QA Framework

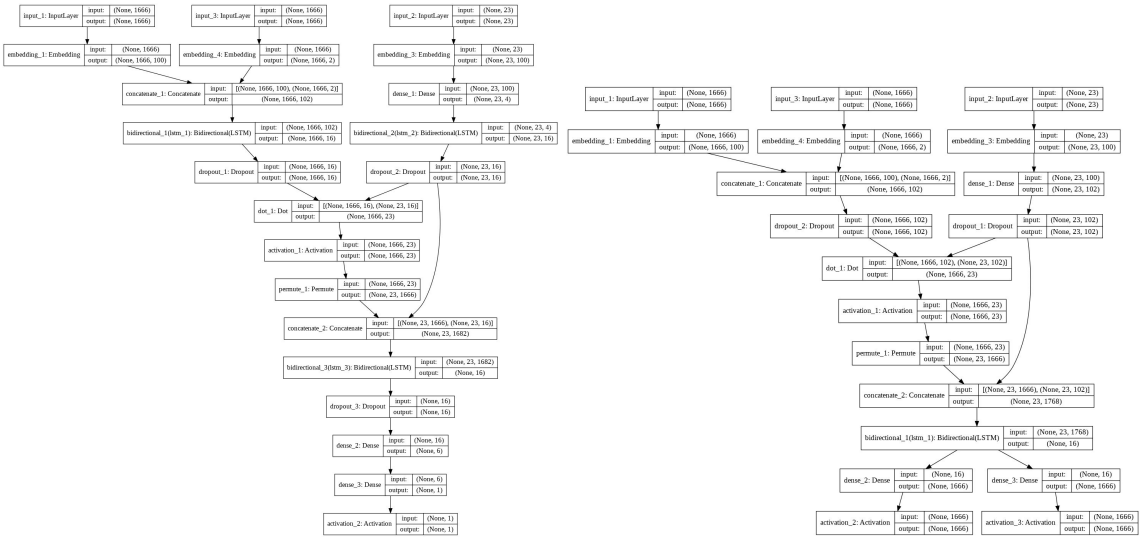

Figure 1: Overview of the architecture of the final model. There are 2 model in total, on the left it is a binary model used to predict if a given doc and question has answer or not. The right figure is a multi-labels model that predict start and end **positions** for start and end token. Both models contains 3 input layer, Doc, word_match and Question.

1) Word Embeddings and Feature Extraction

The pre-trained word embedding layer used is **gloVe** with word vector size = 100, which is the size that is popular in industry and it is reasonable in terms of computational time and performance as having largest size does not seems to improve performance significantly while computational time and memory usage increases the same time.

There are 4 different Features have been extracted from the Dataset. They are the

- PoS tags
- TF-IDF
- Named Entity tags
- Word match feature

As limited by the computational resources such as Ram , GPU memory and limited session period(computational time) in Colab, using 3 - 4 features either took too long computational time or produces memory/ gpu crash. There are only 1-2 features have been used in different models.

To quickly summarise, the best features is **Word match** feature in terms of training accuracy. One potential reason is that Word Match Feature directly exposes mutual information between Document and Question as Word match contains information about tokens appear on both Document and Question.

In many case when Question contains tokens from Documents, it is often the case when the **Answer can be search from Document**, while when there is NO answer found on Document for a given question, this is usually the case when Question does not contain(or only very few words) any tokens on Document.

While another features does not seems to improve the model performance, main reason could be that they do not provide additional meaningful information to the model. For instance PoS tags provide information about part of speech of a given token from Document, while Named Entity tags contain information about name entity.

Intuitively, Knowing whether a token from Document is a Noun or an Organization does not assist the model to learn much from the Question as these are no mutual information between Document and Question. In other word it does not provide additional information for the model to recognize connection between Document and Question. Same case for TF-IDF, just by knowing the frequency of tokens on Document only is not meaningful.

2) Sequence Model (RNN with Attention)

Pipeline architecture

Instead of using one single end to end sequence model, a Pipeline of 2 different sequence models were involved. On figure 1 there are 2 different sequence model, the left hand side model is a Binary Model that classify if a given doc and question has answer or not. While on the right hand side it is a Multi-Label sequence model, that predicts start and end token position given a doc and question. The reason for using a Pipeline is that the labels distribution are highly **Imbalance**. There are 1666 class (i.e. 1666 possible position of start and end tokens), and most of the data contains no answer, thus belongs class 0. **While the rest of the data are spread out over classes 1-1665(i.e. 60% of class 0, with the rest 40% of data spread out over classes 1-1665 thus each class has average of 0.4/1665 proportion),** which is very difficult to train a sequence model with good performance on this highly imbalanced dataset, especially when Colab has limited computational resource for training a high capacity model architecture.

