University College London

DEPARTMENT OF COMPUTER SCIENCE

Question Answering with Paraphrasing as Regularisation

Liana Mikaelyan[1]

MSc Machine Learning

Supervisors: Sebastian Riedel, Patrick Lewis

Submission date: 6 September 2019

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Machine Learning degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

In this work, the hypothesis on whether paraphrasing improves state-of-the-art question answering systems in terms of accuracy, robustness to question paraphrases and robustness to adversarial input is investigated. We first conduct a statistical analysis on the performance of BERT pre-trained language models when paraphrased questions and documents with distracting information are introduced. We show that BERT is oversensitive to such examples and explore potential methods to overcome this issue.

To overcome the drop in performance, we propose to fine-tune BERT incorporating data augmentation and multitask learning: given a question and a context, a set of paraphrases is generated using back-translation; in addition to minimizing the loss between the predicted answer distributions of the original questions, we minimize the supervised loss of the augmented questions and introduce the auxiliary objective of minimization of unsupervised loss between the answer distributions of original and augmented questions. The auxiliary loss is unsupervised because we do not directly use the labels corresponding to the augmented examples. This unsupervised loss is computed as symmetric Kullback-Leibler divergence and the Jenson-Shannin distance, serving as a regularization method for the model.

We trained and evaluated the proposed model on SQuAD 1.1 dataset comparing in to the fully supervised model where augmented examples are simply concatenated to the input and to a semi-supervised model that is similar to the proposed model but without the second supervised loss between the predictions corresponding to the augmented examples and labels.

We observed that the supervised model outperformed the original model and our proposed model. However, our proposed model showed almost 2% improvement in EM and F1 scores over the original model when evaluated on a paraphrased development set that we constructed.

Acknowledgments

I would like to express my gratitude to my supervisors Patrick Lewis and Sebastian Riedel for their continuous support and valuable advice and for providing me with the opportunity to conduct this project. I would also like to acknowledge the Machine Reading group and University College London for making it possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The idea of creating intelligent systems capable of performing tasks at a human level has fascinated people for decades. The definitions of intelligence as well as the standards for systems to be considered intelligent have been evolving over time. It was once thought that chess playing expertise was equivalent to human intelligence due to the complexity of this mentally challenging task [NSS58]. However, it was soon discovered that chess playing can be achieved by a simple algorithm and a system that mastered this skill does not necessarily possess general intelligence. That is, it can show high performance on one specific task or a set of related tasks but does not generalize to other domains or performs poorly when the tasks are slightly changed.

Some tasks that have been used to measure human intelligence such as numerical computation and memory tasks are now trivial for computers and are not used as a defining feature of machine intelligence. At the same time, other tasks that are easy and intuitive for humans such as understanding and responding in nature language, turned out to be very difficult for machines to accomplish. Despite the belief of many experts of the last century, it turned out that manipulating symbolic representations is not enough for many real-word tasks that humans encounter on a daily basis, leading to the development of more learning algorithms that are biologically inspired [Has+17].

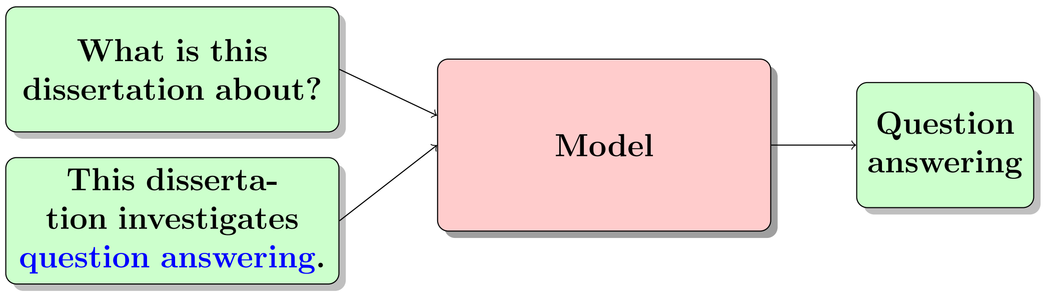

What is this dissertation about?
This dissertation investigates question answering.
Model
Question answering

Figure 1.1: Question Answering

Natural Language Processing (NLP) is concerned with the interpretation and representation of human language by an intelligent system. Due to the complex and ambiguous characteristics of natural language, NLP is often considered as one of the most crucial aspects of artificial intelligence. This is because a system acquiring this skill on a human level is likely to be able to perform any other tasks that human intelligence can tackle [Bos14]. While this is hard to prove until such a system is created, we can certainly argue that understating natural language is a prerequisite to many other prominent tasks such as language generation. This includes, for example, question generation, another extensively studied challenge in the NLP community [Pan+19].

In this project, we focus on question answering (QA), an aspect of NLP which aims to develop machine comprehension systems capable of extracting answers to questions from given texts. Given a document and a question about the document, the task is to determine a subsequence of tokens from the document that answer the question. This is illustrated in Figure 1.1. From a research point of view, this objective provides a method to evaluate how well a machine reading system can comprehend textual information.

Recently, there has been a large increase in QA research leading to the development of models with impressive performance [Che+17; Seo+17; Dev+18; Yu+18]. However, now that we have state-of-the-art machine reading systems that show human performance on some QA datasets, a natural question to ask seems to be: are these systems truly intelligent? Can we tell that they understand the meaning of the questions without relying on superficial cues such as word matching and are able to perform multi-sentence reasoning and inference when questions and documents are worded differently or contain distracting information?

While some successful approaches perform well on specific inputs, they lack robustness in certain areas. In particular, these systems are often sensitive to the phrasing of the questions and may fail to predict the correct answer when a paraphrased version of a question is introduced [GN19]. Moreover, current QA systems are easily deceived if a token of a given question is replaced by a different token or when distracting information is introduced to the passage [JL17]. Indeed, [Sug+18] argue that the ability of reading comprehension systems may be overestimated and it is important to analyze the input documents and questions separating those that can be tackled with simple patterns from those that require sophisticated reasoning and inference.

1.2 Objectives

The primary aim of this project is to propose a method that uses question paraphrases as regularization for QA systems to make them more robust to various question wordings and adversarial examples. In this project, we perform the following steps:

1. Conduct statistical analysis on the robustness of a state-of-the-art QA system in terms of how the distributions over the answers change when paraphrased questions are introduced.
2. Attempt to tackle the observed decreased performance by proposing a regularization method based on multitask learning and data augmentation.
3. Investigate whether the proposed model is able to further reduce the amount of labelled examples while achieving high performance.

In a high level, our proposed multitask augmented model is structured as follows:

1. **Data Augmentation:** Given a question and a context, generate a set of question paraphrases.
2. **Prediction Generation:** For each original and paraphrased question, extract an answer from the text.
3. **Regularization of the Loss:** For each set of answer distributions, minimize a weighted sum of the original supervised loss and the unsupervised loss (defined in terms of some distance metric) between the distributions of the obtained predictions.

1.3 Source Code

The code for this project is publicly available at https://github.com/LianaMikael/MSc-Dissertation-Question-Answering
All experiments presented here can be reproduced in at most 1 hour on a single Google Cloud TPU.

1.4 Key Results

- We show that while BERT Large surpasses human performance on the SQuaAD 1.1 challenge [Raj+16], its ability to comprehend textual information is overestimated; BERT is oversensitive to question paraphrases and adversarial input and requires a large number of labelled training examples for the fine-tuning stage.
- We investigate back-translation as a paraphrase generation method to paraphrase the input questions. We construct three paraphrased development sets containing 54 questions using back-translation with different pivot languages and one development set created manually. We show that the best paraphrase generation method uses two pivot languages one of which originates from a different language family than the original language.
- We propose an architecture for the fine-tuning phase of BERT that incorporates data augmentation and multitask learning as a regularisation method and evaluate its ability to improve the robustness of BERT to question paraphrases and adversarial input. We compare this model to the supervised data augmentation method of simply concatenating new labelled examples and semi-supervised model where the supervised loss between the predictions obtained from the augmented examples and the corresponding labels is not included in the total loss. We discovered that supervised data augmentation method shows higher performance on both the development set and paraphrased examples, however, our proposed model outperforms the original BERT when evaluated on the constructed paraphrased examples.

1.5 Structure of the Dissertation

Chapter 2 presents recent works in QA research considering data augmentation, multitask learning and adversarial training techniques. Chapter 3 outlines the proposed augmented model. Chapter 4 describes all the experiments performed during the project and the justifications for the choices made at each step. Chapter 5 presents a thorough summary and discussion of the results including critique of the proposed model and suggestions for future work.

Chapter 2

Background and Recent Works

This chapter provides an overview of deep learning and NLP approaches that most QA systems are build on and presents state-of-the-art QA models and well-known QA datasets. Furthermore, data augmentation and multitask learning techniques are discussed and approaches that use adversarial training to improve the performances of QA systems are presented.

2.1 Deep Learning for NLP

Deep learning is a thriving field that outperforms the vast majority of classical approaches to tackle NLP tasks. It is an area of machine learning research that focuses on artificial neural networks, frameworks loosely biologically inspired by human brain neurons. A neural network in its simplest form is a mathematical function composed of smaller functions that serve as some representations of the input values. The input is given as a vector of features, encoded information about the input, that are used to compute the output values. In QA, the input typically consists of a sequence of tokens (question + context) that are converted by a learning algorithm into a sequence of other tokens representing the answer. We consider three essential architectures that are extensively deployed for NLP tasks: recurrent neural networks (RNNs) and convolutional neural networks (CNNs) and Transformers.

2.1.1 Sequential Models

Sequential models are popular in NLP due to the sequential nature of language. In contrast to feed-forward neural networks that treat features independently and where information flows from the input layer (groups of neurons) to the hidden layers and finally to the output layer in a chain, in sequential models the input in fed at each time step and the weights are shared [RHW88].

RNN

The architecture of an RNN is designed as follows: given an input vector $\mathbf{x} = (x_1, \ldots, x_T)$, at each time step t, the hidden state h_t is computed using the input x_t and the previous hidden state h_{t-1}. These hidden states are then used to compute the output vector $\mathbf{y} = (y_1, \ldots, y_T)$ using the following equations:

$$\begin{aligned} h_t &= f(W_1^h x_t + W_2^h h_{t-1} + b^h) \\ y_t &= W^y h_t + b^y, \end{aligned}$$

where f is a non-linear activation function (such as tanh). Figure 2.1 shows a vanilla RNN architecture.

The main advantage of an RNN is that has a memory of the past events encoded in its hidden states. Memory is important when processing textual information composed of sequences of words.

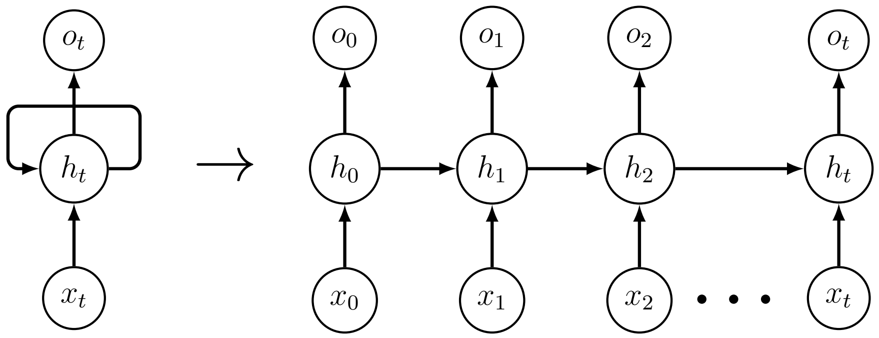


Figure 2.1: Recurrent Neural Network unfolded through time

LSTM

Often when using RNNs, especially when modelling long sequences, we may encounter the problem of vanishing or exploding gradients [BFS93]. In the case of vanishing gradients, that happens more often in practice, memory about past events may get lost. To illustrate this suppose we are trying to predict the last word of a sentence that is directly related to the beginning of the sentence. If a sentence is sufficiently long, an RNN model might loose the information about the beginning of the sequence by the time it gets to analyze the last word creating the issue of long-term dependencies.

Long Short-Term Memory (LSTM) units are an extension to RNNs proposed to solve this problem [HS97]. The architecture allows the information from previous time steps to pass forward using a series of gates.

An LSTM unit is defined with the following equations:

$$
\begin{aligned}
f_t &= \sigma(W^f z_t + b^f) \\
i_t &= \sigma(W^i z_t + b^i) \\
o_t &= \sigma(W^o z_t + b^o) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tanh(W^c z_t + b^c) \\
h_t &= o_t \cdot \tanh(C_t),
\end{aligned}
$$

where $z_t = [x_t; h_{t-1}]$.

