Contextual Answer Validation

Arkadeep Baksi

Contextual Answer Validation

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Arkadeep Baksi

[Roll No: CS2006]

under the guidance of

Dr. Debapriyo Majumdar.

Computer Vision and Pattern Recognition Unit

Indian Statistical Institute
Kolkata-700108, India

July 2022

To my family and my guide

CERTIFICATE

This is to certify that the dissertation entitled **"Contextual Answer Validation"** submitted by **Arkadeep Baksi** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Debapriyo Majumdar
Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Dr. Debapriyo Majumdar*, Computer Vision and Pattern Recognition Unit , Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has literally taught me how to do good research, and motivated me with great insights and innovative ideas.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added valuable suggestions and discussions which added.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Arkadeep Baksi
Indian Statistical Institute
Kolkata - 700108 , India.

l

Abstract

Answer generation for a question, given a context has gained tremendous popularity in the NLP research space. Benchmark datasets like SQuAD[9] have propelled the research and recent years have seen many transformer based models achieving state of the art (SOTA) results on Question Answering tasks even beating human level accuracy. However the second step to a Question Answering System that Contextual Answer Validation is a much less attempted space in NLP.

For the past few years India has seen a tremendous growth in the Edtech industry. These edtech firms are sitting on a gold mine of data primarily in Question Answering space. As a result there is a growing demand for automatic Answer Validation Systems as well which can bypass the norm of human evaluation, automating the process. Apart from these, demand for such systems is also there in the Chatbot space to validate junk/spam responses and smoothen the chatbot experience overall.

In our work we attempted the answer validation problem with the additional constraints of the answer being single sentence long and having 10 words atleast. However due to the unavailability of exact datasets we had to generate synthetic data based on the SQuAD dataset. We build our model inspired from paraphrase detection and fine-tuned it against various datasets clubbed with the synthetic data we generated. Our model on final evaluation even hit an accuracy of **0.83** on the highly complex PAWS dataset which typically contains lexically highly overlapped examples.

Contents

Chapter 1

Introduction

1.1 Natural Language Processing Overview

A combination of Artificial Intelligence and Linguistics, Natural Language Processing paves the way for computers to comprehend literature or text published in human languages. In order to communicate with computers in natural language, as we interact with one another as humans, a field known as Natural Language Processing, or NLP, was created. Consequently, Machine Translation(MT) is one of the most promising areas of research in this space. Building automatic systems that can translate text from one human spoken language to another is the aim of Machine Translation.

One of the most challenging tasks for any computer is processing human text. Any sentence in a language comes with an association of different things like nouns, determinants, adjectives, verbs, etc. which makes the task of processing human text even more difficult. In addition to this, the variation in interpretations of sentences, like in the following example : "the new software version is running faster than the old version" and "these atheletes will be running the marathon" adds to the complexity. In both these sentences, the word "running" has a completely different meaning. The context in which a word is used often affects its meaning, as rightly said by *J.R.Firth* [24] "You shall know a word by the company it keeps". It is challenging for computers to parse these kinds of sentences with ambiguity in word choices.

When we consider the language's own grammar, the task becomes even more challenging. In fact, by introducing new rules, the grammar itself can be enhanced to handle sub-categorization (for instance, the verb would is frequently employed in conditional sentences that begin with the word "If"). However, research have revealed that because of the complexity and ambiguity of the language, the application of rule-based techniques does not produce superior results. Therefore, a new branch of NLP known as Statistical NLP was created to address all of these issues. In this branch, a parsing

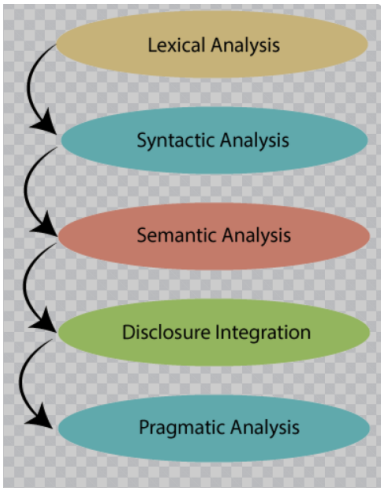

Figure 1.1: Components of NLP[28]

rule is often given a probability with the aid of various machine learning techniques. Many rules are gathered using these probabilities as a base, and some more general rules are then deduced from them. These general rules then aids the process of decision making via statistical decision trees or any other machine learning model which typically encapsulates the principle of Maximum Likelihood. Thus Statistical NLP approaches produced good results for quite a few decades until the introduction of deep learning in NLP.

Five main parts can be considered to make up NLP : Pragmatic Analysis, Disclosure Integration, Semantic Analysis, Syntactic Analysis, and Morphological Analysis as shown in Figure 1.1. Parsing is considered to be the main step in Syntactic Analysis where the words which constitute the text or sentence is analyzed based on the underlying grammar of the language, to determine its syntactic structure.

By arranging the words in the text in a way that makes grammatical sense, Syntax Analysis can be used to extract dictionary meaning from the text. This allows for the evaluation of the text's dictionary meaning based on actual grammatical rules. Semantic analysis is used to evaluate the meaning of the constituent words as well as their grammatical placement and usage within a text or phrase. It is used by NLP to determine a text's precise contextual meaning and semantic organisation. Deep learning has recently transformed NLP and is now receiving a lot of attention from NLP researchers. Traditional methods including Statistical NLP are merely stagnant at this stage where deep learning based NLP methods are achieving State of the Art(SOTA) results in almost all of the NLP tasks like Part-of-speech(POS) tagging[[23],[33]], Named Entity Recognition (NER)[[16],[31]], Text Classification, Text Summarization[12], etc. The major success of deep learning in NLP in the recent

days is majorly attributed to two reasons: one being the availability of huge amount of data and secondly the advancement and availability of fast hardware. These have enabled a plethora of NLP-based applications ranging from *Google's* powerful search engine to *Amazon's Alexa* and finally *Apple's* voice assistant *Siri*. All these are products of the cutting edge NLP research that is being carried out worldwide.

1.2 Deep Learning in NLP

Deep learning involves training deep neural networks on vast amounts of data to carry out tasks that range from straightforward text classification to complex reasoning. It is a collection of processes that, with the ideal input dataset, can extract the best answer to any problem. Or, to put it another way, it simply fits a difficult mathematical function to the information that is given. Deep learning, broadly speaking, is the process of identifying and evaluating significant structures and patterns in data with the goal of developing a solution to a specific problem.

Researchers are now using deep learning architectures and methodologies to produce state-of-the-art NLP outcomes following the Statistical NLP era. Recurrent Neural Networks (RNN)[32], Convolutional Neural Networks (CNN)[42], Long-Short Term Memory (LSTM)[25], and more recently Transformers[40] have all been used in significant NLP applications work.

Some recent developments in deep learning-based natural language processing (NLP) systems and applications were reviewed by *Young et al.*[41] The review and comparison of models and approaches that have produced state-of-the-art (SOTA) results on a variety of NLP tasks, including Machine Translation and Visual Question Answering (Visual-QA)[13], are the main topics of this paper. Often the advancement of NLP is bound to how effectively we have achieved the task of language modelling. A key objective of Statistical Natural Language Processing (NLP) was to offer a probabilistic representation of linguistic word sequences, a challenging undertaking because of the *Curse of Dimensionality*[29]. By learning words to have a distributed representation (embedding) and by giving sequences a probability function, *Bengio et al.*[14] provided some ground-breaking research for avoiding the Curse of Dimensionality. These distributed representations however gained popularity with the advent of Continuous-Bag-of-Words(*CBOW*) and the Skip Gram models[26] as they produced high-quality word embeddings. However only distributional semantics alone was not good enough to cater to the wide range of NLP tasks as they lacked in the flavour of contextuality. Thus with newer tasks being added, a major goal of NLP applications have become to provide good contextual representation of texts, documents. This paved the trail from embeddings to ELMO[35], Transformers, etc. Another key problem in the field of NLP was the problem of transfer learning for downstream

tasks which was again solved by the paper which introduced BERT[21]. Combining all these techniques have paved the way for better language modelling and machine translation, the core goals of Natural Language Processing.

1.3 Answer Validation

Generating answers for a question written, given a context has gained tremendous popularity in the NLP research world. Through contests like NTCIR QAC (Fukumoto et al., 2004) and TREC QA Track (Voorhees, 2004), as well as more recently on the SQuAD[9] dataset by Stanford, research activities in Question-Answering have been encouraged. The opposite procedure, known as Answer Validation, requires computers to simulate human evaluation of QA replies and determine whether or not a response to a question is correct in light of the provided text. Language modeling, machine comprehension and question answering have become a standard problem in NLP with many deep learning models being proposed top solve it. However this particular area of answer validation has not gained much attention from the NLP community as compared to Question Answering.

