

QUESTION ANSWERING SYSTEM USING TRANSFER LEARNING FOR LEGAL DOCUMENTS

A Project Report

Submitted by

Manas Joshi	111703026
Atharva Lohangade	111703033
Atharva Mundada	111707036

*in partial fulfilment for the award of the degree
of*

B.Tech

Computer Engineering

Under the guidance of

Dr. Yashodhara V. Haribhakta

College of Engineering, Pune



**DEPARTMENT OF COMPUTER ENGINEERING
AND
INFORMATION TECHNOLOGY,
COLLEGE OF ENGINEERING, PUNE-5**

May, 2021

**DEPARTMENT OF COMPUTER ENGINEERING
AND
INFORMATION TECHNOLOGY,
COLLEGE OF ENGINEERING, PUNE**

CERTIFICATE

Certified that the project titled, “**Question Answering System using Transfer Learning for Legal Documents**” has been successfully completed by

Manas Joshi	111703026
Atharva Lohangade	111703033
Atharva Mundada	111707036

and is approved for the partial fulfilment of the requirements for the degree of “B.Tech. Computer Engineering”.

Dr. Yashodhara V. Haribhakta

Project Guide

**Department of Computer Engineering
and Information Technology,
College of Engineering Pune,
Shivajinagar, Pune - 5.**

Dr. Vahida Z. Attar

Head

**Department of Computer Engineering
and Information Technology,
College of Engineering Pune,
Shivajinagar, Pune - 5.**

Abstract

Legal Documents are intricate in nature and characterized by long sentences full of nominalizations, passives, oddly inserted adverbials, prepositional phrases and archaic vocabulary. These are typically criticized for being overly complicated, dense, repetitive, and outdated. The complexity of legal documents makes it difficult for lawyers and other legal professionals to respond to problems in a timely and precise manner. It is even difficult to understand these documents for ordinary people who do not know the law.

Question Answering (QA) systems are one of the effective means in Natural Language Processing (NLP) that can deal with such complex documents. In QA systems, a machine learning-based program produces answers to questions from the knowledge base or text paragraphs within a wide range of tasks including information retrieval and entity extraction. These systems provide accurate responses to requests submitted to them, which makes them better than search engines and, consequently, providing an easy way to obtain domain-specific knowledge. Traditionally, most research into QA systems used conventional language NLP techniques. However, recent developments in deep learning, transfer learning, neural networks and attention-based models have shown promise for QA systems.

This project aims to build a Question-Answering system (QAS) using Transfer Learning approach for Legal documents. The project focuses on attention-based models such as BERT that are significantly ahead of the conventional NLP techniques and deliver current state-of-the-art results.

Keywords - QAS, NLP, SQuAD, BERT, Pre-training, Fine tuning, Legal Acts

Contents

List of Tables	ii
List of Figures	iv
List of Symbols	v
1 Introduction	1
1.1 Question Answering System (QAS)	1
1.2 Transfer Learning	1
1.2.1 BERT : The Pre-Trained Model	2
1.3 The Dataset	2
1.3.1 SQuAD : The Pre-Trained Dataset	2
1.3.2 Legal Dataset	3
2 Literature Review	4
2.1 Transformers	4
2.1.1 Introduction to Transformers	4
2.1.2 Self Attention Mechanism	5
2.1.3 Encoder of a Transformer	7
2.1.4 Decoder of a Transformer	9
2.2 BERT	12
2.2.1 Working of BERT	12
2.2.2 Input Data Representations	13

2.2.3	WordPiece Tokenizer	14
2.2.4	BERT Pre-training Strategies	15
2.2.5	Configurations of BERT	17
2.3	SQuAD : Stanford Question Answering Dataset	18
2.4	Exploring BERT Variants for QA System	20
2.4.1	ALBERT : ‘A Lite’ BERT	20
2.4.2	ROBERTa : Robustly Optimized BERT Approach	21
2.4.3	DistilBERT : Distilled version of BERT	22
2.4.4	BERT Variants : Summary	23
2.5	Question Generation	24
2.5.1	Rule-based (Ranking) Approach	24
2.5.2	Sequence-to-sequence Learning Approach	25
2.5.3	Neural Networks Approach	25
2.5.4	Question Generation using GPT-2 Model	26
2.5.5	Google’s T5 Model for Question Generation	27
2.5.6	Question Generation : Summary	28
2.6	Conclusion of Literature Review	29
3	Problem Statement	30
3.1	Problem Statement	30
3.2	Objectives	31
4	Methodology and System Design	32
4.1	Pre-processing and Context Generation	32
4.2	Question Generation	34
4.3	Legal Text Dataset	35
4.4	Question Answering System	36

4.5	Performance Evaluation	38
4.5.1	Weighted Accuracy	38
4.5.2	F1 Score	39
4.5.3	Exact Match	39
4.6	Fine Tuning	40
4.7	Mode of Operations	41
4.7.1	Mode 1	41
4.7.2	Mode 2	43
5	System Requirement Specification	45
5.1	Introduction	45
5.1.1	Document purpose	45
5.1.2	Project scope	45
5.1.3	Intended Audience	46
5.2	Overall Description	46
5.2.1	Product Perspective	46
5.2.2	Product Functions	46
5.2.3	User Classes and Characteristics	46
5.2.4	Operating Environment	47
5.2.5	Design and Implementation Constraints	47
5.2.6	Assumptions and dependencies	47
5.3	External Interface Requirements	48
5.3.1	User Interface requirements	48
5.4	Specific Requirements	48
5.4.1	Hardware Requirements	48
5.4.2	Software Requirements	49

5.5 Other Non-Functional Requirements	49
6 Results and Discussion	50
6.1 BERT Variants Evaluation - I	50
6.1.1 Fine Tuning ALBERT on Legal Dataset	51
6.1.2 Fine Tuning BERT-large on Legal Dataset	55
6.1.3 Fine Tuning DistilBERT on Legal Dataset	56
6.1.4 Fine Tuning RoBERTa on Legal Dataset	57
6.2 BERT Variants Evaluation - II	59
6.2.1 Weighted Accuracy (A_w)	59
6.2.2 F1 Score	60
6.2.3 Exact Match (EM)	60
6.3 BERT Variants Evaluation - III	61
6.3.1 F1 Score Details	61
6.3.2 Weighted Accuracy and Exact Match Details	62
6.4 BERT Pre-Trained Vs. Fine-tuned	62
7 Conclusion	64
8 Future Scope	66
9 Web Application	67
9.1 Mode 1 Operation	67
9.2 Mode 2 Operation	69
10 Project Timeline	71
A Sample Legal Act	72
B Publication Details	75
C Source Code and Dataset	76
C.1 Source Code	76

C.2 Dataset	76
D Issues Resolved	77

List of Tables

2.1	BERT configurations	18
2.2	Comparison in Variants of BERT	23
2.3	Literature Review of Question Generation Systems	28
2.4	Conclusion of Literature Review	29
4.1	Weighted Accuracy Metrics	38
6.1	Comparison on F1 Score	61
6.2	Comparison on A_w and EM	62
6.3	BERT Pretrained vs. Finetuned	63

List of Figures

2.1 Transformer Encoder-Decoder Architecture	5
2.2 Calculation of Self Attention Mechanism [3]	6
2.3 BERT Input Representations	14
2.4 The BERT Model	15
2.5 SQuAD (v1.1 and v2.0) Dataset Sample	19
4.1 Overall System Design	32
4.2 Legal Text Dataset	35
4.3 Question Answering System	36
4.4 Process of Fine Tuning	40
4.5 Mode 1 Block Diagram	43
4.6 Mode 2 Block Diagram	44
6.1 ALBERT-base	51
6.2 ALBERT-large	52
6.3 ALBERT-xlarge	53
6.4 ALBERT-xxlarge	54
6.5 BERT-large	55
6.6 DistilBERT	56
6.7 RoBERTa-base	57

6.8	RoBERTa-large	58
6.9	Weighted Accuracy Comparison	59
6.10	F1 Score Comparison	60
6.11	Exact Match Comparison	61
9.1	Mode - 1 (a)	67
9.2	Mode - 1 (b)	68
9.3	Mode - 2 (a)	69
9.4	Mode - 2 (b)	70
10.1	Project Timeline	71
A.1	A Sample Legal Act	72
A.2	A Sample Legal Act [Index Page]	73
A.3	A Sample Legal Act [Main Content Page]	74
B.1	Mail from ELMNLP 2021	75

List of Abbreviations

ALBERT	A Lite Version of BERT
BERT	Bidirectional Encoder Representation for Transformers
DistilBERT	Distilled version of BERT
DL	Deep Learning
EM	Exact Match
GPT	Generative Pre-trained Transformer
IE	Information Extraction
IR	Information Retrieval
LSTM	Long Short Term Memory
MLM	Masked Language Modeling
NLP	Natural Language Processing
NSP	Next Sentence Prediction
QAS	Question Answering System
QGS	Question Generation System
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT pre-training Approach
SOP	Sentence Order Prediction
SQuAD	Stanford Question Answering Dataset
T5	Text-To-Text Transfer Transformer

Chapter 1

Introduction

1.1 Question Answering System (QAS)

The Question Answer System (QAS) is one of the key tasks of Natural Language Processing (NLP), with wide-ranging applications in the advancement of language based AI systems. The QAS integrates research from various areas, with a common subject, that includes Information Retrieval (IR), Information Extraction (IE), Natural Language Processing (NLP) and Deep Learning (DL). Current search engines, on the other hand, are only engaged in document retrieval; i.e. based on certain keywords, it only returns the relevant classified documents that contain these keywords. They fail to specifically answer to these questions. For this reason, QAS is designed to help individuals find specific answers to specific questions in limited areas such as legal documents.

1.2 Transfer Learning

In recent years, a significant progress has been made in the QA automation through the advancement of end-to-end DL techniques. Transfer Learning is one such technique that attempts to adapt the knowledge gained from one task and apply it to another related task in order to either increase efficiency or reduce fine-tuning data size.

Here, the model is first trained with a large dataset for a certain task and the trained model is saved. For a new related task, instead of initializing a new model with random weights, the model is now initialized with the weights of the previously learned model (pre-trained model). That is, instead of training a new model from scratch for a new task, the pre-trained model is used, and its weights are adjusted (fine-tuned) according to the new task, since the model has been already trained on a large dataset. In other words, the ability to transfer the knowledge of a pre trained model into a new condition is referred to as Transfer Learning.

1.2.1 BERT : The Pre-Trained Model

Language model pre-training has proved to be successful for improving several applied NLP tasks. Among the various pre-trained models, Google's most recent released Bidirectional Encoder Representations from Transformers (BERT) is a technically simple yet operationally powerful pre-trained model. This pre-trained BERT model is further fine-tuned with auxiliary architectures to provide better results for closed-domain specific tasks. As a result, the goal here is to construct an output network on top of the pre-trained BERT model for the specific tasks such as in the QA system.

1.3 The Dataset

1.3.1 SQuAD : The Pre-Trained Dataset

The development of large datasets is a significant aspect in building language models. Until recently, most of the datasets developed for a QAS, either focused solely on answerable questions or used automatically created unanswerable questions that were easy to find. The Stanford Question Answering Dataset (SQuAD) was created to fix these flaws by providing brief answers to questions based on context connected with each question.

SQuAD is used as foundation for pre-training language models, specifically for the QAS because it allows the model to understand a lot about the basic general question answering framework.

1.3.2 Legal Dataset

As this project focuses on developing a QAS on Legal Documents, the pre-trained BERT model will be fine-tuned for legal domain based question answering tasks. Hence, a Legal Text dataset is created using Legal Acts that are amended by the Indian Constitution and the Judiciary. The Legal Text Dataset uses the same format as SQuAD. The pre-trained BERT will be fine-tuned for question answering on this legal text dataset. This fine-tuned BERT will be further used to answer legal domain-based questions.

Overall, this project work proposes on implementing a Question Answering system for Legal Documents using Transfer Learning Approach. The aim is to adapt the BERT model as a baseline model with a linear output layer and design several modules on top of it as task-specific output layers. Building a post-processing encoder-decoder architecture to increase the efficiency of the BERT model in order to create an answering system for legal documents.

Chapter 2

Literature Review

2.1 Transformers

2.1.1 Introduction to Transformers

RNN and LSTM networks are often used in sequential tasks such as next word predicting, machine translation, text generation, and other applications. However, one of the most challenging aspects of the recurrent model is preserving long-term dependency. To address this weakness of RNNs, a new architecture called Transformer was introduced in the paper “Attention Is All You Need” [1]. The invention of the transformer resulted in a significant advance in the field of NLP, as well as the emergence of modern innovative architectures such as BERT, GPT-3, T5, and others. The transformer is currently the state-of-the-art model for several NLP tasks.

The transformer model is entirely dependent on the attention mechanism [2] and totally eliminates recurrence. The transformer employs a special type of attention mechanism known as **self-attention**. The transformer uses encoder-decoder architecture. The encoder receives the input (source) sentence. It learns and sends the representation of the input sentence to the decoder, which produces the output (target) sentence.

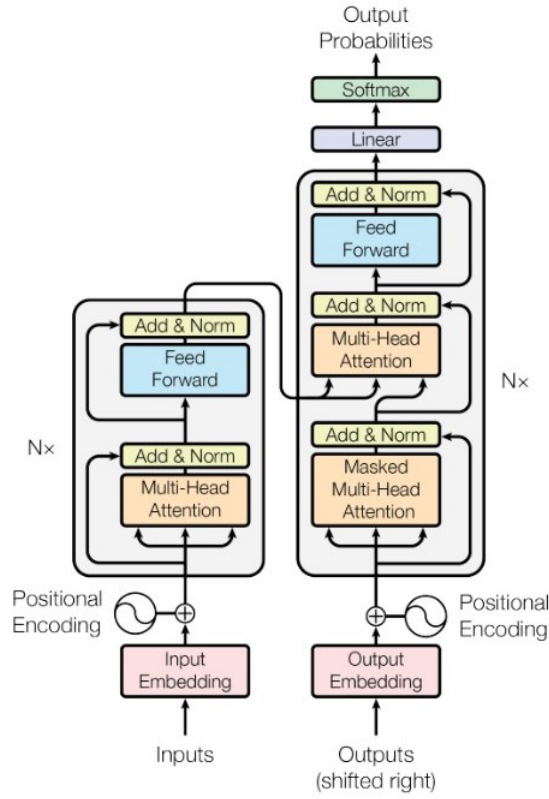


Figure 2.1: Transformer Encoder-Decoder Architecture

2.1.2 Self Attention Mechanism

Self-attention is an attention mechanism that connects different positions in a single sequence to compute a representation of the sequence. Here, the input sentence is represented using the input matrix, X (embedding matrix or input embedding) where each row of the matrix is the embedding of the word. Hence, the dimension of the input matrix is [sentence length x embedding dimension].