The forget gate controls the amount of information passed through from the previous hidden state. Similarly, the input gate modulates the amount of information passed through to the current state. The new internal state C_t is then computed as a linear combination of the forget gate and the input gate.

Bidirectional RNN

So far we have discussed recurrent architectures where at a time step t only the information from the past is taken into account. In many scenarios, however, it is vital to incorporate information from the whole sequence. This is particularly relevant when modelling text; the meaning of a word does not only depend on the previous words but also next few words in the sentence due to the linguistic relations between them. Proposed by [SPG97], a bidirectional RNN is a modified RNN composed of two RNNs, one going forward in time and the other going backward. Many successful QA models have bidirectional recurrent structure incorporating the full context of a question and a sentence, as we will see in more detail in later sections.

2.1.2 Convolutional Models

Convolutional neural networks (CNNs) first proposed by [LeC+90] are neural networks that incorporate the convolution operation replacing matrix multiplications. In contrast to sequence models, CNNs only use the current input to predict the output at the layers and are not designed to store information about the previous inputs. Instead, CNNs focus on relative location and proximity to extract patterns from the data. One advantage of CNNs over RNNs is that computations can be parallelized making them more efficient in practice when using modern hardware. See [Gu+17] for more detailed information about CNNs and recent advances.

While CNNs are more common in the computer vision community, they have proven to be successful in a other tasks such as speech recognition [Wai+89], neural machine translation [Geh+17] and QA [Yu+18].

2.1.3 Attention Mechanisms and Transformers

Neural network architectures encode inputs into fixed-length vectors. The problem with this approach, however, is that the network must be able to compress all the important information into that vector without including irrelevant information. This may be infeasible, especially when long sequences are introduced. The human brain, on the other hand, is able to focus on one instance of the environment at a time ignoring unnecessary information. For example, we can easily distinguish the voice of a person we are speaking to from the background noise. Can we arm neural networks with the ability to select only relevant portions of the input?

Attention

[BCB14] introduces attention mechanisms in order to consider the local context in a sequence ignoring unrelated information. An attention mechanism converts the input into a sequence of vectors that can be viewed as a memory containing a sequence of information. It then exploits the memory to select only those portions that are relevant to a specific task.

The computation of a typical attention mechanism works as follows:

- Compute the similarity between a given query and a key using some similarity metric (such as the dot product):

$$\text{score}\left(Q_t, K_s\right) = Q_t K_s^T$$

- Convert the similarity scores to attention weights (for example using softmax):

$$\alpha_{ts} = \text{softmax}(\text{score}\left(Q_t, K_s\right)) = \frac{\exp\left\{\text{score}\left(Q_t, K_s\right)\right\}}{\sum_{s'=1}^{S} \exp\left(\text{score}\left\{Q_t, K_{s'}\right)\right\}}$$

- Introduce weights to the context vectors for training (for example using tanh):

$$a_{ts} = f\left(V_t, Q_t\right) = \tanh\left(W_v\left[V_t; Q_t\right]\right),$$

  where V_t is the context vector computed as:

$$V_t = \sum_s \alpha_{ts} K_s$$

[CDL16] extends the concept of attention introducing self-attention in order to relate different portions of the same sequence to one another. Self-attention allows to learn complex syntactic relations between tokens of a single sentence. RNNs and CNNs combined with attention or self-attention mechanisms became very powerful in making connections between words and relevant contexts.

Transformers

Recognizing the strength of attention mechanisms, [Vas+17] introduce the Transformer whose architecture is fully composed of attention mechanisms without combining them with CNNs or RNNs. The Transformer consists of two types of attention mechanisms: a scaled dot-product attention and a multi-head attention (See Figure 2.2).

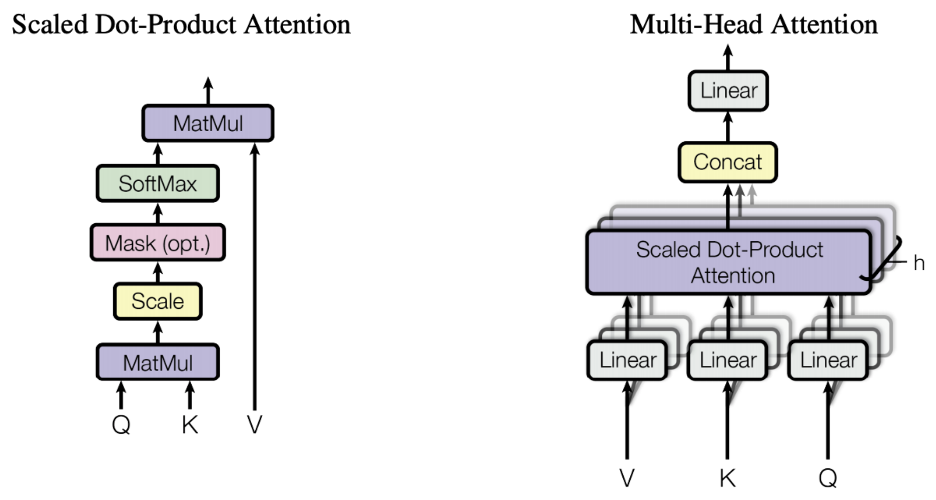


Figure 2.2: Attention mechanisms architectures (retrieved from [Vas+17])

For the scaled dot-product attention, the input is now a query Q, a set of keys K and a value V, and the output is essentially computed as a weighted sum of the values similar to the general attention mechanisms framework but the dot-product similarity scores are scaled by $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of the keys.

The multi-head attention is composed of multiple scaled dot-product mechanisms running in parallel which are then concatenated and passed through a linear layer.

The Transformer combines these two attention mechanisms with fully connected feed-forward layers into a single architecture (see Figure 2.3). Due to a smaller number of computations and parallelization, this architecture is usually faster in practice than its neural network counterparts. The Transformer became an essential component of BERT ([Dev+18]), the main NLP model that we investigate in this work, as we will see in the next sections.

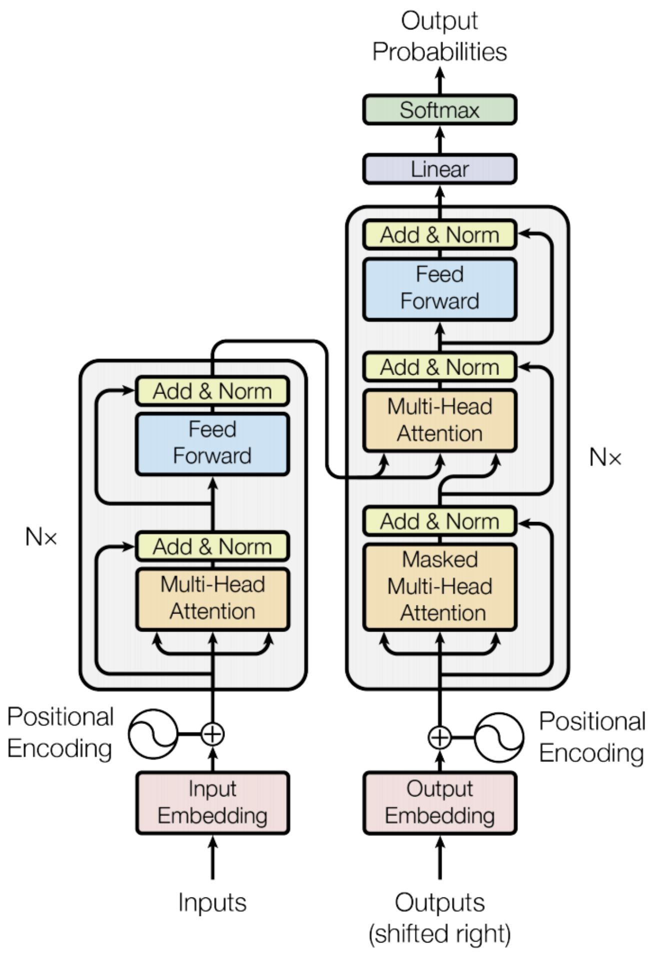


Figure 2.3: Transformer architecture (retrieved from [Vas+17])

2.1.4 Language Models

Most NLP models benefit from language modelling that can be incorporated in many ways. A language model defines a probability distribution over sequences of tokens. That is, given a sequence of tokens $(t_1, t_2, \ldots, t_n)$ that comprises an example, a language model assigns a probability $P(t_1, t_2, \ldots, t_n)$ that the sequence occurs in natural language.

For example, a language model would assign a higher probability to the sentence ***Comprehending natural language requires a high level of reasoning and inference*** than to the sentence ***Language high a requires level inference of comprehending***

reasoning natural and.

N-grams

N-gram language modelling is the most traditional method of language modelling that define a conditionally probability distribution indicating how a token depends on the preceding n tokens:

$$P(t_1, t_2, \dots, t_N) = P(t_1, t_2, \dots, t_{n-1}) \prod_{i=n}^{N} P(t_1, t_2, \dots, t_{i-1})$$

This method, however, has several limitations. First of all, most n-grams will have a probability of zero because they will not appear in the training set, even though those n-grams may be valid sequences of words. In addition, the vocabulary size is often very large and the number of n-grams grows exponentially. This makes n-grams inefficient in many practical scenarios. Moreover, n-grams do not take into account similarities between tokens. Due to the nature of language, some words are more related to each other and others are more different. For example, intuitively ***snow*** is more close in meaning to ***rain*** than to a ***refrigerator*** and we need a way to capture these similarities between words.

Although class-based language models ([Bro+92]) provide a way to overcome some of the limitations of vanilla n-grams by dividing words into clusters, a lot of information is still not represented by this approach.

Word Embeddings

Recently there has been many advances in language modelling over traditional methods. A word embedding can be used to encode a notion of semantic similarity between words. This allows the learned information about one word be used for a similar word instead of treating them independently like with one-hot representations.

Word embeddings map words to points in a lower dimensional space such that related words lie closer to each other while unrelated words lie farther away. In other words, they create numerical representations of words that capture their semantic meanings. These representations can then be used as inputs to learning algorithms for mathematical computations.

For example, Word2Vec model proposed by [Mik+13] is a powerful word embedding that creates vector representations of words given their contexts. See Figure 2.4 for a Word2Vec illustration.

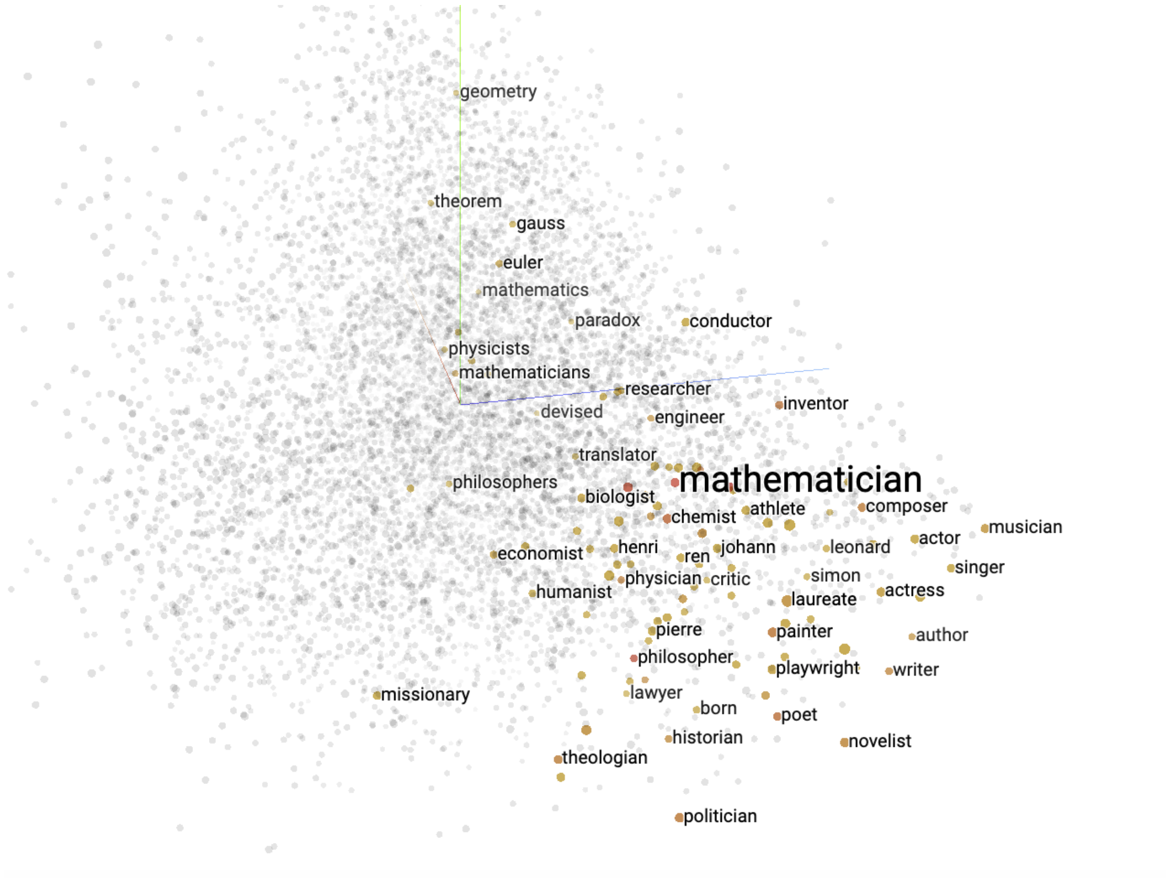


Figure 2.4: Word2Vec illustration. [2]This representation highlights words that are semantically similar to ***mathematician***; more similar words lie closer in the vector space.