In the natural world, questions can be of different types where the answers corresponding to the questions can be varied as well. This makes the task of answer validation seemingly difficult. However answer validation itself is a broad topic and the defining scenario can be varied as well. One such scenario can be that we are given a triplet (Context, Question, Answer) and the task is to check if the answer given is correct according to the text or not. However a little different scenario can be that we are given a context, a question, a correct answer to the question and a candidate answer which we need to validate if it is correct or not. Currently with advances in the question answering problem and some ground breaking research by *Chowdhery et al.*[17], generating a correct answer to a question given a context is feasible atleast at small scales. Hence the second scenario of answer validation seems more frequent in reality and we try to solve this problem in our work.

However even with the defining scenario fixed, still the problem of answer validation can be varied depending on the type of answer particularly depending on their length. In case of factoid questions, the answers typically tend to be short as they are primarily keyword based (from the context). If a correct response to such a question is already provided validating a candidate response just reduces to the problem of *"String Matching"*. Hence such type of questions are of little interest to us and we try to solve the problem where the question is non-factoid and the answer length is typically greater than 9 words and consists of a single sentence. However the case where the answer to the question spans multiple sentences is purposely not dealt in our work owing to resource and time constraints.

1.4 Motivation

The primary motivation propelling our work in answer validation is the intrinsic difficulty of the problem and the rising demand of such answer validation system in the *Edtech* industry. According to [7], "The Indian EdTech industry was valued at USD 750 million in 2020 and is expected to reach USD 4 billion by 2025 at a CAGR of 39.77% ." This exponential growth is primarily driven by the COVID-19 pandemic and the increasing demand for academic and non-academic courses from the tier II and tier III cities in India. The industry is also seeing an ever-growing demand for personalisation as more and more students are getting onboarded to different such platforms for the preparation of various competitive exams. Now naturally this particular industry is sitting on a goldmine of data primarily constituting questions, answers of different types and exams. However research shows that most of the the evaluation even at this scale is done manually. Hence a very effective and state of the art answer validation system can prove to be an asset for any such *Edtech* organization.

However, the scope of such answer validation systems is not just limited to the *Edtech* space. Even within the booming *Chatbot* industry it finds application in validating and responses from concerned users and rejecting the ones intended at baffling the system.

1.5 Problem Statement

The problem which we intended to solve is the Answer Validation problem with the scenario that we are given a quadruplet (Context, Question, Correct Answer, Candidate Answer) and we need to verify whether the potential response is contextually appropriate or not. Additionally as mentioned in the Section 1.3, we restricted the problem scope where the question is not factoid and the answer is typically a single sentence with atleast 10 words.

Chapter 2

Related Work

The exact problem that we have intended to work upon is not a very popular one in the NLP space and to the best of our knowledge there has not been much work catered to the exact problem. However over the last two decades there has been some work contributing to Answer Validation although in entirety it does not align with the problem we are solving. We will discuss some of these works here in this section.

Tonoike et al.[39] proposed an answer validation method by use of **Keyword Association** for validating factoid answers. The answer validation procedure described in the paper can be broken down into two major steps. The first involves extracting pertinent keywords from the given question using features of the word and how strongly they are associated lexically, while the second involves estimating how strongly the keywords and candidate answer are associated based on the hits from the search engine. The author's primary premise was that if the keywords and candidate answers are strongly associated they would co-occur more frequently online. They suggested a Keyword Association-based strategy for factoid response validation based on this understanding.

Answer Validation was approached by *Peñas et al.*[34] as a problem of **Textual Entailment**(TE). The goal of the job is to recognize Textual Entailment i.e. to assess if the veracity of one text necessitates the veracity of another text, referred to as a hypothesis, or, put another way, whether the hypothesis's meaning is incorporated into the text's overall meaning. The implicational link between texts is useful for a variety of tasks, including Question Answering (QA), where the answer to a question must be implied by the text that supports the veracity of this answer, or Automatic Summarization, where a system may omit passages whose meaning is already implied by other passages. To identify whether a collection of text-hypothesis pairings comprised the hypothesis or not, systems that recognize Textual Entailment required to output YES or NOT for each pair. According to the authors, after some reformulations, a Textual Entailment Recognizer may be used to check the responses when QA

systems produce both an answer and a text that demonstrates the accuracy of the answer. The reformulations they proposed are as follows:

- Create a hypothesis by transforming the question and the response into an affirmative statement. For instance, given the response "President of Mexico" to the inquiry "Who is Vicente Fox? ", one viable hypothesis may be "Vicente Fox is the President of Mexico."
- Evaluate the entailment: The answer is anticipated to be valid if the supporting language implies this theory.

Apart from answer validation we have used the work of *Chung et al.*[18] for generating distractors using BERT. The following two restrictions on the current Distractor Generation (DG) techniques are examined in this research. First, the quality of the Distractor Generation approaches currently in use is still far from being useful. The quality of Distractor Generation can yet be improved. Second, the single distractor generation is the focus of most current Distractor Generation systems. Multiple distractions are preferred, though, for effective MCQ preparation. In order to efficiently generate a variety of distractors, the paper offers a unique Distractor Generation approach that blends negative answer training tactics with multitasking. The testing results show that their approach increases the state-of-the-art result (BLEU 1 score) from 28.65 to 39.81, and many of the manufactured distractions are diverse and significantly improve student's capacity to avoid answering multiple-choice questions.

Chapter 3

Proposed Work

3.1 Pre-requisites

3.1.1 Recurrent Neural Networks (RNN)

Humans do not always begin to think from scratch. Even while performing any task, for e.g like reading, we understand each constituent word from our knowledge of previous words. Our thoughts have consistence and we don't start to think from scratch every time.

This is a key bottleneck of traditional neural networks. For instance, a standard neural network might not be able to identify the type of event that is occurring at a specific time in a video since it is unclear how it will utilise the past history of occurrences in the video to do so.

Recurrent Neural Networks [38] try to get over this shortcoming. They are essentially networks with loops in them which allows the persistence of information. In the Figure 3.1, the neural network A examines some input x_t and produces the result h_t. A loop forms the path for information to be flow from one state of the network to the next.

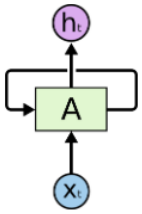

Figure 3.1: Recurrent Neural Network[8]

These loops make Recurrent Neural Networks a bit intimidating at a first glance, but if we carefully look then a Recurrent Neural Network can be thought as multiple copies of the same network with each passing some information to the next and so on. See Figure 3.2 to look at an unrolled recurrent neural network:

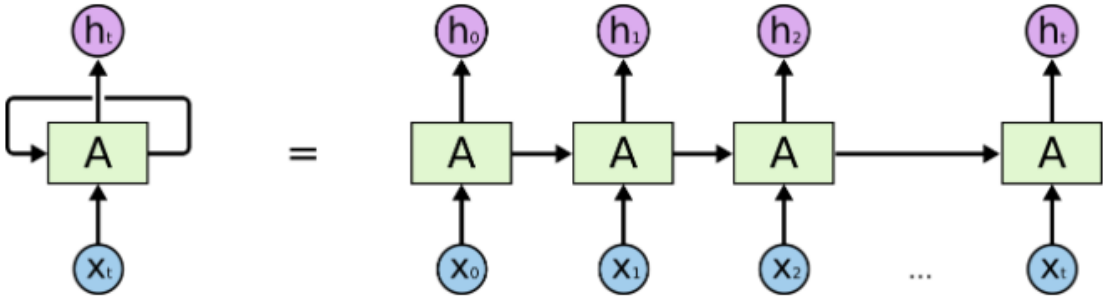

Figure 3.2: Unrolled Recurrent Neural Network[8]

Due to this looping back property, during the last decade there has been tremendous success applying RNNs to a plethora of problems ranging from speech recognition and machine translation to image captioning. This blog [1] by Andrej Karpathy illustrates the effectiveness of RNNs quite comprehensively. Hence readers are encouraged to refer it.

However there are some drawbacks even to Recurrent Neural Networks. Although powerful, Recurrent Neural Networks suffer from the problem of limited memory. Sometimes, to do the work at hand, we only need the current knowledge. RNNs are able to learn to utilise the previous data in situations like this, when there is a little time gap between the relevant information and the location where it is needed. However, there are certain situations when the historical background we actually require is rather far away. Let's say we're attempting to guess the final word in the sentence "I am born and raised in France.....I can speak fluent French." Now here is the catch, based on the recent knowledge from past, the following term is undoubtedly the name of a language, but in order to identify which language, we would need to know France's historical context, which is considerably distant from the present.

As a result, it is clear that the distance between the information we seek and the location where it actually is may be arbitrarily wide. This problem often termed as the Long Term Dependency Problem is caused by the fact that RNNs are unable to connect to the information as the gap widens. If we look at Figure 3.2 mathematically, the arrow indicates that any long-term information must pass sequentially through all of the intermediate cell states before arriving to the current processing cell. This indicates that it may be readily decreased by multiplying it by a large number of small-magnitude gradients. This leads to the **vanishing gradient** problem, which

is the main contributor to the long-term dependency issue. This issue is partially resolved by Long Short Term Memory Networks (LSTM), a modified form of recurrent neural networks.

