CITS4012 QA

Evan Willcocks (22979772) and Nathan Townshend (22970882)

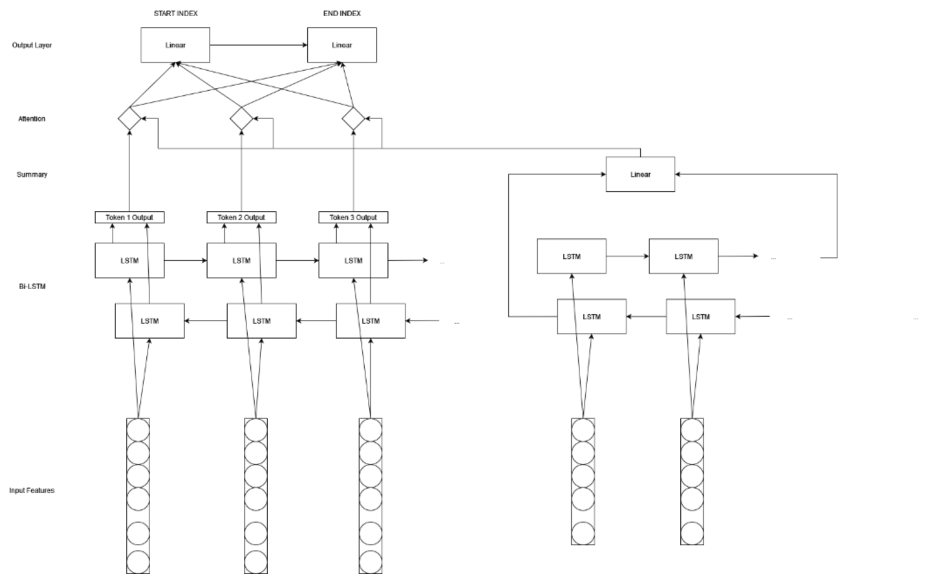
1 Dataset

This project uses a modified WikiQA Corpus, accessed from the assignment specification on GitHub [1]. Basing our processing on the procedure from lab 6, we processed both the documents and questions by removing contractions and punctuation, tokenising them, and padding them with a special pad character. The documents were padded to the maximum length of the documents in the training set, and likewise for the questions. Once tokenised, a list of all vocabulary used across all documents and questions was made and the tokens were processed into the corresponding indices of the token in the vocab list. This has been the standard procedure for preparing the documents in all of the labs. As used in lab 6, torch's nn.Embedding was used later within the model to convert the tokens from the vocab index to a word embedding. This conversion was done within the model using the embedding to reduce the memory that needed to be allocated to the model data.

The target output for the model was initially a tagged version of each token in the document. We initially tried with the {Start Token of the Answer, End Token of the Answer, Inner Token of the Answer, Other} set of tags. Due to the large number of 'other' tags present in the targets, the model would typically produce outputs consisting of just the 'other' tag, producing 'good' results according to the metrics but never actually marking a 'start' or 'end' of the answer. After training for many epochs, it would eventually also tag some tokens as 'inner' but continued to not produce 'start' or 'end' labels. Recognising that it may be difficult for a model to be incentivised to produce the rare 'start' and 'end' tokens, we then attempted the {Before the Answer, In the Answer, After the Answer} set of tags (with questions that had no answer being labelled entirely as 'After the Answer'). As each tag occurred many times within each target, the model performed better than the previous tag set. However, due to the padding causing a larger number of 'After the Answer' tags than the others, there was a bias for the model to select the 'after' tag over the others. To prevent one set of tags from having a bias over the others, we chose to modify the output of the model to be similar to Seo (et al.)'s BiDAF model [2], with now just 2 output values; the start and end index of the answer tokens within the document tokens. The changes that were made to our model and the final structure we used are discussed in second section of this report.

As questions with multiple correct answers are not compatible with a model that outputs only exactly 1 answer per document and question (such as this model that outputs only the start and end indices of the answer, instead of the tag-based output that can have multiple blocks tagged as a potential answer), the first answer within the document was selected as the correct answer. Questions with no answer were labelled with start and end indices of -1.

2 Sequence QA Model



The structure of the QA Model we used based on the one in the project description [1] and the one described on the fourth page of Chen (et al.) 2016 [3] – with a Bi-LSTM layer to encode the document (left) and question (right), a layer to summarise the question into a single (vector) value, and an attention layer weighing the document tokens. As mentioned in the first section of the report, we modified this to produce a start index and end index of the answer within the document by passing the result of the attention to two linear layers [2]. The start index is first calculated, and then passed in addition to the attention results to the end index layer.

