Sentence Encoding Prescreening for Q&A

W266 Spring 2019 Friday Class
Instructor: Mark Butler

Team Members:
Jason Kang
Sonal Thakkar

Abstract

In this paper, we propose a conceptual design for Question and Answer (Q&A), which when used on top of the existing BERT based models can significantly reduce the overall amount of computation. In addition, we shall code 4 prototypes from scratch and prove the feasibility of the concept. We will also share our findings and lessons learnt which we hope will be useful to readers and fellow students who wish to explore the amazing realm of NLP.

Introduction:

Q&A is one of the NLP downstream tasks which is under rapid development and has drawn much attention over the past few years. Now it is a fairly well researched area, and we have seen that the state of art Q&A algorithms developed on BERT (ALBERT + DAAF + Verifier w/ ensemble) scored a staggering 92.77 F1 score on SQUAD leaderboard, a performance matching that of a human reader. So, what is the next research direction? We believe, the paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations" has provided the answer -- to downsize the existing model, making it smaller and faster so that it can run on more devices. In this paper, we propose a conceptual design that can be added on top of the current bert based Q&A solutions and potentially reduce the amount of computation to a fraction of that of a stand alone BERT based model.

Dataset:

For the purpose of proving the feasibility of our proposal, we used version 1.1 to avoid unnecessary complication. (*Refer to Appendix A for more details about the dataset*).

Understand BERT-based Q&A models:

To better appreciate how our proposal can make BERT based Q&A models more efficient, it is necessary to first understand the nature of BERT. As illustrated in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", BERT is a word token based Encoder (also refer to "Attention Is All You Need") model. It is important to note that in the Q&A setting, every single word token of the entire article is passed through the many layers of the BERT encoders to eventually generate the predicted answer which can be simply located within a single sentence. Therefore, if we can do some efficient prescreening to find out the exact sentence that contains the answer, we can feed the BERT based model with only one particular sentence, and eliminate the redundant computations on other word tokens.

Our concept -- SEP:

We propose a conceptual add-on design to tackle the inefficiencies aforementioned -- Sentence Embedding Prescreening (SEP). [It should be noted that during our subsequent research, we found that part of a model presented in paper "Efficient and Robust Question Answering from Minimal Context over Documents"

somewhat related to SEP, though it has a different focus and adopts a different architecture.] Our concept of the light add-on model, SEP, aims to identify the particular sentence that contains the answer, in order to significantly reduce the amount of the input to the subsequent BERT based models.

One paragraph/article may consist of hundreds or even thousands of word or subword tokens, however, it consists of only a few sentences. Thus our prescreening concept adopts sentence embedding instead of token embedding as input to reduce the computation to a fraction of the original amount. The following chart describes the number of sentences in each paragraph for the training data of SQuAD 1.1. On average there are 5.1 sentences per paragraph (Figure 1). Hence, by applying SEP, the input to the subsequent BERT based span classifier can be reduced by an estimated 80.6%. It is noted that in SQuAD's data structure, each question is only associated with one paragraph. In real life, a question may be related to an article consisting of multiple paragraphs. Therefore, the efficiency boost will be even higher.

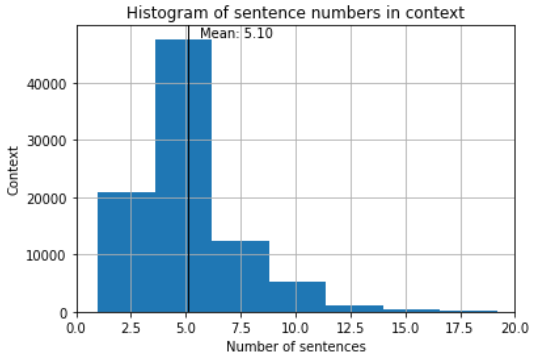

Figure 1: Number of sentences in each paragraph.

In the following sections, we will present the 4 prototypes developed to prove the feasibility of SEP. At the same time, we will also present our development stages and some useful findings.