From the input matrix, X , three new matrices are created: a query matrix Q , a key matrix K , and a value matrix V . The query Q , key K , and value V matrices are created by multiplying the input matrix X , with weight matrices W^Q, W^K, W^V respectively. The weight matrices W^Q, W^K, W^V are randomly initialized and their optimal weights are learned during training. As the optimal weights are learned, more accurate query,

key, and value matrices are obtained. The self-attention mechanism relates a word to all the words in the sentence using the query, key, and value matrices. This includes the following four steps:

1. First, the dot product between the query matrix Q and the key matrix K transpose, $Q.K^T$ is computed and the similarity scores are obtained.
2. Next, $Q.K^T$ is divided by the square root of the dimension of the key vector $\sqrt{d_k}$ for stable gradients.
3. Then, the softmax function is applied to normalize the scores (in range of 0 to 1) and obtain the score matrix.
4. At the end, the attention matrix, Z , is computed by multiplying the score matrix by value matrix V which contains the attention values for each word in the sentence.

$$Z = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

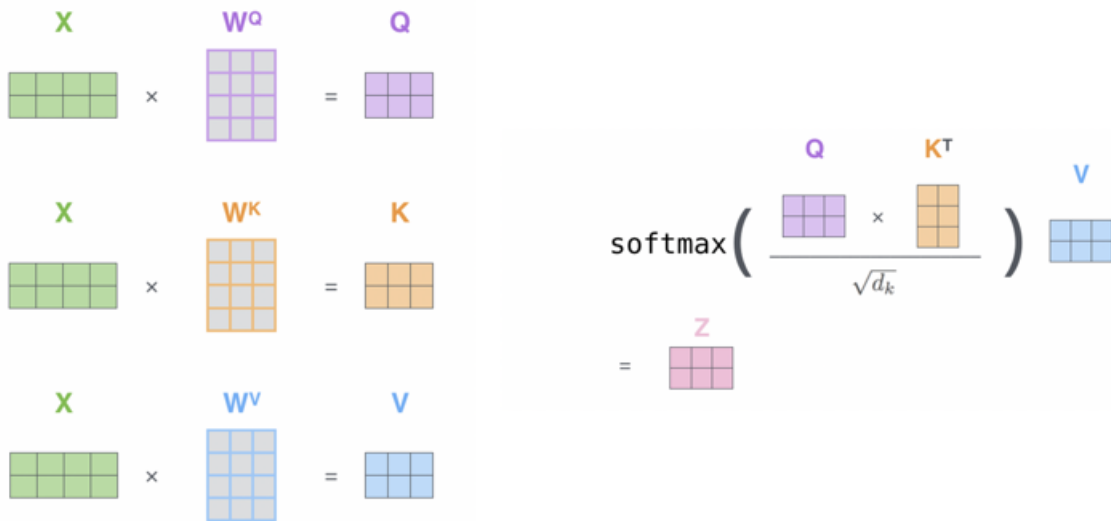


Figure 2.2: Calculation of Self Attention Mechanism [3]

2.1.3 Encoder of a Transformer

The transformer constitutes of a stack of N ([1] considered $N=6$) encoders. The output of one encoder is fed into the encoder above it as input. As output, the final encoder returns the representation of the given source sentence. Each encoder block is identical, and each encoder block is made up of two sub-layers: Multi-head attention and Feed forward network. The stack of encoders in a transformer performs the following operations :

1. First, the input sentence is converted to an embedding matrix, and then the position encoding is added to it and fed as input to the bottom-most encoder.
2. The first encoder takes the input and sends it to the multi-head attention sub-layer, which returns the attention matrix, Z , as output.
3. The attention matrix, Z , is fed as input to the next sub-layer, which is the feed-forward network. The feed-forward network takes the attention matrix as input and returns the encoder representation as output.
4. Next, the output obtained from encoder is fed as input to the encoder above it.
5. This encoder carries the same process until the topmost encoder and returns the encoder representation of the given input sentence as output.

Multi-head Attention Sub Layer

In the case of a multi-head mechanism, multiple attention heads are used instead of a single attention head. This is useful in situations where the meaning of the actual word is ambiguous, that is, if the value vector of other words dominates the actual word; otherwise, understanding the correct meaning of the word would be difficult. Here, several attention matrices are calculated and then their values are concatenated to ensure that the results are correct. This improves the accuracy of the attention matrix.

$$\text{MultiHead}(Q, K, V) = \text{Concatenate}(\text{head}_1, \dots, \text{head}_h).W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Feed Forward Network Sub Layer

The feed-forward network consists of two dense layers with ReLU activations. The parameters of the feed-forward network are the same over the different positions of the sentence and different over the encoder blocks.

Positional Encoding

In RNNs, the sentence is fed word by word so that the network understands the sentence completely. But with the transformer network, the recurrence mechanism is not followed. So, in transformers, all the words in the sentence are fed in parallel to the network which helps in decreasing the training time and also helps in learning the long-term dependency. However, it would be difficult to understand the meaning of the sentence if the word order (position of the word) is not retained. Hence, to retain the order of words of the sentence, positional encoding is introduced. Positional encoding, as the name suggests, is an encoding indicating the position of the word in a sentence (word order).

Positional encoding of each word embedding is calculated as :

$$P(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad \text{when } i \text{ is even}$$

$$P(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad \text{when } i \text{ is odd}$$

where, pos implies the position of the word and in a sentence, and i implies the position of the embedding. These calculations are stored in a positional encoding matrix, P , added with the input matrix, X , and then fed it as input to the network.

Add and norm component

This component connects the input and output of a sub-layer. It is basically a residual connection followed by layer normalization. Layer normalization promotes faster training by preventing the values in each layer from changing heavily.

2.1.4 Decoder of a Transformer

The decoder takes the encoder representation as input and generates the target sentence. Similar to the encoder, a stack of decoders is used ([1] uses $N = 6$) by the transformer, where output of one decoder is sent as the input to the decoder above it. A decoder receives two inputs: one is from the previous decoder, and the other is the encoder's representation (encoder output).

The decoder generates the target sentence in time steps. At the first time step, it takes $\langle \text{sos} \rangle$ (start of sentence) as an input and generates the first word. Subsequently, on every time step, the decoder combines the newly generated word to the input and predicts the next word. This is performed until the decoder generates the $\langle \text{eos} \rangle$ (end of sentence) token, indicating that the decoder has completed generating the target sentence.

The decoder block is similar to the encoder and consists of three sub-layers: Masked multi-head attention, Multi-head attention, Feed forward network. The stack of decoders in a transformer performs the following operations :

1. The input to the decoder is converted into an embedding matrix and add the position encoding to it and fed it as input to the bottom-most decoder.
2. The decoder takes the input and sends it to the masked multi-head attention layer, which returns the attention matrix as output.

3. This attention matrix and the encoder representation are fed as input to the multi-head attention layer, which again outputs the new attention matrix.
4. This new attention matrix is fed as input to the next sub-layer of feed-forward network which returns the decoder representation as output.
5. Next, the output obtained from decoder is fed as input to the decoder above it.
6. This decoder carries the same process until the topmost decoder and returns the decoder representation of the target sentence as output.
7. At last, the decoder representation of the target sentence is fed to the linear and softmax layers and the predicted word is obtained.

Masked Multi-head Attention Sub Layer

The decoder predicts the target sentence word by word in each time step, but this applies only during testing. Since the correct target sentence is known during training, the whole target sentence is simply fed as input to the decoder with a small modification. This modification is carried out in the masked multi-head attention mechanism, which is identical to the multi-head attention mechanism but differs slightly.

Since, during training, the right target sentence is known, the attention mechanism should relate the words only until the word to be predicted and not the next other words. To do this, all words on the right that have not yet been predicted by the model are masked. Masking words help the self-attention mechanism to attend only to the words that would be available to the model during testing. For masked multi-head attention mechanism, masking is done before applying softmax function (step 3) to each *head* where all the words to the right are replaced with $-\infty$ and the final attention matrix, M , is fed to the next sub-layer, which is another multi-head attention layer.

Multi-head Attention Sub Layer

The multi-head attention sub-layer in each decoder receives two inputs: one is from the previous sub-layer, masked multi-head attention, and the other is the encoder representation. Since there is an interaction between the encoder and decode, this layer is also called an **encoder-decoder attention layer**.

The first step in the multi-head attention mechanism is creating the query, key, and value matrices. The query matrix, Q , is created by multiplying weight matrix W_i^Q by the attention matrix, M , obtained from the previous sub-layer which holds the representation of the target sentence and the key, K and value matrices, V , obtained by multiplying the encoder representation with weight matrices W_i^K and W_i^V holding the representation of source sentence. The final attention matrix is constructed in the same way as constructed in the encoder, and fed to the feed forward network layer.

Feed Forward Network Sub Layer

The feed forward layer in the decoder works exactly the same as in the encoder. Also, just like with the encoder, the add and norm component connects the input and output of a sub-layer.

Linear and Softmax Layers

The linear layer generates the logits whose size is equal to the vocabulary size. Next, the logits are converted into a probability using the softmax function, and then the decoder outputs the word whose index has a high probability value.

2.2 BERT

BERT stands for Bidirectional Encoder Representation from Transformer. It is the state-of-the-art embedding model published by Google [4]. It has created a major breakthrough in the field of NLP by providing greater results in many NLP tasks, such as question answering, text generation, sentence classification, and many more besides. One of the major reasons for the success of BERT is that it is a context-based embedding model, which generates dynamic embeddings based on the context, unlike other popular static embedding models, such as word2vec, which are context-free.

2.2.1 Working of BERT

Bidirectional Encoder Representation from Transformer (BERT), as the name suggests, is based on the transformer model. BERT can be perceived as the transformer, but only with the encoder. The sentence is fed as input to the transformer's encoder and it returns the representation for each word in the sentence as an output. That's exactly what BERT is – an Encoder Representation from Transformer. The encoder of the transformer is bidirectional in nature since it can read a sentence in both directions. Thus, BERT is basically the Bidirectional Encoder Representation obtained from the Transformer.

In the BERT model, the input sentence is fed to the transformer's encoder and the contextual representation (embedding) of each word in the sentence is obtained as an output. Once the sentence is fed as an input to the encoder, the encoder understands the context of each word in the sentence using the **multi-head attention mechanism** (relates each word in the sentence to all the words in the sentence to learn the relationship and contextual meaning of words) and returns the contextual representation of each word in the sentence as an output.

2.2.2 Input Data Representations

Before feeding the input to BERT, the input is converted into embeddings using the 3 embedding layers indicated as: Token, Segment and Position embedding.

Token Embedding

First, there is a token embedding layer. Here, the sentences are tokenized (using tokenizer as discussed in section 3.2.3) and the tokens are obtained. Next, the [CLS] token is added at the beginning of the first sentence, and the [SEP] token is added at the end of every sentence. The [CLS] token is used for classification tasks and the [SEP] token is used to indicate the end of every sentence. Now, before feeding all the tokens to BERT, the tokens are converted into embeddings using an embedding layer called token embedding, where the value of token embeddings will be learned during training.

Segment Embedding

Segment embedding is used to distinguish between the two given sentences. Apart from the [SEP] token, some sort of indicator need to be given to the model to distinguish between the two sentences. To do this, the input tokens are fed to the segment embedding layer. The segment embedding layer returns only either of the embeddings of a sentence as an output. That is, if the input token belongs to sentence A, then the token will be mapped to that embedding.

Position Embedding

BERT is essentially the transformer's encoder, and so the information about the position of the words (tokens) in sentence is needed to be given before feeding them directly to BERT. So, a layer called the position embedding layer is used and position embedding for each token in our sentence is obtained.

Final Representation

First the given input sentences are converted to tokens and fed to the token embedding, segment embedding, and position embedding layers and obtain the embeddings. Next, all the embeddings together are sum up and fed as input to BERT:

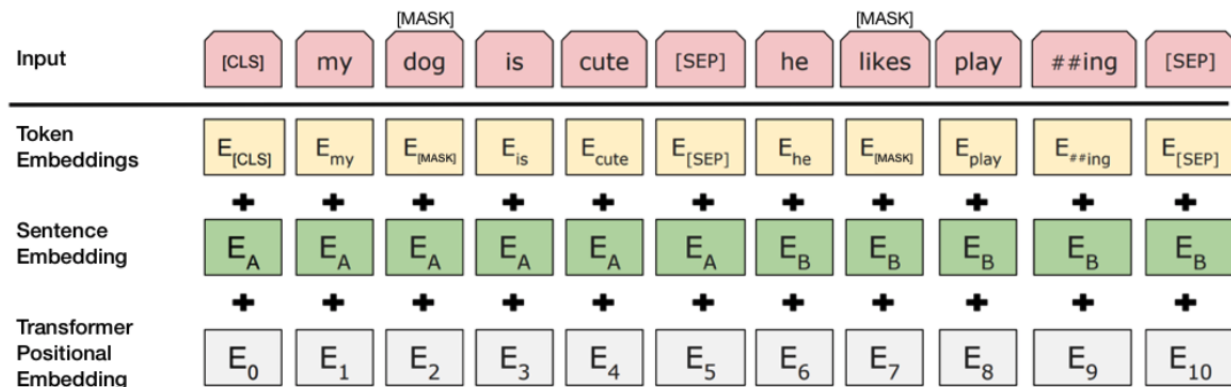


Figure 2.3: BERT Input Representations

2.2.3 WordPiece Tokenizer

BERT uses a special type of tokenizer called a WordPiece tokenizer. It follows the subword tokenization scheme. When the word is tokenized using the WordPiece tokenizer, first it is checked whether the word is present in the vocabulary (BERT has vocabulary size of 30K tokens). If the word is present in the vocabulary, then it is used as a token. If the word is not present in the vocabulary, then the word is split into subwords and it is checked whether the subword is present in the vocabulary. If the subword is present in the vocabulary, then it is used as a token. But if the subword is not present in the vocabulary, then the subword is again split and checked whether it is present in the vocabulary. If it is present in the vocabulary, then it is used as a token, otherwise split it again. In this way, the subword is kept splitting and checked with the vocabulary until individual characters are reached. This is effective in handling the **out-of-vocabulary (OOV)** words.

2.2.4 BERT Pre-training Strategies

The BERT model is pre-trained on the two tasks : Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). During pre-training, in the initial iterations, the model does not return the correct probability because the weights of the feed forward network and encoder layers of BERT are not optimal. However, over a series of iterations, with backpropagation, the weights of the feed forward network and encoder layers of BERT are updated and the optimal weights are learned. After pre-training, the weights of the model are saved and this pre-trained model is further used for fine-tuning on downstream tasks.

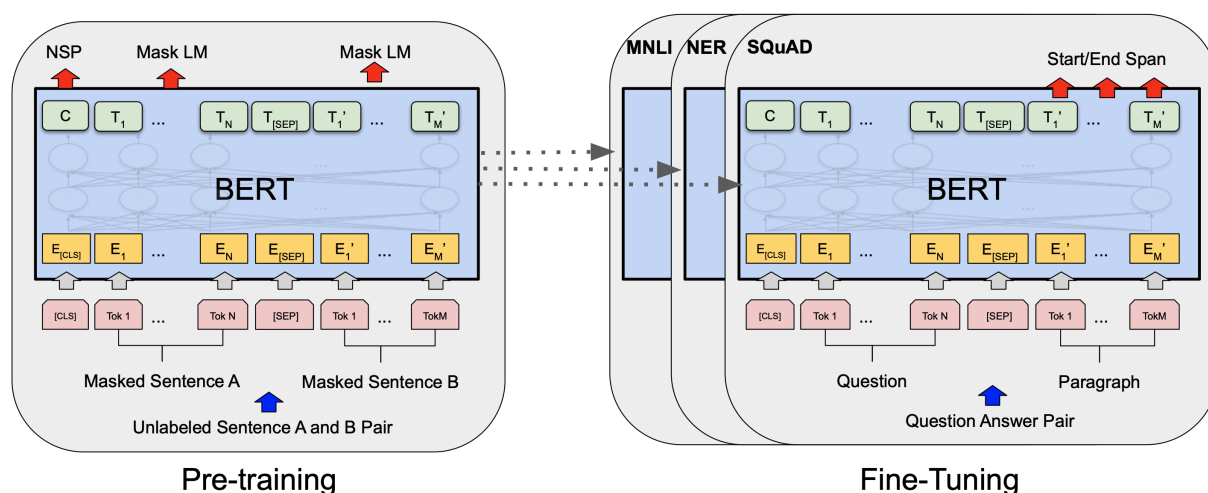


Figure 2.4: The BERT Model

Masked Language Modeling (MLM)

BERT is an auto-encoding language model, meaning that it reads the sentence in both directions to make a prediction. In a masked language modeling task (or cloze task), in a given input sentence, 15% of the words are randomly masked and the network is trained to predict the masked words. To predict the masked words, the model reads the sentence in both directions and tries to predict the masked words.