The input features for both documents were based on the tokens generated in the data preparation step. GloVe word embeddings were used like in both papers mentioned, although due to hardware considerations only 25-dimensional embeddings were used instead of larger ones such as the 100-dimensional embeddings used in [3]. A variety of features were extracted from the dataset and tested (as discussed in the following section of the report), such as PoS Tags, TF-IDF, NER, and Word-Matching. They were evaluated for each token in the document or question (except for Word-Matching, which was only evaluated for the tokens in the document), and when a model used a feature, the value for each token was concatenated on to the word embedding before being passed to the model. From the results of the tests, using no features (except for the word embedding) for each token yielded the highest values on the metrics we tested with, and so no features were selected for the final model.

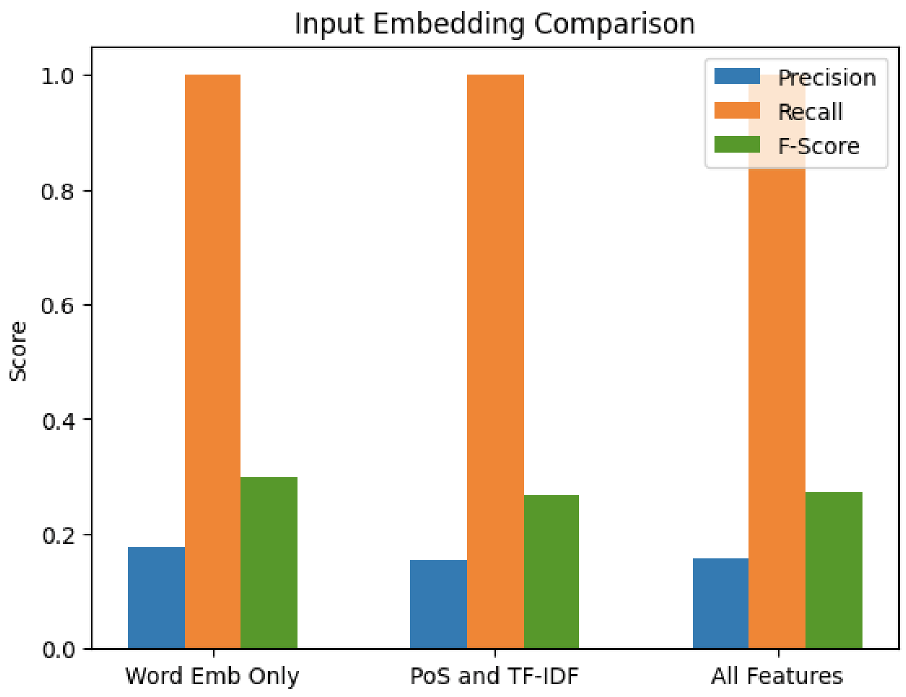
3 Model Testing

3.1 Input Embedding Ablation Study

In this section we studied the performance of the model with different variations of input embeddings. We trained and evaluated one model with only the word embeddings as input. Another with the word embeddings as well as Part of Speech tags (tag encodings) and Term Frequency – Inverse Document Frequency values. And the final model took all features as input (PoS tags, TF-IDF, NER, Word Match).

Each model also used dot product attention and trained for 50 epochs.

We can see from the generated bar plot below that the separate models performed similarly, however the model with word embeddings only performed best, followed by the model with all features. This result is likely partly due to the poor encoding of our features, particularly the PoS tags and NER tags. We encoded these features by converting their tags to indexes. This form of encoding is clearly of low dimension (one integer) and so they most likely aren't representative of the underlying words. Therefore, some of the feature encodings may have caused this negative impact on the model.