Baseline:

We use the most common class as our baseline. From the training data, we identified that the 1st sentence of each article is most likely to contain the answer. (Figure 2 below shows a histogram of the SQuAD 1.1 training dataset sentences that contains the answer for each question and context). Based on this class, the baseline scored 32% accuracy on the development data.

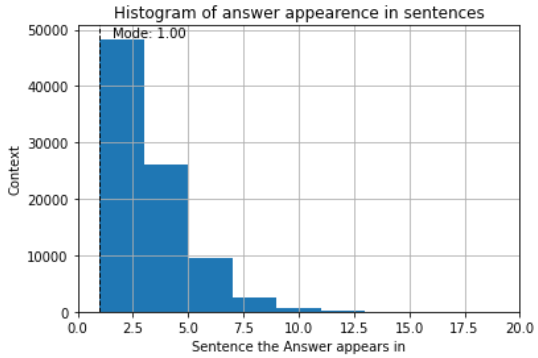

Figure 2: Sentence 1 is most likely to contain the answer.

Prototype 1 -- Transformer Architecture:

Transformer architecture (refer to "Attention Is All You Need") is arguably one of the most influential breakthroughs in NLP since 2017. Hence we placed our first bet on the transformer design. The following is the architecture we implemented:

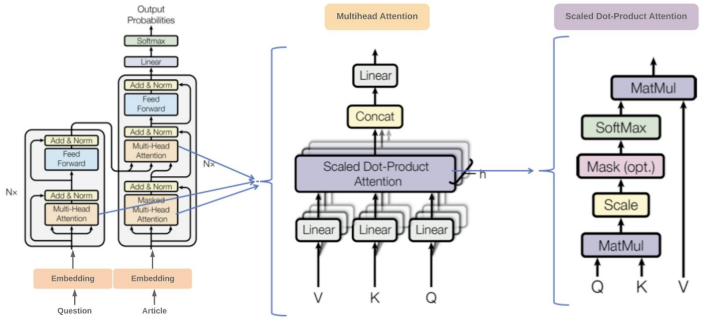

Figure 3: Prototype 1 architecture.

To speed up training and development, we adopted a simple sentence encoder "Wiki-words-250-with-normalization" from TensorFlow Hub. In the above architecture, we feed the question embedding through the encoder, and we feed the paragraph sentence encodings through the decoder. We then feed the results of the sentence vectors returned by the decoder to a dense layer. Subsequently, the results are reshaped and passed through a softmax layer to represent the probability of each sentence containing the correct answer.

The first prototype scored only 24.8% accuracy on the development data, which is worse than the baseline. Upon careful examination, we obtained an interesting finding -- the order of input for the transformer architecture is directional. In other words, if we swap the encoder and decoder inputs, the resulting performance will be totally different. This property is the result of the 2nd tier multi-head attention block in the decoder. The following is a simple mathematical illustration. Note that the formula is denominated in matrix dimensions instead of values so as to make it easier to visualize.

The result of the 1st tier multi-head attention in the decoder produces the article sentence matrix using self-attention mechanism.

Article Sentence Vectors : $\mathbf{N}_{\text{article\_sentences}}$ X **d_model**

Where $\mathbf{N}_{\text{article\_sentences}}$ represents the number of sentences in a particular article, and **d_model** is the dimension of the sentence encoding.

Encoded Question Vector: **1** X **d_model**

In the 2nd tier multi-head attention block, the encoded question vector together with the article sentence vectors form the input. There are two major types of computations in the 2nd tier multi-head attention block:

(1) Attention weights $= \text{softmax}\left(\text{Matmul}\left((\mathbf{N}_{\text{article\_sentences}} \text{ X } \mathbf{d\_model}), (\mathbf{1} \text{ X } \mathbf{d\_model})^T\right)\right)$
$= \mathbf{N}_{\text{article\_sentences}} \text{ X } \mathbf{1}$

