CITS4012 Project

Group 28: Max Chatfield and Nicholas Clarke

University of Western Australia, Australia
22176321@student.uwa.edu.au
10417777@student.uwa.edu.au

1 Dataset

We used a predefined split of training and test data from the WikiQA dataset. This dataset was assembled from Bing query logs from May 1st 2010 to July 31st 2011 and the summary sections of Wikipedia pages linked to these queries [1]. In this data, candidate answers to the question were labeled 0 and 1, with 1 indicating that the sentence was judged an answer to the question. There were some queries for which there were multiple, or no, sentences which were labeled as answering the question.

The dataset consisted of one row for each sentence of the document, we extracted the question sentence, and collated all sentences associated with that question id along with their answer labels.

Some of the text had irregular spacing, which caused issues for our embedding functions, so we converted all multiple whitespaces to single spaces through a simple regex substitution. We also used this step to space out punctuation from adjacent words.

We chose to keep most punctuation symbols in the data, because we felt that things like brackets were necessary for the context and meaning of certain sentences, and full stops would help demarcate separation of sentences. We also noticed that there was a question “what is the @ sign called?” which implied that for some questions in the dataset certain non-alphanumeric characters might play an important role in identifying the answer.

The sentences were tokenized into their individual terms, and later the embeddings of each term were collated into one list of embeddings for the entire document.

We chose to take the approach of labeling all individual words within the document as being within the answer or not, and using that as the target of our model. We trained our model to predict which span of words in a document were the answer, rather than modeling answer classification sentence by sentence.
We tested two possible systems of labeling:

- An inside-outside (IO) labeling system in which answer words were labeled ‘IOA’ and non-answers labeled ‘OOA’
- A before-after-answer system where words before the answer sentence were labeled as ‘BA’, all words within an answer were labeled ‘IA’ and all words after an answer sentence were labeled ‘AA’.

Initially we began with an inside-outside-beginning-end (IOBE) classification, where the start and finish of the answer would be labeled as ‘BOA’ and ‘EOA’ respectively.

However, we later decided that there would be insufficient support for the beginning and end tokens, and thus shifted to a simpler IO classification.

The second system trialed, classifying the words before and after the answer with two different labels was explored for a similar reason. Our initial training run produced a model which predicted zero answer tokens, and while we later improved and refined our model training process, as discussed in subsequent sections, these results led us to explore alternative labeling systems with more even proportions of the classes. By splitting non-answer sentences into two groups, before and after answer, the proportions of labels in the data would not be as unbalanced as before. We do not present the before-after-answer system in this paper as we focused our efforts on the simpler case.

We decided to cut the documents to the maximum length of 256 words. Although this is quite limited as the documents are quite large, we found that computing over the entire document space was too difficult in time and compute.

2 Sequence QA model

We embedded each word using gensim's pretrained glove-wiki-gigaword-100 model. This word embedding was chosen because it was a medium-sized model trained on 2014 Wikipedia and the Gigaword datasets. As this was partially trained on Wikipedia text, it was thought that its word embeddings may be more suitable than those of a model trained only on news or twitter corpora.

For feature extraction, we chose to perform part-of-speech tagging, term-frequency-inverse-document-frequency, named entity recognition, and implement a word match function on the document. We used POS and NER on the question.

Of these, the word match between question and answer seemed like it might be the most helpful. The question text was cleaned of common function words and simple punctuation using a list of 'stop words', before being passed to the Natural Language Toolkit WordNet lemmatizer to return a list of lemmas for content words within the sentence. The lemmatized words of the document were then compared to this list of question lemmas to identify which words matched with the lemma of a question word.

TF-IDF was performed on the document, using term frequency of a word within each sentence. Document frequency for the purposes of the calculation was number of sentences containing that word within the document.

Both the query and the document were tagged with POS and NER. POS tagging was performed using NLTK's POS tagging model, with the output tags then converted to integer labels using a predefined index.

NER tagging was done using spaCy's pretrained en_core_web_sm model. Words were embedded with both the entity type and the entity IOB predicted by the model.

We choose the architecture corresponding to the best model found through iteratively training large number of models over the parameter space. The models/batch_training.py file was used to perform this search. The criteria for selecting the best model was the model that had the best f1 score over the test set. After this process, the best model discovered was a single-directional GRU based model, with two hidden layers and an attention dot product layer. The GRU hidden size was 100. The embeddings used on this model are document NER and question POS. This means that all other embeddings were found to decrease the performance of the model, although slightly, and thus are not included in the final model.

