CITS-4012: Wiki Question & Answering Framework

Daniel Tan (22684196), Finn Murphy (22237879), Nicholas Choong (21980614) and Tom Walker (21979744)

[1] University of Western Australia, Australia
22684196@student.uwa.edu.au
22237879@student.uwa.edu.au
21980614@student.uwa.edu.au
21979744@student.uwa.edu.au

1 Dataset

The dataset used for this project is a portion of the Microsoft Research WikiQA Corpus [1]. For the purpose of this project, the corpus used contains 2,747 questions, 26,463 candidate answers, and 1,330 correct answers sampled from Bing query logs. Based upon user clicks, each question is linked to the summary paragraph of a Wikipedia page. Each sentence of this summary paragraph is considered to be a candidate answer to the question. The correct answer to the question (if present) has been labelled manually (Label=1), however only 48.4% of questions have a correct answer.

The dataset was divided into a training set and a test set, a summary of which is provided in the table below.

	Training Set	**Test Set**	**Total**
Dimension	7 x 20,347	7 x 6,116	7 x 26,463
Unique Questions	2,117	630	2,747
Answers	20,347	6,116	26,463
Correct Answers	1,039	291	1,330
Incorrect Answers	19,308	5,825	25,133

Unique questions from both the training and test set were extracted. Document ID's and correct answers for each unique question were then extracted and combined into a dataframe (Figure 1A). If no correct answer was present for a question, then the correct answer was listed as 'No answer'.

	Question	DocumentID	Answer
0	how are glacier caves formed?	D1	A glacier cave is a cave formed within the ice...
1	How are the directions of the velocity and for...	D2	No answer
2	how did apollo creed die	D5	No answer
3	how long is the term for federal judges	D6	No answer
4	how a beretta model 21 pistols magazines works	D7	No answer

(Figure 1A. Question, Document ID, Correct Answer: First 4 Questions for Train Set)

Next, using the function get_documents, for each document ID, all candidate answers are compiled by concatenating each sentence into a document example shown below.

"A partly submerged glacier cave on Perito Moreno Glacier . The ice facade is approximately 60 m high Ice formations in the Titlis glacier cave A glacier cave is a cave formed within the ice of a glacier . Glacier caves are often called ice caves , but this term is properly used to describe bedrock caves that contain year-round ice."

The data is cleaned using the function preprocess. This changes all text to lowercase and removes all punctuation and symbols. Then, common contractions such as "don't" or "doesn't" are then also expanded into "do not" and "does not". The main idea for our method of preprocessing is to allow for more accurate and precise word embeddings for the CBOW model. This ensures that the words are standardized and makes sure that the same word is not over-represented.

The resulting preprocessed data is next passed to the function 'labelling' which is used to tokenize each document and then tag each token of the document as one of 4 token types based upon whether the token is part of the correct answer to a question. The first token of the correct answer in the document is labelled with 'S' (start token of answer), each further token of the correct answer is labelled with 'I' (inner token of answer), and the token at the end of the correct answer is labelled with 'E' (end token of the answer). Each other token in the document which is not part of the correct answer is marked with 'N' (None).

Each tagged document is finally paired with its relevant question in a data frame which is now ready to undergo embedding.

2 Sequence QA model

Word embedding

The word embedding technique we chose to use is the Word2Vec Continuous Bag of Words (CBOW) model. In the CBOW model, a neural network takes numerous words as input and makes a prediction of the target word which is closely related to the context of the numerous input words. When selecting CBOW for word embedding we took into account several different factors. First of all, prediction-based models such as CBOW have been shown to consistently outperform count-base models on a variety of tasks and are generally applied in recommendation systems and knowledge discovery [2]. When comparing the Word2Vec CBOW and Skip-Gram models, it is reported that for models using large corpora, the Skip-Gram model yields a higher accuracy however is more computationally expensive. Given the relatively small size of our corpora and the

limited hardware at our disposal we chose to use CBOW as the less expensive model with similar accuracy results [3].