(2) Decoded vectors $= \text{Matmul}\left((\mathbf{N}_{\text{article\_sentences}} \text{ X } \mathbf{1}), (\mathbf{1} \text{ X } \mathbf{d\_model})\right)$
$= \mathbf{N}_{\text{article\_sentences}} \text{ X } \mathbf{d\_model}$

From the above two calculations, we realize that when we feed the question into the encoder and paragraph into the decoder, the model cannot make use of the attention mechanism, instead, it merely performs dot product between the question and article vectors. This is evidenced by the result in (1). The dimension of the attention weights is 1, thus, after the softmax, the dimension (representing the question in this case) will always receive 100% attention. This completely defeats the purpose of the attention mechanism, as we want no single component to receive all attention, and also we want the article sentence instead of the question sentence to receive attention. The above illustration explains why prototype 1 failed.

Prototype 2 -- Transformer Architecture + attention weights output:

To fix the issue identified above, we produced Prototype 2, which uses articles as input for the encoder, and questions as input for the decoder. In addition, we used the attention weights instead of the resulting vectors as the answer classifier, so that the attention weights can be directly used as the prediction output, and its shape is dynamic allowing it to fit articles of various length.

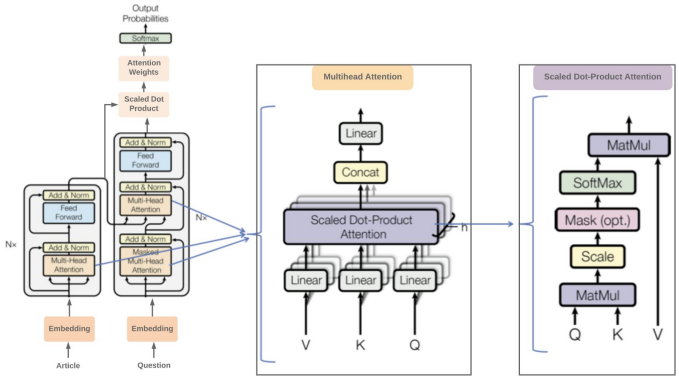

Figure 4: Prototype 2 architecture.

As we expected, upon fixing the prior issue identified, the accuracy is pushed to 56.00% on the training data and 56.41% on the development data, an almost 125% boost compared to Prototype 1, and 75% boost to the baseline performance. Prototype 2 sets us on the right track. However, we wonder if a full transformer architecture is necessary. By referencing BERT's implementation, we noted that BERT is able to achieve state of art performance with only encoder blocks. The fact suggests that some components in the transformer architecture are playing a more important role than the rest of the components in the Q&A setting.

The above hypothesis can be mathematically proven:

Computing in the encoder:

(1) **Attention_weight_art**= Softmax(Matmul (**Article_Matrix_a**, **Article_Matrix_b**^{T}))

(2) **MHA_output_art** = Matmul (**Attention_weight_art** , **Article_Matrix_c**)

(3) **Encoder_output_art** = Normalize (f_{Dense} (**MHA_output_art**) + **MHA_output_art**)

Where Article_Matrix_a, Article_Matrix_b and Article_Matrix_c are the three variations of the article

embedding produced by three different layers of linear transformation in the encoder.

Computing in the decoder:

(4) **Attention_weight_ques**= Softmax(Matmul (**Question_Matrix_a**, **Question_Matrix_b**^{T}))

(5) **MHA_output_ques** = Matmul (**Attention_weight_ques** , **Question_Matrix_c**)

(6) **Norm_ques** = Normalize (f_{Dense} (**MHA_output_ques**) + **MHA_output_ques**)

(7) **Attention_weight_comb**=Softmax(Matmul (**Question_Matrix_a**, **Encoder_output_art**^{T}))

(8) **MHA_output_comb** = Matmul ((**Attention_weight_comb**, **Encoder_output_art**)

(9) **Decoder_output** = Normalize (f_{Dense} (**MHA_output_comb**) + **MHA_output_comb**)

Where Question_Matrix_a, Question_Matrix_b, and Question_Matrix_c are the three variations of the question embedding produced by three different layers of linear transformation in the decoder.