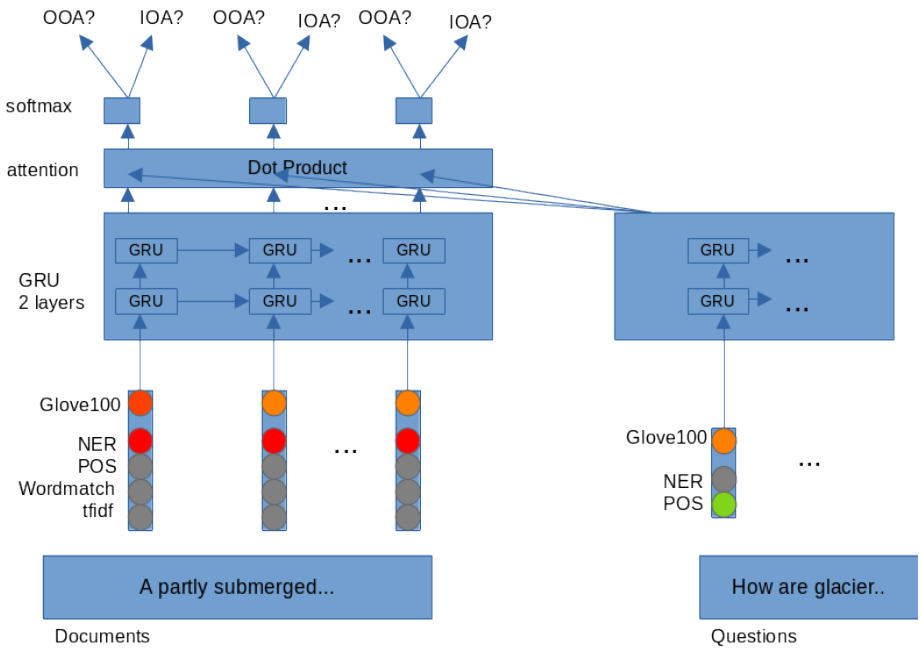
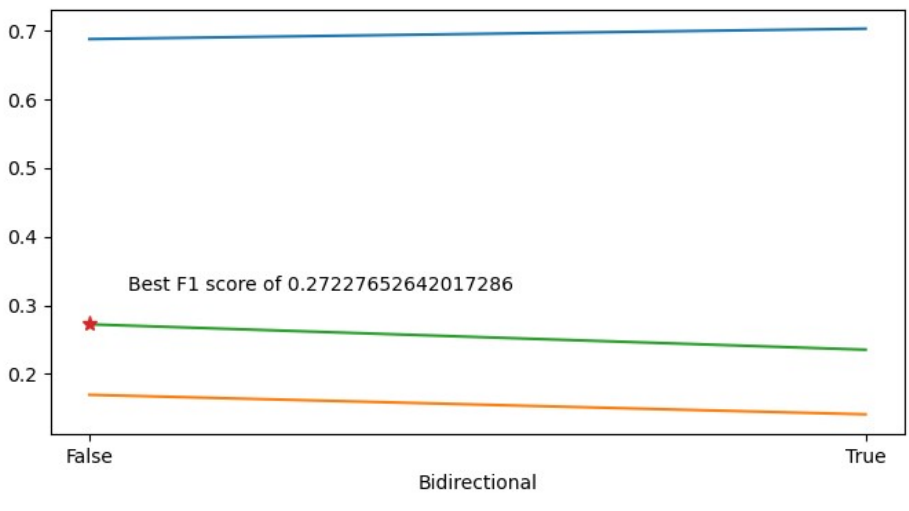
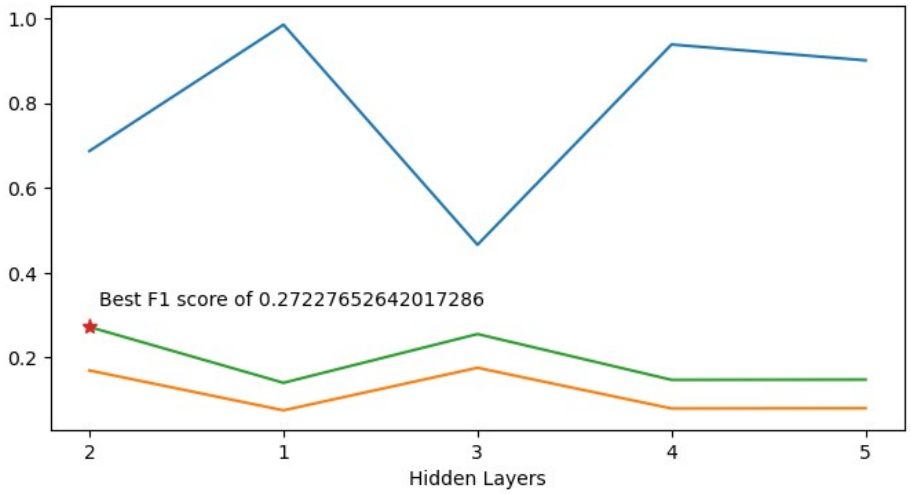