3.1.2 Long Short Term Memory (LSTM)

Long Short Term Memory networks [25] often abbreviated as "LSTMs" are a special kind of Recurrent Neural Networks, which are architecturally capable of learning long-term dependencies. Introduced by *Hochreiter* and *Schmidhuber* in 1997, LSTMs have been in popularity especially in the last decade due to their effectiveness in machine translation tasks.

LSTMs were explicitly designed to solve the problem of long term dependency. Similar to RNNs, LSTMs also have a chain like structure but the module which gets repeated has a different structure than that of RNNs. Instead of a single tanh layer, there are four neural networks which interact with each other in a special way. Considering the Diagram 3.3, we can see a horizontal line which runs along the entire chain. This is termed as the ***cell state*** and is the key to a LSTM. It is very easy for information to just flow along it unchanged since it has very few linear interactions. This actually allows a LSTM to avoid the problem of ***vanishing gradient*** by short-circuiting the path of information flow.

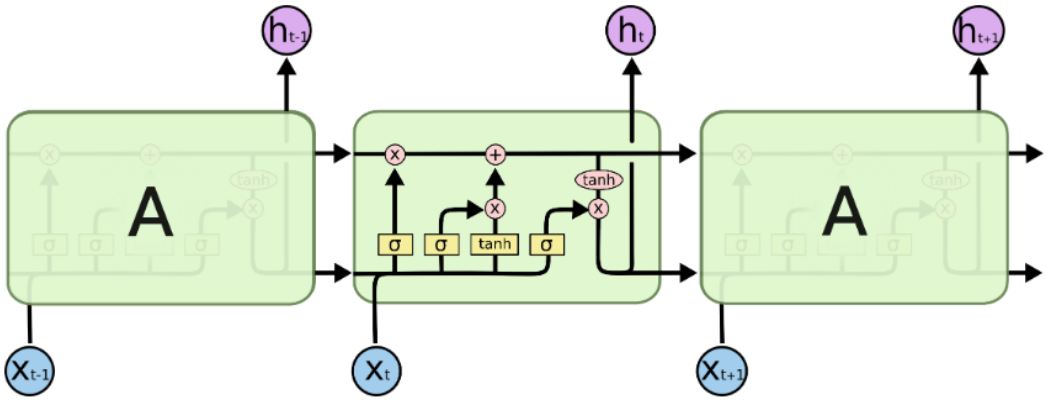

Figure 3.3: Long Short Term Network[8]

LSTMs can add or delete information from the cell state by utilising certain regulatory structures known as **gates**. They are built via a point-wise multiplication technique and a sigmoid neural network. These gates allow information to move across LSTMs on a whim. To regulate the state of the cell, an LSTM contains three different sorts of these gates.

The three types of regulatory gates in a LSTM cell are the forget gate, the input gate and the output gate respectively.

- A LSTM cell's forget gate determines which information from the cell state should be kept and which should be deleted. The sigmoid function receives input from the hidden state h_{t-1} and the current state X_t, producing a value between 0 and 1. By producing a number that is near to 0, the sigmoid function determines whether the old output can be eliminated or should be retained (by generating a value close to 1). The cell will eventually utilise this $f(t)$ value for point-by-point multiplication.
- By changing the cell state, the input gate influences the values we will store in the cell. The sigmoid layer must first be fed the current state X_t and the prior hidden state h_{t-1} in order to determine which values we will update. The tanh layer then creates a vector of fresh candidate values $\tilde{C}_t$ to control the network. The cell state is updated by combining the two outputs at the end.
- The output gate ultimately determines the value of the next hidden state and what we are going to output. To choose which portions of the cell state we will output, we initially pass the current state and prior hidden state through the third sigmoid function. The cell state is then multiplied point-by-point with the output of the sigmoid gate after being sent via a tanh layer to scope the outputs between -1 and 1. This guarantees that we only output the sections we intended to.

Finally the new cell state and new hidden state are carried forward to the next timestamp. Although this mitigates the problem of long term dependency, experimentally it has been found that even LSTMs suffer from the same problem when the sequence length crosses a certain threshold. This is primarily because at the core, LSTM is still a sequential model. Readers are encouraged to refer this blog [8] by Christopher Olah which illustrates the internal workings and limitations of LSTM quite comprehensively.

3.1.3 Transformers

Proposed by *Vaswani et al.*[40] the transformer is a neural network architecture, which constitutes an encoder and decoder within itself. The architecture managed to produce state of the art (SOTA) results for many Machine Translation tasks just with the help of a concept called ***attention*** and some dense ***normalization layers***. In addition, the transformer architecture is faster to train and allows the scope for parallelization which was simply not possible with RNNs and LSTMs. These properties

actually brought down the computational cost and have been the base for most of the recent researches in the NLP space.

Now lets digress a bit into the working of a transformer. The base concept which a Transformer uses is ***self-attention***. The Transformer is the first transduction model to calculate representations of its input and output only via self-attention, without the use of convolution or sequence-aligned RNNs. Simply speaking self-attention can be imagined as a mechanism which allows inputs to interact with each other via some mathematical operation and decide whom to pay more attention to. Without getting into the mathematical intricacies, self-attention can be simply perceived as an embedding vector of one of the words in the input sequence. This embedding vector then undergoes calculations with itself as well as with other embeddings of different words from the same sequence and form interrelations among themselves. Since we entire process goes in parallel, the embedding vectors of the words are encoded with positional embeddings, which are calculated to represent the order of appearance of the words in the original sequence.

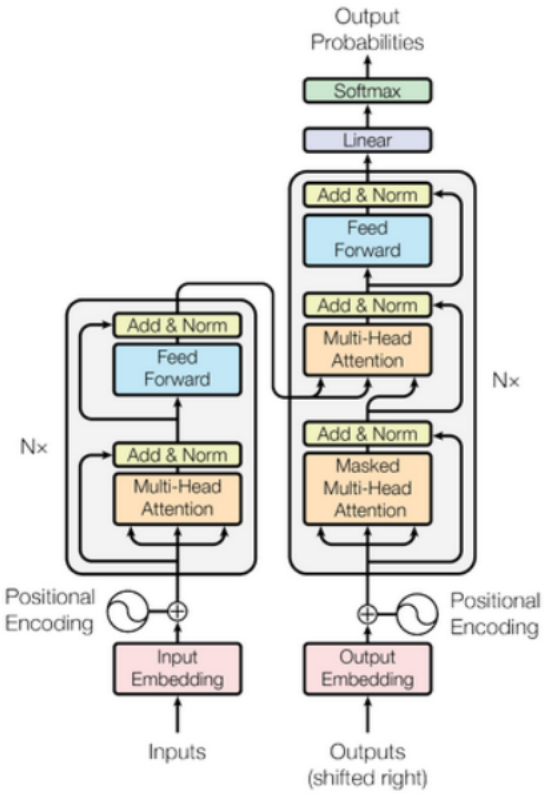

Figure 3.4: Transformer[40]

Architecture: In the Figure 3.4 the encoder is shown on the left side of the diagram,

and the decoder is shown on the right side. Both parts generate the required output from sequences that are input as embeddings. The Encoder encodes each word into a 512-dimensional embedding vector after taking a batch of sentences expressed as sequences of word tokens (numerical values supplied for input corpus of words to compute). As a result, the encoder requires [batch size, max input sentence length] as input, which results in [batch size, max input sentence length, 512] after encoding. On the other hand, the decoder uses the inputs from the encoder as well as the target phrase (which is likewise represented as a series of word tokens) to train the network. The final output from the encoder is given to the decoder at each of the 'N' levels since the top portion of the decoder is stacked 'N' times.

Up to this point, we have a brief understanding of the Transformer's internal connections and basic operation. However, before ending, there are a few points that are worth a quick introduction. The parts of the architecture that we can see but have not discussed are as follows:

- **Multi-head attention**: We can imagine multi-head attention as a multi-tasker from a surface-level perspective. You talk to yourself about the need for the transformer to learn the following word. For the same word, multi-head attention performs various parallel computations to produce various results. The best possible word is then produced by concatenating these findings with *SoftMax*. These simultaneous calculations may take into account the word's tense, the text's context, the word's type (verb or noun, etc.), and other factors. These are then combined, and the *SoftMax* determines the word with the highest probability.

- **Masked multi-head attention**: Similar to the previously mentioned multi-head attention, except that it conceals the following words in the sequence with respect to the word decoder that is currently at work. The transformer cannot realistically learn from the data because of this masking, which prohibits it from peering into the future.

- **Residual connection**: The directed arrows that connect one "Add and Norm" to another without travelling via the attention module are known as skip connections. This helps keep the network from degrading and keeps the gradient flow consistent throughout the network for effective training.