In short the Pipeline are worked as follow. The test data were first fed into the binary model, if the model predict an given data as 1, which means there is answer for the given question, it then fed into the next Multi-label sequence model as to predict it's corresponding positions of start and end tokens. Thus the Multi-label sequence model will only predict those who are filtered by the previous Binary Model as they are highly chance in having an answer.

Sequence Model architecture

The 2 models contains 3 input layers, doc, word_match feature and question.

BiDirectional LSTM layers, Attention and Embedding layers have been used on both models. And From the figure the input layers, *Input_1*, *Input_3* are referring to Document layer and Word_match layers respectively. Both are sequence input and are converted to word

vector via Embedding layers(with pretrained gloVe 100 for Document input layer) and the resulting word vectors are later concatenated via *concatanate_2* as shown on the Figure 1.

Binary model on the left

After experience different number of Bidirectional LSTM layers, I was settled with 2 Bidirectional LSTM layers for each 16 hidden units. The main reason for this number is that under this setting the model converge fast as well as spending reasonable computational time and resources. I have tried 2-3 layers with 16-128 hidden units but it either lead to memory crash, GPU crash or very long training time on Colab WITHOUT significant improvement in terms of converging speed and training performance. Under some setting with more layers the model just stuck on a certain training accuracy such as 0.1 remains over 100 epochs. While the main reason for Attention position is similar. The attention layer started in DOT product layer and it somehow lead to faster converge time and reasonable accuracy. Thus the optimal number of layers and position are based on training practice experience in terms of comverge speed, local optimal issue(stuck on some accuracy) and computational resources.

Multi-labels model on the right

Similar to the Binary Model, the multi-labels model also has small number of Bidirectional LSTM layer with small amount of hidden units i.e. One Bidirectional LSTM layer that contains 16 hidden units located after *concatanate_2* layer and before output layers for start and end tokens position. The reason for the setting are the same as Binary Models that is experienced based on training practices, which lead to faster converge time, better performance, reasonable training time without memory crashing issue.

Section B: Results and Discussions

Word_match(Final model)	pos	Tf idf
Precision: 0.449393 Recall: 0.460581 F1 score: 0.454918	Precision: 0.467290 Recall: 0.414938 F1 score: 0.439560	Precision: 0.453642 Recall: 0.568465 F1 score: 0.504604

Table 1: Binary model score

	precision recall f1-score (start token)			precision recall f1-score (end token)		
Word_match(Final model)	micro avg	0.44	0.44	0.44	micro avg	0.44	0.44	0.44
	macro avg	0.00	0.00	0.00	macro avg	0.00	0.00	0.00
	weighted avg	0.47	0.44	0.46	weighted avg	0.47	0.44	0.46

pos	micro avg	0.38	0.38	0.38	micro avg	0.39	0.39	0.39
	macro avg	0.00	0.00	0.00	macro avg	0.00	0.00	0.00
	weighted avg	0.35	0.38	0.37	weighted avg	0.37	0.39	0.38
Tf idf	micro avg	0.36	0.36	0.36	micro avg	0.36	0.36	0.36
	macro avg	0.00	0.00	0.00	macro avg	0.00	0.00	0.00
	weighted avg	0.31	0.36	0.33	weighted avg	0.30	0.36	0.33

Table 2: Multilabel model with start and end token prediction avg scores

The 2 tables shows scores for binary model and multi-labels model respectively. From table 1 Word_match does not seems outperform others. However from table 2, when predicting start and end token position, it has outperformed average score for all precision, recall, and f1-score than other 2 i.e. word_match feature extraction model is the only one who > 40% score on all weighted avg on both start and end tokens prediction.

As mention, the reason is that Word Match Feature directly share mutual information between Document and Question as Word match count tokens occurance on both Document and Question. Also, for the document that contain answers, they often have “keywords” appear on both question and document, e.g. what is Obama children name?, the keyword could be obama and children, and when these keyword from question also appear in document, that means we have higher chance to extract the answer from the documents.

While knowing pos and name entity might not help much as it only contains information on Document but not the question as well.