Feature Extractions

POS Tags

POS tags, or better referred to as part-of-speech tags, are used to give context to each word in a sentence by labelling them with – "Noun", "Verb", etc. This allows us to better represent context of a word, especially in a question and answer model, and to differentiate the same word from 1 document to another, with different meanings/context. In the presented model, we used the pre-trained model from NLTK to tag each word in both all our documents and questions, for both the training and the test sets.
We can see how the POS tags are attributed to each word from a sample below.

```
[('how', 'WRB'),
 ('many', 'JJ'),
 ('schools', 'NNS'),
 ('are', 'VBP'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('big', 'JJ'),
 ('ten', 'NN')]
```

In our corpus, we identified 36 unique types of POS tags. To be able to represent these tags numerically, which is what will be fed into the neural network, we represent each tag using a sparse representation of a one-hot vector. For each word, the one-hot vector (OHV) corresponding each unique POS tag is then concatenated to the vector of word embeddings.

NER Tags

NER tags, also known as Named Entity Recognition is another method of feature extraction. The intent of including NER is to give our word representations more meaning. This is especially important in the context of a QA model, where named entities are generally more frequent. For example as below – in the question "How much are the Harry Potter movies worth?", the NER tag will identify and tag "Harry Potter movies" as "Work of Art" providing valuable further context to our model.

```
[('how', ''),
 ('much', ''),
 ('are', ''),
 ('the', ''),
 ('harry', 'WORK_OF_ART'),
 ('potter', 'WORK_OF_ART'),
 ('movies', 'WORK_OF_ART'),
 ('worth', 'WORK_OF_ART')]
```

In this project we use the pre-trained Spacy NER to automate this assignment of NER tags. This allows us to provide more meaning to the word then just using word embeddings. We employ a similar workflow as aforementioned in the POS part, where NER tags of one-hot vectors corresponding to each word in the document or question, and then concatenating the OHV to the end of the vectors.

TF-IDF

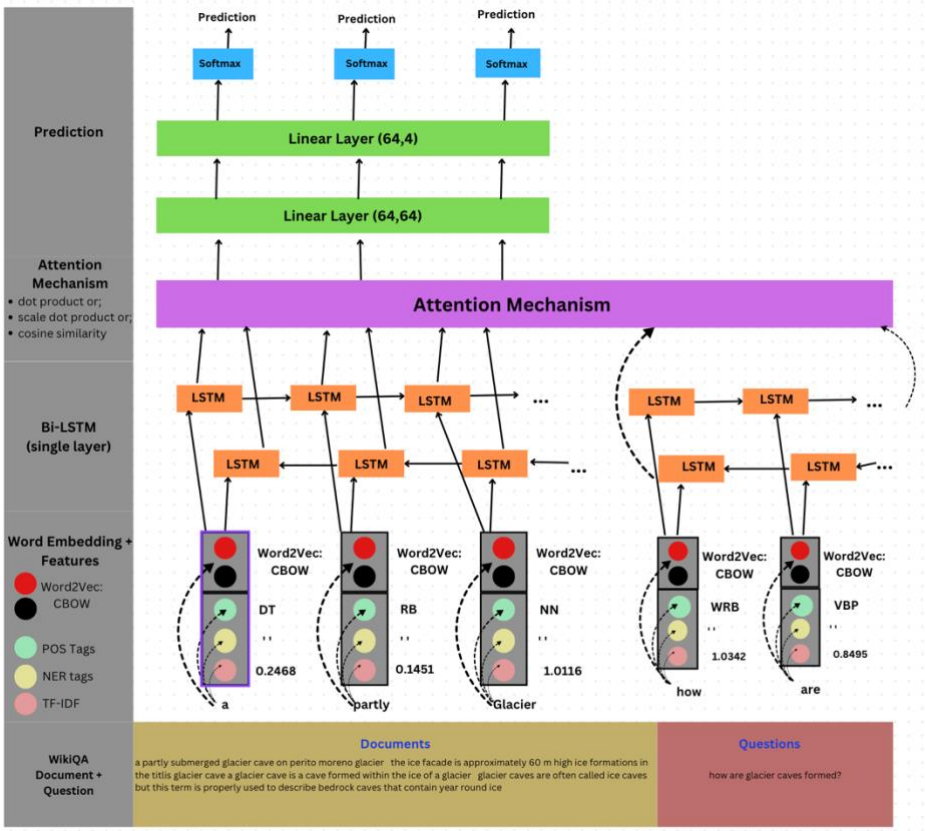
Term Frequency – Inverse Document Frequency also known as TF-IDF is an algorithm which uses the frequency of words to compute a statistic representing how relevant different words are to a given document. The TF-IDF value of a word is expected to increase in proportion with how many times a word appears in a document but is balanced by the frequency of the word in the corpus. This provides a very useful feature for representing the importance of a word to our model.