Transformers are used to construct the majority of cutting-edge models that are currently in use around the globe like converting talking speech into text, machine translation, text generation, paraphrase, question-answering, sentiment analysis, and many more. Readers are encouraged to refer this blog[6] by Jay Alammar which provides a very thorough explanation of the internal operations and the mathematics behind Transformers.

3.1.4 BERT

Devlin et al.[21] introduced BERT(Bidirectional Encoder Representations from Transformers) in 2018 and since then it has been the latest trend in the NLP space. In the realm of computer vision, transfer learning is crucial, as all are aware. For instance, a deep learning model that has previously been trained on the ImageNet dataset[37], may be fine-tuned for a new task and still deliver reasonable results with a small labelled dataset. Similar to this, language model pre-training has been effective in improving a number of tasks[20] involving natural language processing.

Pre-training and fine-tuning techniques are used in the BERT framework to create cutting-edge models for a range of applications. Some of these responsibilities include question-answering systems, sentiment analysis, and language inference. BERT makes use of a multi-layer bidirectional Transformer encoder. Its self-attention layer performs self-attention in both directions.

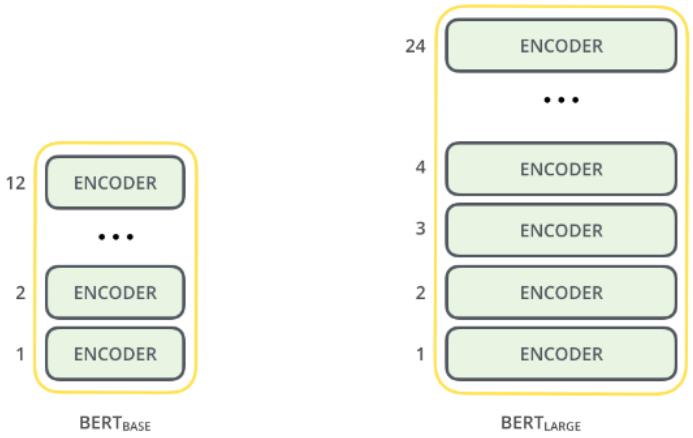

Figure 3.5: Bert base and large[5]

The authors have released two variants of the BERT model:

- **BERT Base:** Number of Transformers layers = 12, Total Parameters = 110M
- **BERT Large:** Number of Transformers layers = 24, Total Parameters = 340M

By pre-training on two tasks, **Next Sentence Prediction** and **Masked Language Model**, BERT employs bidirectionality.

- **Masked Language Modeling (MLM)** : In Masked Language Modeling, initially a part of the input tokens are masked randomly. Then the objective is

to predict the actual vocabulary index of the masked token solely based on its context. In contrast to left-to-right language model pre-training, the MLM goal allows the representation to merge the left and the right context, thus enabling the pre-training of a deep bidirectional Transformer.
In each sequence, the authors randomly masked 15% of the words. The task was to anticipate these hidden words. However the [MASK] token would never emerge during fine-tuning, therefore the masked words were not always substituted with the [MASK] tokens. The researchers therefore employed the following method:

- 80% of the time, the masked token [MASK] was used in place of the words.
- 10% of the time, random words were used in place of the terms.
- 10% of the time, the words remained unchanged.

.

- **Next Sentence Prediction**: Language models sometimes overlook how sentences that are followed by one another relate to one another. BERT was also pre-trained for this task. Sentence pairs serve as the training data for BERT's language model pre-training algorithm. Each pair's selection of sentences is pretty intriguing. Let's use an illustration to try to comprehend it.
Imagine that we want to use a dataset of 100,000 text phrases to pre-train a BERT language model. Therefore, there will be 50,000 training instances or sentence pairs in the training data. In half of the pairings, the second sentence would actually come after the first sentence. In the remaining 50% of the pairings, a randomly chosen sentence from the corpus would be the second sentence.

As demonstrated by architectures like BERT, Unsupervised pre-training and Unsupervised fine-tuning will play a significant role in many language comprehension systems. These deep bidirectional topologies can be quite beneficial for low resource activities in particular. Readers are again encouraged to refer this blog[6] by Jay Alammar which illustrates the details of BERT comprehensively with visualizations.

3.1.5 T5-Transformer

Let us consider the example of a BERT-style architecture that is enhanced for downstream tasks after being pre-trained on a Masked Language Model and Next Sentence Prediction target (like predicting the class label for a classification task or detecting the starting and ending of the answer in QA). Here, we individually fine-tune a number of instances of pre-trained models for a range of downstream tasks.

Contrarily, the text-to-text framework advises applying the same model, loss function, and hyperparameters to all NLP applications. In this method, the inputs are modelled such that the model will be able to identify a task, and the output is only a text representation of what is intended to happen.
Pre-training language models using sizable unlabelled datasets has been common by now. Among these is the **Common Crawl dataset**[2]. It is acquired via scraping web pages, skipping over any HTML markup. Each month, it generates about 20TB of data via scraping. However, a significant quantity of Common Crawl contains duplicate or garbage material, such as menus and error notices. Additionally, there is a considerable amount of pointless content related to our activities, such as foul language, placeholder text, or source codes. This led to the creation of a highly clean dataset of 750GB, that is comparatively larger than the majority of pre-training datasets existing.

Input and Output Representations : This is one of T5's top goals since it makes it possible to use the unified text-to-text technique. In order to have the same model for all downstream tasks, the model's initial input is augmented with a task-specific text prefix. This text prefix is also thought of as a hyperparameter.
The proposed model is basically an encoder-decoder transformer[40] with minor architectural changes. These changes include adding a Normalization layer before a sub layer and then appending the initial input to the sub-layer output: also termed as pre-norm. The model configuration is also comparable to the BERT base[21].

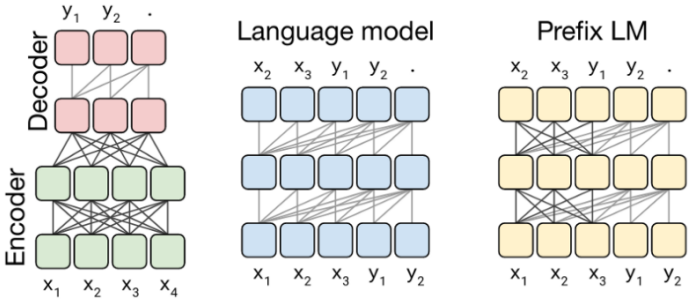

Figure 3.6: Model Structures from [36]

Architecture : There are numerous alternatives for choosing the training strategy at the architectural level: The paper[36] provides a thorough analysis of numerous contemporary methods for interpreting language. The comparison and exploration of numerous architectural requirements has hence resulted.

- **Encoder-Decoder (Left)** : The decoder is trained in a causal fashion inspired from GPT's, whereas the encoder is trained in a fully observable, BERT-style

manner where each token have some contribution towards the calculation of the attention values of every other token in the same sequence. The conventional seq2seq architecture, or the traditional encoder-decoder is used here.

- **Language Model (Middle)** : This mechanism for causal attention is essentially the same as the one that was previously discussed. It bases its methods on autoregressive modelling.
- **Prefix LM (Right)** : The BERT-style and language model techniques are combined in this. For instance, a BERT-style attention could be placed on the work of translating from English to German: "translate English to German: That is good. target:". The translation "Das ist gut." will subsequently be attended autoregressively after that.

The authors have investigated some of the practical approaches with regard to the pre-training goal as well:

- **Language Modeling** : This strategy focuses primarily on the causal prediction problem, or guessing the next word in a sentence while taking into account all the words that came before it.
- **Deshuffling** : The model is trained to predict the original text after all the words in a sentence are randomly shuffled.
- **Corrupting Spans** : Removing some words from the sentence, then teaching the model to anticipate those removed terms,also termed as a denoising objective.

The authors remarked that the denoising objective have shown the best results after experimentations.

T5 has surpassed benchmarks for translation and summarization as well as the state of the art in several GLUE[4] and SuperGLUE[10] tasks. T5 uses a fill-in-the-blanks task that it is accustomed to from pre-training to produce realistic text. This is why we also employed T5 for some text generation task in our work.

3.2 Overview of our Approach

Reiterating from Section 1.5, we are solving the problem of Answer Validation where we assume that we are given a quadruplet (Context, Question, Correct Answer, Candidate Answer) and we need to verify whether the potential response is contextually appropriate or not. Also since we are not interested in factoid questions we limit the scope of the problem to only those cases where the answer is a single sentence with atleast 10 words.

To solve such a problem we need to fix a model architecture that could solve our task and then we need to search for suitable datasets (if available) to train the model. Now to fix the architecture we take an even closer look at the problem. If we think intuitively we have a correct answer and we need to validate if the candidate answer is semantically correct or not . This situation closely resembles the problem of paraphrase detection. For readers who are not familiar with the term of paraphrase detection: It is nothing but the problem to identify if two sentences which are linguistically different but semantically convey the same meaning.