It is straightforward to see that the entire encoder [(1) to (2)] processed self-attention only to the article, and the first multi-head attention block [(4) to (5)] in the decoder processed self-attention only to the question. The only interaction between the question and article takes place in the second multi-head attention block [(7) to (8)]. Therefore, the 2nd multi-head attention block in the decoder is identified as the core component in this Q&A setting. We will further prove this theory by implementing an extremely light Prototype 3 with only the 2nd multi-head attention block in the decoder structure.

Prototype 3 -- Single Layer Attention Block:

In Prototype 3, we downsized the Prototype 2 by getting rid of all non-essential components as aforementioned; keeping only the multi-head attention, as illustrated in the following flow-chart:

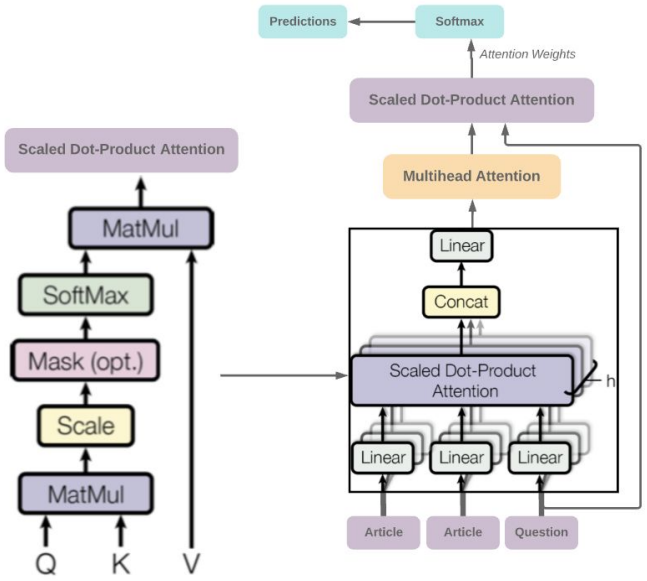

Figure 5: Prototype 3 architecture.

In Prototype 3, we continue using single dot attention as the answer classifier with attention weights as the prediction result. The downsized Prototype 3 achieved an accuracy of 50% on development data which is fairly close to the result of the full transformer architecture as implemented in Prototype 2. Further, Prototype 3 achieved this performance with only a fraction of the computing power as used in Prototype 2. This result proved that our mathematical reasoning is correct and that the attention mechanism is the core component enabling the Q&A. From this trial we learnt an important lesson that complex architecture is not necessarily superior to simple architecture, and we have to identify the core component that is really helpful to achieve the goal.

Despite the above achievement, we noted that Prototype 3 converges significantly slower than Prototype 2. It takes more than 60 epochs for Prototype 3 to converge, whereas it takes less than 10 epochs for Prototype 2 to converge. We believe that the normalization layer in Prototype 2 makes the model converge faster. The theory is well illustrated in paper "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", and we shall not repeat the reasoning here. As a result, we decided to apply the batch normalization in the next prototype.

In addition, by examining the implementation of BERT, we found that BERT heavily leverages self-attention, whereas in our Prototype 3, we used paragraph and question as seperate inputs to feed the multi-head attention. Hence, we wonder whether self-attention of the combined input is superior to the attention using question and article as separate inputs. To find out, we conducted the following mathematical analysis:

Prototype 3 mechanism:

(1) **Attention_weight_proto**=Softmax(Matmul ($f_{Linear0}$(**Question_Matrix**), $f_{Linear1}$(**Article_Matrix**)^T))
(2) **MHA_output_proto** = Matmul (**Attention_weight_proto**, $f_{Linear2}$(**Article_Matrix**))

BERT mechanism:

(3) **Attention_weight_BERT**= Softmax(Matmul ($f_{Linear3}$(**Combined_Matrix**), $f_{Linear4}$(**Combined_Matrix**)^T))
(4) **MHA_output_BERT** = Matmul (**Attention_weight_BERT**, $f_{Linear5}$(**Combined_Matrix**))
(5) **BERT_output** = Normalize (f_{Dense} (**MHA_output_BERT**) + **MHA_output_BERT**)