Chosen Sequence QA Model (Recurrent Model with Attention)

Number of Layers

While investigating the literature to gain an understanding of the need for multiple layers, we came across evidence that the performance gains from adding a second or third layer are often minimal, if present at all, and occur at the expense of computational efficiency [4]. This means that a single hidden layer is sufficient for the large majority of projects [4,5]. Furthermore, as we are adding an attention mechanism to our model, the attention mechanism should improve the performance of the model enough that we do not require multiple layers [6]. There is evidence in the literature that the addition of a convolutional layer improves the performance of Bi-LSTM models for more classification problems but in this project, we did not investigate this further [6]. Therefore, for the purpose of this project, the optimal number of layers is 1.

Bi-LSTM Model Architecture



(Figure 2.0: Bi- LSTM with Attention Mechanism Architecture)

Bi-LSTM

The sequence model we chose to use in this project was a Bidirectional Long-Short-Term-Memory Recurrent Neural Network model or bi-LSTM with 3 different kinds of attention mechanisms. LSTM's have received substantial research attention due to their strong performance on many sequence-based classification tasks and outperformance of traditional RNN models [7,4]. This is due to their ability to track long-term contextual information, and deal with the drawbacks of vanishing gradients [8]; both attributes which are essential for contextual NLP problems, and in particular, QA models. However, one limitation of standard LSTMs is the failure to extract context information related to future tokens and the inability to extract local context information [7]. Bi-LSTM models allow us to do this [9]. In the context of a reading comprehension task using a document model and question model, a common approach is to use a combination of a document encoder and a question encoder. The document encoder processes the context or document, while the question encoder processes the question.

These encoders have been implemented using Bi-LSTM. The outputs from the document and question encoders are combined and passed through additional layers to generate the answer. For us this involves attention layers and linear layers.

Attention Mechanism

After the embeddings are passed through the bi-LSTM model, the representation of the sequence which emerges from the bi-LSTM is passed to an attention mechanism. The attention mechanism is placed in this position because it allows the model to selectively utilize the information generated from the Bi-LSTM. The attention mechanism processes the output and chooses which features/words are highly correlated with the question by assigning them a weighting [7]. This process improves the prediction accuracy considerably. In this project we explored 3 different kinds of attention mechanisms including: dot product, scaled dot product and cosine similarity. Dot product attention and scaled dot product attention are both very similar mechanisms which perform similarly at low values of queries and keys however at high values of queries and keys, the dot products grow in magnitude and the gradients of the softmax function become extremely small [10]. The scaling counteracts this effect. One limitation associated with dot product attention is the fact that the results of dot product are unbounded therefore there is a risk of large variance [11]. To counteract this, we also explore an attention model known as cosine similarity. Cosine similarity acts to bind dot product attention and normalize them which reduces the risk of high variance [11].

Finally, the output of the attention mechanism is passed through 2 linear layers before it is passed to the softmax function, and a class prediction is made based upon this value. The model is now ready for testing.

3 Model Testing

In this section, the aim is to study how the model performs under different circumstances, more specifically when changing the word embeddings features, attention mechanism, and finding the best hyperparameters in the RNN model to give us the best performance. In order to test the model, we chose to evaluate the model's ability to classify tokens outside the correct answer (encoded as 0), the correct starting token of the answer (1), the correct inside tokens of the answer (2), and the correct end token of the answer, (marked as 3). Given the large class imbalances between the 4 classes of tokens in the test set, with 80,177 occurrences of '0', 230 occurrences of '1', 5,295 occurrences of '2' and 229 occurrences of '3' we chose to use the macro average of the performance metrics for difference classes. If we were to use the weighted average metric this would heavily weight the model's performance towards predicting tokens outside of the correct answer, which is not what we are trying to achieve and would skew the model's performance to be higher than it is for predicting the answer.

The macro averages for the performance of the model under different conditions are displayed in figures 3.0, 3.1 and 3.2 below.