However, is the problem that we are solving just a mere example of paraphrase detection? At a first glance although it might seem affirmative but the answer is no. This is because there is an additional constraint of contextuality, i.e. the candidate answer should not only be correct but also has to be contextual with respect to the given context. So a natural question arises here, that if the correct answer and the candidate answer are paraphrases doesn't this imply that the candidate answer is also contextual? Unfortunately no because two sentences might be paraphrases but not contextual and vice versa.

Now the question arises that which model to choose, and how to train such a model which can identify contextual paraphrases. We tackle this additional constraint of contextuality via data itself. We propose fine-tuning a paraphrase detection model on data which contain both positive and negative samples for contextual paraphrases. This way we can transfer-learn the ability to detect such paraphrases and such a model can then be effectively used to determine if the candidate answer is contextually and semantically same as that of the correct answer. However although data for paraphrase detection is readily available and is a widely researched topic, dataset we are wanting is not available to the best of my knowledge. Hence we had no other option but to generate synthetic data.

We fine-tune the BERT base uncased Sequence Classifier[21] for paraphrase identification. 12 layers of Transformer encoder, 12 attention heads, 768 hidden size, and 110M parameters make up the BERT base uncased model. We fine-tune it on the

famous MRPC[22] dataset. Created by *Dolan et al.*, the Microsoft Research paraphrase Corpus (MRPC) is a dataset that includes 4583 pairs of sentences with labels that specify whether or not they are paraphrases. This serves as the foundational or baseline model for our work.

Now we need to fine-tune our baseline model with such data so that the ability to identify contextual paraphrases is learned by it. For this we use the widely known SQuAD[9] dataset. The SQuAD dataset contains tuples with a context, a question, a correct answer and the starting position of the answer in the context. We first sampled out only those answers which are atleast 10 words long. This reduced the dataset size from 86K to 6213. Now clearly the SQuAD dataset alone is not enough to achieve the objective we are aiming for as it contains tuples with correct answers only. We need both positive(paraphrase) and negative(distractor) samples for the SQuAD dataset. For obtaining positive samples we used the technique of *Back Translation*[15]. The process of converting a target text to some other language and back again to the source language is termed as Back Translation. We back translated the answers from each tuple of the SQuAD dataset to obtain contextual paraphrases or positive samples. For generating negative samples or distractors we used the BERT model as described in this paper[18]. This way we formed a dataset of contextual paraphrases and distractors. We fine-tuned our baseline model again on this augmented dataset.

Finally we want our model to be able to detect paraphrases which have high lexical overlap. For this we use the *PAWS*(Paraphrase Adversaries for Word Scrambling) Wiki dataset developed by *Zhang et al.*[43]. Below are two examples from the dataset:

Sentence 1	Sentence 2	Label
Kristoffer together with Karen had eight children.	Kristoffer had eight known children together with Karen.	1
Saunders defeated Dan Barrera at by unanimous decision.	By unanimous decision Dan Barrera defeated Saunders.	0

Table 3.1: Lexically highly overlapped examples

The example shows that PAWS Wiki contains positive and negative samples of paraphrases which are lexically highly overlapped. But again to add tackle the constraint of contextuality we need PAWS like paraphrases for the answers in SQuAD dataset. For this we employed the text-to-text T5-Transformer[36] for generating such paraphrases. For such text generation tasks, the text-to-text T5-Transformer is widely renowned for producing state of the art results. Hence we fine-tuned it first on

the PAWS English dataset and then used this fine-tuned T5-Transformer to generate PAWS like paraphrases for the SQuAD answers. Finally we appended these synthetic data with the original PAWS English dataset and used it to fine-tune the paraphrase detection model. The final results which we achieved were very promising.

3.3 Datasets

3.3.1 SQuAD

A span of text from the corresponding reading passage serves as the answer to each question in the Stanford Question Answering Dataset (SQuAD)[9], a reading comprehension dataset made up of questions prepared by crowdworkers on a collection of Wikipedia articles. The question itself may also be unanswerable.

In addition to the 100,000 questions from SQuAD1.1, SQuAD2.0 contains 50,000 more questions that cannot be answered but have been crafted by crowdworkers to resemble the answerable ones. Hence systems must be able to recognise when an answer is not supported by the paragraph and refrain from responding in order to do well on SQuAD2.0

The top 10,000 Wikipedia articles were sampled for SQuAD, with 536 selected by the creators. They extracted a total of 23,215 distinct paragraphs from each of these chosen articles (making sure to filter for paragraphs that were too small). They divided the dataset into training, development and testing sets, with 80% of the articles going into the training set, 10% into the development set, and 10% into the testing set.

Annotating SQuAD : Annotating SQuAD was the most crucial component that Mechanical Turk workers did. These workers were only chosen if they have a track record of producing excellent work (as measured by the HIT approval rate). The employees were tasked with coming up with and responding to 5 questions regarding the content of each chosen paragraph. They might put their question in a text box given and then highlight the responses in the paragraph. SQuAD's designers took extra care to ensure that the questions the employees came up with were written in their own terms, even turning off the copy-paste feature.

Categories for responses : The categories "date," "other numeric," "person," "place," "other entity," "common noun phrase," "adjective phrase," "verb phrase," "clause," and "other" were used to group each response. They discovered that nouns made up 32.6%, noun-phrases made up 31.8%,miscellaneous categories made up 15.8% of the responses and Dates and numbers made up the remaining 19.8% percent of the answers.

Reasons to choose SQuAD :

- **SQuAD is big** : Deep Read and MCTest, two more reading comprehension datasets, are both too small to support extensive and complicated models. Only

2,640 questions make up the MCTest, and only 600 questions make up Deep Read. These datasets are dominated by SQuAD, which has more than 100,000 questions.

- **SQuAD is challenging** : The solution to a particular question can be found in several documents in other document-based question-answering datasets that emphasise answer extraction. However, with SQuAD, the model can only access a single passage, making the task much more challenging because it is less forgiving if the response is incorrect.

- **SQuAD requires reasoning** : The Cloze dataset, which asks a model to anticipate a missing word in a text, is a common form of dataset. These datasets are sizable and present a task that is fairly similar to SQuAD. The main change that SQuAD introduces in this area is that SQuAD is better at assessing model understanding and capabilities since SQuAD's solutions are more complicated and demand more rigorous reasoning.

3.3.2 MRPC

The Microsoft Research Paraphrase Corpus(MRPC) is derived from a collection of 13,127,938 sentence pairs, taken from 9,516,684 sentences in 32,408 news clusters gathered from the World Wide Web in two years. *Dolan et al.*[22] explain the methods and hypotheses utilised to construct this initial data set. To create the first database, two algorithms based on shared lexical features and sentence position in the text were used. A training subset of 4,076 sentence pairs, of which 2,753 are paraphrases, are taken from the entire corpus of 5801 sentences, and a test subset is taken from the remaining sentences (1,725 pairs of which 1,147 are paraphrases).

We used the MRPC dataset to build our baseline paraphrase detection model.

3.3.3 PAWS

Sentence pairings with significant lexical overlap that are not paraphrases are scarce in the existing datasets for paraphrase identification. Models developed using such data are unable to distinguish between pairs of flights, such as those from Mumbai to Kolkata and Kolkata to Mumbai. Here, the 108,463 well-formed paraphrase and non-paraphrase pairs with significant lexical similarity from the *PAWS*(Paraphrase Adversaries from Word Scrambling) dataset[43] are discussed.

Controlled word swapping and reverse translation are employed to make difficult pairs, which are then evaluated by human validators for paraphrasing. The accuracy

of contemporary models trained on current datasets is only approximately 40% on PAWS, but when these models also incorporate PAWS training data, their accuracy rises to 83% while retaining accuracy on existing tasks. However, models that do not take contextual data that is not local into consideration suffer even with PAWS training data. As a result, PAWS provides a useful tool for developing models that utilise structure, context, and pairwise comparisons more efficiently.

This dataset serves as an example of the value of word order, context, and structural modelling for the task of paraphrase recognition. There are 656k noisy labelled pairings and 108,463 human-labeled pairs in total. The Quora Question Pairs (QQP) dataset[9] and Wikipedia, respectively, are the bases for two subsets of the dataset.

Below are two examples from the dataset:
The first is a paraphrase while the second pair is semantically different.

Sentence 1	Sentence 2	Label
Kristoffer together with Karen had eight children.	Kristoffer had eight known children together with Karen.	1
Saunders defeated Dan Barrera at by unanimous decision.	By unanimous decision Dan Barrera defeated Saunders.	0

Table 3.2: Lexically highly overlapped examples from PAWS

PAWS-Wiki :

- **PAWS-Wiki Labeled (Final)** : Comprised of pairings produced using back translation and word swapping techniques. All pairs are divided into Train/Dev/Test parts and have received human evaluations for both fluency and paraphrase.
- **PAWS-Wiki Labeled (Swap-only)** : Including pairs that are not part of the first set because they lack equivalents in the back translation. They can be used as an additional training set because they are high-quality pairings with human judgments on both fluency and paraphrase.
- **PAWS-Wiki Unlabeled (Final)** : The pairs in this collection can also be utilised as an additional training set because they have noisy labels without human evaluations. Both word swapping and back translation techniques are used to produce them.