The above formula reveals two major differences between BERT's implementation of the multi-head attention and that of ours. First, in (3), the question and article as one combined input undergoes a dot multiplication with a variation of itself (It is a variation because the question + article embeddings are transformed using a different dense layer). In this process each article sentence embedding is also multiplied by the transposed variation of other sentence embeddings. In other words, the attention weights thus derived consists of the relationship not only between the question and the article but also among the article sentences themselves. Whereas in (1), the relationship is only between the article and the question. Second, in (3) and (4), we noticed that the same question and article pair is transformed three times using three different linear transformations; whereas in (1) and (2) the article is only transformed twice and the question is transformed using a linear transformation different from that performed on the article.

The above mathematical examination prompts us to think that the creation of additional relationships among the article sentences, the adoption of the same linear transformation to the article and the question embedding, and the additional variation of linear transformations performed on the input embeddings played an important role making BERT state-of-the-art. However, it is not straightforward to mathematically prove the impact of these differences. Therefore, we decided to design Prototype 4 to test it out.

Prototype 4 -- Encoder Architecture:

In Prototype 4, we implemented several changes. First, we added batch normalization layers as discussed above to help the training process converge faster. Second, we combined the question and article sentence embedding as one input to leverage on self-attention, and we transformed the input three times using three different linear transformations. The architecture implemented is as following:

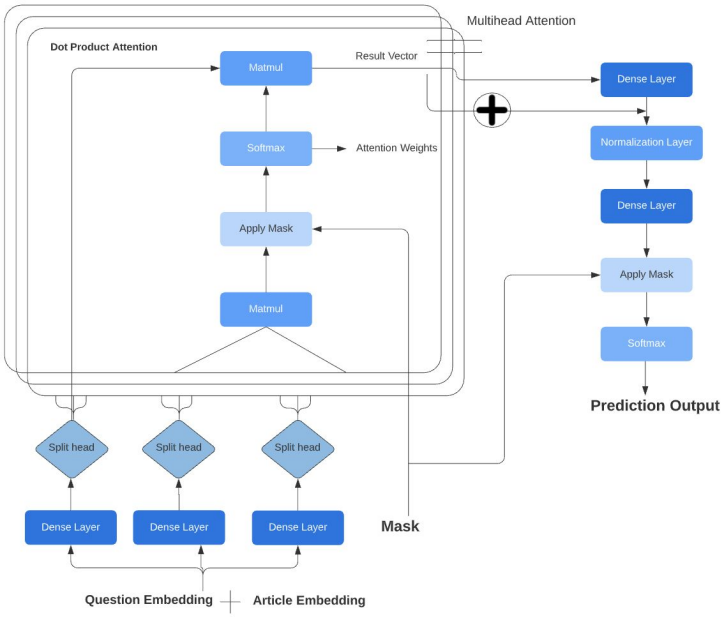

Figure 6: Prototype 4 architecture.

The normalized output is reshaped and fed into a [d_model, 1] dimension dense layer. The results are reshaped back and passed through a softmax layer to produce predictions indicating the probability that each sentence contains the answer.

The new prototype converges significantly faster than Prototype 3, proving the effectiveness of the batch normalization in terms of speeding up the training process. It takes only 4 epochs to converge instead of more than 60 epochs for Prototype 3. In addition, Prototype 4 scored 73.6% accuracy on training data and 73.5% accuracy on development data representing a 47% relative performance boost, and 23.5% absolute increase. The leap in accuracy proved our prior suspicion that the two major differences are what makes

BERT based models outstanding in downstream Q&A tasks. The performance of Prototype 4 also serves as a strong evidence to prove the feasibility of our proposed concept - **SEP**.