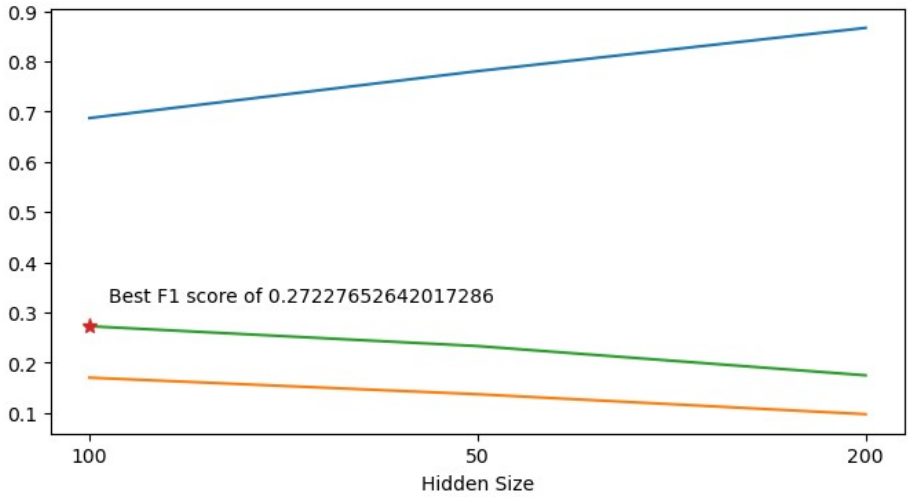
Figure 1: Architecture of the final model. The grayed out embeddings mean they were not included.

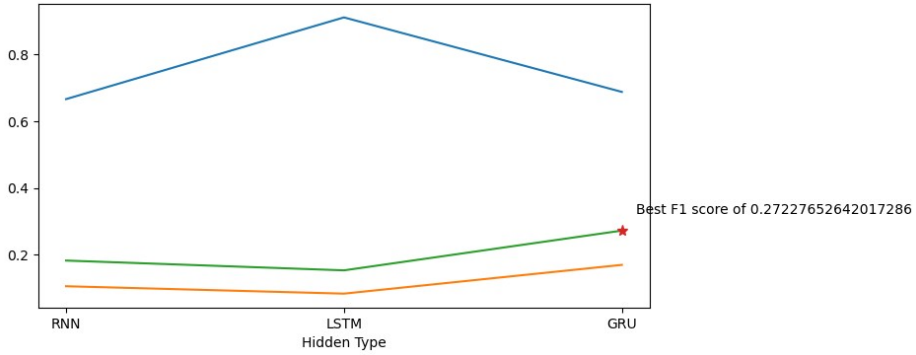
We chose to put the attention layer after the recurrent layer. We did this as it follows established models used by others in the NLP sphere[2] and is simpler conceptually and easier to implement.
All parameters were determined experimentally according to the criteria described above. We include some plots demonstrating the best model parameters beats other choices.



4


1.0
0.8
0.6
0.4
0.2
Best F1 score of 0.27227652642017286
2
1
3
4
5
Hidden Layers


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Best F1 score of 0.27227652642017286
100
50
200
Hidden Size



The GRU has the best performance by far. It is able to generalize better, but as for why its performance is so much better we are unsure. Empirical results [3] suggest the LSTMs and GRUs perform similarly in a music and speech modeling scenario.

We were surprised to find that the GRU generalized best on a smaller hidden size and smaller number of layers. Specifically above, we also found that bi-directional models perform worse than single-directional models. This trend seemed to hold over a wide class of models.

3 Model Testing

Performance of the models were assessed by comparing the predicted classifications of each word in the document to the ground truth.

3.1 Input Embedding Ablation Study

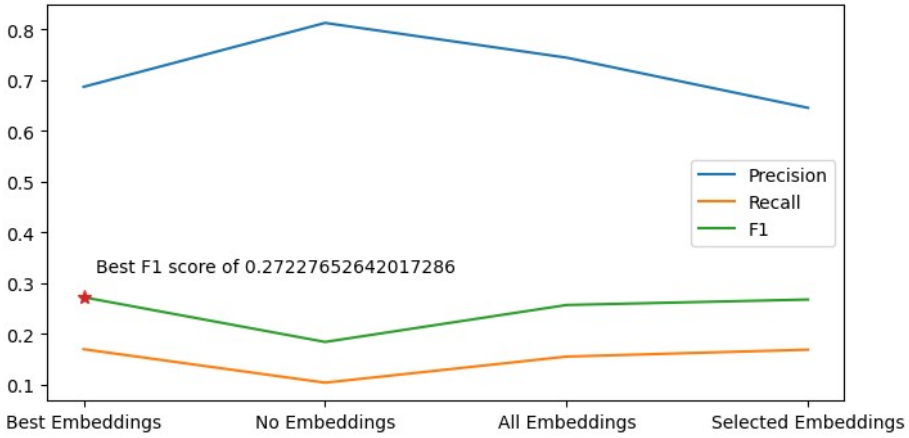


Figure 2: Albation over the input embeddings. Selected Embeddings have some embeddings chosen such as document tfidf and word match, and question NER and POS.