Masking tokens randomly will create a discrepancy between pre-training and fine-tuning. That is, BERT is trained by predicting the [MASK] token. After training, during fine-tuning, there will be no [MASK] tokens in the input. So it will cause a mismatch between the way in which BERT is pre-trained and how it is used for fine-tuning. To alleviate this issue, 80-10-10% rule is applied where, for 80% of the time, the token (actual word) is replaced with the [MASK] token, for 10% of the time, the token (actual word) is replaced with a random token (random word) and for 10%, no changes are made.

Following tokenization and masking, the input tokens are fed to the token, segment, and position embedding layers and the input embeddings are obtained which are fed to BERT. BERT takes the input and returns a representation of each token (along with the masked token) as an output. To predict the masked token, the representation of the masked token returned by BERT is fed to the feed forward network with a softmax activation. The feed forward network takes it as input and returns the probability of all the words in the vocabulary to be the masked word. The highest probable word is returned as the predicted word by the model.

Next Sentence Prediction (NSP)

Next sentence prediction (NSP) is another interesting strategy used for training the BERT model. In the NSP task, two sentences are fed to BERT and it has to predict whether the second sentence is the follow-up (next sentence) of the first sentence. In the NSP task, the goal of the model is to predict whether the sentence pair belongs to the `isNext` or `notNext` category. The sentence pair (A and B) is fed to BERT and it is train to predict whether sentence B follows on from sentence A. The model returns `isNext` if sentence B follows on from sentence A, otherwise, it will return `notNext` as an output. Thus, NSP is essentially a binary classification task.

By performing the NSP task, the model can understand the relation between the two sentences. Understanding the relation between two sentences is useful in the case of many downstream tasks, such as question answering and text generation. For NSP task, first, the sentence pair is tokenized. A [CLS] token is added just at the beginning of the first sentence, and an [SEP] token at the end of every sentence. Now, the input tokens are fed to the token, segment, and position embedding layers and the input embeddings are obtained. Then, the input embeddings are fed to BERT and the representation of each token are obtained. To perform classification, the representation of only the [CLS] token is taken as it holds the aggregate representation of all the tokens and fed it to the feed forward network with the softmax function, which then returns the probability of the sentence pair being `notNext` and `notNext`.

2.2.5 Configurations of BERT

Devlin, et. al.(2019) [4] presented the BERT model in two standard configurations: -

BERT-base

BERT-base consists of 12 encoder layers, each stacked one on top of the other. All the encoders use 12 attention heads. The feed forward network in the encoder consists of 768 hidden units. Thus, the size of the representation obtained from BERT-base is 768. The total number of parameters in BERT-base is 110 million.

BERT-large

BERT-large consists of 24 encoder layers, each stacked one on top of the other. All the encoders use 16 attention heads. The feed forward network in the encoder consists of 1,024 hidden units. Thus, the size of the representation obtained from BERT-large will be 1,024. The total number of parameters in BERT-large is 340 million.

Model	BERT-base	BERT-large
Encoder Layers (L)	12	24
Attention Heads (A)	12	16
Hidden Units (H)	768	1024
Total Parameters	110 mil	340 mil

Table 2.1: BERT configurations

2.3 SQuAD : Stanford Question Answering Dataset

Rajpurkar, et. al.(2016) [5] published the Stanford Question Answering Dataset (SQuAD), a reading comprehension dataset comprised of crowdworkers questions posed on a series of Wikipedia articles, with the answer to each question being a fragment of text, or span, from the corresponding reading segment. SQuAD is much larger than previous reading comprehension datasets, with 100,000+ question answer pairs spread over 500+ pages.

SQuAD dataset was developed because existing high-quality datasets (MCTest; ProRead) were too small for training modern data-intensive models, and those that were large (CNN/Daily Mail; CBT) were semi-synthetic and lacked the same features as explicit reading comprehension issues. Furthermore, unlike previous datasets such as WikiQA, CBT, and MCN, SQuAD does not have a list of answer choices for each question. Rather, systems must choose an answer from all possible spans in the passage. Though questions with span-based answers are more restricted than questions with more interpretative answers found in more advanced standardised tests, SQuAD still contains a wide range of question and answer types.

It was observed that understanding the context and type-matching heuristics helped models perform well on SQuAD, but it did not guarantee the robustness of SQuAD on distorting sentences. The emphasis of SQuAD on questions for which a correct answer is guaranteed to appear in the reference document is one of the root causes of these issues. As a result, models only need to choose the span that seems to be more related to the question, rather than ensuring that the answer is necessarily implied by the text. Hence, Rajpurkar, et. al.(2018) [6] developed SQuAD 2.0, a new dataset that integrates answerable questions from SQuAD 1.1 with 53,775 new, unanswerable questions about the same paragraphs. These questions were designed by crowdworkers, which are applicable to the paragraph and have a logical answer (something of the same sort) to the question asked.

SQuAD (v1.1 and v2.0) has proven to be an effective dataset for pre-training BERT, especially for question answering tasks. When pre-trained with SQuAD, BERT is able to understand the general question-answering structure, which helps in fine-tuning it for a particular QA task.

Paragraph: In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under *gravity*. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other raindrops or ice crystals *within a cloud*. Short, intense periods of rain in scattered locations are called "showers".

Question 1: What causes precipitation to fall?
Answer: *gravity*

Question 2: Where do water droplets collide with ice crystals to form precipitation?
Answer: *within a cloud*

Article: Endangered Species Act

Paragraph: " ... Other legislation followed, including the Migratory Bird Conservation Act of 1929, a *1937 treaty* prohibiting the hunting of right and gray whales, and the *Bald Eagle Protection Act of 1940*. These *later laws* had a low cost to society—the species were relatively rare—and little *opposition* was raised."

Question 1: "Which laws faced significant *opposition*?"
Plausible Answer: *later laws*

Question 2: "What was the name of the *1937 treaty*?"
Plausible Answer: *Bald Eagle Protection Act*

(a): Question-answer pairs for a sample passage in the SQuAD(v1.1) dataset. Each of the answers is a segment of text from the passage.

(b): Two unanswerable questions written by crowdworkers, along with plausible (but incorrect) answers as mentioned in SQuAD(v2.0) dataset.

Figure 2.5: SQuAD (v1.1 and v2.0) Dataset Sample

2.4 Exploring BERT Variants for QA System

2.4.1 ALBERT : ‘A Lite’ BERT

Lan, Chen (2019) et. al. [7] present ALBERT, “A lite” version of BERT. One of the difficulties with BERT is that it contains millions of parameters. BERT-base has 110 million parameters, making it difficult to train and requiring a long inference time. Although increasing the model size yields positive results, it places limitations on the computational resources available. ALBERT was created to counteract this. ALBERT is a simplified version of BERT that has fewer parameters than BERT. To minimise the number of parameters, it employs the cross-layer parameter sharing and factorized embedding layer parameterization techniques.

During training, the BERT model learns all of the encoder layers’ parameters. Instead of learning the parameters of all the encoder layers, cross-layer parameter sharing only learns the parameters of the first encoder layer, which are then shared with the parameters of all the other encoder layers. In addition, the BERT model makes the WordPiece embedding size the same as the hidden layer embedding size, resulting in a greater number of parameters to learn. Factorized embedding layer parameterization, in which the embedding matrix is factorised into smaller matrices, is used to prevent this.

The ALBERT model is pre-trained using the MLM task, but rather than using the NSP task like BERT, ALBERT uses a new task called sentence order prediction (SOP). SOP is a classification task where its goal is to classify whether a sentence pair belongs to a positive class (sentence order not swapped) or a negative class (sentence order swapped). SOP is based on inter-sentence coherence rather than topic prediction.

2.4.2 ROBERTa : Robustly Optimized BERT Approach

Liu, Ott (2020) et. al.[8] introduced another interesting and popular variant of BERT with a Robustly Optimized BERT pre-training Approach, or ROBERTa. BERT was observed to be severely under-trained and hence, several approaches were implemented to pre-train the BERT model. ROBERTa is one such approach with BERT with the following changes in pre-training : Dynamic masking instead of static masking in the MLM task, removal of the NSP task and training with just the MLM task, training with a large batch size and using byte-level BPE (BBPE) as a tokenizer.

BERT uses static masking where the masking is done only once during the pre-processing step and the model is trained over several epochs to predict the same masked token. ROBERTa uses dynamic masking instead. Here, each sentence is duplicated several times and tokens are masked randomly in all of these duplicates of the sentence. When the model is trained for several epochs; for each epoch, the sentence with different tokens masked is fed in. The ROBERTa model is trained only with the MLM task and not with the NSP task and the input consists of a full sentence, which is sampled continuously from one or more documents. The input consists of at most 512 tokens. If the end of one document is reached, then sampling begins from the next document.

Also, the ROBERTa model is pre-trained using five datasets [Toronto BookCorpus, English Wikipedia datasets, CC-News (Common Crawl-News), Open WebText, and Stories (subset of Common Crawl) datasets] with the sum of the total size of these five datasets as 160 GB. Unlike BERT which uses WordPiece tokenizer, ROBERTa uses BBPE as a tokenizer which uses byte-level sequence instead of character-level sequence and has a vocabulary size of 50,000.

2.4.3 DistilBERT : Distilled version of BERT

One of the challenges of using the pre-trained BERT model is that it is computationally expensive and running the model with minimal resources is difficult. The pre-trained BERT model has a large number of parameters as well as a high inference time, which makes it harder to use it on edge devices like mobile phones. To alleviate this issue, Knowledge Distillation is used to transfer the pre-trained knowledge from a large pre-trained BERT to a small BERT. Knowledge distillation (Hinton, 2015) [9] is a compression technique in which a small model (the student) is trained to replicate the working of a larger model (the teacher) or a group of models.

DistilBERT, was introduced by Sanh, Debut (2019) [10] which uses Knowledge Distillation technique for BERT. DistilBERT is a smaller, faster, cheaper, and lighter version of BERT. The ultimate idea of DistilBERT is to take a large pre-trained BERT model (teacher BERT) and transfer its knowledge to a small BERT (student BERT) through knowledge distillation. Since the small BERT (student BERT) acquires its knowledge from the large pre-trained BERT (teacher BERT) through distillation, the small BERT is referred as DistilBERT. DistilBERT is 60% faster and its size is 40% smaller compared to large BERT models.

The teacher BERT is a large pre-trained BERT-Base model, pre-trained using the MLM task. Hence, the teacher BERT model is used to predict the masked word, which gives the probability distribution of all the words in the vocabulary of the masked word when given a masked input sentence. This probability distribution contains dark knowledge (weights) that must be transferred to the student BERT. The student BERT, unlike the teacher BERT, has not been pre-trained. The student BERT has to learn from the

teacher. In comparison to the teacher BERT, the student BERT is smaller and has fewer layers. The teacher BERT consists of 110 million parameters, but the student BERT consists of only 66 million parameters.

2.4.4 BERT Variants : Summary

The table 2.2 shows a comparison of the BERT and its variant models :

	BERT	ALBERT ^{1,2}	ROBERTa	DistilBERT
Parameters	[B]: 110M [L] : 340M	[B]: 12M [L] : 18M	[B]: 110M [L] : 355M	[B]: 66M
(L) / (H) / (A)	[B]: 12/768/12 [L] : 24/1024/16	[B]: 12/768/12 [L] : 24/1024/16	[B]: 12/768/12 [L] : 24/1024/16	[B]: 6/768/12
Pre-Training Data	[16 GB]	[16 GB]	[160 GB]	[16 GB]
Method	Bidirectional Trans- former, MLM, NSP	BERT with reduced parameters, SOP	Dynamic Masking, Only MLM, BBPE	Knowledge Distilla- tion

Table 2.2: Comparison in Variants of BERT

¹ALBERT has 2 more configurations with their parameters as - [xL] : 60M and [xxL] : 235M.

²The (L)/(H)/(A) for both configurations are [xL] : 24/2048/32 and [xxL] : 12/4096/64.

2.5 Question Generation

Question Generation (QG) is one of the most critical component of the QAS. It aims to generate natural questions from a given a sentence or paragraph. The questions generation process may follow various approaches like traditional rule-based or recent neural models. In this section, literature that focuses on natural question generation and developments taken place in this domain over time are reviewed.

2.5.1 Rule-based (Ranking) Approach

Various rule-based approaches to question generation have been used in the past. The construction of well-designed rules for declarative-to-interrogative sentence translation is vital to the effectiveness of these approaches [11]. They are usually focused on in-depth linguistic understanding, such as how declarative sentences are constructed and what parts of speech may match a certain part of speech. These rules define the syntactic transformation for declarative sentences.

Hielmann (2010) [12] proposed an overgenerate-and-rank method that produces questions from an input sentence using a rule-based approach. It then ranks them using a supervised learning-based ranker to improve over a purely rule-based approach. The steps used to generate questions were: sentence simplification, answer phrase selection, key verb decomposition, subject-auxiliary inversion, question phrase insertion and statistical ranking. The produced questions were ranked based on the predictions of a logistic regression model which describes a binomial distribution ($P(R = q, t)$) with a binary random variable (R), given the query q and the source text t . Ranking helps in generating appropriate queries but is highly reliant on a manually constructed feature set, and the questions created often duplicate the tokens in the input sentence word for word.

2.5.2 Sequence-to-sequence Learning Approach

Du, Shao (2017) [13] suggested a sequence-to-sequence learning approach for generating questions and investigated on the impact of sentence encoding vs. paragraph level information. Unlike previous work, the model does not focus on hand-crafted rules or a complex NLP pipeline; instead, it can be trained from start to end using sequence-to-sequence learning.

This study proposes a global attention mechanism and an RNN encoder-decoder architecture model. The encoder comes in two forms: one that only encodes sentences and the other that encodes both sentence and paragraph-level information. The aim of the QG task is to maximise the conditional log-likelihood probability $P(y|x)$, where y represents the expected query sequence and x represents the given input sentence. To encode the sentence or the paragraph containing the sentence, the encoder uses a bi-directional LSTM. A standard LSTM network is used as the decoder. When compared to conventional rule-based methods, this model produced substantially better performance.

2.5.3 Neural Networks Approach

Duan, Tang (2017) [14] demonstrated how to use neural networks to produce questions from given passages, using QA (question-answer) pairs, which were crawled and analysed from the Community-QA website as training data. The research aimed to generate questions from given passages using neural networks, with three intended goals: no or little human effort for training data; questions based on natural passages and common-intended questions and generated questions to be useful for QA tasks. The architecture proposed, composed of four different parts.

The first is Query Pattern Mining, which derives FAQ (frequently asked question) patterns from large-scale CQA questions. The second part is Query Pattern Prediction, which predicts the top-N question patterns given a passage S . Prediction can be interpreted in two ways; retrieval-based (uses CNNs) and generation-based (uses RNNs). The third part, Question Topic Selection uses a predicted question form Q_p to choose a word Q_t from S , the question topic. To form a complete question, Q_t is filled into Q_p . The last part, Question Ranking assigns a score to each produced question based on a feature set. As S could contain several details, multiple questions with different intentions could be produced.

2.5.4 Question Generation using GPT-2 Model

Lopez (2020) [15] showed the use of Generative Pre-trained Transformer 2 (GPT-2) in question generation task. The paper demonstrated that using transformer-based fine-tuning techniques, powerful question generation systems can be created with only a single pre-trained language model. GPT-2 was then fine-tuned in a similar manner to how it was trained on language modelling. SQuAD dataset was formatted such that each testing sample is interpreted as a “context [SEP] question,” effectively eliminating the concept of response knowledge during training.

Further, the GPT-2 model was tested if the model benefited from answer awareness. Answer tokens were used to mark the answer span within the context ([ANSS] for start and [ANSE] for end). However, as opposed to the model without answer awareness, this model worked poorly. This was due to the fact that the model had no innate idea what to do with the answer-awareness knowledge and there was no clear framework in the model that prioritised the presence of answer-awareness.

2.5.5 Google's T5 Model for Question Generation

Google researchers recently published a T5 model [16] (Text-To-Text Transfer Transformer) in July 2020, which explored the landscape of transfer learning strategies for NLP by adding a single system that transforms any language challenge into a text-to-text format. The model was pre-trained using the 750GB 'Colossal Clean Crawled Corpus (C4)' dataset. It has performed well in a variety of tasks, including summarization, natural language generation, and translation.