However, Prototype 4 is still far from perfect, there are many areas that can be further improved. First, we used a humble 250 dimension sentence embedding developed based on wiki articles, whereas there are many pre-trained sentence encoders that offer embedding in higher dimensions, such as universal sentence encoder which offers 512 dimension embedding and is trained on an enormous amount of text. We expect to see further improvement when a better embedding layer is used. Second, by expanding the amount of variables used, it is likely to achieve better results. The impact of the number of variables on model performance has been well discussed in paper "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", which we will not repeat here. Third, by implementing beam search which retains top k choices, there is a higher chance of scoring a significant performance boost, however, at the cost of increased amount of input to subsequent Q&A models. To test these out we further conducted four more trials based on the architecture in Prototype 4. For instance, we explored a light version of the Universal Sentence Encoder and a large version of the Universal Sentence Encoder instead of the 250-dimension wiki embedding; increased the amount of variables by using 8 encoder layers instead of 4, and added a beam layer of k=2 on top of the predictions. All showed continued improvements. Here we present a summary of all prototypes, trials and results:

Performance Summary -- All models

Model	Architecture	Remarks	Training Accuracy	Development Accuracy
Baseline	Most common class		31.9%	32.0%
Prototype 1	Full Transformer w/ 4-layer design	Question injected into encoder and article injected into the decoder	27.3%	24.8%
Prototype 2	Full Transformer w/ 4-layer design	Article injected into encoder and question injected into the decoder	56.0%	56.41%
Prototype 3	Single-layer Multi-head Attention	Question and Article used as separate input	52.6%	50.0%
Prototype 4	4-Layer Encoder using MHA	Question and Article used together as one input	73.6%	73.5%
Additional trial A	4-Layer Encoder using MHA	Used light version of Universal Sentence Encoder	77.8%	77.5%
Additional trial B	8-Layer Encoder w/ 8-head MHA	Used large version of Universal Sentence Encoder	80.2%	81.2%
Additional trial C	8-Layer Encoder w/ 16-head MHA	Used large version of Universal Sentence Encoder	84.1%	83.0%
Additional trial D	8-Layer Encoder w/ 16-head MHA + Beam k=2	For articles containing more than 2 sentences, select top 2 output	-	**92.0%**

We trust that there are many other potential ways to better implement the SEP concept and push the accuracy score even higher, for example, by using the 5th layer output of ELMO which offers 1024 dimension sentence encoding (the 5th layer of ELMO is a pooling layer of the word embeddings. ELMO may be a better choice not because of its pooling design, but because of its contextualized word embeddings and the larger dimension which allows it to possible capture richer context), by implementing the encoder architecture in a more refined manner such as using customized leaky Relu instead of the standard Relu activation function, by using non-linear transformation for sentence embeddings instead of linear transformation, and by exploring other architectures etc. We could not exhaust all potential experiments partly because of the project deadline and hardware constraints, and partly because the goal of our paper is to prove the feasibility of the SEP concept, which is already achieved.

Conclusion:

In this paper, we proposed a conceptual design, SEP, which when used on the existing token based Q&A models can improve the overall efficiency by an estimated 80.6%; the efficiency boost can be even higher when used on longer articles. We further proved the feasibility of the SEP concept by the four prototypes and the additional four trials. When SEP is used concurrently with other BERT based models, the overall accuracy can be measured by $\mathbf{P}_{SEP} \times \mathbf{P}_{BERT\_QA|SEP}$. We want to further highlight that there is a possibility that $\mathbf{P}_{BERT\_QA \mid SEP}$ may be higher than $\mathbf{P}_{BERT\_QA}$, because the SEP add-on will eliminate the irrelevant sentences and narrow the input range for BERT based models making them impossible to pick wrong answers outside the narrowed input range. We did not design experiments to test this point, and the fun shall be left for the readers and fellow students to explore. We hope that this paper can inspire more data scientists and students working on SEP or similar ideas to make current Q&A models more efficient, so that more devices can benefit from the state-of-art performance without using the state-of-art hardware.