3.1 Input Embedding Ablation Study

Embedding Variant	Metric			
	Precision	**Recall**	**F1**	**Ranking**
Word2Vec (CBOW)	0.29	0.65	0.23	1
Word2Vec (CBOW) + TF-IDF	0.28	0.65	0.20	3
Word2Vec (CBOW) + TF-IDF + POS tags + NER tags	0.28	0.67	0.22	2

(Figure 3.1: Input Embedding Ablation Study Results)

As we can observe from Figure 3.0, while all embedding variants tested displayed highly similar performance, the best model was the simplest with only basic CBOW word embeddings which yielded an F1 score of 0.23. This is followed by the model using all embedding techniques (CBOW, TF-IDF, POS tags, NER tags) which returned a very close F1 score of 0.22 and finally by the CBOW+TF-IDF embedded model with an F1 score of 0.20. The fact that all the embedding variations have such similar performance may suggest that the CBOW embeddings provide enough contextual for this relatively simple QA model that further features are not required.

3.2 Attention Ablation Study

Attention Mechanism	Metric			
	Precision	**Recall**	**F1**	**Ranking**
Dot Product	0.28	0.68	0.16	3
Scaled Dot Product	0.28	0.65	0.18	2
Cosine similarity	0.29	0.65	0.23	1

(Figure 3.2: Attention Ablation Study Results)

As we can observe from Figure 3.1. the use of different Attention mechanisms varied model performance significantly more than varying embedding techniques. This may suggest that model performance is more sensitive to the attention mechanism than it is to the embedding technique. Cosine similarity provided the best F1 score by around 5%, followed by Scaled Dot Product and then Dot Product. This suggests that the dot product attention was resulting in high variance which was reducing the performance of the model.

3.3 Hyper Parameter Testing

(Figure 3.3: Model Performance at Different Learning Rates)

In Figure 3.3, we can observe the performance of the model on the test set when trained with differing learning rates. We can see that as the learning rate increases from 0.00001, the model performance increases until it reaches its maximum performance at 0.01, after which it begins to decrease again. This suggests that at learning rates of under 0.01 the model is learning too slowly, while at learning rates of over 0.01 the model is converging too quickly and is not finding the optimal solution. We can therefore determine that the optimal learning rate, all other hyperparameters held even, is 0.01.

References

1. Yang, Y., Yih, W-t. & Meek, C.: “WikiQA: A Challenge Dataset for Open-Domain Question Answering”. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2013-2018 (2015).
2. Baroni, M., Dinu, G., Kruszewski, G.: “Don’t count, predict! A systematic comparison of context-counting vs. Context-predicting semantic vectors”. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. Vol 1. 238-247 (2014).
3. Mikolov, T., Chen, Kai., Corrado, G., Dean, J.: “Efficient Estimation of Word Representations in Vector Space”. arXiv:1301.3781. (2013).
4. Hameed, Z., Garcia-Zapirain, B.: “Sentiment Classification Using a Single-Layered BiLSTM Model”. In: IEEE Access. Vol 8. 73992-74001 (2020).
5. Odhiambo, P. “Generative Chatbots – How many LSTM Layers should you have?”. 2019. https://www.linkedin.com/pulse/generative-chatbots-how-many-lstm-layers-should-you#:~:text=The%20vanilla%20LSTM%20network%20has,a%20standard%20feedforward%20output%20layer. Last accessed 20/05/2023.
6. Liu, G., Guo, J.: “Bidirectional LSTM with attention mechanism and convolutional layer for text classification”. Neurocomputing. Vol 337. 325-338 (2019).
7. Jang, B., Kim, M., Harerimana, G., Kang, S-u.: “Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism. Applied Sciences. Vol 10. 5841-5855 (2020).
8. Sal, H., Senior, A., Beaufays, F.: “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition”. arXiv:1402.1128. (2014).
9. Zhao, A., Kim, S.: “Question Answering Using Bi-Directional rNN”. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2758402.pdf (2018).
10. Vaswani, A., Shazeer, N., Parmer, N., Uszkoreit, J., Jones, L., Gomaz, A.N., Kaiser, L., Polosukhin, I.: “Attention is All You Need.” arXiv:1706.03762. (2017).
11. Luo, C., Zhan, J., Xue, X., Wang, L., Ren, Rui., Yang, Q.: “Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks”. Lecture Notes in Computer Science. Vol 11139. (2018).