2.1.5 Pre-trained Contextual Representations

Word embedding like Word2Vec always assign the same vector to a word irregardless of the context. This is problematic for words that have very distinct meanings depending on the context they appear.

[2]https://projector.tensorflow.org/

In contract to word embedding like Word2Vec, contextual representations take into account the context of a word. One such model is ELMO develop by [Pet+18] that incorporates a bi-directional LSTM.

In recent years, pre-trained contextual representations trained in unsupervised manner have had large success in many NLP tasks. These pre-trained models are fine-tuned using supervised data on downstream NLP tasks such as text classification, natural language inference, QA and others. Here we describe two of such models: OpenAI GPT [KSS18] and BERT [Dev+18].

OpenAI GPT

OpenAI GPT is developed to tackle multiple language tasks at a time motivated by the fact that most NLP systems are designed to tackle single tasks lacking generalization capabilities [KSS18].

The GPT training consists of two phases (see Figure 2.5):

- **Unsupervised pre-training:** the goal is to learn a powerful language model: predict the next word given all the previous words in the span of text using a multi-layer Transformer decoder, a type of Transformer that uses multi-head self-attention followed by a feed-forward layer proposed by [Liu+18].
- **Supervised fine-tuning:** the goal is to adapt the model with task-specific objectives given labeled data. The final Transformer block output obtained from the pre-trained model is passes through a linear output layer to predict the labels.

BERT

Developed and open-sourced by Google AI, BERT (Bidirectional Encoder Representation for Transformers) is a powerful pre-trained model for QA and other prominent tasks in NLP [Dev+18]. Similar to OpenAI's GPT, BERT consists of the pre-training and fine-tuning stages. It is pre-trained on Wikipedia paragraphs and only needs to be fine-tuned for a downstream NLP task, such as QA. Unlike OpenAI GPT, however, BERT has a bidirectional nature which allows to incorporate the full context of a sentence. The input contains additional tokens $[CLS]$ that is placed at the beginning of each sequence and $[SEP]$ that is placed to separate segments of the sequence.

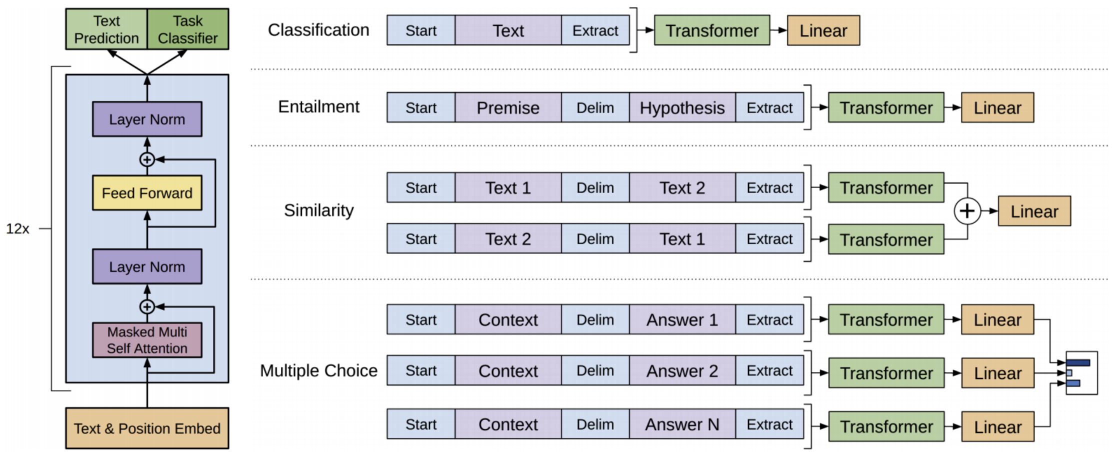


Figure 2.5: OpenAI's GPT architecture (retrieved from [KSS18])

BERT architecture (shown in Figure 2.6) has the following structure:

- **Unsupervised pre-training:**
  - **Masked language modelling:** Mask (replace) randomly chosen tokens and predict what those tokens are based on the full context of the sentence.
  - **Next sentence prediction:** Incorporate the context across the sentences in order for the model to understand relationships between sentences.
- **Supervised fine-tuning:** the pre-trained parameters are used to initialize the model and the task is to minize the loss between the predictions and labels for each downstream task.

  Let T be the output of the pre-trained BERT model. The fine-tuning procedure for QA introduces a start vector S and end vector E representing the location of the prediction. The probability of the i_{th} word being a start or an end is computed as a softmax of the dot product between T_i and S and T_i and E respectively: $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$. The probability of a subsequence being the answer is computed as the sum of log-likelihood of the start and end locations. This comprises the objective function that the model aims to minimize during fine-tuning.

Let L be the number of Transformer blocks, H be the number hidden size and A be the number of self-attention heads. There are three publicly available versions of BERT:

- **BERT Base:** L=12, H=768, A=12, Total Parameters=110M

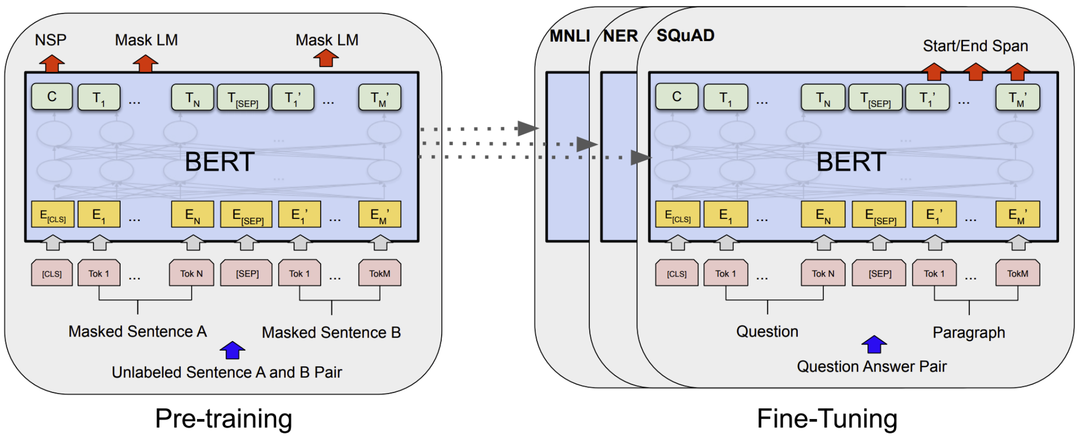


Figure 2.6: BERT architecture (retrieved from [Dev+18])

- **BERT Large:** L=24, H=1024,A=16, Total Parameters=340M
- **BERT Large (Whole Word Masking):** is constructed using the same parameters as BERT Large but now during training all tokens of a word are masked instead of only portions of a word.

In this work, we focus on improving the fine-tuning step of BERT Large for QA. In particular, we investigate how the objective function can be regularized using question paraphrases.

2.2 QA datasets

There have been many datasets created for the purpose of tackling the QA task. In this work we focus on SQuAD dataset [Raj+16], one of the most widely used QA dataset that many successful models are trained and evaluated on. We start by briefly describing some powerful QA datasets containing factual questions about Wikipedia articles. We then describe SQuAD and evaluate its quality.

2.2.1 TriviaQA

TriviaQA, developed by [Jos+17] consists of over $650,000$ question-answer-evidence tuples. [3] The dataset is created by collecting question-answer pairs from quizzes and then retrieving documents from the Web that contain answers to the corresponding questions. TriviaQA contains many challenging questions that require reasoning across sentences and making comparisons.

2.2.2 Natural Question Dataset

Natural Questions dataset [4] released by Google is constructed using real queries to the Google search engine [Kwi+19]. There are $307,373$ training examples, 7,830 development examples and 7,842 test examples. The dataset is composed of *(question, Wikipedia page, long answer, short answer)* quadruples. The advantage of the Natural Question dataset is that it is based on real human queries and thus is more real-word applicable than datasets that are constructed synthetically.

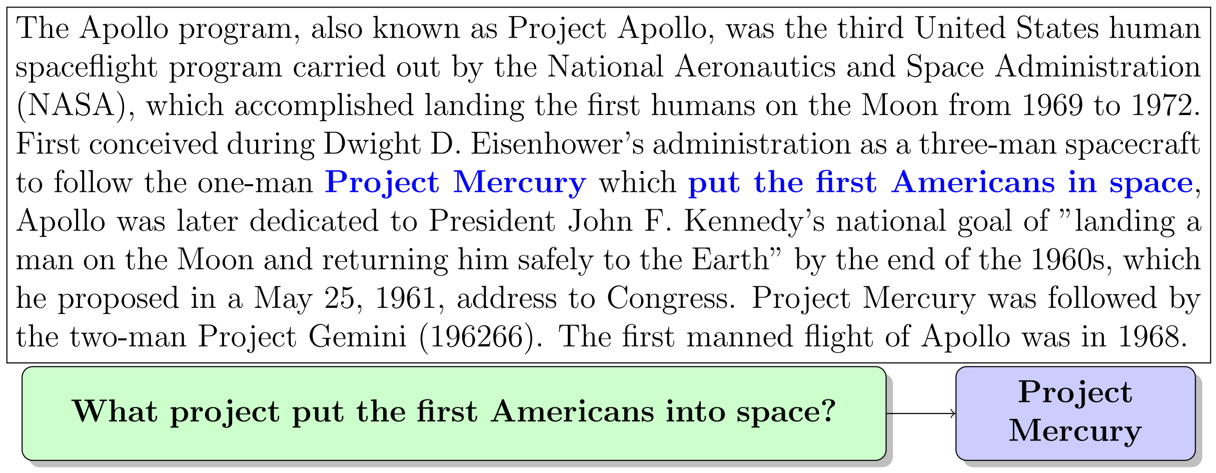
2.2.3 SQuAD 1.1

One of the most popular QA dataset, SQuAD 1.1, consists of $107,785$ sets of questions, passages and answers designed to train and evaluate QA systems [Raj+16]. The passages are taken from Wikipedia and the questions about the passages are created manually. The answer is a segment of the passage and the task is to find its correct location. There are multiple question-answers pairs about a single paragraph.

The dataset is separated into a training set (80%), a validation set (10%) and a test set (10%). Only training and validation sets are publicly available.[5] and we base our experiments on these sets.

The difficulty of the SQuAD 1.1 questions greatly varies. Most questions can be answered with simple key matching or searching for tokens of specific types while only a few questions would require high-level reasoning and inference across multiple sentences. Consider examples in Figures 2.7, 2.8.

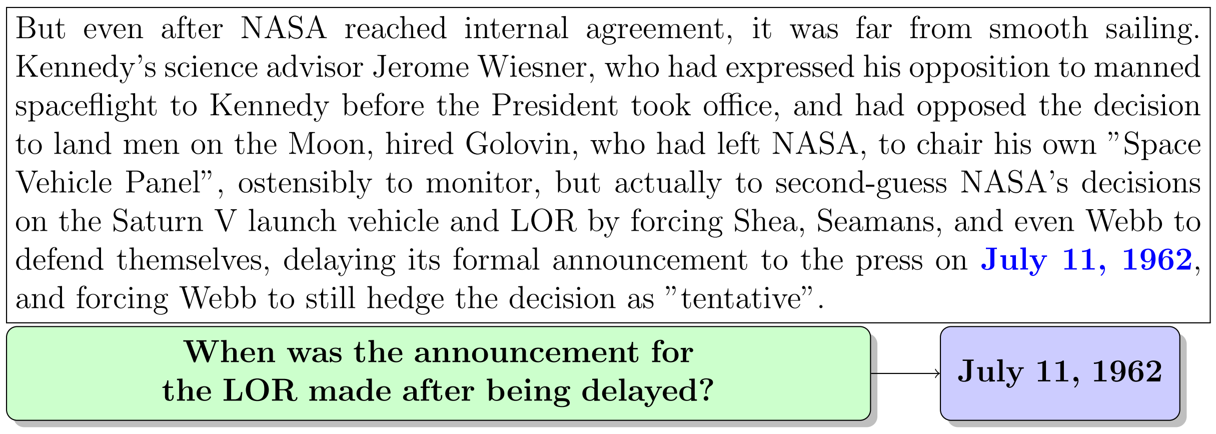
[3] http://nlp.cs.washington.edu/triviaqa/
[4] https://ai.google.com/research/NaturalQuestions
[5] https://rajpurkar.github.io/SQuAD-explorer/



The Apollo program, also known as Project Apollo, was the third United States human spaceflight program carried out by the National Aeronautics and Space Administration (NASA), which accomplished landing the first humans on the Moon from 1969 to 1972. First conceived during Dwight D. Eisenhower's administration as a three-man spacecraft to follow the one-man **Project Mercury** which **put the first Americans in space**, Apollo was later dedicated to President John F. Kennedy's national goal of "landing a man on the Moon and returning him safely to the Earth" by the end of the 1960s, which he proposed in a May 25, 1961, address to Congress. Project Mercury was followed by the two-man Project Gemini (196266). The first manned flight of Apollo was in 1968.