Several high-level approaches like language modeling, BERT-style training and de-shuffling were also analyzed for training T5 model. Out of these, BERT-style de-noising objective performed the best and thus, further corruption strategies like masking tokens, replacing spans and dropping tokens were applied with variable corruption rate and corruption span-length. Unlike BERT, T5 is an encoder-decoder model. In both supervised and unsupervised settings, T5 can be fine-tuned. Hence, T5 became very resilient, particularly when it came to **text generation tasks** and could be suitably used for QG Tasks.

BERT over T5 for Answering Task

T5 is an unified framework for converting any language problem into text-to-text format. It is a language generative model that is well suited for language generation tasks such as language translation, summarization, and so on. BERT is more of a language understanding and extractive model, understanding minute textual meanings and determining which section of the text is most relevant. The legal domain contains more formal terms, vocabulary, and cross references. The answers to such closed domain questions are unique and must be retrieved efficiently by encoding the necessary information. Since BERT is an encoder model, it does this much better than the T5 model.

2.5.6 Question Generation : Summary

Literature Title	Techniques/Method used
The First Question Generation Shared Task Evaluation Challenge. [11]	<p>Task 1: Provide input in the form of a paragraph to a group of participants.</p> <p>Task 2: A group of participants is given input consisting of a single sentence and Wh-words.</p> <p>Examine the question relevance, form, syntactic correctness, ambiguity, and variety.</p>
Good question! Statistical Ranking for Question Generation [12]	To generate more relevant questions, a rule-based question generation scheme along with a ranking system was introduced.
Learning to Ask: Neural Question Generation for Reading Comprehension [13]	<p>An encoder-decoder architecture approach was implemented.</p> <p>The effect of encoding sentence vs. paragraph level information was investigated using a sequence-to-sequence learning technique.</p>
Question Generation for Question Answering [14]	Performed question pattern mining on community-QA website, used the pattern to find relevant sentences from the paragraph, selected the question topics from those sentences and generated the question, and ranked them.
Transformer-based End-to-End Question Generation.[15]	A GPT-2 model was used. Showed that using only a single pre-trained language model and no answer awareness, transformer-based fine-tuning techniques can be used to build robust question generation schemes.
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [16]	Introduced and trained T5 encoder-decoder architecture transformer model on mega C4 dataset which showed good results particularly on language generation tasks.

Table 2.3: Literature Review of Question Generation Systems

2.6 Conclusion of Literature Review

This section summarizes the various literature reviewed and the methods/techniques grasped from these literature that were used in our system:

Topic	Methods/Techniques used in our System
Attention Mechanism ^[2]	The mechanism on which the transformers are based. It focuses on the part of the data is more important than others depends on the context.
Transformers ^{[1][3]}	Encoder-decoder architecture. The encoder receives the input sentence, learns and sends the representation of the input sentence to the decoder, which produces the output sentence.
BERT ^[4]	Pre-trained model based on transformer's encoder. Sentence is fed as input to the encoder and it returns the representation for each word in the sentence as an output
SQuAD ^{[5][6]}	Foundation for pre-training language models, specifically for the QAS. Allows the model to understand a lot about the basic general question answering framework.
ALBERT ^[7]	A simplified version of BERT that has fewer parameters than BERT
RoBERTa ^[8]	BERT with robustly optimized approaches to improve its performance
DistilBERT ^{[9][10]}	Transfer knowledge from larger BERT to a small BERT through knowledge distillation.
Question Generation (T5 Model) ^[16]	Language generative model that is well suited for QG tasks.

Table 2.4: Conclusion of Literature Review

Chapter 3

Problem Statement

3.1 Problem Statement

“To build a Question Answering System (QAS) for Legal Documents using Transfer Learning Approach to make it easier and faster for lawyers, legal practitioners and even common people to find solutions to their legal issues with the help of Indian Legal documents, particularly Central Legal Acts, as amended by the Indian Constitution and the Judiciary”

3.2 Objectives

1. Studying different pre-trained Deep Learning (DL) Models for implementing a Question Answering System (QAS).
2. Scraping and Generating Dataset of Legal Documents from the Constitution of India and the Judiciary.
3. Storing and Indexing the Legal Text Dataset generated from Legal Documents.
4. Building Factoid Questions from the Legal Text Dataset.
5. Intent Detection from the Natural Language (NL) Queries (related to the Legal Documents).
6. Extraction of relevant answers (with evidence) using the QA Model.
7. Performance Evaluation of the Question Answering System (QAS).
8. Developing a web application demonstrating the use of the above objectives.

Chapter 4

Methodology and System Design

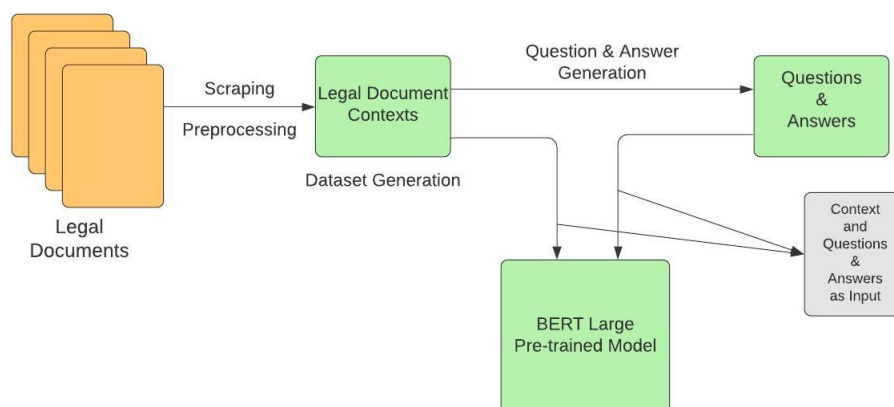


Figure 4.1: Overall System Design

4.1 Pre-processing and Context Generation

This is the first stage of the system which includes pre-processing the Indian Legal Acts and generating context paragraphs required for question generation and answering. Indian legal acts are available in PDF format at India-Code ([website](#)), a database of all Central enactments. This PDF data should be converted into text data. A typical legal act consists of title of the act, index pages, chapters, sections, subsections and footer notes. As a result, pre-processing is needed to ensure that only the most relevant content data is extracted.

Each act is categorised into chapters, which are further subdivided into parts and subsections. As shown in figure A.2, each act has an index in the first few pages that lists the titles of all the parts present in the act. In addition, some pages have footer notes that list any amendments or substitutions made to the act. The title, index pages, footer notes, and chapter headers do not include any substantial details that would be useful in answering, so they are discarded. The relevant information is found in the sections and subsections that serve as the context.

As shown in Figure A.3, one such section is “*Short title and commencement*” and the data listed under it in point-wise format [(1)..., (2)...] forms the subsections. The data in such sections is pertinent and is further divided into chunks of no more than 350 words, each of which is referred to as a ‘context’. The steps followed are:

1. The document name is initially passed to the Python `pdftotext` module, which extracts the text from the pdf legal act file.
2. The first main content page number is then determined by comparing the title on the index page with the title on the first main content page. This is done in order to remove the index pages and keep only the content pages.
3. The title present on the first content page, the footer notes and the chapter headers with capital letters are deleted.
4. The content from all of the pages is then concatenated to create a single text data entity and this text is being filtered to delete any unnecessary symbols.
5. The text is divided into chunks based on the portions of the act. If the context is longer than 350 words, we break it again into chunks of less than or equal to 350 words, retaining the subsections boundary for each chunk.

4.2 Question Generation

Question Generation is an auxiliary component of our system to generate questions in absence of the actual user interaction and also to generate a SQuAD-like legal dataset required for fine-tuning. For this, SQuAD fine-tune Google's T5 base model is used.

The pre-processed contexts are given as input for question generation. For generating questions, the model needs to know what should be the span of answer from the context for which the question needs to be generated. So initially, the model tries to extract possible answer spans. For this, we convert the input given to the model context into list of text such that each list element is the context with a particular sentence within that context highlighted by `< hl >` tokens. This is to inform the model which sentences it should look for answer span extraction. A sample example is as given below:

Context: "This Act is called the RTI Act, 2005. It extends to the whole of India."

Modified i/p: ["`<h/`> This Act is called the RTI Act, 2005. `<h/`> It extends to the whole of India.", "This Act is called the RTI Act, 2005. `<h/`> It extends to the whole of India. `<h/`>"]

The extracted answer spans and the corresponding context they are present in are given to the model. For this, the input format is :

`answer : < answer_text > context : < context_text >`

This QG model will generate the question for that answer by referring the specified context. Further, these questions generated are used in generating the machine-generated question and answer pairs required for building the Legal Dataset.

4.3 Legal Text Dataset

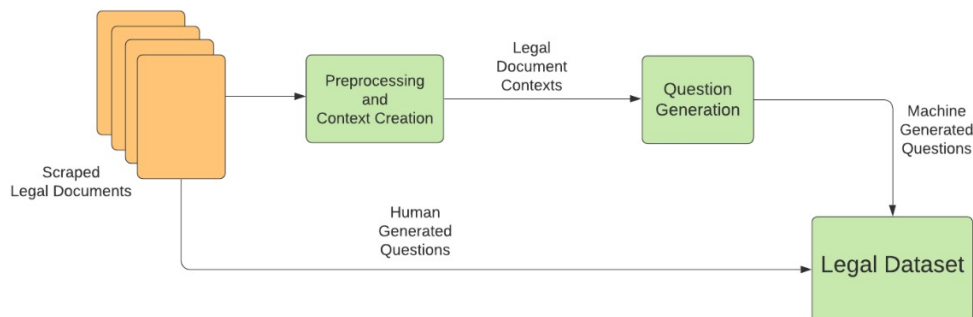


Figure 4.2: Legal Text Dataset

The state-of-the-art BERT model is a general purpose model which shows good results for the task of Question Answering on the SQuAD Dataset. However, to make it more legal specific, fine-tuning of the model is required, on a Legal Dataset. Hence this phase of the methodology discusses the generation of the legal dataset.

The Legal Text Dataset is a collection of wide range of questions on various Indian Legal Acts. The questions are of two types, namely, Machine Generated and Human Generated Questions. The Machine Generated Questions are produced by the Question Generator on varied Legal Acts and the Human Generated Questions are created manually for testing the reliability and versatile nature of the system. Each question is organised in the form of a dictionary containing other required fields as shown in the example below. A list of such question entries in the json format, forms the Legal Dataset. Each field in question-answer pair entry as shown in 1 below specifies the following:

- title: the title of the Legal Act on which the question is asked.
- id: a unique.id for each question in the dataset.
- context: the ‘chunk’ of the Legal Act containing the relevant answer.

- question: the question to be asked to the answering system.
- answer: a dictionary containing two fields:
 - answer_start: the character index that starts the actual answer (in context)
 - answer_end: the character index that ends the actual answer (in context)

```
{
  "title": <title_of_the_Act>,
  "id": <unique_entry_id>,
  "context": <context_containing_the_answer>,
  "question": <question>,
  "answer": {
    "answer_start": <answer_start_index>,
    "answer_end": <answer_end_index>
  }
}
```

Listing 1: Legal Question-Answer Entry Example

4.4 Question Answering System

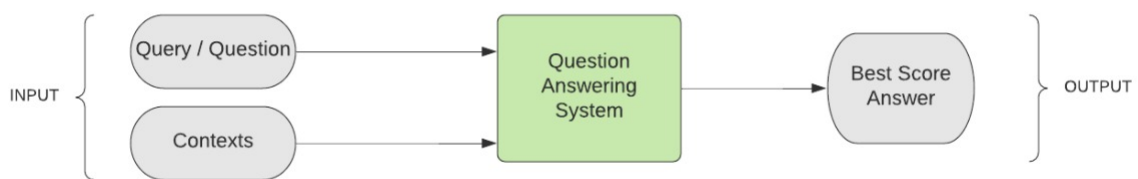


Figure 4.3: Question Answering System

The BERT(large) Model, fine-tuned on the Legal Dataset, is the major component used for answering the questions in the Question Answering system (QAS). The BERT Model takes as input, a question to be answered and a context or a list of contexts which contain the answer to the given question.

On processing the contexts, it outputs the span of the best answer in the form of indexes for the start and end word of the predicted answer. These indexes are further processed by the QAS to find the best answer in the textual and interpretative format. The answer to a question is fetched using the technique of batch processing:

1. A list of contexts, generated by the preprocessing component, are fed into the BERT model along with the question. The question is replicated, forming a list of questions, with one question item for each context, allowing the batch to be processed in parallel.
2. After processing, the model generates a list of vector scores for all the tokens of every input context.
3. All the vector scores are multiplied with the fine-tuned start and end vectors present in the token classification layer of the model to generate the start and end scores for each token. The start and end scores signify if these tokens are likely to be the beginning or end of the answer for the given question.
4. The model processes one context at a time to find the best answer from the respective context. The best answer is determined using a final score which is calculated as the sum of the start and end scores for the answer.
 - 4.1 The token indexes giving the top 20 start and end scores are fetched.
 - 4.2 All start-end combinations between these top 40 indexes are applied and the final scores of each valid answer is recorded.
 - 4.3 The answer with the highest final score is returned (along with the final score) as the answer from the context.
5. After processing all contexts from the batch as in the previous step, the system finds the best batch answer based on the final scores of the context answers. Finally, the batch answers are sorted in the same way to find the best answer from the document.

4.5 Performance Evaluation

Question answering systems are evaluated using two common evaluation metrics, namely **EM** and **F1 score**. We have used the same two metrics to evaluate the model. We also used a **Weighted Accuracy** mechanism to test the model, as discussed below:

4.5.1 Weighted Accuracy

Accuracy defines how accurate the model works. It is defined as the ratio of number of questions answered correctly (or sum of weights of the answers) to the total number of questions, expressed as a percentage. For each question, a weight between 0 and 4 is chosen on the matched percentage. The weight allocated to each answer is determined by how closely the predicted and actual answers align.

Weight	Answer Type	Matched Percentage
Weight 0	Irrelevant Answer	< 25%
Weight 1	Wrong Answer	$\geq 25\%$ and < 50%
Weight 2	Correct Answer	$\geq 50\%$ and < 75%
Weight 3	Specific but Incomplete Answer	$\geq 75\%$ and < 100%
Weight 4	Complete Answer	= 100%

Table 4.1: Weighted Accuracy Metrics

The weighted Accuracy (A_w) can be calculated as :

$$A_w = \frac{w_1 + w_2 + \dots + w_n}{\max(w_i) * N}$$

where:

w_i = Weight assigned to an answer i

N = Total number of questions

4.5.2 F1 Score

The F1 score is a common metric for evaluating Question Answering Systems. For QA systems, the F1 score is calculated using individual words from both the predicted and actual answers. It also tests the model's accuracy by combining precision and recall, providing a more accurate metric than absolute accuracy. F1 is determined by the number of shared words between the actual and predicted answers.

F1 is defined as the harmonic mean of precision and recall. Precision is defined as the ratio of the number of shared words to the total number of words in the predicted answer. It indicates the preciseness of the model based on the predicted answer. On the other hand, Recall is defined as the ratio of the number of shared words to the total number of words in the actual answer. It indicates the ability of the model to answer in a way similar to the actual answer.