Each file in the dataset has four columns and is in the tsv format:.

Column Name	Data
id	A unique id for each pair
sentence1	The first sentence
sentence2	The second sentence
(noisy_)label	(Noisy) label for each pair

Table 3.3: Table to test captions and labels.

There are two possible values for each label: 0 denotes that the pair is different semantically, and 1 denotes that it is a paraphrase. Details of the PAWS-Wiki dataset are shown below:

Data	Train	Dev	Test	Yes%
Labeled (Final)	49,401	8,000	8,000	44.2%
Labeled (Swap-only)	30,397	—	—	9.6%
Unlabeled (Final)	645,652	10,000	—	50.0%

Table 3.4: Proportion of paraphrase (Yes%) pairs in PAWS-Wiki

PAWS-QQP The pairs in this corpus were built using the Quora Question Pairs corpus. We are unable to directly offer the original PAWS-QQP data due to the QQP licensing. However one can recreate the data by running some scripts on the raw file (which can be downloaded) and then attaching the corresponding labels.

Data	Train	Dev and Test	Yes%
PAWS-QQP	11,988	677	31.3%

Table 3.5: Proportion of paraphrase (Yes%) pairs in PAWS-QQP

PAWS-X The PAWS examples are translated into six typologically different languages in this corpus: French, Spanish, German, Chinese, Japanese, and Korean.

We used the PAWS-Wiki(labelled version) dataset for our work which can be easily downloaded using the Tensorflow Dataset API[11].

3.4 Dataset Pre-Processing

We used three datasets in our work namely the MRPC[22] dataset, SQuAD1.1[9] dataset and the PAWS-Wiki [43] dataset. The MRPC and PAWS are typical paraphrase detection datasets while SQuAD1.1 is a Question Answering dataset. We mainly needed to perform some data pre-processing step on the SQuAD dataset only.

We are solving the problem of answer validation where we constrained ourselves to only non-factoid questions with answers comprising of a single sentence with atleast 10 words. Now we need to make sure that these constraints are satisfied by the data on which we are fine-tuning the model. The examples of MRPC and PAWS followed these constraints but the answers of the SQuAD dataset didn't. Following is the distribution of the answer length in SQuAD1.1 dataset :

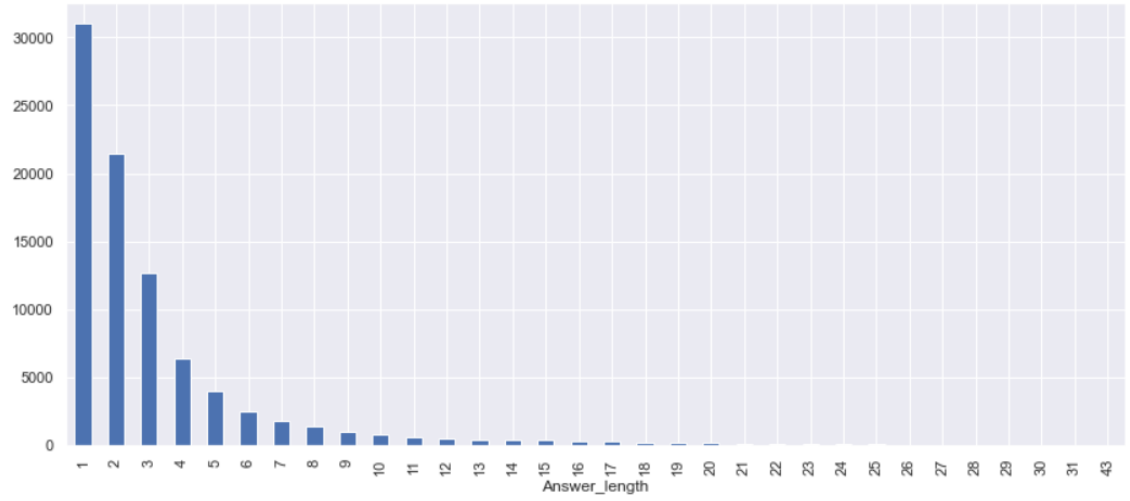

Figure 3.7: Answer Length Frequency of SQuAD1.1 Dataset

So we sampled out only those data points for which the answer length was atleast 10 words. The sampled data from SQuAD1.1 had 6217 data points.

3.5 Methodology/Approach

3.5.1 Baseline Model Fine-Tuning

For the baseline model we fine-tune the BERT base uncased Sequence Classifier[21] for paraphrase identification. We fine-tune it on the famous MRPC dataset. Created by *Dolan et al*[22], the Microsoft Research paraphraseCorpus (MRPC) is a dataset containing 4583 pair of sentences along with labels dicating whether the pair is a paraphrase or not.

Model Architecture : We use the the Bert base uncased model for paraphrase identification task. Architecturally the numbers for Bert base uncased are as follows:- Number of Layers L=12, Size of the hidden layer, H=768, and Self-attention heads, A=12 with Total Parameters=110M.

Fine-Tuning inputs :

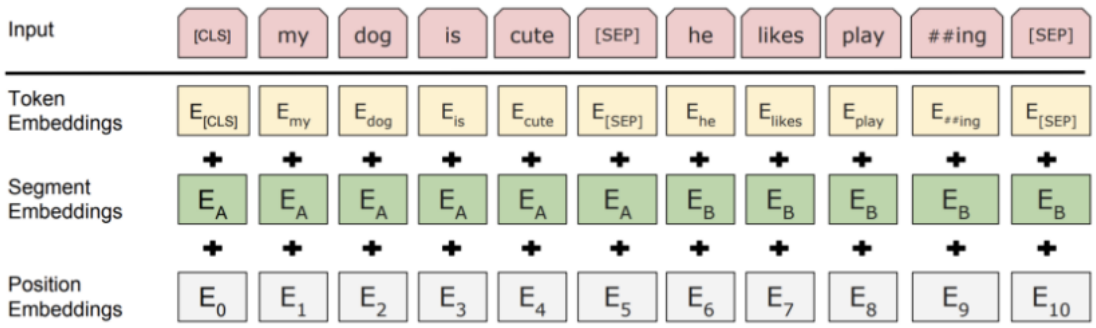

Figure 3.8: Training Input Format [21]

Using the shown format, we provide input to BERT. Two unique tokens [CLS] and [SEP] along with a pair of sentences collectively referred to as sequences make up the input.

Therefore, in this example, for the two lines "my dog is cute" and "he likes playing" BERT first utilises wordpiece tokenization to break the sequence into tokens, adding the [CLS] token at the beginning and the [SEP] token at the end and begining of the second sentence.

Therefore, in the example-pair, "my dog is cute" and "he enjoys playing", BERT uses wordpiece tokenization to first divide the sequence into tokens, adding the [CLS] token at the start and the [SEP] token at the end and beginning of the second sentence.

Words like playing must necessarily be broken down into "play" and "ing" due to the

wordpiece tokenization utilised in BERT. It does it in two ways:

- It helps to limit the size of our vocabulary by eliminating terms like solving, solves, solver, etc.
- It aids us in learning unfamiliar words. For instance, even though plays are absent from the language, play and s may still have embedded meanings.

Token Embeddings : The Token embeddings are then obtained by indexing a 30000x768 Matrix (H). After wordpiece tokenization, the vocabulary length in this case is 30000. This matrix's weights would be learned throughout training.

Fine Tuning : Finally we fine-tune the BERT on the MRPC dataset for the sentence pair classification task by adding a $Linear + Softmax$ layer atop the 768 sized [CLS] token.
Thus at the end of this stage we are in possession of our baseline model fine-tuned on the MRPC dataset and capable of identifying paraphrases. However this model will need further fine-tuning on contextual paraphrases generated from SQuAD dataset.

Evaluation Method : Accuracy is the evaluation method used for performance assessment. It is calculated by the formula :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where TN stands for True Negative, FN for False Negative, TP for True Positive and FP for False Positive.

3.5.2 Positive and Negative Sample Generation for SQuAD Dataset

Positive Sample Generation : We first need to be clear what does positive sample mean in this context. So now we have our baseline model and the sampled SQuAD dataset with us. We are using SQuAD1.1 solely to make our baseline model learn to identify contextual paraphrases. However SQuAD is a Question Answering dataset and the model we are fine-tuning is a paraphrase detection model. So we need to convert the SQuAD dataset into a paraphrase dataset. For this we need to generate paraphrases (positive samples) for the answers in the sampled SQuAD dataset. Similarly we need to generate negative samples as well so that while fine-tuning we do not introduce any bias.