What project put the first Americans into space? → **Project Mercury**

Figure 2.7: SQuAD 1.1 example. The question contains very similar structure and wording as the paragraph. Simply searching for the tokens in the question may be sufficient to locate the correct answer.



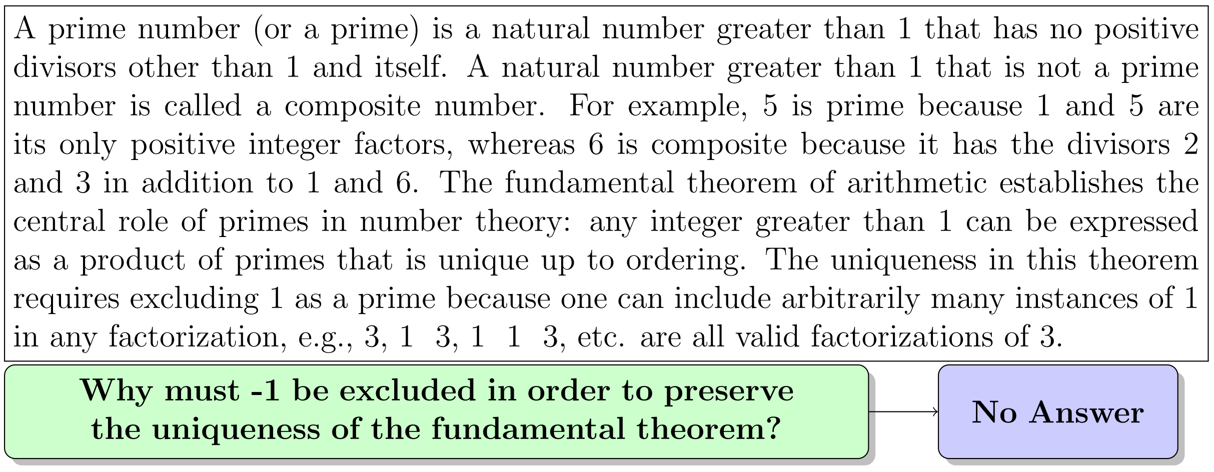
But even after NASA reached internal agreement, it was far from smooth sailing. Kennedy's science advisor Jerome Wiesner, who had expressed his opposition to manned spaceflight to Kennedy before the President took office, and had opposed the decision to land men on the Moon, hired Golovin, who had left NASA, to chair his own "Space Vehicle Panel", ostensibly to monitor, but actually to second-guess NASA's decisions on the Saturn V launch vehicle and LOR by forcing Shea, Seamans, and even Webb to defend themselves, delaying its formal announcement to the press on **July 11, 1962**, and forcing Webb to still hedge the decision as "tentative".

When was the announcement for the LOR made after being delayed? → **July 11, 1962**

Figure 2.8: SQuAD 1.1 example. Since the question start with ***When***, a QA model might pick up the fact that the answer contains a time frame and the only sequence of tokens that refers to a specific date is ***July 11, 1962***.

2.2.4 SQuAD 2.0

SQuAD 2.0 is an extension to SQuAD 1.1 released by [RJL18]. The new release made the SQuAD challenge more complex as now the dataset contains questions that cannot be answered by the given passages. See Figure 2.9 for an example.



A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number. For example, 5 is prime because 1 and 5 are its only positive integer factors, whereas 6 is composite because it has the divisors 2 and 3 in addition to 1 and 6. The fundamental theorem of arithmetic establishes the central role of primes in number theory: any integer greater than 1 can be expressed as a product of primes that is unique up to ordering. The uniqueness in this theorem requires excluding 1 as a prime because one can include arbitrarily many instances of 1 in any factorization, e.g., 3, 1 3, 1 1 3, etc. are all valid factorizations of 3.

Why must -1 be excluded in order to preserve the uniqueness of the fundamental theorem?

No Answer

Figure 2.9: SQuAD 2.0 example with a question that cannot be answered by the passage. A QA model should be able to recognize the lack or irrelevance of the information in the passage and abstain from answering.

2.2.5 Performance Evaluation

Evaluation metrics serve are a way to access how well a learning algorithm performs on a given dataset. One of the most popular evaluation metrics for learning algorithms is ***accuracy***, defined as the number of correct predictions divided by the number of total examples. Other metrics include ***precision*** that determines the proportion of positive identifications that were indeed correct, while ***recall*** determines the proportion of true positives that were identified correctly.

In the context of QA, the evaluation metrics are ***Exact Match*** (EM) and ***F1*** scores (the official SQuAD evaluation metrics [Raj+16; RJL18]).

EM is a binary metric that counts how many predictions were exactly the same as the corresponding labels.

F1 is a less strict metric defined as:

$$\frac{2 \times precision \times recall}{precision + recall}$$

For model predictions on SQuAD, the F1 score is calculated between the tokens and the labels and is averaged across all the instances.

2.3 Other Successful QA Systems

Here we present QA systems that show high performance on the task, most of which achieved state-of-the-art results on SQuAD 1.1 challenge at the time of their submission.

2.3.1 BiDAF

Most standard QA model architectures consist of RNNs with an attention component. A successful example of such a system is Bidirectional Attention Flow (BiDAF) [Seo+17]. The structure of this model consists of six layers performing the following: character-level, word-level and contextual embeddings and a bidirectional attention flow (both query to context and context to query). The BiDAF model architecture is shown in Figure 2.10.

At the time of its submission, BiDAF outperformed all previous QA models on the SQuAD 1.1 challenge. Despite being able to successfully tackle the task, however, this approach is slow due to its recurrent structure. The high training and inference times usually lead to the difficulty and reluctance for further research and deployment in real-world scenarios.

2.3.2 QANet

The limitation of recurrent models lead to the development of faster QA models without recurrent structure. For example, the architecture of QANet is exclusively built on convolutional and self-attention mechanisms [Yu+18]. The QANet model architecture is shown

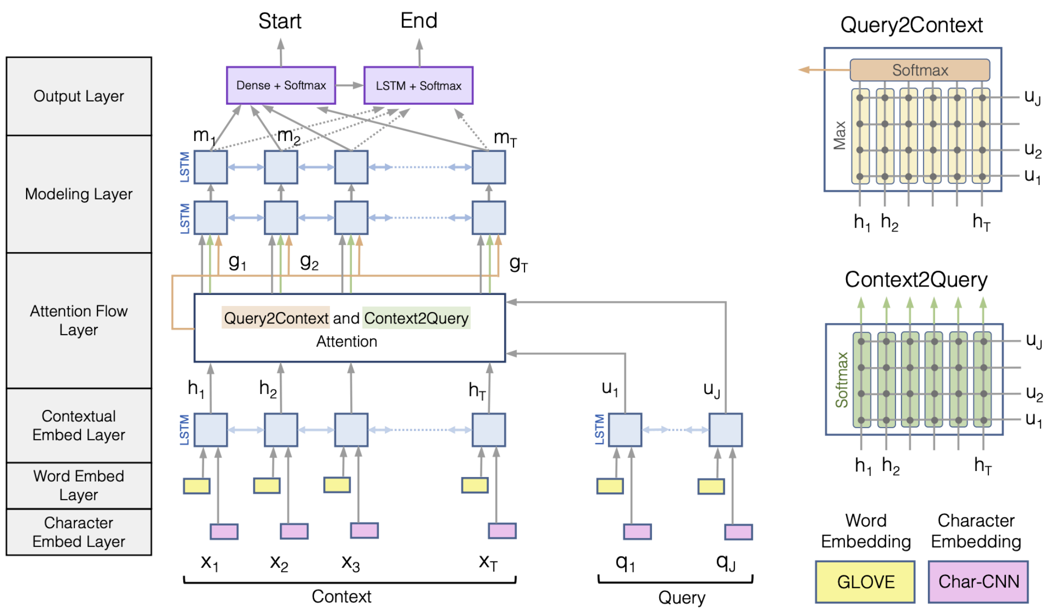


Figure 2.10: BiDAF architecture (retrieved from [Seo+17])

in Figure 2.11. Not only was QANet faster than BiDAF, it also achieved the new highest scores on the SQuAD 1.1 challenge. QANet performs even better when regularization techniques are introduced; we revisit this model in the data augmentation section.

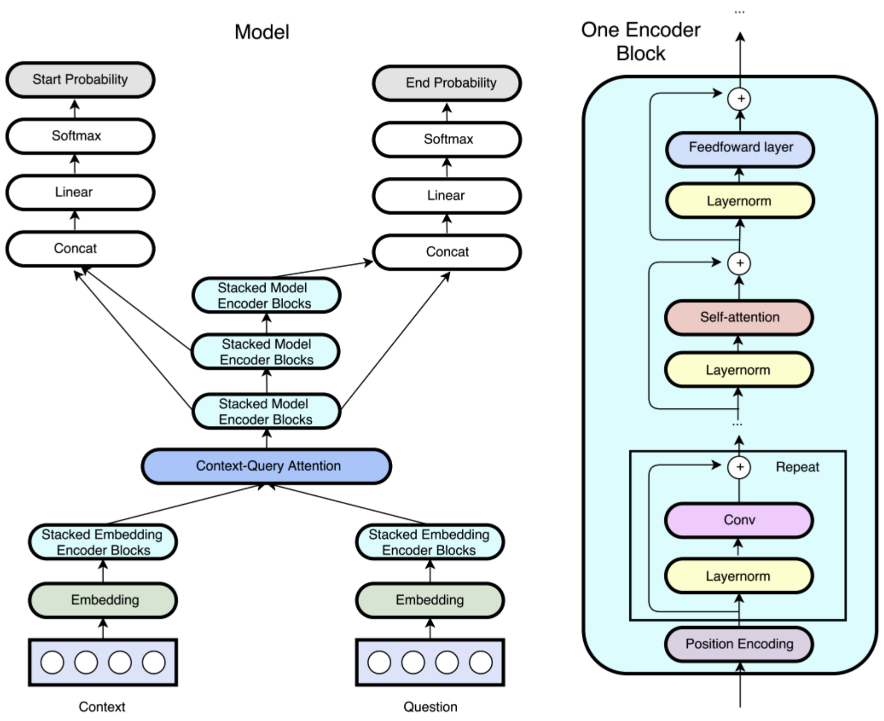


Figure 2.11: QANet architecture (retrieved from [Yu+18])

2.4 Regularization for QA systems

Regularization is a set of strategies applied to a learning algorithm, usually its objective function, to reduce its generalization error. In this section, we consider regularization methods particularly relevant to QA: multitask learning, data augmentation and adversarial training.

2.4.1 Multitask Learning

Learning to perform multiple tasks and successfully generalizing to other tasks are essential components of an intelligent system. Multitask learning, introduced by [Car93], is motivated by the idea that humans learn many complex tasks simultaneously rather than in isolation. In the context of machine learning, we can view this approach as follows: if the model is designed in such a way that it performs well on multiple related tasks, it is more likely to show better generalization capabilities by pushing the parameters to benefit all the tasks. Moreover, when large amount of data is not available, learning tasks jointly may provide a way to combine data from those tasks. This idea lead to the development of GLUE and superGLUE benchmarks, tools for evaluating general-purpose NLP models instead of evaluating the performance on a single task [Wan+18; Wan+19].

MQAN

[McC+18] proposes a Multitask Question Answering Network (MQAN) designed to perform ten NLP tasks ranging from text summarization to machine translation. The model is composed of bidirectional LSTMs and attention components. It is trained on decaNLP, a dataset specifically designed for the model. Unlike previously discussed QA models, MQAN does not assume that the answer is extracted from the context. The architecture of MQAN is illustrated in Figure 2.12.