References:

1. **SQuAD: 100,000+ Questions for Machine Comprehension of Text**
   https://arxiv.org/pdf/1606.05250.pdf

2. **ALBERT: A Lite BERT for Self-supervised Learning of Language Representations**
   https://arxiv.org/pdf/1909.11942.pdf

3. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**
   https://arxiv.org/pdf/1810.04805.pdf

4. **Attention Is All You Need**
   https://arxiv.org/abs/1706.03762

5. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**
   https://arxiv.org/abs/1502.03167

6. **Efficient and Robust Question Answering from Minimal Context over Documents**
   https://arxiv.org/pdf/1805.08092.pdf

7. **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**
   https://arxiv.org/pdf/1910.10683.pdf

8. **ELMo helps to further improve your sentence embeddings**
   https://towardsdatascience.com/elmo-helps-to-further-improve-your-word-embeddings-c6ed2c9df95f

Appendix A: Dataset

The **S**tanford **Q**uestion **A**nswering **D**ataset (SQuAD) is a labelled reading comprehension dataset comprising thousands of paragraphs of Wiki articles and relevant questions and answers posed by crowdworkers. Data science community is tasked to produce algorithms that can identify the answers based on the given article and question. Currently, SQuAD has two versions -- 1.1 and 2.0. The difference is that, in version 2.0, some questions may not have an answer.

Appendix B: Pre-processing Training Dataset for SEP Models

In SQuAD dataset, the article is given in format of string, and the start of the answer span is denoted as an integer indicating the location index in the article string. To prepare training data for SEP models, the challenge is to convert the location index of the answer span to an index indicating the number of the sentence that contains the answer.

To achieve that, we first insert a special token at the answer location index. Second, we use Punkt Sentence Tokenizer from NLTK library to convert the article into a list of sentences. Last, we locate the sentence which contains the special token inserted. The sentence number then becomes the target value for the purpose of training the SEP models.

Appendix C: Prototype 4 predictions

The followings are some illustrations of the Prototype 4 predictions:

Example 1:

Article:

1."Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season."

2. "The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title."

3. "The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California."

4. 'As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.'

Q&A:

1. 'Which NFL team represented the AFC at Super Bowl 50?'
   Prediction **2**
   Target **2**

2. 'Which NFL team represented the NFC at Super Bowl 50?'
   Prediction **2**
   Target **2**
3. 'What venue did Super Bowl 50 take place in?'
   Prediction **3**
   Target **3**
4. 'What team was the champion?'
   Prediction **2**
   Target **2**

Example 2:

Article:

1.'I went to work in the morning and went home at night.'

2. 'I like to eat chocolate, but I hate to eat bananas.'

Q&A:

1. 'What did I do in the morning?'
   Prediction **1**
   Target **1**
2. 'When did I go to work?'
   Prediction **1**
   Target **1**
3. 'When do I like to eat?'
   Prediction **2**
   Target **2**
4. 'When do I hate?'
   Prediction **2**
   Target **2**

Example 3:

Article:

1.'I had hot dogs for breakfast.'

2. 'I ate steak for dinner.'

Q&A:

1. 'What did I have for dinner?'
   Prediction **2**
   Target **2**
2. 'What did I eat in the morning?'
   Prediction **1**
   Target **1**

Example 4:

Article:

1. 'A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used.'

2. 'The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying the amount of resources needed to solve them, such as time and storage.'

3. 'Other complexity measures are also used, such as the amount of communication (used in communication complexity), the number of gates in a circuit (used in circuit complexity) and the number of processors (used in parallel computing).'

4. 'One of the roles of computational complexity theory is to determine the practical limits on what computers can and cannot do.'

Q&A:

1. 'What measure of a computational problem broadly defines the inherent difficulty of the solution?'
   Prediction **1**
   Target **1**
2. 'What method is used to intuitively assess or quantify the amount of resources required to solve a computational problem?'
   Prediction **2**
   Target **2**
3. 'What are two basic primary resources used to gauge complexity?'
   Prediction **3**
   Target **2**
4. 'What unit is measured to determine circuit complexity?'
   Prediction **3**
   Target **3**

*Note that the above error can be addressed by selecting top two predictions instead of only the top one.