For the generation of positive samples we used the technique of *Back Translation*[15].

Back translation, also known as reverse translation, is the process of literally translating text back into the source language from the target language. To achieve this we used ***MarianMT***[27] Machine Translation Model via **HuggingFace** Model API to translate the english answers of SQuAD1.1 to Russian and then again back to English. This way we achieved high quality paraphrases (positive samples) for the answers. However some back translated answers were exactly equal to the original answers and hence we found it best to remove them. This reduced the size of sampled SQuAD from 6217 to 5239. To really check if the positive samples generated were semantically really the same, we took the help of external validator, details of which are added in the Experiments section

Negative Sample Generation :
We used a BERT based model as described in [19] to generate negative samples also termed as **distractors**.

Model : Tasks involving language modelling at the sequence-level classification and token-level prediction levels can both be completed using BERT. For these jobs, there are three distinct tokens: [C], [S], and [M]. The token embedding [C] is meant to be used as the representation of the aggregated sequence for classification tasks. Given that the input token sequence may consist of a pack of many sentences, the [S] is made to discriminate between the various sentences in a sequence of tokens (to convey or indicate data from several sentences). The [M] token, on the other hand, is intended for use in prediction at token-level (for example, using the context to predict the masked token or predicting the probability of the starting or ending position for span-based tasks like QA tests).

BERT is essentially a language modelling auto-encoder that attempts to recover the original data from distorted inputs. BERT frequently yields illogical and rambling results when asked to anticipate a series of related masked tokens. For instance, the identical prediction outcome for the tokens is frequently seen when using BERT to forecast three successive [M] masked tokens. The resulting sentences are incoherent since the context (the data used to forecast the tokens) for the masked tokens is almost identical but for the position embedding. As a result, while decoding the subsequent distractor token, we take into account the previous decoding outcomes.
Three inputs are provided in a situation where a distractor is generated: a paragraph P, a question Q and a response A. Let us take C as the Context Sequence which refers to the token order produced by joining P, Q, and A. Distractor tokens are produced by the BDG (Bert Distractor Generation) model in an auto-regressive manner. In particular, the BDG model makes token-by-token predictions based on (1) the context sequence C that has been provided and (2) the previously anticipated distractor tokens. To produce a distractor, the BDG model goes through several rounds. Note that the model predicts a token based on C and the previously generated tokens at

each iteration. When [S] is anticipated, the generation comes to an end.

Input : The input sequence X_i to BERT is specifically X_i =([C], C, [S], d_1,..., d_i, [M]) at iteration i. Let's say that $h_{[M]} \in R^h$ stands for the hidden representation of [M] of X_i that the BERT transformer stacks have returned. The following linear layer transformation $W_{DG} \in R^{h \times V}$ and softmax activation to all vocabulary dimensions provide the prediction of d_i.

$$p(w|X_i) = softmax(h_{[M]} \times W_{DG} + b_{DG}) \tag{3.2}$$

$$d_{i+1} = argmax_w Pr(w|X_i) \tag{3.3}$$

The freshly created token d_i is then attached to the new X_{i+1}, and the distractor creation procedure is continued using the new X_{i+1}. The procedure comes to an end when [S] is predicted.

Loss Function : The loss function used is shown below:

$$\underset{\theta}{minimize} \sum_{(C,D)} \sum_{i=0}^{|D|} (log_2 p(d_{i+1}|C, d_{1:i}; \theta)) \tag{3.4}$$

Using this model we generated distractors or negative samples for the answers of our sampled SQuAD dataset. First of all some of the distractors generated were exactly same as the original answers and we removed them. This downsized the dataset from 6217 to 5839 data points. Secondly to really check if the distractors were really authentic and conveyed negative meaning we took the help of external validator, details of which are added in the Experiments section. Next we clubbed the positive and negative samples generated and fine tuned the baseline model on this. However before fine-tuning we observed the performance of the baseline model on this synthetic data and again observed its performance after fine tuning. Results observed where affirmative and showed that our model improved.

3.5.3 Generation of PAWS-like positive samples for SQuAD and fine-tuning

Experimentations were needed to see how our model performs when subjected to lexically highly overlapped sentences like given in Table 3.1. We ran inference over the PAWS dataset and found that the model performance is utterly bad. The accuracy merely hit 0.50

Fine-tuning was the only option to have the model learn to classify such lexically overlapped sentences. However we needed to make sure that the model is also fine-tuned on contextual PAWS like examples. We decided to follow the same path as

before and generate some PAWS like samples for SQuAD. For this we employed the T5 transformer discussed in detail in Section 3.1.5 as it produces state of the art(SOTA) results on text generation tasks. We used a T5-Transformer which was already fine-tuned on PAWS dataset from **HuggingFace** to generate paraphrases for the sampled SQuAD1.1 dataset. We validated a subset of these synthetic paraphrases by an external validator, details of which are added in the Experiments section. Again some of the paraphrases generated were exactly same as the original answer text and hence they were dropped from consideration. Overall the generated dataset equalled 5189 data points and we clubbed it with the PAWS dataset. Finally we fine-tuned our model with the dataset we formed. Results observed were very positive and the accuracy hit 0.83 after fine-tuning.

Chapter 4

Experiments and Results

4.1 Baseline Model

The baseline model is a BERT base uncased model fine-tuned on the MRPC dataset. The MRPC dataset is splitted into train,test and val sets with the train set having 3668 data points, test set having 1725 data points and the validation set having 408 data points. Following is the summary of the training and validation loss for the baseline model: So we have our baseline model giving an accuracy of 0.86 on

Epoch	Step	Training Loss	Validation loss	Accuracy
1	230	0.5981	0.4548	0.7798
2	460	0.3739	0.3806	0.8345
3	690	0.1991	0.2345	0.8529
4	920	0.1286	0.2023	0.8529
5	**1150**	**0.0812**	**0.2187**	**0.8601**

Table 4.1: Baseline Model fine-tuning summary

the MRPC dataset. We also experimented with the hyperparameters and after fine tuning the following values gave us the best results :

- learning rate: 3e-05
- training batch size: 16
- eval batch size: 8
- lr scheduler type: linear

- num of epochs: 5
- optimizer: Adam with betas=(0.9,0.999) and epsilon=2e-08
- seed: 47

4.2 Generation of positive and negative samples for SQuAD and fine-tuning

At this stage we have a baseline model giving an accuracy of 0.86 on the MRPC dataset. We generated both positive samples or paraphrases and negative samples or distractors for the SQuAD dataset. However for both the positive and negative samples some were exactly same as the original answers in SQuAD and hence were dropped. Overall there were 5239 positive samples and 5839 negative samples which were generated and used for further experimentations. However to validate the authenticity of the samples generated we first have a subset of them validated by an external validator.

The external validator is a **Computer Science Engineering** graduate with **proficiency** in English. However this was a voluntary support on part of the validator and there were no financial contract. The strategy that we use is simple we first sample out a subset of 500 positive and 500 negative samples. We tried to maintain the distribution of answer length in these subsets. Among each subset of 500 samples we again divided it into 3 sets I, II, III of 200, 200, 100 samples each. Set I of 200 samples was validated by us while the Set II of 200 samples was validated by the external validator. Finally the remaining set of 100 was validated by both of us individually. Note here validation means that we just marked whether the corresponding sample was really a positive or negative sample

For the positive samples, the following was the summary of the validation experimentation: So in set III for positive samples which was validated by both of us, we

Set Type	Validator	No.of samples	Generated Correct	Generated Wrong	Doubtful
I	External	200	197	2	1
II	Us	200	198	2	0
III	Both	100	97(By both)	3(By both)	0

Table 4.2: Validation summary for positive samples

were in consensus that 97 of them generated correctly while 3 generated were not correct. We also made an observation that all the samples which were incorrectly generated were sentences which had more than 27 words. This hinted us that the quality of back translation might not have been good for very long sentences.

For the negative samples, the following was the summary of the validation experimentation: So in set III for negative samples which was validated by both of us, we

Set Type	Validator	No.of samples	Generated Correct	Generated Wrong	Doubtful
I	External	200	193	6	1
II	Us	200	194	5	1
III	Both	100	93(By both)	7(By both)	0

Table 4.3: Validation summary for negative samples

were in consensus that 93 of them were generated correctly while 7 generated were not correct. We also made a similar observation like before that all the samples which were incorrectly generated were sentences which had more than 25 words. This hinted us that the quality of distractor generation might not have been good for very long sentences.

The validation exercises instilled confidence about the generation of the synthetic samples. However since nearly 4% of the negative samples were generated wrong in the subset of 500 validated, so we randomly removed 4% of the negative samples with all the samples removed were atleast 25 words long. With the remaining generated samples we went ahead for fine-tuning the model. We now had a total of 5239 positive and 5606 negative samples totalling to 10,845 samples which was splitted into train, test and eval sets in the ratio 70:20:10. In all there were 7591 samples in the train set, 2169 samples in the test set and remaining 1085 samples in eval set. Focus was given to maintain class balance in all the sets. Before fine-tuning our baseline model gave an accuracy of 0.8219 on the test set of the generated samples.