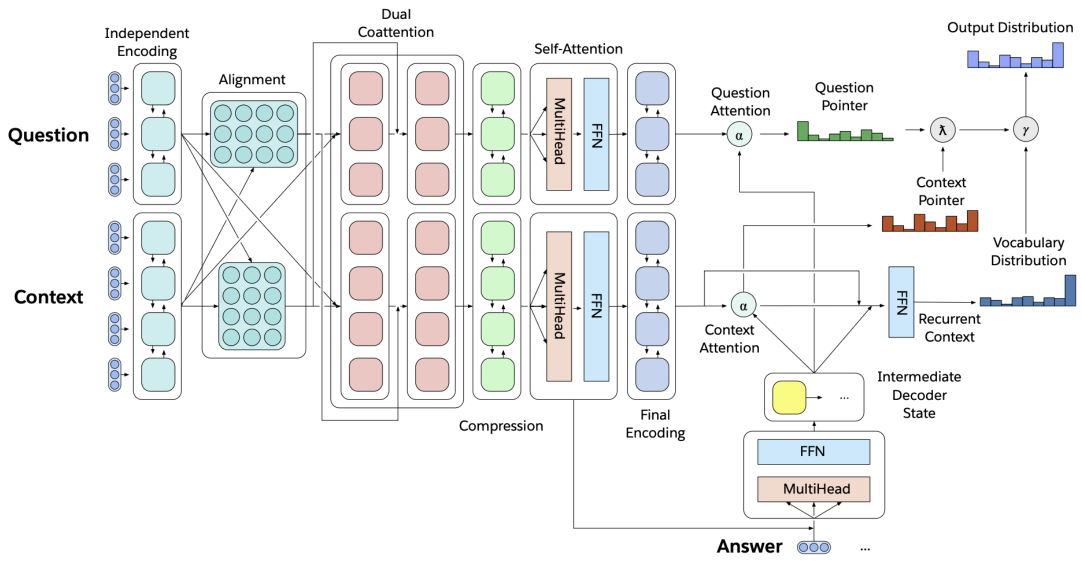


Figure 2.12: MQAN architecture (retrieved from [McC+18])

While achieving high performance on some of the tasks, MQAN contributes to performances drop in others.

MT-SAN

[Xu+19] claims that simply adding multiple tasks in one network does not lead to a better performance and presents a machine comprehension model Multi-Task SAN that focuses on re-assigning weights in the loss function. The idea is to use data from other domains to improve the performance on the target task. They propose to sample a task and a corresponding mini-batch according to their importance. They propose to weight the tasks either using grid search or by incorporating language models.

MT-DNN

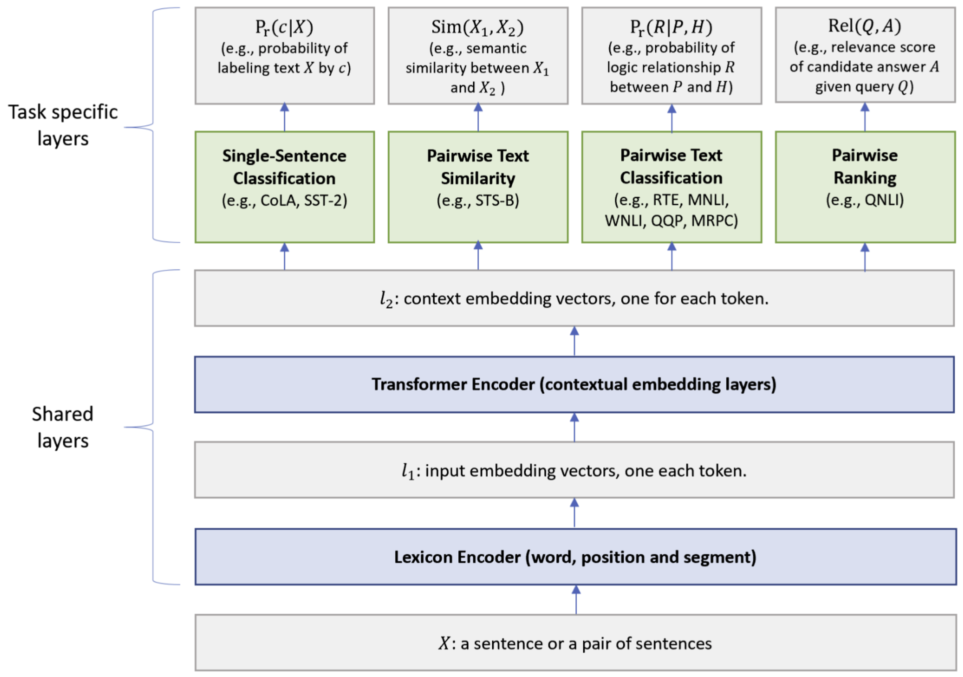
Multi-Task Deep Neural Network (MT-DNN) ([Liu+19b; Liu+19a]) combines advantages of both multitask learning and pre-trained language representations. The model consists of layers shared across four tasks (single-sentence classification, pairwise text classification, text similarity scoring and relevance ranking) at the bottom and task-specific layers on the top. The model produces multiple outputs, each corresponding to one of the four tasks. The pre-training phase consists of the same steps as in BERT. In the multitask phase, for each tasks a mini-batch is selected and the model gets updated according to the corresponding task. The architecture of MT-DNN is shown in Figure 2.13.

MT-DNN achieved state-of-the-art results on eight GLUE tasks (2.2% absolute improvement over BERT Large).

2.4.2 Data Augmentation

Data augmentation refers to the technique of producing realistic examples such that they serve as different representations of the original examples without having to collect additional data. This can be used to increase the amount of training data and as a regularization method encouraging the model to learn a function that is robust for small changes in the input. Data augmentation is convenient when we can find a transformation that keeps some properties of the data invariant. This approach is popular in computer vision where images are often cropped, scaled, flipped, padded and combined with other images to enrich datasets.

In NLP, however, data augmentation is more difficult to accomplish due to the complexity



$P_r(c|X)$ (e.g., probability of labeling text X by c)
$\mathrm{Sim}(X_1, X_2)$ (e.g., semantic similarity between X_1 and X_2)
$P_r(R|P,H)$ (e.g., probability of logic relationship R between P and H)
$\mathrm{Rel}(Q,A)$ (e.g., relevance score of candidate answer A given query Q)
Task specific layers
Single-Sentence Classification (e.g., CoLA, SST-2)
Pairwise Text Similarity (e.g., STS-B)
Pairwise Text Classification (e.g., RTE, MNLI, WNLI, QQP, MRPC)
Pairwise Ranking (e.g., QNLI)
l_2: context embedding vectors, one for each token.
Transformer Encoder (contextual embedding layers)
Shared layers
l_1: input embedding vectors, one each token.
Lexicon Encoder (word, position and segment)
X: a sentence or a pair of sentences

Figure 2.13: MT-DNN architecture (retrieved from [Liu+19b])

of language and its discrete nature. Transformations to inputs of NLP models that can be performed for data augmentation include text generation, word replacement with synonyms, word replacement with the nearest word using word embeddings and paraphrase generation.

Supervised Data Augmentation

Models that apply data augmentation with any of these techniques usually do so in supervised learning setting. This means that the augmented examples are simply added to the training set and the model's objective remains minimization of the loss between the examples and the labels without any modifications to the model architecture.

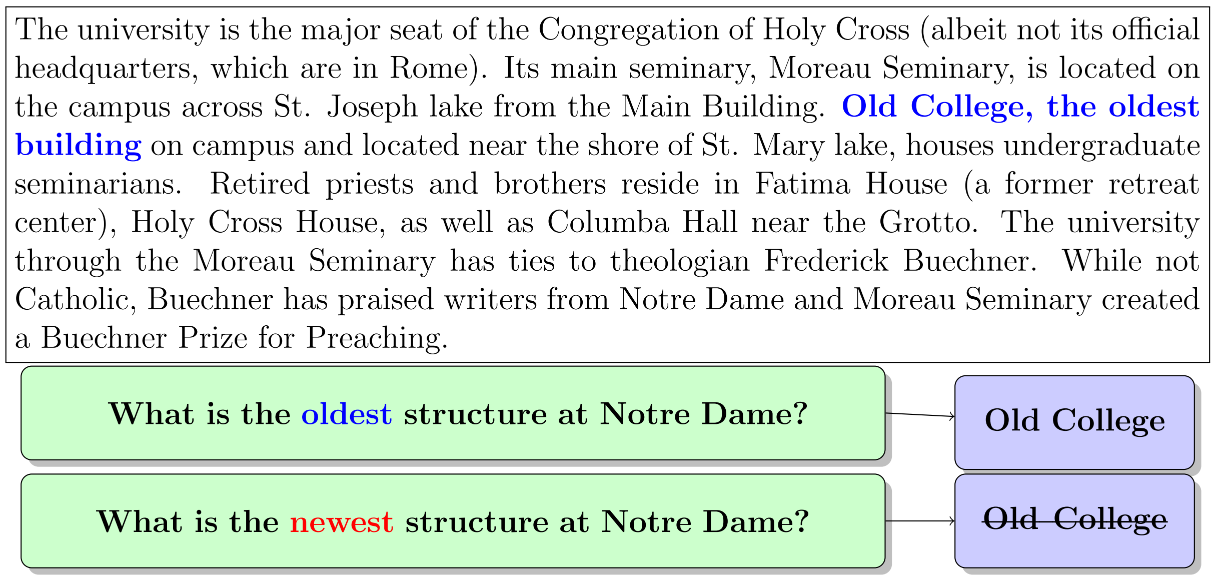
For example, QANet generates context paraphrases by back-translation [Yu+18]. Back-translation refers to the technique of translating an English text into a pivot language and then translating back to English. The idea is that the translated text has the same meaning as the original text but is likely to be worded differently. Back-translation provides two advantages: increasing the size of the dataset and introducing more diverse sentences. As shown in [Yu+18], QANet with supervised data augmentation achieves higher EM and F1 scores compared to the original QANet. The authors, however, do not investigate question paraphrases. While data augmentation in supervised setting certainly improves the performance in most cases due to higher variations in language, it can only be applied if labelled examples are available.

UDA

[Xie+19] propose a novel approach of applying data augmentation to state-of-the-art models in an unsupervised learning manner, named Unsupervised Data Augmentation (UDA). The key idea is encouraging the model predictions between an unlabeled example and an augmented example constructed by data augmentation to be consistent. In the context of QA that means that the predicted locations of answers from a question and its augmented alternative should be as close as possible. The motivation for this technique is to extend data augmentation approaches to unsupervised data that is usually available in larger quantities and is easier to obtain.

2.4.3 Adversarial Training

When a model shows high performance on a task according to the defined evaluation metrics, it does not necessarily imply that the model has a high level of understating of the task. It is important to analyze examples that the model predicts incorrectly. For this purpose, researchers often construct adversarial examples, that is, examples made to confuse the model and then use them to make the model robust to such examples. In contrast to data augmentation, where we have to make sure that the applied transformation of the input does not change the identify of the label, in the process of adversarial training the label may or may not change depending on how we wish to confuse the model. For example, in QA, replacing a token or even one character in a question may drastically change its meaning and hence the model must be able to change the predictions accordingly. If the context does not contain the answer to the modified question, a model capable of sophisticated reasoning and comprehension of the text should be robust enough to return no answer, as we saw previously with SQuAD 2.0 dataset. This is illustrated in the example in Figure 2.14.



The university is the major seat of the Congregation of Holy Cross (albeit not its official headquarters, which are in Rome). Its main seminary, Moreau Seminary, is located on the campus across St. Joseph lake from the Main Building. **Old College, the oldest building** on campus and located near the shore of St. Mary lake, houses undergraduate seminarians. Retired priests and brothers reside in Fatima House (a former retreat center), Holy Cross House, as well as Columba Hall near the Grotto. The university through the Moreau Seminary has ties to theologian Frederick Buechner. While not Catholic, Buechner has praised writers from Notre Dame and Moreau Seminary created a Buechner Prize for Preaching.

