence-models
3. https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/
4. https://cloud.google.com/storage/docs/gsutil_install

Chapter 8

Publication Details

Our research work over the year, has been drafted in the form of a journal paper, and has been submitted for review at the

Conference.