Following is the summary of the training and validation loss for the fine-tuning of the model:

Epoch	Training Loss	Validation loss	Accuracy
1	0.4971	0.3548	0.8306
2	0.3739	0.3006	0.8415
3	0.2345	0.2465	0.8528
4	0.1261	0.2023	0.8567
5	**0.1012**	**0.2063**	**0.8678**
6	0.0916	0.2879	0.8688

Table 4.4: Model fine-tuning on Generated samples summary

Thus after fine-tuning the model accuracy on the Generated samples improved from **0.82** to **0.86**. The hyperparameter values which we listed above in Section 4.1 gave us the best results again.

4.3 Fine-tuning on PAWS and generation of PAWS-like samples

Before we began fine-tuning we first evaluated our current model on the PAWS dataset. The accuracy figures merely hit **0.50**. So we had to fine-tune the model on the PAWS datset. However to meet the constraint of contextuality we also generated some PAWS-like paraphrases for the SQuAD dataset using the T5-Transformer(fine-tuned on PAWS) for paraphrase generation. But just like we have seen before, some of the samples generated were exactly same as the answers in SQuAD and hence were dropped reducing the number of generated samples to 5189.

Just like before we followed a validation exercise combining us and an external validator in this case as well. Following was the summary of the validation experimentation :

Set Type	Validator	No.of samples	Generated Correct	Generated Wrong	Doubtful
I	External	200	190	10	0
II	Us	200	192	7	1
III	Both	100	95(By both)	5(By both)	0

Table 4.5: Validation summary for PAWS-like generated samples

Again we had a similar observation like before that all the samples which were incorrectly generated were sentences which had more than 22 words. This hinted us that the quality of sample generation might not have been good for very long sentences. Since nearly 4% of the samples were generated wrong in the subset of 500 validated, so we randomly removed 4% of the generated samples with all the samples removed having atleast 22 words. With the remaining generated samples of 4982 data points we went ahead for fine-tuning the model.

Clubbing the generated samples with PAWS english dataset we had 52,886 training samples, 8997 test samples and 8499 validation samples. Following is the summary of the training and validation loss for the fine-tuning of the model: The hyperpa-

Epoch	Training Loss	Validation loss	Accuracy
1	0.7249	0.6978	0.5096
2	0.5045	0.5007	0.6435
3	0.4099	0.3878	0.7028
4	0.2967	0.2866	0.7576
5	0.2345	0.2142	0.8016
6	0.2045	0.1945	0.8233
7	**0.1675**	**0.1610**	**0.8399**

Table 4.6: Model fine-tuning on PAWS summary

rameters which are mentioned in Section 4.1 gave the best results for this fine-tuning experiment as well. Finally as evident from Table 4.6, we were able to improve the accuracy of our model from **0.50** to **0.83** which is a great leap in terms of accuracy. This is the final model that we settle with.

Chapter 5

Conclusion and Future Work

In our work we attempted to solve the problem of contextual answer validation where we scoped out that the answer should be atleast 10 words long. We started with a baseline model inspired from paraphrase detection which achieved state of the art accuracy of **0.86**. However due to the unavailability of exact datasets for this task, we generated synthetic data based on the SQuAD dataset using techniques like back translation etc. Finally we even generated PAWS like synthetic data for SQuAD and clubbed it with the PAWS dataset to fine tune our model. This greatly improved our model's accuracy from **0.50** to **0.83** for PAWS dataset which is a great leap. The final answer validation model thus has the ability to validate lexically highly overlapped contextual answers as correct or not.

In our work we primarily focussed on validating answers which are single sentence long and have atleast 10 words. However in a typical real world setting, this doesn't hold necessarily and often we find answers to span multiple sentences. Hence there is a tremendous scope of work where one can try the answer validation problem for long answers. To aid to the fortune, there are also readily available contextual Question Answering datasets which cater to long answers like **Eli5** (Explain me like I am 5) dataset[3] from Facebook, **Natural Questions** dataset[30] from Google, etc.

Bibliography

[1] Andrej karpathy's blog on effectiveness of rnns. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, accessed: 2022-06-29

[2] Common crawl dataset. https://commoncrawl.org/the-data/, accessed: 2022-06-29

[3] Eli5 dataset from facebook. https://facebookresearch.github.io/ELI5/, accessed: 2022-06-30

[4] Glue tasks. https://gluebenchmark.com/tasks, accessed: 2022-06-29

[5] Illustrated bert. https://jalammar.github.io/illustrated-bert/, accessed: 2022-06-29

[6] Illustrated transformer. https://jalammar.github.io/illustrated-transformer/, accessed: 2022-06-29

[7] MS Windows NT kernel description. https://www.ibef.org/blogs/india-to-become-the-edtech-capital-of-the-world, accessed: 2022-06-30

[8] Olah's blog on rnn and lstm. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed: 2022-06-29

[9] Squad competition stanford. https://nlp.stanford.edu/blog/cs224n-competition-on-the-stanford-question-answering-dataset-with-codalab/, accessed: 2022-06-29

[10] Superglue tasks. https://super.gluebenchmark.com/tasks, accessed: 2022-06-29

[11] Tensorflow dataset hyperlink. https://www.tensorflow.org/datasets, accessed: 2022-06-29

[12] Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., Gupta, S.: Better fine-tuning by reducing representational collapse. arXiv preprint arXiv:2008.03156 (2020)

[13] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: Vqa: Visual question answering. In: Proceedings of the IEEE international conference on computer vision. pp. 2425–2433 (2015)

[14] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. journal of machine learning research, vol. 3, no (2003)

[15] Brislin, R.W.: Back-translation for cross-cultural research. Journal of cross-cultural psychology 1(3), 185–216 (1970)

[16] Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Transactions of the association for computational linguistics 4, 357–370 (2016)

[17] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311 (2022)

[18] Chung, H.L., Chan, Y.H., Fan, Y.C.: A BERT-based distractor generation scheme with multi-tasking and negative answer training strategies. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings. pp. 4390–4400. Association for Computational Linguistics, Online (Nov 2020), https://www.aclweb.org/anthology/2020.findings-emnlp.393

[19] Chung, H.L., Chan, Y.H., Fan, Y.C.: A bert-based distractor generation scheme with multi-tasking and negative answer training strategies. arXiv preprint arXiv:2010.05384 (2020)

[20] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)

[21] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[22] Dolan, B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Third International Workshop on Paraphrasing (IWP2005) (2005)

[23] Dos Santos, C., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: International Conference on Machine Learning. pp. 1818–1826. PMLR (2014)

[24] Firth, J.: A synopsis of linguistic theory 1930-1955 in studies in linguistic analysis, philological society. Links (1957)

[25] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)

[26] İrsoy, O., Benton, A., Stratos, K.: Corrected cbow performs as well as skip-gram. arXiv preprint arXiv:2012.15332 (2020)

[27] Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A.F., Bogoychev, N., et al.: Marian: Fast neural machine translation in c++. arXiv preprint arXiv:1804.00344 (2018)

[28] Khurana, D., Koli, A., Khatter, K., Singh, S.: Natural language processing: State of the art, current trends and challenges. arXiv preprint arXiv:1708.05148 (2017)

[29] Kuo, F.Y., Sloan, I.H.: Lifting the curse of dimensionality. Notices of the AMS 52(11), 1320–1328 (2005)

[30] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al.: Natural questions: a benchmark for question answering research. Transactions of the Association for Computational Linguistics 7, 453–466 (2019)

[31] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)

[32] Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)

[33] Manning, C.D.: Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In: International conference on intelligent text processing and computational linguistics. pp. 171–189. Springer (2011)

[34] Peñas, A., Rodrigo, A., Sama, V., Verdejo, F.: Testing the reasoning for question answering validation. Journal of Logic and Computation 18(3), 459–474 (2008)

[35] Peters Matthew, E., Mark, N., Mohit, I., Matt, G., Christopher, C., Kenton, L.: Deep contextualized word representations.(2018) (1802)

[36] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. 21(140), 1–67 (2020)

[37] Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th international conference on Machine learning. pp. 759–766 (2007)

[38] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985)

[39] Tonoike, M., Utsuro, T., Sato, S.: Answer validation by keyword association. In: Proceedings of the 3rd workshop on RObust Methods in Analysis of Natural Language Data (ROMAND 2004). pp. 95–103 (2004)

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems 30 (2017)

[41] Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. ieee Computational intelligenCe magazine 13(3), 55–75 (2018)

[42] Zhang, Y., Wallace, B.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820 (2015)

[43] Zhang, Y., Baldridge, J., He, L.: Paws: Paraphrase adversaries from word scrambling. arXiv preprint arXiv:1904.01130 (2019)