What is the oldest structure at Notre Dame? → **Old College**

What is the newest structure at Notre Dame? → ~~**Old College**~~

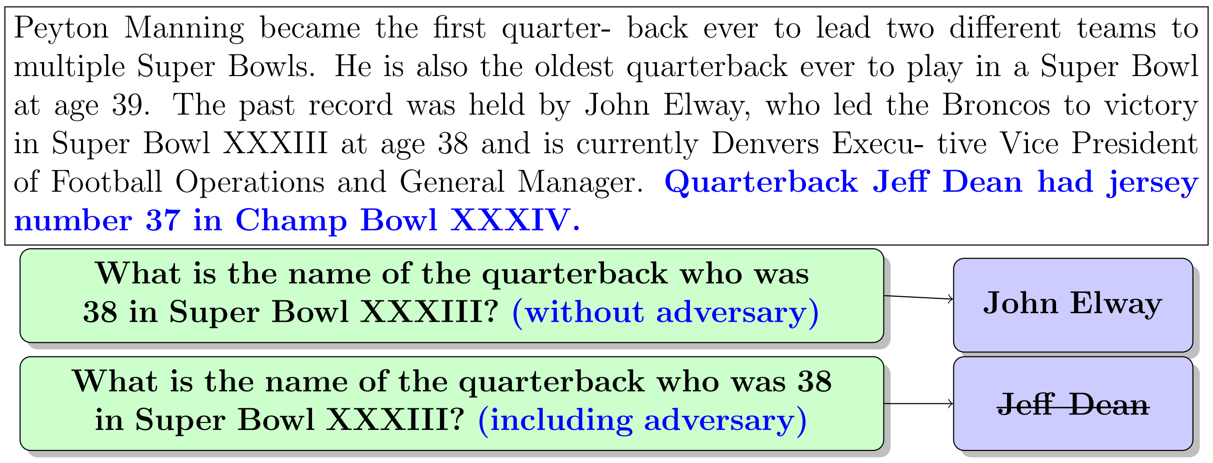
Figure 2.14: Adversarial Question Example. Here replacing a single word with its antonym significantly changes the meaning of the question such that the passage does not contain an answer to the new question. A powerful machine comprehension model should not be fooled by the similar structure and words and be able to abstain from answering.

To the best of our knowledge, [JL17] is the first work on adversarial example generation in QA. Their approach involves adding sentences to the end of the input passages such that they have words in common with the questions but do not contain answers to the questions. Those sentences serve as distractions making QA models more likely to output an incorrect answer. This is an instance of concatenative adversaries since we are only adding sentences without modified the passage or the questions. [JL17] create two adversaries based on SQuAD 1.1:

- AddSent: adds grammatically correct sentences that contain some words from the questions.
- AddAny: adds random sequences of words.

See Figure 2.15 for an example of AddSent adversary applied to BiDAF. From this example it is obvious that the model relies on superficial cues without high-level reasoning and inference about the text. Constructing adversarial examples on SQuAD 1.1 dataset and evaluating this approach on BiDAF, the authors observe significantly lower F1 scores.

One of the most recent works done in parallel with our work that investigates data aug-


Peyton Manning became the first quarter- back ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denvers Execu- tive Vice President of Football Operations and General Manager. **Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.**

What is the name of the quarterback who was 38 in Super Bowl XXXIII? (without adversary) → **John Elway**

What is the name of the quarterback who was 38 in Super Bowl XXXIII? (including adversary) → **~~Jeff Dean~~**

Figure 2.15: Concatenative Adversarial Example (adapted from [JL17]) Note that the additional sentence (in blue) does not contradict the question and the label. The distracting sentence confused BiDAF to output the wrong prediction even though initially the output was correct.

mentation and adversarial training for improving the robustness of QA models to question paraphrasing is [GN19]. They do so by manually creating two development sets: 1) a set of paraphrased questions similar to the original SQuAD 1.1 dataset and 2) a set of adversarial questions that use context words near an incorrect answer.

It is discovered that the performance of BERT and BiDAF significantly worsens when evaluated on the two datasets indicating that these models are very sensitive to question wordings. The models are then retrained using the original training set and additional 25,000 paraphrased questions in a supervised manner to improve the performance.

Chapter 3

Augmented Model Architecture

In this work, we focus on improving fine-tuning strategies of BERT Large for QA. We propose an augmented model architecture and attempt to solve some of the limitation of the fine-tuning stage of BERT:

- over-sensitivity to question paraphrases
- over-sensitivity to passages with distracting sentences
- the requirement to have a large number of labelled training examples

3.1 Model Design

We propose to combine supervised data augmentation with a method similar to UDA [Xie+19]. To address the limitations of BERT, we perform data augmentation and multi-task learning as a regularisation method for the fine-tuning stage of BERT. The architecture of the proposed augmented model is shown in Figure 3.1.

- **Data augmentation** is performed by constructing additional training examples by paraphrasing the original set using back-translation.

  More formally, given the input tuple (c, q_{orig}, a), where c stands for context, q_{orig} is a factual question about the context and a is the labeled answer, we construct q_{par} which is a paraphrased alternative of q_{orig}.

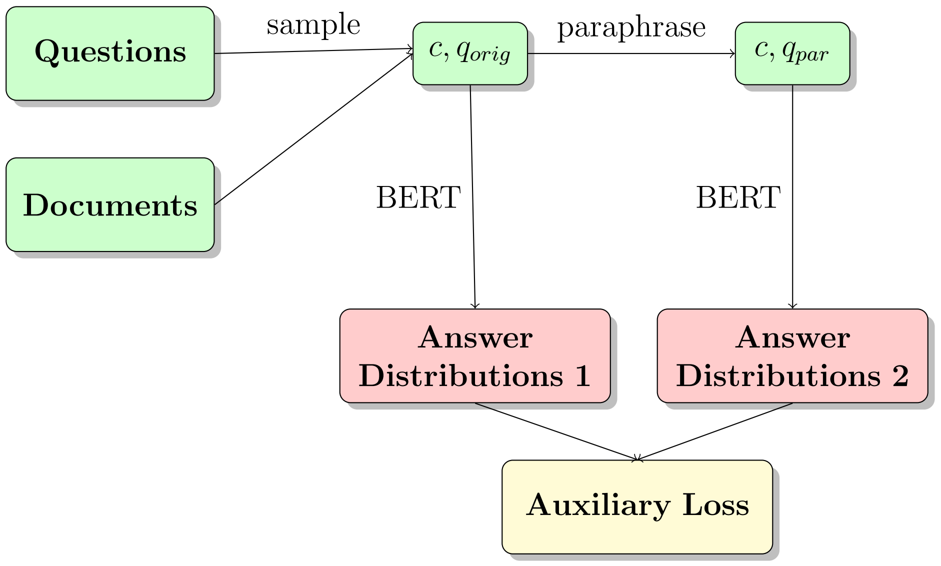


Figure 3.1: Augmented Model Architecture

- **Multitask learning** is performed by introducing an auxiliary objective of minimizing the loss between two sets of predictions (predictions obtained by feeding the original questions q_{orig} and those obtained by feeding the corresponding paraphrased questions q_{par}), in addition to the original task of predicting an answer a' minimizing the supervised loss between a' and a.

3.2 Auxiliary Unsupervised Loss

Unlike [Xie+19] that only uses Kullback-Leibler (KL) divergence to compute the loss between original predictions and their augmented counterparts, we propose to use symmetric KL divergence or the Jenson-Shannon (JS) distance (as defined below).

KL divergence for discrete probability distributions $P(x)$ and $Q(x)$ over the same random variable x is defined in 3.1.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{Q(x)}{P(x)} \tag{3.1}$$

KL divergence is equal to 0 only in the case of the two discrete distributions being equal. A successful QA system should ideally predict the same answer distributions for two questions that have the same meaning. That is, if P represents an answer distribution predicted from an original question and Q represents an answer distribution predicted from a paraphrased question, then we should expect to obtain $D_{KL}(P||Q) = 0$. As we show in the next chapter, this is not often the case with BERT. Hence, our proposed model incorporates the auxiliary task of minimizing this loss.

Note that KL divergence is not a true distance measure because in general $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. We wish, however, to construct a symmetric loss such that the order at which we feed the answer distributions are fed into the loss function does not matter because a question and its paraphrase should be treated as being equivalent.

In order to do this, we calculate the sum of KL divergences in the following way:

$$D(P||Q) = \frac{1}{2}\left[[D_{KL}(P||Q) + D_{KL}(Q||P)\right] = \frac{1}{2}\left[\sum_{x \in X} P(x) \log \frac{Q(x)}{P(x)} + \sum_{x \in X} Q(x) \log \frac{P(x)}{Q(x)}\right] \tag{3.2}$$

Jenson-Shannon (JS) distance introduced by [Lin91] is another way to symmetrify the KL divergence:

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) = \frac{1}{2}\left[\sum_{x \in X} P(x) \log \frac{2P(x)}{M(x)} + \sum_{x \in X} Q(x) \log \frac{2Q(x)}{M(x)}\right]$$

where $M(x) = \frac{P(x)+Q(x)}{2}$.

Experiments comparing the performance of the proposed model trained using each of the distance metrics are presented and discussed in the next chapter.

3.3 Total Loss

The total loss is defined as a weighted sum of a supervised and auxiliary unsupervised loss (Formula 3.3). The supervised loss L_s is the loss between the predictions and the labels as defined in the original BERT fine-tuning model. The unsupervised loss L_u is defined as the distance between the predictions obtained from the original training examples and paraphrases computed using one of the distance metrics discussed above. The total loss is computed as

$$L = \alpha L_s + (1 - \alpha) L_u, \tag{3.3}$$

where is a hyperparameter that sets the extent at which the model emphasizes each loss. During experiments we perform hyperparamter tuning in order to determine the value for α under which the model performs the best.

Chapter 4

Experiments and Analysis

4.1 Initial Hypotheses Testing

4.1.1 Testing Paraphrasing Techniques

Initially, we test the hypothesis that back-translation provides a successful way of generating paraphrases and attempt determine which intermediate languages should be used for better paraphrase quality. We hypothesize that multiple pivot languages one of which is from a different language family produce paraphrases with diverse wordings. To test this, we construct four sets of paraphrased questions using SQuAD 1.1 development set [1]:

- **Back-translated set 1**: using Chinese (Mandarin) and Dutch as intermediate languages. The idea is that Chinese being from a different language family would enrich the paraphrased questions with more diverse structures.
- **Back-translated set 2**: using Vietnamese and Indonesian as intermediate languages. Here all the languages are from different language families. We hypothesize that using back-translation with multiple languages of very different origins may lead to drastic changes of the meanings of the questions. Hence, this method would produce poor question paraphrases.

[1]We choose SQuAD 1.1 dataset because existing QA models already show human performance on the task in terms of EM and F1 scores. We argue that this evaluation is not sufficient to conclude that the machine comprehension challenge is solved and further investigations are needed.

- **Back-translated set 3**: using French as a single intermediate language. Several works such as [Yu+18] use back-translation with French. We hypothesize that using one pivot language that is close in origin to the language of the original questions may cause very small changes to the input questions. Thus, this method may also be a poor way of generating paraphrases since the goal is to enrich the dataset with diverse examples.
- **Manually paraphrased set**: manually paraphrasing the questions for the purpose of having a human baseline to access the quality of other methods.

We use human evaluation to compare the quality of generated paraphrases. We calculate the percentage of questions that were not changed by the paraphrasing method (including the cases where the change was insignificant such as the addition of *the* to a title). We also calculate the percentage of questions that are valid paraphrases. For a paraphrase to be valid, it must: i) retain the meaning of the original question, ii) be expressed using different words and/or structure and iii) be grammatically correct.

The human evaluation results are given in Table 4.1.1. Note that the first back-translated set contained the largest number of acceptable paraphrases (over 90% out of which 61.11% are valid an 29.63% are unchanged). The second method contains a large number of incorrect paraphrases that change the meaning of the questions and the last method did not paraphrase most of the questions (75.93%) as we expected. We noticed however that none of the methods resulted in particularly diverse paraphrases such as those that were produced manually. Hence, in the future more complex paraphrasing technique may be more beneficial.

In the subsequent experiments we deploy the first method that uses Chinese and Dutch as intermediate languages in order to generate question paraphrases.

Table 4.1: Human Evaluation of Paraphrasing Techniques Based on Back-Translation

Paraphrasing Method	**Valid**	**Unchanged**	**Total**	**Valid/Total**	**Unchanged/Total**
Back-Translated Set 1	33	16	54	61.11%	29.63%
Back-Translated Set 2	16	15	54	29.63%	27.78%
Back-Translated Set 3	10	41	54	18.52%	75.93%

4.1.2 Testing Robustness of BERT to Paraphrases

The original BERT Large model was trained on the SQuAD 1.1 training dataset using default hyperparameters and evaluated on the original SQuAD 1.1 development set, achieving the EM score of 83.86 and F1 score of 90.50, consistent with the results reported in [Dev+18]. While this is higher than human performance ([Raj+16]), we hypothesize that BERT may be performing word-matching or relying on other simple patterns to locate the answers and thus will show lower performance if the questions are paraphrased. In this part, we test the robustness of BERT to the generated paraphrased development set.