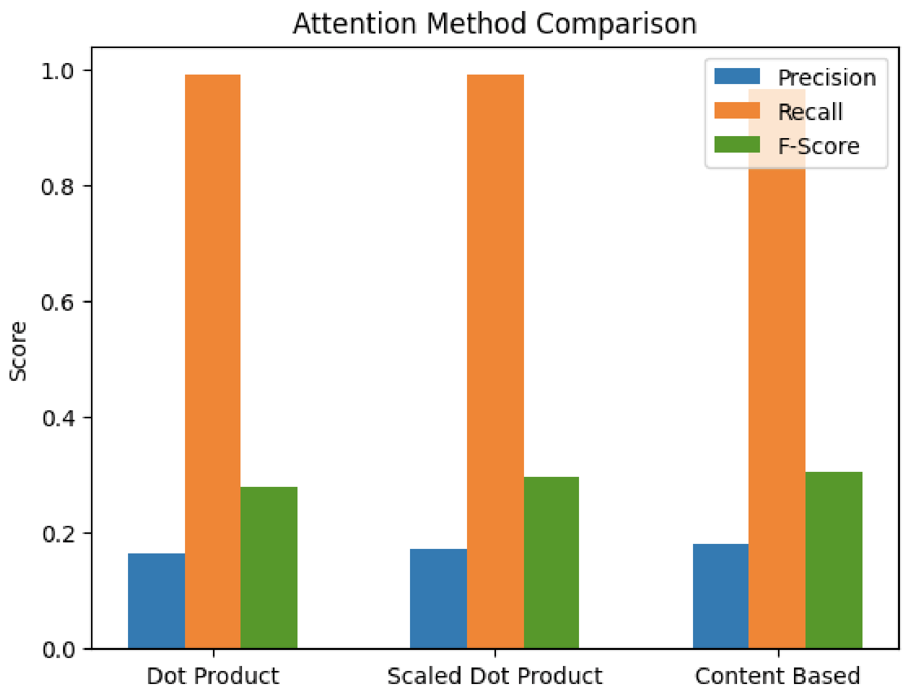


3.2 Attention Ablation Study

In this section we studied the performance of the model with different attention calculation methods including dot product, scaled dot product, and content-based (cosine similarity).

Each model was given only the word embeddings and trained for 50 epochs.

From the bar plot, we can see that the content-based method performed the best, followed by scaled dot product. This makes sense as it uses cosine similarity which is more useful for understanding the similarity between the context and the summary as it considers vector orientations rather than just magnitude.

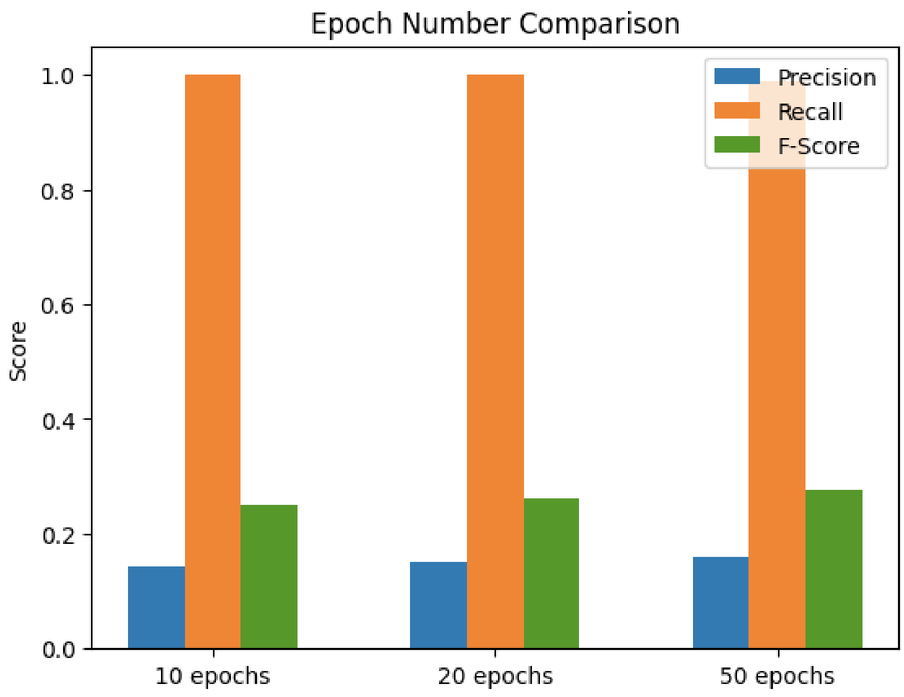


3.3 Hyper Parameter Testing

In this section we studied the performance of the model with different variations of total training epochs.

The model was given only word embeddings and used dot product attention.

We can see that the 50 epoch model performed better. This is intuitive as it had more iterations to learn over, so naturally it should perform better.



4 References

1. adlnlp. *CITS4012 Assignment Specification*. 2023 [cited 2023 20/05/2023]; Available from: https://github.com/adlnlp/CITS4012_2023.
2. Seo, M., et al., *Bidirectional Attention Flow for Machine Comprehension.* 2016.
3. Chen, D., J. Bolton, and C.D. Manning, *A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task.* 2016.