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.5.3 Exact Match

Exact Match(EM) calculates the accuracy based on the number of exact answers provided by the model. It is defined as the percentage of questions which are answered definitely.

$$EM = \frac{exact_N}{total_N} * 100$$

where:

$exact_N$ = Number of exact answers

$total_N$ = Total number of answers

According to the weighted accuracy defined above, the questions classified as *weight 4* are the exactly answered questions given by the model

4.6 Fine Tuning

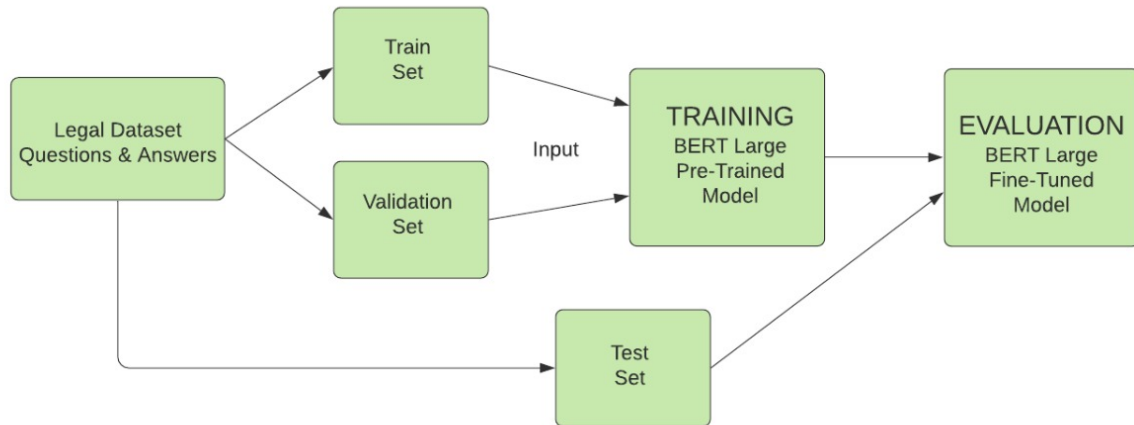


Figure 4.4: Process of Fine Tuning

Fine Tuning is a process to train and adjust the model parameters to make it more task-specific. To achieve better results on the Answering Questions task on the Legal Acts, the pre-trained model was fine-tuned on the Legal Dataset, which consists of about 2500 legal questions. The Legal Dataset is split into the following three datasets in the percentage of 80%-10%-10%. The splitting is managed in a way that the validation and the test set contain more Human Generated Questions to test the versatility of the system.

1. **Train Set**(80%) - Set used for training the model for the legal domain.
2. **Validation Set**(10%) - Set for validating the model after each epoch of training.
3. **Test Set**(10%) - Set for testing the fine-tuned model after training.

After the splitting of the dataset, the Fine-Tuning of the Model is carried out in 4 steps :

1. **Load the split datasets** - In the fine-tuning model, load the three datasets: train, validation, and test. The datasets are of the same format as the Legal Dataset, meaning, a list of question dictionaries, as stated earlier.

2. **Preprocessing the datasets** - The data items from these train and validate datasets are preprocessed into input features which are further used to train the model.
3. **Trainer Module for Fine-tuning** - The trainer module fine-tunes the given model based on the features of the dataset. At each epoch, it tweaks the model parameters to reduce the error in the training results. It also validates the model against the validation set after each epoch, with the goal of lowering the validation loss. The model is trained in 4 epochs at a recommended learning rate of $2e - 5$. After each epoch, the training and validation loss are recorded.
4. **Model Evaluation** - After fine tuning, the model is evaluated using three performance metrics, namely Exact Match, Weighted Accuracy and the F1-score. The evaluation is performed using the test set containing an equal measure of Machine Generated and Human Generated Questions. The answers predicted by the model are compared against the reference answers in the test set and based on the number of common words between the two, the model is evaluated.

4.7 Mode of Operations

4.7.1 Mode 1

The first mode of operation is the **generate-and-answer-question-on-document** mode. In this mode, the system expects a legal document as input, and automatically generates a set of questions on the inputted document using the QG component. The user can then select any question from this list of generated questions to get a specific answer, from the QA system. This mode is particularly useful when users have no explicit questions of their own, to be asked on the legal document. The working of Mode 1 is as follows:

Front End

1. Users are prompted to upload a Legal Act or select one of the Legal Acts from the list displayed on the User Interface.
2. Users are displayed with a set of questions based on the uploaded or selected document.
3. They are allowed to select a particular question from the list to get the corresponding answer and the referred context on the screen.
4. The users can further choose another question from the list or click on the option to generate more questions for the same act.

Back End

1. Once the document is selected or uploaded by the user, it is given as input to the pre-processing and context creation component to process it into smaller contexts.
2. The contexts are further given as input to the QG component to generate various related natural questions.
3. The generated questions along with their contexts are given as input to the QA component which finds the relevant answers.
4. When a user selects a question, the fetched answer and context are shown on the UI.
5. When a question is changed, the relevant answer from the list of already fetched answers is displayed.
6. If the option to generate more questions is selected, the QG model is re-run, along with the QA, to generate new questions and find answers to these new questions.
7. On a change in document, all the above steps run again to process the new document.

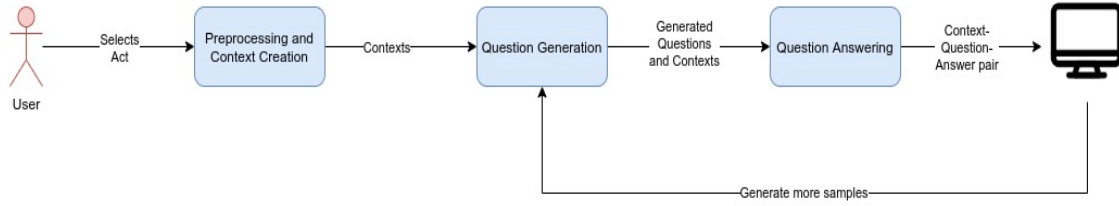


Figure 4.5: Mode 1 Block Diagram

4.7.2 Mode 2

The second mode of operation is the **answer-question-on-document** mode. In this mode, the system not only expects a document as input but also expects an explicit user question on the document. The QA system answers the given input question with the help of the preprocessing component. The system also generates a list of possible questions on the document, using the QG component, as suggested questions to the user.

Front End

1. Users are prompted to upload a Legal Act or choose one from the list and enter the question they want to inquire about it.
2. On clicking the answer button, the specific answer is displayed on the UI. The system-generated questions, which are displayed to users as recommended questions, are also shown alongside the answer.
3. Users may select a question from the suggestions or ask another explicit question to the system on the same document. They can also change the document and the question related to it for fetching a new answer.

Back End

1. The user selects or uploads a document, which is then provided as input to the preprocessing and context creation component, to process it into smaller contexts.

2. All of the contexts, as well as the user's question, are fed into the QA component, to find the best answer from the entire document.
3. The document is provided to the QG component to produce different potential questions for the suggested questions.
4. When a question is changed, the QA model receives the new question along with the preprocessed contexts to retrieve the answer.
5. On a change in document, all the above steps run again to process the new document.

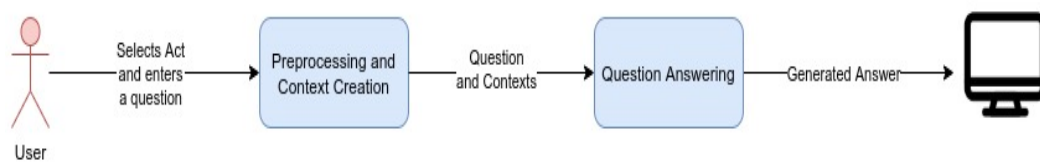


Figure 4.6: Mode 2 Block Diagram

Chapter 5

System Requirement Specification

5.1 Introduction

5.1.1 Document purpose

Legal documents are complex in nature and said to be too complicated, dense, repetitive and out of date. These make it difficult for lawyers and other law professionals to address issues in a timely and accurate manner. The document lists the specifications of our software which uses Transfer Learning approach for building an efficient question answering system using Indian Legal Documents.

5.1.2 Project scope

The project is aimed to deliver an efficient QA system using state-of-art NLP technologies specifically for Indian Legal domain, in order to resolve legal problems in a more timely and precise manner. The project mainly strives on exploring transfer learning approach and using it for the BERT Model for Legal QA task, minimising the training time but also providing correct outcomes. The project will test and demonstrate the robustness and accuracy of the model for Legal domain while providing the service via a web page in a client-server system.

5.1.3 Intended Audience

This software is intended to facilitate lawyers and other legal practitioners in resolving legal disputes. This application can also be used by common people for answering any of their legal issues allowing them to protect themselves, assert their rights and seek justice when they have been wronged. The application will fulfill the demand of users by allowing them to quickly pose questions and get quicker and more precise responses.

5.2 Overall Description

5.2.1 Product Perspective

Everyone, whether an individual, a worker, or a business owner, is subject to and must follow the laws of the country. Laws are formalised in legal documents, which are massive in size and often difficult for ordinary citizens to comprehend. As a result, in order to resolve legal concerns, this project employs a new state-of-the-art model approach for building a QAS on legal documents.

5.2.2 Product Functions

In a broader perspective the main functions performed by the product are :

1. Provide a simple and intuitive user interface to input required data from the user.
2. Pre-process the legal act content and create good contexts.
3. Generate fact-based questions from contexts.
4. Output the most relevant answer to the user query or the system-generated questions.

5.2.3 User Classes and Characteristics

Users – The system is solely for users, who can enter questions and/or legal documents to get the corresponding answer.

5.2.4 Operating Environment

This software is designed on an Unix-based operating system but is also compatible to run on any other Windows or MAC-OS platforms. After a successful training and testing of model, a web server is developed in Python to handle Question-Answer requests.

1. **For client/web page** - The web application is compatible on any browser on any platform (mobile, tablet or PC) with input capabilities.
2. **For Server and training the model** - Environment with support for NVIDIA CUDA, Python v3.6+, Deep learning libraries for python, Streamlit framework library for developing UI, Networking capabilities and internet connectivity.

5.2.5 Design and Implementation Constraints

1. **Model training time** - May vary from hardware to hardware. Hardware without an NVIDIA GPU can take up to ten times longer to train than hardware with it.
2. **Language** - For the server language used is Python, whereas, for the client, languages will be in HTML, CSS and JavaScript.
3. **Communication** between client and server will be over HTTP using REST API

5.2.6 Assumptions and dependencies

The following assumptions made by the application and its dependencies :

1. The server will run on a 64 bit Linux/Windows OS with support for Python v3.6+ and support for High-end GPUs.
2. Python libraries such as StreamLit web framework, Threading Support along with Database connectivity and Deep learning libraries as Transformers, Pytorch and nltk along with SentencePiece Tokenizer for Answering System.

5.3 External Interface Requirements

5.3.1 User Interface requirements

A user-friendly web application with following functionalities :

1. Allows the user to select the mode of operation.
2. Allows the user to choose a legal act from a list or upload of his/her own choice.
3. Allows the user to enter the query or, if necessary, pick a question from the given list.
4. Displays the system-generated questions as suggestions.
5. Displays the system answer.

5.4 Specific Requirements

5.4.1 Hardware Requirements

The following are the hardware requirements required for the system **Server** side:

1. Supported device types : Windows or Linux based system.
2. High end GPUs with at least 12 GB RAM and 4 CPU cores.
3. Good internet connection.
4. Support for NVIDIA CUDA to exploit the capabilities of the GPUs.
5. Support for HTTP web protocol to accept the client requests and respond to them.

The following are the hardware requirements required for the user or **Client** side:

1. Supported device types : All devices with a web browser.
2. Good internet connection.
3. Support for HTTP web protocol to send the requests to the server and receive the responses.

5.4.2 Software Requirements

- Python3 for implementation of back-end.
- Libraries such as `pdftotext`, `re` for Preprocessing of Legal Acts.
- Libraries like `Transformers`, `pytorch`, `numpy` and `nltk` along with `sentencepiece` Tokenizer for QA and QG.
- `Streamlit` library for developing user interface.
- `Ngrok` for accessing the system from the browser.
- `IPython` notebooks for visualizing the results of fine-tuning and interactive working.

5.5 Other Non-Functional Requirements

- Performance highly depends on whether GPUs are used or not. GPUs like NVIDIA K80 and T4 give better performance.
- System should be responsive in real time.
- System should be robust, easy-to-use, convenient and accessible from anywhere.
- System should not collect any user specific information.
- System should be reusable for other developers.
- System should carry out various activities in synchronized manner.
- System should have modular design and must be easy to maintain, update and test.
- System should be up 24 x 7.
- System should be portable and easily deployed.

Chapter 6

Results and Discussion

Experimentation were carried out on the Legal Text Dataset, generated on the Legal Acts. This dataset was used to fine-tune the BERT model, as well as its variants ALBERT, RoBERTa, and DistilBERT, in all of their configurations. All of the models used for fine tuning, were pre-trained on SQuAD 2.0 dataset. The performance of each model was evaluated using F1 score, Weighted Accuracy (A_w) and Exact Match (EM) metrics and the following results were obtained.

6.1 BERT Variants Evaluation - I

The training and validation losses values obtained while fine tuning the models, as well as the weighted-accuracy class counts obtained when testing these models on the Legal Text Dataset, are shown in the figures below. For each model :

- Figure-(a) illustrates a **line curve** that represents the training and validation losses obtained while training the model for over 4 epochs on the training and validation datasets.
- Figure-(b) illustrates a **bar chart** that represents the no. of questions answered on the test dataset according to the predefined classes of the A_w evaluation.

6.1.1 Fine Tuning ALBERT on Legal Dataset

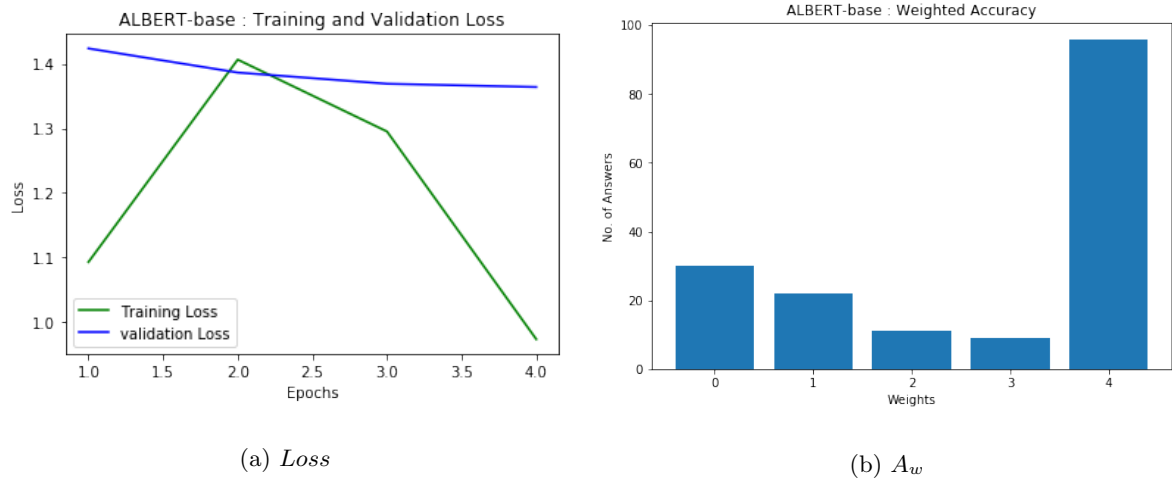


Figure 6.1: ALBERT-base

Observation

Figure 6.1a shows the training and validation losses when ALBERT-base model¹, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss was varying but validation loss kept on decreasing over 4 epochs, showing that model was learning and weights were adjusted accordingly.

The Weighted Accuracy distribution shown in figure 6.1b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by ALBERT-base model were complete and specific, which was expected, but there were also a substantial number of answers answered in the other classes. For other 4 Weight-Classes, that is, for Weight-Classes 0, 1, 2 and 3 the number of answers kept on decreasing as the Weight-Classes were increasing.

¹Pre-trained ALBERT Model used : `twmkn9/albert-base-v2-squad`.