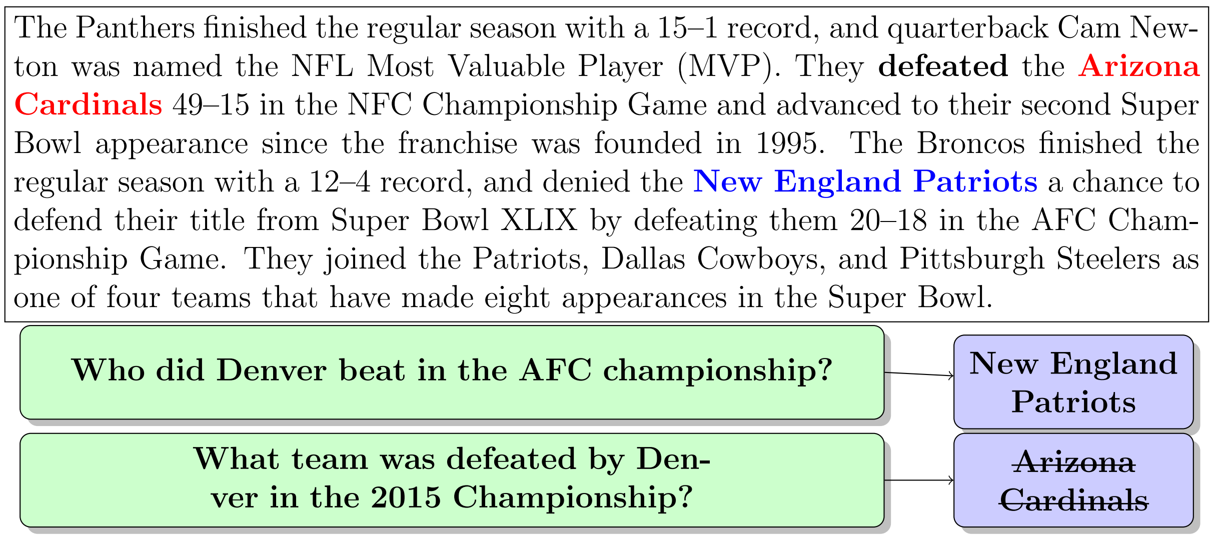
In addition to the EM and F1 evaluation metrics, the mean values for the KL divergence and the Wassertstein distance between the predicted answer distributions of the original set and the augmented set are computed. Wassertstein distance (or Earth Mover's distance) is a measure of dissimilarity between two distributions that computes how much *work* needs to be done for one distribution to be transported to obtain the second distribution. For more information see [RTG98].

We also calculate the confidence of BERT by computing the mean of the probabilities of the start and end locations of the predictions. We analyze predicted answer distributions since high F1 score can be achieved by models that look for simple predictive patterns without necessarily having high level of reasoning and inference [JL17]. Thus, we make a deeper analysis of resilience of BERT to question paraphrases and adversarial examples than [GN19].

Figure 4.1.2 shows an example when the model predicted the correct answer for the original input but produced an incorrect prediction when the input question was manually paraphrased.

Table 4.2: Results of BERT evaluated on 54 original and 54 paraphrased questions (SQuAD 1.1)

Development Data	**EM**	**F1**	**Mean KL**	**Mean Wasserstein**	**Confidence**
Original Set	90.74	95.97	-	-	81.51%
Back-Translated Set 1	83.33	90.34	6.05	2.46×10^{-4}	99.88%
Manually Paraphrased Set	83.33	88.02	8.31	2.56×10^{-4}	82.94%


The Panthers finished the regular season with a 15–1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP). They **defeated** the **Arizona Cardinals** 49–15 in the NFC Championship Game and advanced to their second Super Bowl appearance since the franchise was founded in 1995. The Broncos finished the regular season with a 12–4 record, and denied the **New England Patriots** a chance to defend their title from Super Bowl XLIX by defeating them 20–18 in the AFC Championship Game. They joined the Patriots, Dallas Cowboys, and Pittsburgh Steelers as one of four teams that have made eight appearances in the Super Bowl.

Who did Denver beat in the AFC championship?
New England Patriots
What team was defeated by Denver in the 2015 Championship?
~~Arizona Cardinals~~

Figure 4.1: Example of BERT's over-sensitivity to question paraphrasing. We hypothesize that due to the presence of the word ***defeated*** in the paraphrase and ***defeated the Arizona Cardinals*** in the context, BERT was fooled to extract the wrong team name without considering the full context. In addition, most questions in SQuAD training set are given in active voice, so we hypothesize that BERT is not familiar with the structure of passive voice and is more likely to fail to predict answers correctly if the questions are structured in passive voice.

Observe that in both cases the performance or BERT drops significantly, indicating that BERT is sensitive to wordings of the questions. In particular, we can identify the following:

- The F1 score of BERT evaluated on the back-translated set is closer to the score of BERT evaluated on the original set compared to the one on the manually paraphrased set because 30% of the questions being unchanged.
- From the Wassertstein distance calculations, we can see that the Wassertstein distance does not capture the difference in the answer distributions as the value in all cases in very small. Thus, using it for optimization purposes in the augmented model may not be beneficial unless we scale it.
- BERT can be overconfident as is the case of the back-translated set. However, these experiments do not show high correlation between confidence and the performance of BERT.
- KL divergence for the manually paraphrased set is higher than for the back-translated

set as expected. In the subsequent section, we use JS distance and symmetric KL divergence to train the augmented model and compare it to the same model trained with the KL divergence.

4.1.3 Testing Robustness of BERT to Adversarial Input

We test the performance of original BERT Large when adversarial development examples are introduced. We select the adversarial set that includes distracting sentences in the input passages of the SQuAD dataset constructed by [JL17].

Table 4.3 shows the original BERT evaluated on the adversarial test set compared to the same model evaluated on the original questions.

Table 4.3: Performance of BERT Large on original and corresponding adversarial questions

Trained on the full training set			**Trained on 10% of the training set**		
Development Data	**EM**	**F1**	**Development Data**	**EM**	**F1**
Original Set	83.00	89.61	**Original Set**	75.80	83.54
Adversarial Set	51.10	56.27	**Adversarial Set**	39.80	45.14

Observe the significant decrease in performance due to the distracting sentences. We hypothesize that BERT gets confused by the additional sentences because they contain similar words and structure as the questions and BERT picks us simple patterns to be able to locate tokens that look like answers.

4.2 Augmented Model Experiments

This section summarizes the experiments conducted using the proposed model architecture with data augmentation to alleviate the performance drop observed when the model is evaluated on paraphrased questions and adversareial passages.

4.2.1 Experiments on SQuAD 1.1

Implementation Details

We start by setting the default hyperparameters, specifically batch sizes of 24 examples, 2 epochs (corresponding to 7299 training steps), and the maximum sequence length of 384 tokens, as suggested on the official BERT Tensorflow github page. [2]

We then change the maximum sequence length to 386 and the batch size to 16. We also experiment with different epochs since we are essentially doubling the amounts of training data when performing fully supervised model. The need to change the hyperparameters is explained when describing the semi-supervised data augmentation experiments.

Supervised Data Augmentation Experiments

In the first set of experiments (Tables 4.4 and 4.5), the original model is trained to get the baseline for the default hyperparameters. This is compared to the model trained to minimize the loss between the original examples and predictions and the augmented examples and predictions in a fully supervised manner. We implement the fully supervised model by feeding each example individually treating the augmented examples independently from the original examples. Here we computed symmetric KL and JS distances between the predictions of the model with supervised data augmentation and the predictions of the original model. Essentially, BERT with supervised data augmentation is a model similar to BERT trained with a larger number of training examples (although very similar). Thus, we are comparing them with different number of training steps.

Key observations:

- Observe that all the obtained EM and F1 scores are within the range of possible scores reported in [Dev+18]. The confidence and the symmetric KL and JS distances slightly increase as the number of training steps increases but are not substantially different. We hypothesize that the performances of BERT with and without data augmentation is consistent due to the lack of diversity of the augmented examples created by the chosen back-translation approach.

[2] https://github.com/google-research/bert

Table 4.4: Performance of the fully supervised model with data augmentation with default hyperparamters for various number of training steps (SQuAD 1.1).

Fully Supervised Setting						
Model	**Training Steps**	**EM**	**F1**	**Confidence**	**KL**	**JS**
Original BERT Large	7299	**83.86**	**90.50**	82.13%	-	-
Original BERT Large	10949	83.11	90.45	86.17%	-	-
Original BERT Large	14599	82.24	89.88	90.6%	-	-
BERT Large + DA	7299	83.88	90.42	79.18%	0.24	0.17
BERT Large + DA	10949	**83.96**	**90.79**	84.89%	0.40	0.23
BERT Large + DA	14599	82.91	90.28	89.04%	0.55	0.29

- As expected, when training on 4 epochs (14599 training steps), the original BERT Large starts to overfit decreasing both EM and F1 scores. Note, however, that the performance of BERT with data augmentation does not decrease as rapidly.

In order to test whether the augmented answers help to more substantial changes when the training data is limited, we significantly reduce the size of the training data to 8560 examples (10%) and perform the same experiments (Table 4.5). With smaller number of training examples, the advantages of using supervised data augmentation becomes more apparent, in particular, the EM value increases by almost 5% from 73.86 to 78.81 and the F1 score increases by over 3.5% from 83.26 to 86.83.

Table 4.5: Performance of the fully supervised model with data augmentation with default hyperparamters and 10% of the training data for various number of training steps (SQuAD 1.1).

Fully Supervised Setting						
Model	**Training Steps**	**EM**	**F1**	**Confidence**	**KL**	**JS**
Original BERT Large	730	72.96	82.51	79.42%	-	-
Original BERT Large	1095	**73.86**	**83.26**	84.95%	-	-
Original BERT Large	1460	73.91	83.16	87.56%	-	-
BERT Large + DA	730	72.19	81.87	75.52%	0.50	0.25
BERT Large + DA	1095	**78.81**	**86.83**	95.53%	0.91	0.33
BERT Large + DA	1460	73.46	82.76	86.09%	0.84	0.28

For comparison purposes, we retrain the fully supervised model with updated hyperparameters and observe similar trend.

Semi-Supervised Data Augmentation Experiments

Now we evaluate the performance of the proposed augmented model to test whether it improves the original BERT model as well as BERT with supervised data augmentation.

In the second set of experiments, the model is trained to minimize both the loss between the original examples and labels and the unsupervised loss between the original example predictions and the corresponding augmented example predictions.

During the implementation, we encountered the following issue: recall that the maximum sequence length defines an upper bound for the length of each input example. If an example was longer than the upper bound, then it would be separated into two smaller examples. Due to the default maximum sequence length of 384 and the original and paraphrased questions being of different lengths, it was possible that the model would produce different number of examples for each set. This could cause difficulties when computing the loss between the original example predictions and the corresponding augmented example predictions since the model is accessing each example in order. Hence, it was vital to find a maximum sequence size such that both the original examples would exactly correspond to the paraphrased counterparts. The initial proposed solution was to increase the maximum sequence size. However, we observed that increasing the sequence size up to 390 did not solve the issue, and increasing the maximum sequence size to a larger value lead to out-of-memory issues even with smaller batch sizes.

We then sampled 10% of the training set and performed the same search. We observed that setting the maximum sequence length to 386 would allow us to avoid this issue since in this case the model created the same number of examples for each set. Due to the limited computational resources, increasing maximum sequence length leads to out-of-memory issues. Due to the available computational resources, we will use 16 as the training batch size for all subsequent experiments.

Table 4.6 shows the performance of semi-supervised model. The model is trained with 3 epochs since we observed that this produces the highest EM and F1 scores.

Key results:

- Observe that the confidence decreases significantly as we increase α. This means that previously the predicted answer distributions had a high peak at one token (for each the start and end locations) but the proposed model contributed to smoother

Table 4.6: Performance of the semi-supervised model with data augmentation with new hyperparameters and 10% of the training data for 3 epochs (1605 training steps).

Semi-Supervised Model						
Training Data	**Training Steps**	**Loss**	α	**EM**	**F1**	**Confidence**
8560 (10%)	1605	KL	0.2	70.18	80.16	26.12%
8560 (10%)	1605	KL	0.5	72.21	81.53	40.59%
8560 (10%)	1605	KL	0.8	**72.93**	**82.09**	61.32%
8560 (10%)	1605	Symmetric KL	0.2	70.06	79.95	29.53%
8560 (10%)	1605	Symmetric KL	0.5	71.80	81.15	43.88%
8560 (10%)	1605	Symmetric KL	0.8	**72.03**	**81.43**	58.97%
8560 (10%)	1605	JS	0.2	71.64	81.45	37.84%
8560 (10%)	1605	JS	0.5	**72.53**	**82.08**	54.51%
8560 (10%)	1605	JS	0.8	72.71	81.99	69.18%

distributions.

- Recall that higher α means that more weight is put on the supervised loss in the total objective function. Since increasing α corresponds to the performance increase, this means that our auxiliary loss actually contributes to lowering the model capabilities on the QA task disproving the hypothesis.