We found that the specified combination of embeddings (Glove100, Document NER and question POS) gave the best model performance. It is very interesting that not including some embeddings such as word match and tf-idf led to better performance. Although it seems difficult to hypothesize how tf-idf may help or hamper the model, we thought that word match may not increase model performance as questions usually are general, and so it doesn't matter if some words match between the document and question.

3.2 Attention Ablation Study

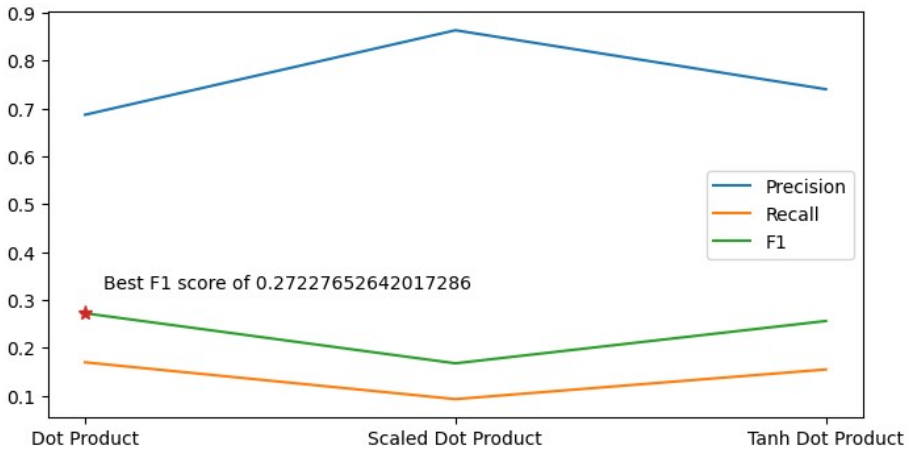


Figure 3: Attention ablation. The best model uses Dot Product. The results between the Tanh Dot Product and Dot Product are very similar.

We chose simple attention variants. Scaled Dot Product seems to decrease the performance of the model drastically. As we can see from the high precision score and lower recall score, the model using Scaled Dot Product seems to over predict the positive (IOA) class.

3.3 Hyper Parameter Testing

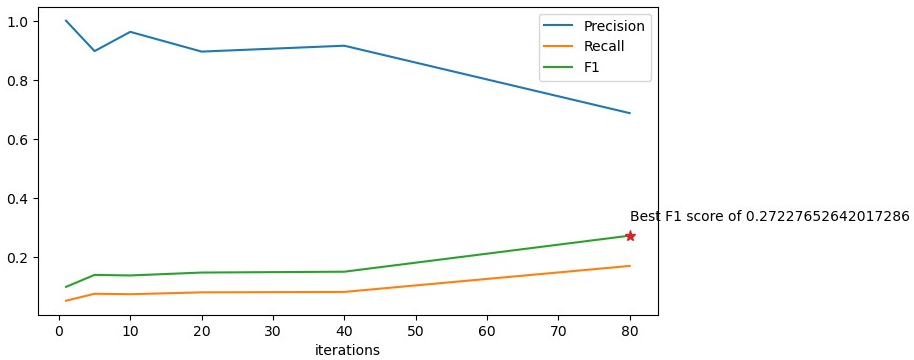


Figure 4: Iteration ablation for the best model. The model achieves the best f1 seen at iteration 80.

We sampled the model at 6 iterations: 1,5,10,20,40 and 80. Initially as we were training models, we saw after 40 to 80 iterations, that the model seemed to be overfitting. In this case, the GRU model seems to be able to generalize well and could train for longer, achieving better performance. We stop here due to time and compute concerns.

References

1. Yang, Y., Yih, W., Meek, C.: WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018. Association for Computational Linguistics, Lisbon (2015).
2. Luong, Minh-Thang, et al: Effective Approaches to Attention-Based Neural Machine Translation. *arXiv.Org*, arxiv.org/abs/1508.04025. (2015)
3. Chung, Junyoung, et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence* arxiv.org/pdf/1412.3555v1.pdf. (2014)