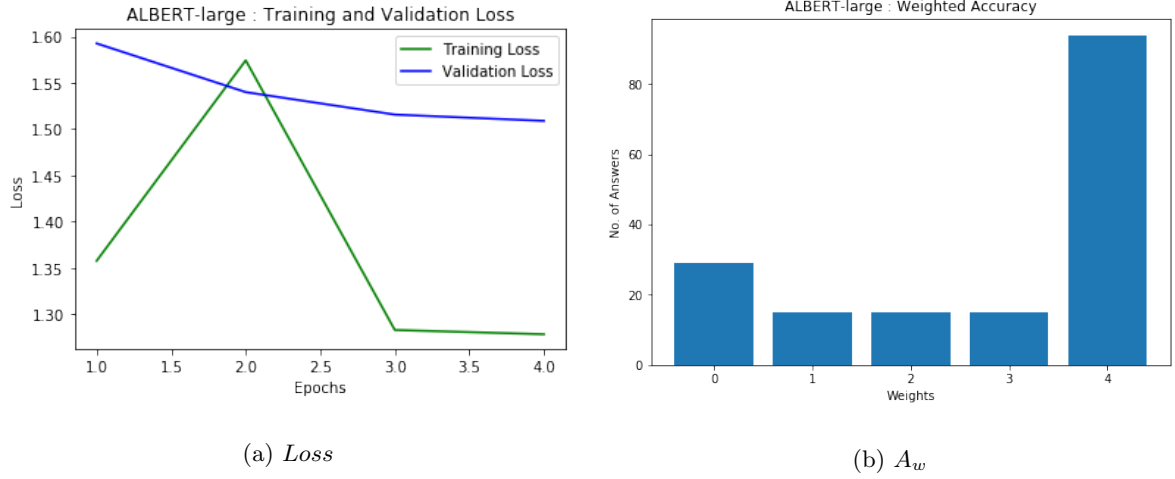


Figure 6.2: ALBERT-large

Observation

Figure 6.2a shows the training and validation losses when ALBERT-large model², pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss increased on second epoch, but decreased drastically and minimal losses were encountered until the fourth epoch. The validation loss kept on decreasing over 4 epochs, showing that model was learning and weights were adjusted accordingly.

The Weighted Accuracy distribution shown in figure 6.2b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by ALBERT-large model were complete and specific, which was expected, but there were also a substantial number of answers answered in the other classes. Surprisingly, the number of class 4 answers in ALBERT-large model were slightly less than ALBERT-base model, which was not expected as ALBERT-large has more parameters and more encoding layers as compared to ALBERT-base (Refer table 2.2).

²Pre-trained ALBERT Model used : `elgeish/cs224n-squad-albert-large-v2`.

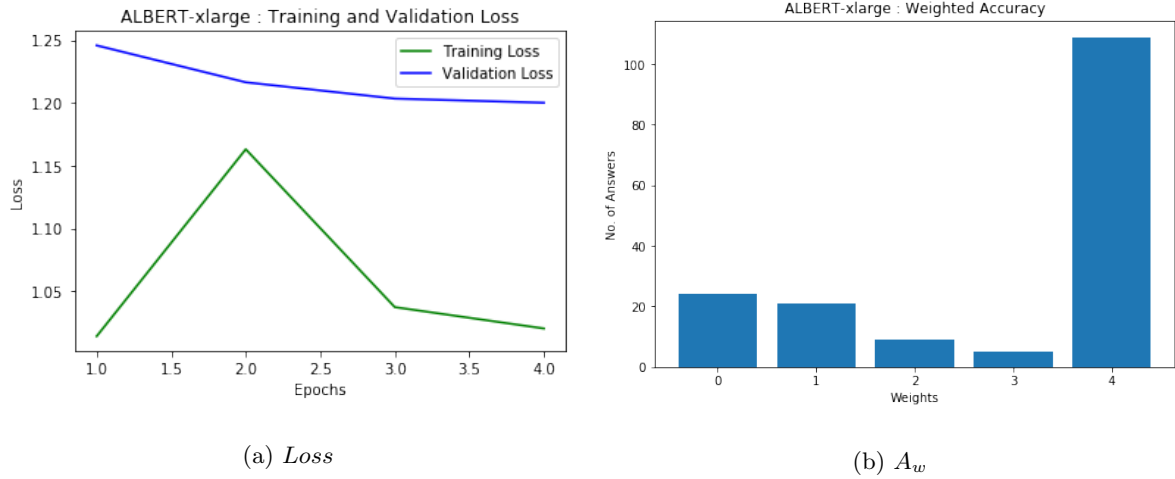


Figure 6.3: ALBERT-xlarge

Observation

Figure 6.3a shows the training and validation losses when ALBERT-xlarge model³, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss increased on second epoch, but kept on decreasing until the fourth epoch. The validation loss kept on decreasing over 4 epochs, showing that model was learning and weights were adjusted accordingly.

The Weighted Accuracy distribution shown in figure 6.3b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by ALBERT-xlarge model were complete and specific, which was expected, but there were also a substantial number of answers answered in the other classes. As expected, The number of Weight-Class 4 answers in ALBERT-xlarge model were slightly more than ALBERT-base model and ALBERT-large model as ALBERT-large has more parameters and more encoding layers as compared to other 2 models (Refer table 2.2).

³Pre-trained ALBERT Model used : [ktrapeznikov/albert-xlarge-v2-squad](https://huggingface.co/ktrapeznikov/albert-xlarge-v2-squad).

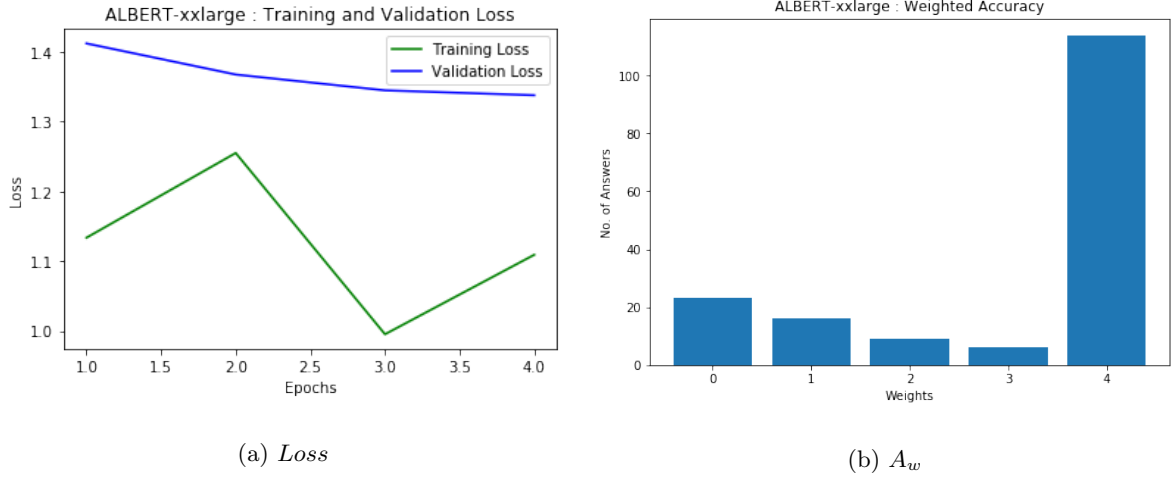


Figure 6.4: ALBERT-xxlarge

Observation

Figure 6.4a shows the training and validation losses when ALBERT-xxlarge model⁴, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss increased on second epoch, decreased drastically and slightly increased at the fourth epoch, while, the validation loss kept on decreasing over 4 epochs, showing that model was learning and weights were adjusted accordingly.

The Weighted Accuracy distribution shown in figure 6.4b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by ALBERT-xxlarge model were complete and specific, which was expected, but there were also a substantial number of answers answered in the other classes. To the expectation, ALBERT-xxlarge model performed the best among its variants with the most number of Weight-Class 4 answers answered as compared to other three. The main reason may be an increase in parameters and encoding layers, (Refer table 2.2) which help in extracting more specific answers.

⁴Pre-trained ALBERT Model used : mfeb/albert-xxlarge-v2-squad.

6.1.2 Fine Tuning BERT-large on Legal Dataset

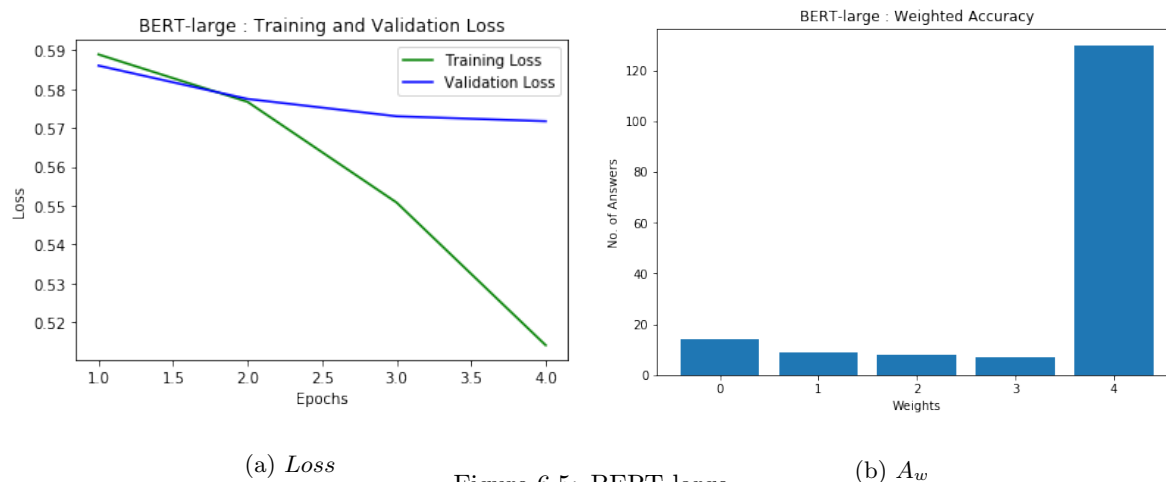


Figure 6.5: BERT-large

Observation

Figure 6.5a shows the training and validation losses when BERT-large model⁵, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss was least in this case and it went on decreasing as with the epochs till its threshold epoch of 4. The validation loss also showed a significant decrease over 4 epochs, demonstrating that the model was improving and that the weights were modified appropriately.

The Weighted Accuracy distribution shown in figure 6.5b represents the distribution of weights, when the model was tested on test set. Almost all of the answers produced by BERT-large model were complete and specific, which was expected, making it the best model to choose for the real time use. But, there were also some answers answered incompletely by the BERT-large model. The incomplete answers, were below 10, which were significantly less as compared to the other models.

⁵Pre-trained BERT Model used : bert-large-uncased-whole-word-masking-finetuned-squad.

6.1.3 Fine Tuning DistilBERT on Legal Dataset

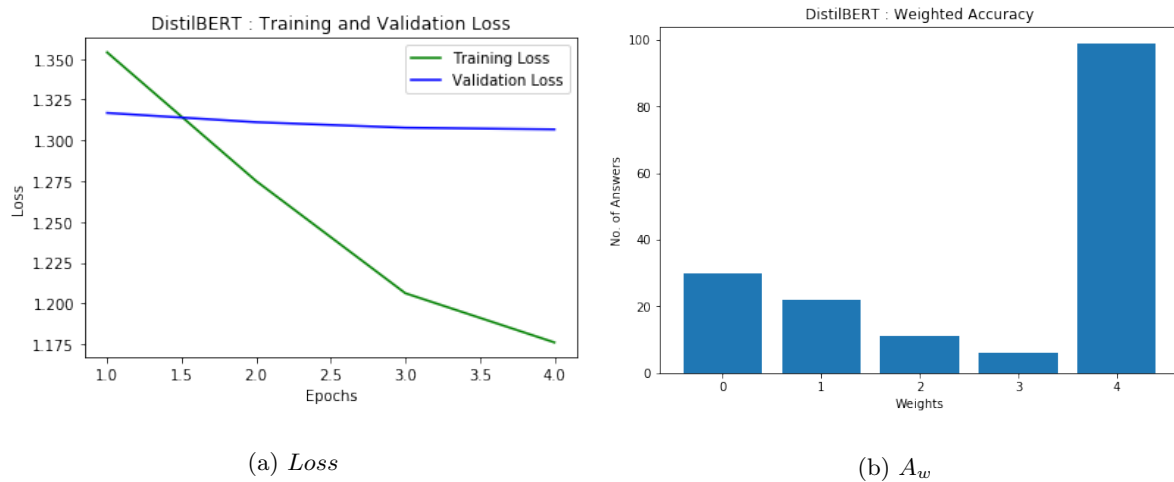


Figure 6.6: DistilBERT

Observation

Figure 6.6a shows the training and validation losses when DistilBERT model⁶, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss went on decreasing as with the epochs till its threshold epoch of 4. The validation loss showed a little decrease over 4 epochs showing that the model was improving. This slight decrease may be due to DistilBERT extracting the most relevant context and leaving out the minute facts that are most needed in the legal domain.

The Weighted Accuracy distribution shown in figure 6.6b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by DistilBERT model were complete and specific, which was expected, but there were also a substantial number of answers answered in the other classes.

⁶Pre-trained DistilBERT Model used : distilbert-base-uncased-distilled-squad.

6.1.4 Fine Tuning RoBERTa on Legal Dataset

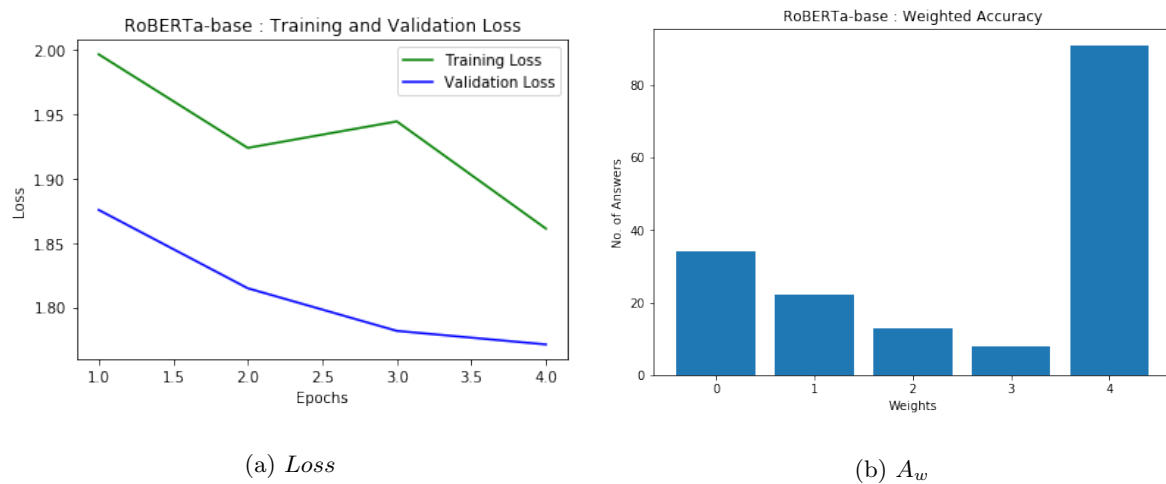


Figure 6.7: RoBERTa-base

Observation

Figure 6.7a shows the training and validation losses when DistilBERT model⁷, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss decreased, then increased a little and again decreased, till the fourth epoch. The validation loss kept on decreasing over 4 epochs showing that the model was improving and learning the weights.

The Weighted Accuracy distribution shown in figure 6.7b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by RoBERTa-base model were complete and specific, which was expected, but these were the least as compared to the other models. The reason may be the dynamic masking approach and pre-training only on MLM task used in RoBERTa.

⁷Pre-trained RoBERTa Model used : csarron/roberta-base-squad-v1.