Combined Data Augmentation Experiments

In the third set of experiments (Table 4.7), the model combines the two previous approaches minimizing the full supervised loss and the unsupervised loss for each pair of examples.

Table 4.7: Results on SQuaD 1.1 on the combined model with the supervised and unsupervised losses.

Combined Model						
Training Data	**Training Steps**	**Loss**	α	**EM**	**F1**	**Confidence**
8560 (10%)	1605	KL	0.2	70.00	80.03	28.54%
8560 (10%)	1605	KL	0.5	71.03	80.89	44.15%
8560 (10%)	1605	KL	0.8	**71.81**	**81.12**	65.50%
8560 (10%)	1605	Symmetric KL	0.2	69.83	79.88	28.05%
8560 (10%)	1605	Symmetric KL	0.5	70.63	80.42	42.56%
8560 (10%)	1605	Symmetric KL	0.8	**71.84**	**81.27**	66.68%
8560 (10%)	1605	JS	0.2	70.62	80.36	33.79%
8560 (10%)	1605	JS	0.5	71.51	81.18	55.61%
8560 (10%)	1605	JS	0.8	**72.69**	**81.81**	64.45%

Best Model Experiments

Having identified the best hyperparameters for the fully supervised, semi-supervised and the combined models, we now compare them with the original BERT evaluating each model on the paraphrased development set created using back-translation.

Table 4.8: Performance of the all models trained with the highest performing hyperparameters and 10% of the training set (SQuAD 1.1).

Best Models Evaluated on Paraphrased Dataset			
Model	**EM**	**F1**	**Confidence**
Original BERT Large	83.33	88.47	99.96%
BERT Large + DA (supervised)	**87.04**	**92.16**	99.96%
BERT Large + DA (semi-supervised)	88.33	87.28	99.86%
BERT Large + DA (combined)	**85.19**	**90.12**	99.83%

The results are given in Table 4.8.

Key observations:

- The original BERT Large model shows lower performance than the same model trained on the full training set (F1 scores of 88.47% compared to 95.7% in Table 4.2). This is expected since fine-tuning of BERT requires a large amount of training examples. The supervised data augmentation model trained for the same number of training steps, however, outperforms original BERT improving the EM and F1 scores by almost 4%, lowering the gap between the original BERT trained on the full training set and on 10% of the dataset.
- Previously, combined model showed lower performance than the original development set, however, it performs better on the constructed paraphrased set. This may be because the model learned patterns corresponding to the structure of the paraphrased generation method.
- The models are still overconfident showing over 99% confidence on average on all predictions, although the combined model does slightly decrease the confidence.

Table 4.9: Performance of the all models trained with the highest performing hyperparameters and 10% of the training set (SQuAD 1.1).

Best Models Evaluated on Adversarial Dataset				
Original Questions			**Adversarial Questions**	
Model	**EM**	**F1**	**EM**	**F1**
Original BERT Large	**75.80**	**83.54**	**39.80**	**45.14**
BERT Large + DA (supervised)	73.80	81.56	39.40	44.92
BERT Large + DA (semi-supervised)	69.6	78.82	35.7	41.55
BERT Large + DA (combined)	71.40	79.52	35.1	40.53

Now we evaluate the models on the avdersarial daatset with distracting sentences. The results are shown in Figure 4.9. Observe that the fully supervised model performs worse when evaluated both on the original and the adversarial set (although very close in performance in the case with the adversarial set). We hypothesize that this is due to the model overfitting to the language structure and wordings used in the training set. In future work, it may be beneficial to produce adversarial examples on the training set with distracting sentences and retrain the model using those as well.

Chapter 5

Conclusion and Future Work

5.1 Summary of Investigations and Results

In this work, we posed multiple research questions that we attempted to answer with our investigations. In particular, we performed the following:

1. The robustness of BERT Large fine-tuned on the reading comprehension task to questions paraphrases and adversarial examples was investigated. The hypothesis that BERT's ability to surpass human performance on this task was overestimated was supported. We observed that while BERT consistently showed high performance on the SQuAD 1.1 dataset, the performance significantly dropped when the inputs were changed. That is, the model learned useful patterns to be able to correctly locate most of the answers in the original SQuAD 1.1, however, it did not posses high level reasoning and inference skills to show resilience to question paraphrases and passages containing distracting information. Hence, the machine comprehension challenge is far from being solved.

2. We briefly investigated back-translation techniques for paraphrase generation. Using human evaluation, we showed that back-translation with one pivot language that has the same origin as the original language, a method used by multiple NLP approaches, is a poor way of generating paraphrases since a large portion of the input text remains unchanged. In our evaluation, over 75% of input questions were not show substantial changed compared to under 30% resulted from a back-translation

method that uses two pivot languages one of which originates from a different language family. Using two significantly different pivot languages resulted in a smaller number of valid paraphrases. Preserving semantic equivalence of questions and their paraphrased counterparts is essential to our model due to the auxiliary objective.

3. A regularisation method to attempt to improve the robustness of BERT to question paraphrases was introduced. The idea incorporated data augmentation and multitask learning.

   The choice of data augmentation was inspired by the need to enrich the dataset with more diverse examples so that the model would be exposed to more language structures and vocabulary. The choice of multitask learning was made to go beyond simply concatenating more training examples to incorporating an auxiliary objective to the main task. This objective was to feed the original questions and their paraphrased alternatives into the model such that they would result in consistent predictions.

   The hypothesis that the proposed regularisation method would increase robustness over the original method was not supported. We observed an almost 2% increase in the EM and F1 scores when the model was evaluated on the constructed paraphrase set, although it showed lower performance on the original development set. We believe that the model overfitted to the new language structure and wordings introduced by the paraphrased method.

   The hypothesis that the proposed regularisation method would outperform the supervised data augmentation method was not supported either. Supervised data augmentation method outperformed the original method by almost 4%, which is 2% higher than the performance of the combined augmented model.

5.2 Critique and Future Work

In this section we describe limitations of our approaches and suggest potential improvements for future work. There are multiple issues with our proposed augmented model that should be addressed in future work:

1. We used the full development set to evaluated the model's performance on unseen data. A better practice would be to separate it into validation and test sets. This

would allows to perform the following:

- incorporate early stopping into the algorithm to help avoiding overfitting. Early stopping would allow to obtain a model with the lowest validation set error by terminating the training process when the validation loss starts to increase.
- incorporate k-fold cross-validation when performing hyperparameter tuning.

2. In order to construct more diverse examples, it would be beneficial to use more complex paraphrase generation methods other than back-translation.
3. Instead of producing only one paraphrased input, in the future we can produce multiple augmented examples and change the auxiliary loss to be a weighted sum of pairwise losses.
4. We used a fixed α to weight the losses, however, incorporating weight annealing such that α starts being closer to 1 and slowly decreases approaching 0 as the model makes more predictions can be done in the future work.

Bibliography

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[BFS93] Yoshua Bengio, Paolo Frascouni, and Patrice Simard. "The Problem of Learning Long-term Dependencies in Recurrent Networks". In: *IEEE International Conference on Neural Networks* (1993), pp. 12183–1195.

[Bos14] Nick Bostrom. *Superintelligence: Paths, dangers, strategies.* 2014.

[Bro+92] Peter F. Brown et al. "Class-based N-gram Models of Natural Language". In: *Computational Linguistics* 18.4 (1992), pp. 467–479.

[Car93] Rich Caruana. "Multitask Connectionist Learning". In: *Connectionist Models Summer School.* 1993, pp. 372–379.

[Che+17] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: *arXiv preprint arXiv:1704.00051* (2017).

[CDL16] Jianpeng Cheng, Li Dong, and Mirella Lapata. "Long Short-Term Memory-Networks for Machine Reading". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.* Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 551–561. DOI: 10.18653/v1/D16-1053.

[Dev+18] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[GN19] Wee Chung Gan and Hwee Tou Ng. "Improving the Robustness of Question Answering Systems to Question Paraphrasing". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6065–6075.

[Geh+17] Jonas Gehring et al. “Convolutional Sequence to Sequence Learning”. In: *arXiv preprint arXiv:1705.03122* (2017).

[Gu+17] Jiuxiang Gu et al. “Recent Advances in Convolutional Neural Networks”. In: *arXiv preprint arXiv:1512.07108* (2017).

[Has+17] Demis Hassabis et al. “Neuroscience-Inspired Artificial Intelligence”. In: *Neuron* 95 (2017), pp. 245–258.

[HS97] Sepp Hochreiter and Jurgen Schminhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (8 Sept. 1997), pp. 1735–1780.

[JL17] R. Jia and P Liang. “Adversarial Examples for Evaluating Reading Comprehension Systems”. In: *arXiv preprint arXiv:1707.07328* (2017).

[Jos+17] Mandar Joshi et al. “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada: Association for Computational Linguistics, July 2017.

[KSS18] Alec Radford anda Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. “Improving Language Understanding by Generative Pre-Training”. In: *arxiv*. 2018.

[Kwi+19] Tom Kwiatkowski et al. “Natural Questions: a Benchmark for Question Answering Research”. In: *Transactions of the Association of Computational Linguistics* (2019). URL: `https://tomkwiat.users.x20web.corp.google.com/papers/natural-questions/main-1455-kwiatkowski.pdf`.

[LeC+90] Yann LeCun et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1990, pp. 396–404. URL: `http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf`.

[Lin91] Jianhua Lin. “Divergence Measures Based on the Shannon Entropy”. In: *IEEE Transactions on Information Theory* 37.1 (Jan. 1991), pp. 145–151.

[Liu+18] Peter J. Liu et al. “Generating Wikipedia by Summarizing Long Sequences”. In: *arXiv preprint arXiv:1801.10198* (2018).

[Liu+19a] Xiaodong Liu et al. “Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding”. In: *arXiv preprint arXiv:1904.09482* (2019).

[Liu+19b] Xiaodong Liu et al. “Multi-Task Deep Neural Networks for Natural Language Understanding”. In: *arXiv preprint arXiv:1901.11504* (2019).

[McC+18] Bryan McCann et al. “The Natural Language Decathlon: Multitask Learning as Question Answering”. In: *arXiv preprint arXiv:1806.08730* (2018).

[Mik+13] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada, 2013, pp. 3111–3119.

[NSS58] Allen Newell, J. C. Shaw, and Herbert A. Simon. “Chess-Playing Programs and the Problem of Complexity”. In: *IBM Journal of Research and Development* 2 (1958), pp. 320–335.

[Pan+19] Liangming Pan et al. “Recent Advances in Neural Question Generation”. In: *arXiv preprint arXiv:1905.08949* (2019).

[Pet+18] Matthew E. Peters et al. “Deep contextualized word representations”. In: *Proc. of NAACL*. 2018.

[RJL18] Pranav Rajpurkar, Robin Jia, and Percy Liang. “Know What You Don’t Know: Unanswerable Questions for SQuAD”. In: *arXiv preprint arXiv:1806.03822* (2018).

[Raj+16] Pranav Rajpurkar et al. “Squad: 100,000+ questions for machine comprehension of text”. In: *arXiv preprint arXiv:1606.05250* (2016).

[RTG98] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. “A metric for distributions with applications to image databases”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)* (1998), pp. 59–66.

[RHW88] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Neurocomputing: Foundations of Research”. In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. Learning Representations by Back-propagating Errors, pp. 696–699. ISBN: 0-262-01097-6.

[SPG97] Mike Schuster, Kuldip K. Paliwal, and A. General. “Bidirectional Recurrent Neural Networks”. In: *IEEE Transactions on Signal Processing* (1997).

[Seo+17] Minjoon Seo et al. “Bidirectional Attention Flow for Machine Comprehension”. In: *arXiv preprint arXiv:1611.01603* (2017).

[Sug+18] Saku Sugawara et al. “What Makes Reading Comprehension Questions Easie”. In: *arXiv preprint arXiv:1808.09384* (2018).

[Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5998–6008. URL: `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`.

[Wai+89] Alex Waibel et al. “Phoneme Recognition using Timedelay Neural Networks.” In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37(3) (1989).

[Wan+18] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *arXiv preprint arXiv:1804.07461* (2018).

[Wan+19] Alex Wang et al. “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems”. In: *arXiv preprint arXiv:1905.00537* (2019).

[Xie+19] Qizhe Xie et al. “Unsupervised Data Augmentation”. In: *arXiv preprint arXiv:1904.12848* (2019).

[Xu+19] Yichong Xu et al. “Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension”. In: *arXiv preprint arXiv:1809.06963* (2019).

[Yu+18] Adams Wei Yu et al. “QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension”. In: *CoRR* abs/1804.09541 (2018). arXiv: `1804.09541`.