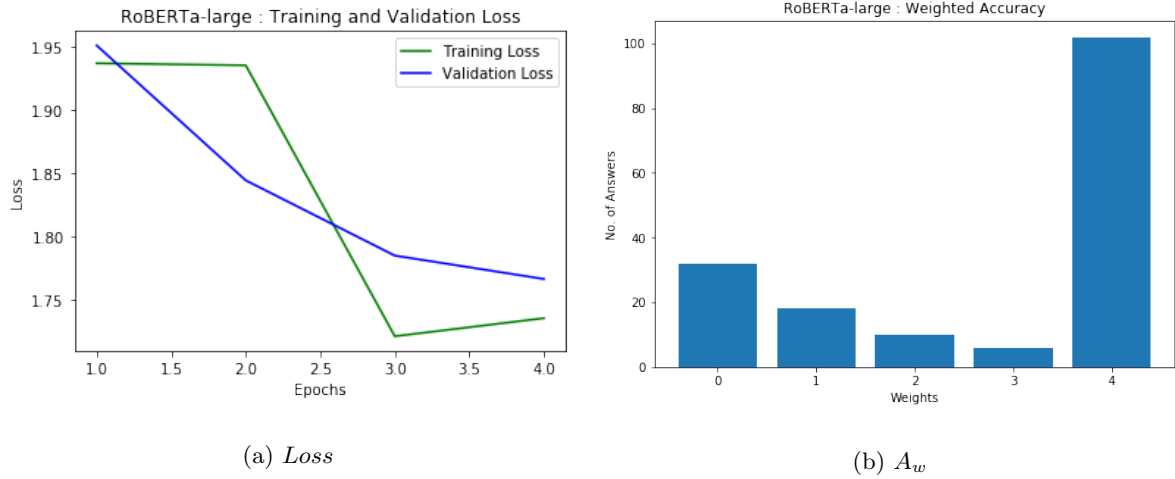


Figure 6.8: RoBERTa-large

Observation

Figure 6.8a shows the training and validation losses when DistilBERT model⁸, pre-trained on SQuAD, was fine tuned on Legal Text Dataset. From the figure, it can be observed that training loss dropped slightly on the first epoch, then decreased sharply on the second epoch, and increased marginally till the fourth epoch. The validation loss kept on significantly decreasing over 4 epochs showing that the model was improving and learning the weights.

The Weighted Accuracy distribution shown in figure 6.8b represents the distribution of weights, when the model was tested on test set. Most of the answers produced by RoBERTa-large model were complete and specific, which was expected. This number was more as compared to RoBERTa-base model, due to increase in the number of parameters. But the dynamic masking approach and pre-training only on MLM task used in RoBERTa, still limit the number of correct answers.

⁸Pre-trained RoBERTa Model used : csarron/roberta-large-squad-v1.

6.2 BERT Variants Evaluation - II

The results obtained on experimenting various BERT variants, after fine-tuning them on the Legal Dataset, are compared below to find the best model for the QAS on Legal documents. These are compared using three performance metrics, namely, Weighted Accuracy(A_w), Exact Match(EM) and F1 score.

6.2.1 Weighted Accuracy (A_w)

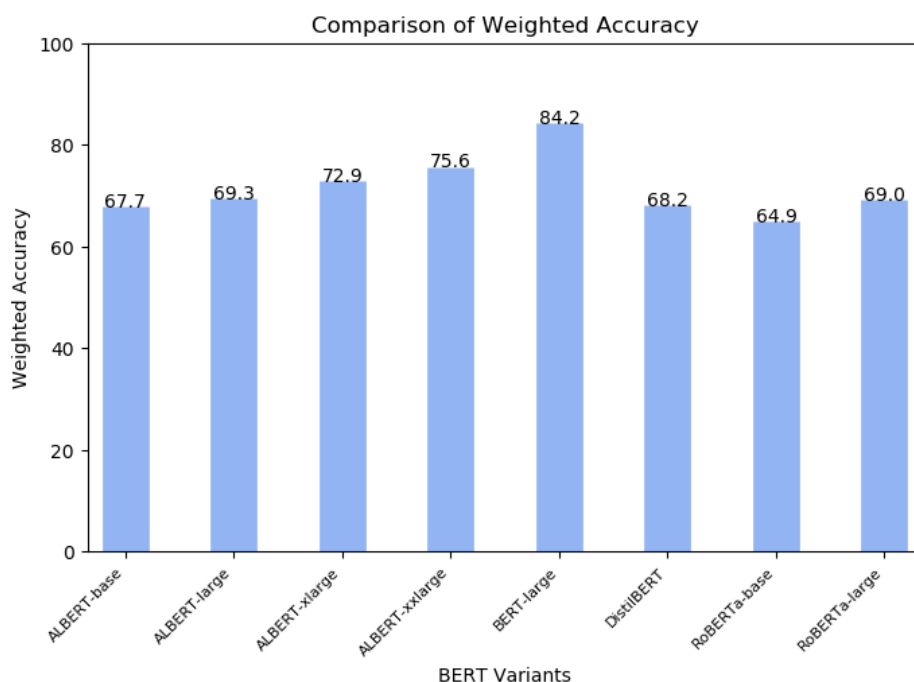


Figure 6.9: Weighted Accuracy Comparison

As we can see, from the figure 6.9, **BERT-large** clearly outperformed its other variants in terms of A_w . One possible reason can be the number of parameters in BERT-Large which are highest as compared all other variants. But this cannot be the only reason as, DistilBERT (66M) has less parameters than RoBERTa-base (110M) model, but the A_w is just reverse. Hence, it also depends on the approach used by these variants. The dynamic masking approach and pre-training RoBERTa only on MLM task, does not show satisfactory results for QA.

6.2.2 F1 Score

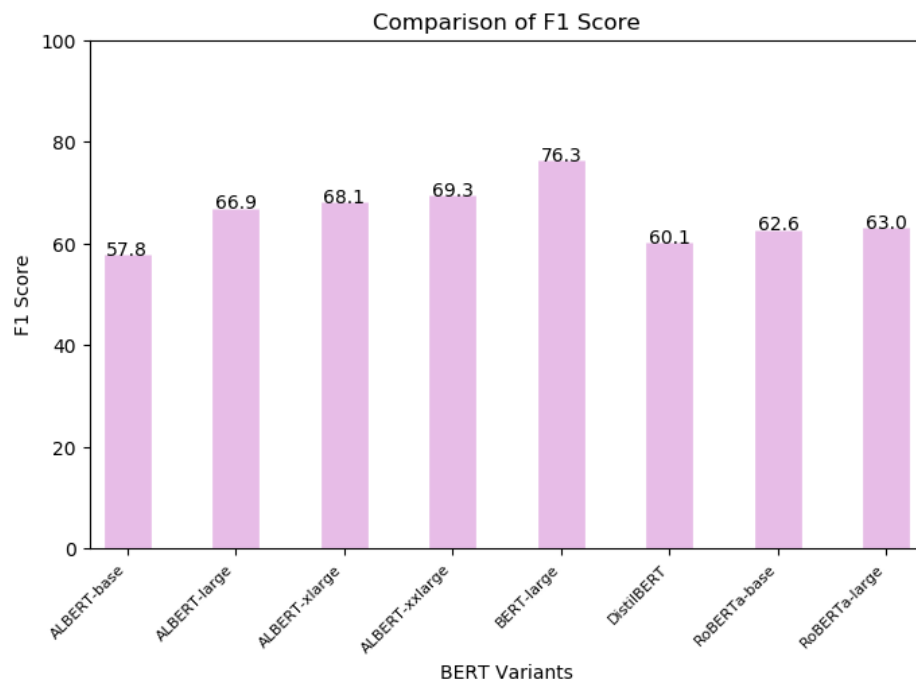


Figure 6.10: F1 Score Comparison

It can be clearly seen from the figure 6.10 that **BERT-large** outperformed its other variants in terms of F1-score. It can be seen that the variants of the BERT were developed with an aim to optimize or to get a light weight version of BERT while trying to attain the similar accuracy as that of BERT. But in doing so, the performance or accuracy of those models is compromised to an extent depending on the methodologies they used.

6.2.3 Exact Match (EM)

From the figure 6.11, it can be observed that **BERT-large** outperformed its other variants in terms of EM too. The possible reasons can be the number of parameters used, the underlying architecture and methodology used in each variant. Interestingly, ALBERT-large under-performed as compared to ALBERT-base model. This was not the expected result with respect to ALBERT model.

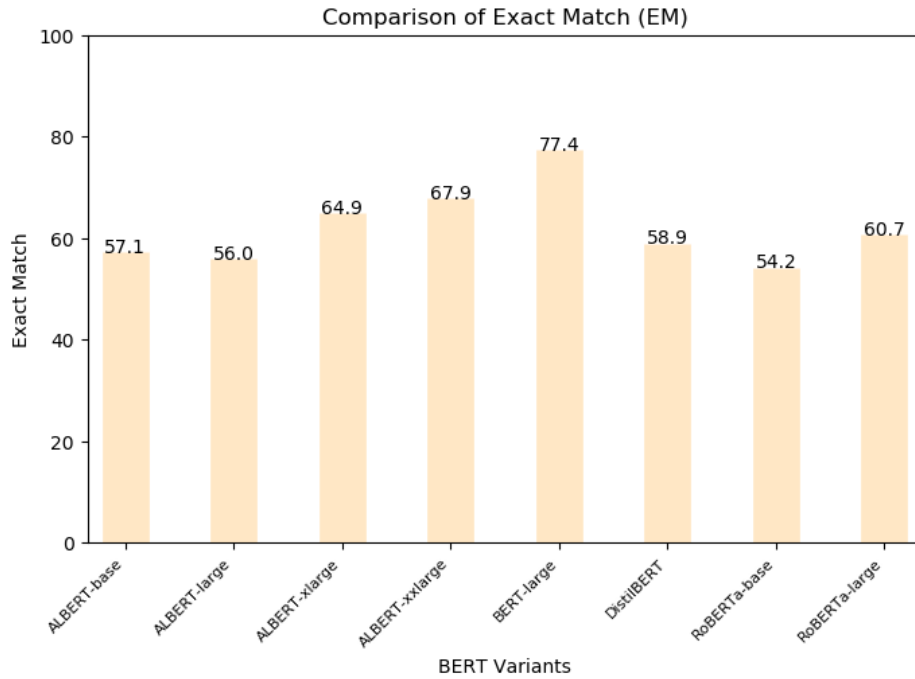


Figure 6.11: Exact Match Comparison

6.3 BERT Variants Evaluation - III

The following tables summarizes the exact numerical metrics values achieved while evaluating all the fine-tuned models on the test dataset :

6.3.1 F1 Score Details

Models	Precision	Recall	F1 score
ALBERT-base	0.74918	0.47052	57.80142
ALBERT-large	0.86048	0.54680	66.86838
ALBERT-xlarge	0.84568	0.56948	68.06309
ALBERT-xxlarge	0.83930	0.59010	69.29782
BERT-large	0.93529	0.64371	76.25794
DistilBERT	0.81338	0.47629	60.07802
RoBERTa-base	0.88051	0.48619	62.64612
RoBERTa-large	0.84243	0.50268	62.96488

Table 6.1: Comparison on F1 Score

Interestingly, from table 6.1, precision was highest in ALBERT-large among its variants and RoBERTa-base has higher precision than RoBERTa-large, but again, BERT-large outperformed all. But, recall followed an expected trend, and was increasing with the new configuration. This affected overall F1-score where the trend was as observed and F1-scores were pre-dominantly increasing as number of parameters in each variant.

6.3.2 Weighted Accuracy and Exact Match Details

Models	W0	W1	W2	W3	W4	A_w	EM
ALBERT-base	30	22	11	9	96	67.70833	57.1429
ALBERT-large	29	15	15	15	94	69.34524	55.9524
ALBERT-xlarge	24	21	9	5	109	72.91667	64.881
ALBERT-xxlarge	23	16	9	6	114	75.59524	67.8571
BERT-large	14	9	8	7	130	84.22619	77.381
DistilBERT	30	22	11	6	99	68.15476	58.9286
RoBERTa-base	34	22	13	8	91	64.88095	54.16667
RoBERTa-large	32	18	10	6	102	69.04762	60.7143

Table 6.2: Comparison on A_w and EM

The table 6.2 showed A_w details, where the details of each Weight-Class can be seen. The first 4 classes had mixed results, but the class 5, which determined EM, had results as expected and **BERT-large** showed the best results, making it the best suitable model.

6.4 BERT Pre-Trained Vs. Fine-tuned

The following are the results obtained when BERT-large pre-trained and fine-tuned were evaluated on test set of Legal Text Dataset. It can be observed that the fine tuning of the BERT-large Model improved its performance on the Legal Domain.

Parameters	BERT Pre-Trained	BERT Fine-tuned
Precision	0.93643	0.93529
Recall	0.64289	0.64371
F1-score	76.2347	76.25794
A_w	83.9286%	84.22618%
EM	76.1904	77.381

Table 6.3: BERT Pretrained vs. Finetuned

Observation

Fine-Tuning is a process which updates and adjusts the parameters of the model to increase the performance of the model in solving a particular problem, making them more task-specific. The fine-tuning of the BERT Pre-Trained Model displayed improved results in answering the questions of the Legal Domain as shown in the above table.

The F1 score increased in the evaluation of the Fine-Tuned BERT, largely due to the increase in the Recall of the answers by the model. The model learnt to answer questions with a few more details as in the reference answer, as expected, as shown in the example.:

Question	In what form can we obtain the information under the RTI Act?
Reference Answer	<i>diskettes floppies tapes video cassettes or in any other electronic mode</i>
BERT Pre-trained	diskettes floppies tapes video cassettes
BERT Fine-Tuned	diskettes floppies tapes video cassettes or in any other electronic mode

The number of words common between the reference and predicted answer were more for the Fine-Tuned Model as it learnt to answer the questions specific to the Legal Domain during its training.

Chapter 7

Conclusion

The project aimed to build a robust Question Answering system for legal documents, particularly the Indian legal acts, by following the Transfer Learning approach. The QA systems built using traditional NLP and deep linguistic techniques fail to assimilate context information required for answering. On the other hand, the modern neural models, advanced in the way they are trained, and the massive datasets on which they are trained, enable them to understand the general English language model and apply it to various downstream tasks. The project discussed and demonstrated the use of such a state-of-the-art BERT model for answering questions on complex Legal Acts.

The subtle language and intricate terminologies of the legal documents make them difficult to comprehend even for legal practitioners. A domain specific QA System can reduce the level of these complexities. The system uses the pre-trained BERT Model, fine-tuned on a Legal Dataset, to address the problem by applying the approach of Transfer Learning. The legal documents contain some insignificant information along with the valuable content. This necessitates several preprocessing steps in order to focus and retrieve only the relevant data. Further, the auxiliary Question Generation component demonstrates automatic question generation using Google's T5 model.

The Legal Dataset consisting of a mixture of machine-generated and human generated questions, incorporates a diversity in the question types, making the model more robust and versatile. The evaluation of the fine-tuned BERT and its variants provide insight into how these models behave for domain-restricted QA tasks and proves that the BERT Large model gives reliable results for the Legal Domain, proving to be the best choice for the system. Last but not the least, the project merges all these system components into a QAS making the Legal Acts easier to understand.

Chapter 8

Future Scope

1. Increasing the scope of the system to other Legal Documents like journals, contracts, legal cases and affidavits.
2. Building a Smart Retrieval System to find the relevant contexts and documents for answering the question.
3. Making the answering system smarter for generating answers in a simplified language.
4. Extending the system capability of answering personal-level questions.
5. Expanding the dataset to encompass more Legal Acts to achieve better results.

Chapter 9

Web Application

9.1 Mode 1 Operation

Question Answering system for Legal Documents

Select the mode of operation :

- ☒ Generate and Answer Question on Document
☐ Answer Question on Document

Select a Legal Document :

The Juvenile Justice (Care and Protection of Children) Act, 2015

Choose a file



Drag and drop file here
Limit 200MB per file • PDF

Browse files

Generate more samples

Generated Questions :

When was the Juvenile Justice (Care and Protection of Children) Act, 2000 enacted?

Figure 9.1: Mode - 1 (a)

Question :

When was the Juvenile Justice (Care and Protection of Children) Act, 2000 enacted?

Answer :

Sixty-sixth Year of the Republic of India

Context :

An Act to consolidate and amend the law relating to children alleged and found to be in conflict with law and children in need of care and protection by catering to their basic needs through proper care, protection, development, treatment, social re-integration, by adopting a child-friendly approach in the adjudication and disposal of matters in the best interest of children and for their rehabilitation through processes provided, and institutions and bodies established, herein under and for matters connected therewith or incidental thereto. WHEREAS, the provisions of the Constitution confer powers and impose duties, under clause (3) of article 15, clauses (e) and (f) of article 39, article 45 and article 47, on the State to ensure that all the needs of children are met and that their basic human rights are fully protected; AND WHEREAS, the Government of India has acceded on the 11th December, 1992 to the Convention on the Rights of the Child, adopted by the General Assembly of United Nations, which has prescribed a set of standards to be adhered to by all State parties in securing the best interest of the child; AND WHEREAS, it is expedient to re-enact the Juvenile Justice (Care and Protection of Children) Act, 2000 (56 of 2000) to make comprehensive provisions for children alleged and found to be in conflict with law and children in need of care and protection, taking into consideration the standards prescribed in the Convention on the Rights of the Child, the United Nations Standard Minimum Rules for the Administration of Juvenile Justice, 1985 (the Beijing Rules), the United Nations Rules for the Protection of Juveniles Deprived of their Liberty (1990), the Hague Convention on Protection of Children and Co-operation in Respect of Inter-country Adoption (1993), and other related international instruments. BE it enacted by Parliament in the Sixty-sixth Year of the Republic of India as follows:-

Figure 9.2: Mode - 1 (b)

9.2 Mode 2 Operation

Question Answering system for Legal Documents

Select the mode of operation :

- ☐ Generate and Answer Question on Document
☒ Answer Question on Document

Select a Legal Document :

Select Act ▾

Choose a file



Drag and drop file here
Limit 200MB per file • PDF

Browse files



2019 - The Consumer Protection Act, 2019.pdf 0.7MB



Enter the Question :

What does "advertisement" mean ?

Get the Answer

Answer :

any audio or visual publicity, representation, endorsement or pronouncement made by means of light, sound, smoke, gas, print, electronic media, internet or website

Figure 9.3: Mode - 2 (a)

Some suggested questions :

What is the only state that is excluded from the scope of the Consumer Protection Act, 2019? ▾

Question :

What is the only state that is excluded from the scope of the Consumer Protection Act, 2019?

Answer :

Jammu and Kashmir

Figure 9.4: Mode - 2 (b)

Chapter 10

Project Timeline

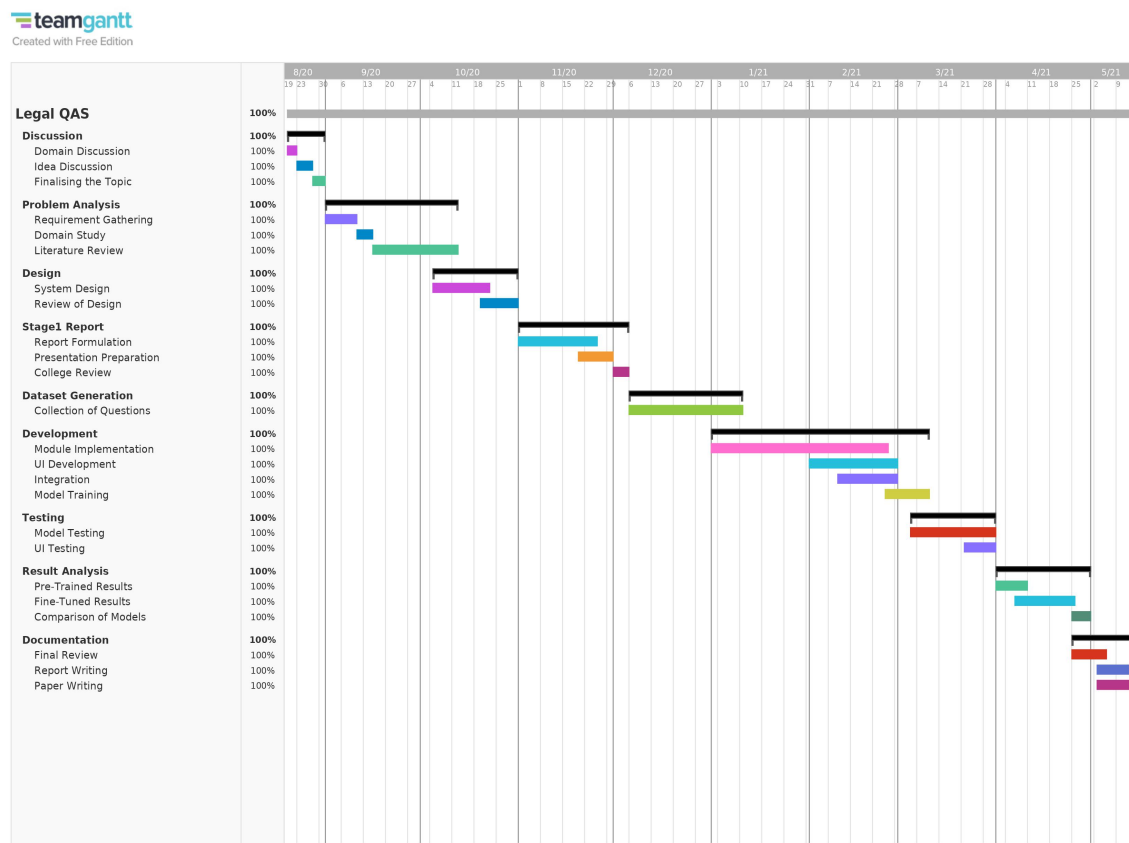



Figure 10.1: Project Timeline

Appendix A

Sample Legal Act

बिस्वी के सी एल (एन) 04/0007/2003-05	REGISTERED NO. DL (N) 04/0007/2003-05
--------------------------------------	---------------------------------------


भारत का राजपत्र
The Gazette of India
असाधारण
EXTRAORDINARY
भाग II — खण्ड 1
PART II — Section 1
प्राधिकार से प्रकाशित
PUBLISHED BY AUTHORITY

क्र. 25]	नई दिल्ली, मंगलवार, जून 21, 2005/ज्यैष्ठ 31, 1927
No. 25]	NEW DELHI, TUESDAY, JUNE 21, 2005/JYAISTHA 31, 1927

इस भाग में विभिन्न पृष्ठ संख्या दी जाती है जिससे कि यह अलग संकलन के रूप में रखा जा सके।
Separate paging is given to this Part in order that it may be filed as a separate compilation.

MINISTRY OF LAW AND JUSTICE
(Legislative Department)
New Delhi, the 21st June, 2005/Jyaistha 31, 1927 (Saka)

The following Act of Parliament received the assent of the President on the 15th June, 2005, and is hereby published for general information:—

THE RIGHT TO INFORMATION ACT, 2005
No. 22 of 2005

[15th June, 2005.]

An Act to provide for setting out the practical regime of right to information for citizens to secure access to information under the control of public authorities, in order to promote transparency and accountability in the working of every public authority, the constitution of a Central Information Commission and State Information Commissions and for matters connected therewith or incidental thereto.

WHEREAS the Constitution of India has established democratic Republic;

AND WHEREAS democracy requires an informed citizenry and transparency of information which are vital to its functioning and also to contain corruption and to hold Governments and their instrumentalities accountable to the governed;

AND WHEREAS revelation of information in actual practice is likely to conflict with other public interests including efficient operations of the Governments, optimum use of limited fiscal resources and the preservation of confidentiality of sensitive information;

Figure A.1: A Sample Legal Act

THE RIGHT TO INFORMATION ACT, 2005

ARRANGEMENT OF SECTIONS

CHAPTER I

PRELIMINARY

SECTIONS

1. Short title, extent and commencement.
2. Definitions.

CHAPTER II

RIGHT TO INFORMATION AND OBLIGATIONS OF PUBLIC AUTHORITIES

3. Right to information.
4. Obligations of public authorities.
5. Designation of Public Information Officers.
6. Request for obtaining information.
7. Disposal of request.
8. Exemption from disclosure of information.
9. Grounds for rejection to access in certain cases.
10. Severability.
11. Third party information.

CHAPTER III

THE CENTRAL INFORMATION COMMISSION

12. Constitution of Central Information Commission.
13. Terms of office and conditions of service.
14. Removal of Chief Information Commissioner or Information Commissioner.

CHAPTER IV

THE STATE INFORMATION COMMISSION

15. Constitution of State Information Commission.
16. Term of office and conditions of service.
17. Removal of State Chief Information Commissioner or State Information Commissioner.

CHAPTER V

POWERS AND FUNCTIONS OF THE INFORMATION COMMISSIONS, APPEAL AND PENALTIES

18. Powers and functions of Information Commissions.

THE RIGHT TO INFORMATION ACT, 2005

ACT No. 22 OF 2005

[15th June, 2005.]

An Act to provide for setting out the practical regime of right to information for citizens to secure access to information under the control of public authorities, in order to promote transparency and accountability in the working of every public authority, the constitution of a Central Information Commission and State Information Commissions and for matters connected therewith or incidental thereto.

WHEREAS the Constitution of India has established democratic Republic;

AND WHEREAS democracy requires an informed citizenry and transparency of information which are vital to its functioning and also to contain corruption and to hold Governments and their instrumentalities accountable to the governed;

AND WHEREAS revelation of information in actual practice is likely to conflict with other public interests including efficient operations of the Governments, optimum use of limited fiscal resources and the preservation of confidentiality of sensitive information;

AND WHEREAS it is necessary to harmonise these conflicting interests while preserving the paramountcy of the democratic ideal;

Now, THEREFORE, it is expedient to provide for furnishing certain information to citizens who desire to have it.

BE it enacted by Parliament in the Fifty-sixth Year of the Republic of India as follows:—

CHAPTER I

PRELIMINARY

1. Short title, extent and commencement.—(1) This Act may be called the Right to Information Act, 2005.

(2) It extends to the whole of India^{1***}.

(3) The provisions of sub-section (1) of section 4, sub-sections (1) and (2) of section 5, sections 12, 13, 15, 16, 24, 27 and 28 shall come into force at once, and the remaining provisions of this Act shall come into force on the one hundred and twentieth day of its enactment.

2. Definitions.—In this Act, unless the context otherwise requires,—

(a) "appropriate Government" means in relation to a public authority which is established, constituted, owned, controlled or substantially financed by funds provided directly or indirectly—

(i) by the Central Government or the Union territory administration, the Central Government;

(ii) by the State Government, the State Government;

(b) "Central Information Commission" means the Central Information Commission constituted under sub-section (1) of section 12;

(c) "Central Public Information Officer" means the Central Public Information Officer designated under sub-section (1) and includes a Central Assistant Public Information Officer designated as such under sub-section (2) of section 5;

(d) "Chief Information Commissioner" and "Information Commissioner" mean the Chief Information Commissioner and Information Commissioner appointed under sub-section (3) of section 12;

1. The words "except the State of Jammu and Kashmir" omitted by Act 34 of 2019, s. 95 and the Fifth Schedule (w.e.f. 31-10-2019).

Appendix B

Publication Details

1. **ELMNLP 2021** : The project is submitted for review in The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021). As in previous years, some of the conference presentations will be for papers published for publication in the Transactions of the ACL (TACL) and Computational Linguistics (CL) journals.

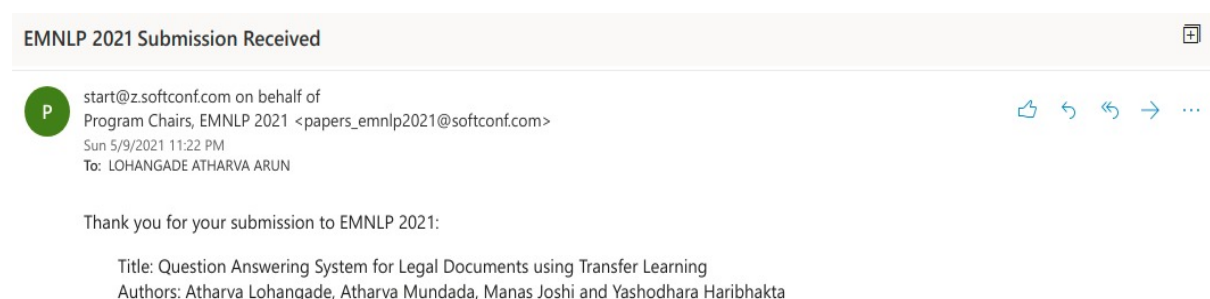


Figure B.1: Mail from ELMNLP 2021

Appendix C

Source Code and Dataset

C.1 Source Code

The source code for the entire project “Question Answer System using Transfer Learning Approach for Legal Documents” is available at [\[Github Link\]](#) under MIT License.

C.2 Dataset

Legal Text Dataset: Legal Text Dataset, built using Indian Legal Acts, consists of around 2500 questions, is available at [\[Dataset Link\]](#).

Appendix D

Issues Resolved

Following are the issues which were observed in the previous system (during MidSem) but are resolved in the current system :

- ✓ The T5 model and the BERT model, used for the question generation and the answering system, respectively, have a maximum of 512 token length input. On the other hand, legal documents are very large, which poses the problem of sectorizing the document and finding the right context for a question efficiently.
- ✓ The sectorizing of the document leads to question batches which need to be given as input to the BERT model. This requires two additional steps of padding and masking which BERT does not handle and has to be handled explicitly and without any error.
- ✓ The answer extraction T5 model was developed using the transformers library of version 3.0.0 and the question generation T5 model was developed using version 4.3.3. Before version 4, the sentencepiece tokenizer was included in the transformer library itself as a required dependency. In the release of v4 of transformers, the integrated sentencepiece tokenizer module was removed and thus we need to install it explicitly. But by doing so, the answer extraction model which is dependent on transformer v3 is not able to instantiate the new sentencepiece tokenizer and it fails to extract the answer spans. This issue will be resolved in further work.

Bibliography

- [1] Ashish Vaswani, Llion Jones, Noam Shazeer, Niki Parmar, Aidan N. Gomez, Jakob Uszkoreit, Łukasz Kaiser, Illia Polosukhin. 2017. “Attention Is All You Need.” *arXiv:1706.03762*.
- [2] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, Phil Blunsom, 2015. “Teach Machines to Read and Comprehend” *arXiv:1506.03340*
- [3] Alammr, Jay. 2018. “The Illustrated Transformer.” [Blog].
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *arXiv:1810.04805*
- [5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. “SQuAD: 100,000+ questions for machine comprehension of text.” *arXiv:1606.05250*.
- [6] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. “Know what you don’t know: Unanswerable questions for SQuAD.” *arXiv:1806.03822*.
- [7] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.” *arXiv:1909.11942*.

- [8] Yinhan Liu, and Myle Ott. 2019. “RoBERTa: A Robustly Optimized BERT Pre-training Approach.” *arXiv:1907.11692*.
- [9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. “Distilling the Knowledge in a Neural Network.” *arXiv:1503.02531*.
- [10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” *arXiv:1910.01108*
- [11] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. “The first question generation shared task evaluation challenge.” *6th INLGC*.
- [12] Michael Heilman and Noah A. Smith. 2010. “Good question! statistical ranking for question generation.” *ACL 2010*
- [13] Xinya Du, Junru Shao, Claire Cardie. 2017. “Learning to Ask: Neural Question Generation for Reading Comprehension.” *arXiv:1705.00106*.
- [14] Nan Duan, Duyu Tang, Peng Chen, Ming Zhou. 2017. “Question Generation for Question Answering.” *ELMNLP 2017*
- [15] Luis Enrico Lopez, Diane Kathryn Cruz, Jan Christian Blaise Cruz, Charibeth Cheng. 2020. “Transformer-based End-to-End Question Generation.” *arXiv:2005.01107*
- [16] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu. 2020. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” *arXiv:1910.10683v3*.
- [17] The Constitution of India. [Link].

- [18] HuggingFace transformers documentation for pre-trained models, [Link].
- [19] Sudharsan Ravichandiran. January 2021. “Getting Started with Google BERT.”
ISBN 978-1-83882-159-3.