JAM for HotpotQA

Mars Huang **Ashton Teng** **Juan Manuel Zambrano**

{mschuang, ashteng, jmz}@stanford.edu

Abstract

HotpotQA is a recently released question-answering dataset that involves multi-hop reasoning over multiple paragraphs of information to produce an answer. A successful model must not only report answers as yes/no or a span within the text but also identify supporting facts. In this project we set out to develop a pipeline model for this dataset, consisting of two components: a supporting facts classifier that produces relevant sentences which are grouped and used by a question answering model. For the former we have modified the baseline model implemented in the original publication to output supporting facts and for the question answering module we have pretrained an implementation of Bidirectional Encoder Representations from Transformers (BERT) using the SQuAD dataset and fine-tuned it using the HotpotQA dataset. We achieve with our best model a joint F1 score for supporting facts and answers of 57.40, a marked increase over the baseline implementation (joint F1 40.86).

1 Introduction

Question answering is an active area of natural language processing research. HotpotQA is a question answering dataset that features questions that can only be answered from multiple supporting facts within a text [Yang et al., 2018], and is designed to test a system's ability to do multi-hop reasoning (reasoning over many supporting sentences). The dataset consists of 112,779 questions, the vast majority of which involve reasoning from supporting facts embedded within two paragraphs of text, where each paragraph corresponds to the first paragraph of 5 million+ Wikipedia articles. It differs from other previously published question-answering datasets such as SQuAD [Rajpurkar et al., 2016], which involve reasoning over a single supporting fact to produce an answer span within the text (or as of SQuAD 2.0, declare the question unanswerable). Furthermore, it differs from other multihop question datasets [Welbl et al., 2017] because the questions are not constrained to a pre-formed knowledge base. In addition, the dataset introduces the additional challenges of answering yes/no questions, comparison questions, and the need to predict supporting sentences.

Since recent leaderboard submissions have already achieved close to human-level performance on the SQuAD 2.0 dataset, a more interesting challenge for the field is multi-hop question answering, which requires much more natural language understanding abilities from the system. HotpotQA features two modes of model evaluation: distractor and full wiki setting. In the former, two paragraphs that contain supporting facts for the answer are shuffled among 8 other selected paragraphs selected from the corpus as the most similar as determined by bigram tf-idf [Chen et al., 2017] which are meant to represent noise. In the full wiki setting, designed to evaluate the performance of the model in the wild, the initial state is the full set of paragraphs. For our project we have focused on the distractor setting. In this project we aimed to develop a question answering deep learning model for HotpotQA using components that have reached state of the art results in other datasets. Our approach is to build a pipeline system based on joining two models: one to extract the relevant supporting facts from a series of paragraphs and the second to use a concatenated set of supporting facts as an input paragraph to produce an answer to the question. The full model outputs the supporting fact sentences that are required to answer the question, and the answer itself (which can be yes/no or a text span). We achieve a significant improvement compared to the only published model based on several metrics.

2 Related Work

The baseline model implemented in the original publication of the dataset [Yang et al., 2018], based on a well performing current model for multi-hop question answering [Clark and Gardner, 2017] in previous datasets, consists of an end-to-end model with components that have been successful in recent question-answering datasets. These include commonly used text representations that are a combination of both word embeddings that correspond to Glove vectors [Pennington et al., 2014] as well as character level CNN embeddings that are modified during training, such as that implemented by Kim et al. [2015]. Other incremental improvements on single-hop question answering datasets refine vector representations of words. An example of this is contextualized vectors (CoVe) McCann et al. [2017], where an LSTM with attention is trained on a translation dataset to produce the vectors and then implementations of transfer learning are shown. Another recent improvement on the machine comprehension task is the use of attention mechanisms. One particularly successful implementation is the bi-directional attention flow (BiDAF) model [Seo et al., 2016], which is used to build a query-aware context representation without text summarization. In this model, attention scores between the context and the query are calculated and concatenated and passed through subsequent layers. This is an important component of our model and is described with more detail in the subsequent sections.

Three additional models have recently competed for top positions in question answering datasets and have proven themselves relevant in other contexts through successful implementations of transfer learning. [Peters et al., 2018] developed a character-level embedding, deep neural network mainly comprised of many CNN and LSTM layers whose final representation is contextual termed ELMo. The QANet [Yu et al., 2018] uses a faster-to-train CNN combined with self attention to produce state of the art results at the time of its publication. More recently, the best performing models on leaderboards of question answering datasets [squ, qua] feature models that achieve near-human performance using BERT [Devlin et al., 2018], which uses at its core the Transformer [Vaswani et al., 2017] to create a language model representation that achieves state of the art results in question answering and other several tasks.

The relative success of deep learning on machine comprehension demonstrated in previous questioning answer datasets is now to be improved upon more complex, multi-hop reasoning question databases, such as HotpotQA. Other attemps to model more complex forms of reasoning include single and multi-hop trivia questions (TriviaQA) [Joshi et al., 2017], knowledge based approaches (QAngaroo) [Welbl et al., 2017], reasoning over dialogues and (QuAC) [Choi et al., 2018], and the recently released Natural Questions, comprised of 307,373 Google searches [Kwiatkowski et al., 2019].

3 Approach

3.1 Overall Model

HotpotQA's original baseline model [Yang et al., 2018] was trained to simultaneously predict supporting facts, as a well as the answer to the question (yes/no, text span). This may be the cause of their lower score in the supporting fact part of the task (21.95 EM, 66.66 F1). We believe that this end-to-end architecture is challenging for the model to learn well, since one error is used to update the weights for two different tasks, and the model is not necessarily focusing on the supporting facts in order to generate the answer. Thus, to solve some of these issues, our proposed model has two components: 1) A supporting facts classifier module: The support evidence identifier looks at each sentence for all input paragraphs and with a binary classifier predicts whether or not they can be considered as supporting evidence to answer the question. 2) A SQuAD-like QA model that takes all supporting sentences concatenated as one paragraph as input, and predicts the correct answer to the question (yes/no, or a text span).

Our system combines these two models, aggregating the sentences predicted by the support evidence identifier into a paragraph as input into the QA model. Figure 1 shows the details of our system, to be elaborated in the sections below.

3.2 Supporting Facts Classifier Module

The aim of the supporting facts module is to decide, from an input text of ten paragraphs, whether each sentence is a supporting fact for answering the question at hand. For this module, we modified the original open source HotpotQA model [Yang et al., 2018], to only output supporting facts. We removed layers that predict features related to the final answer question type (yes/no/span), and start and end indices. Corresponding changes were made to the loss function and the evaluation scripts. The code was also adapted to Pytorch 1.0, in order to fully integrate with the QA Module. The details of the model are as follows:

Each datapoint consists of one question and one context. The context includes words from all sentences in ten paragraphs. Let h_i be the i'th passage word, and q_j be the j'th question word.

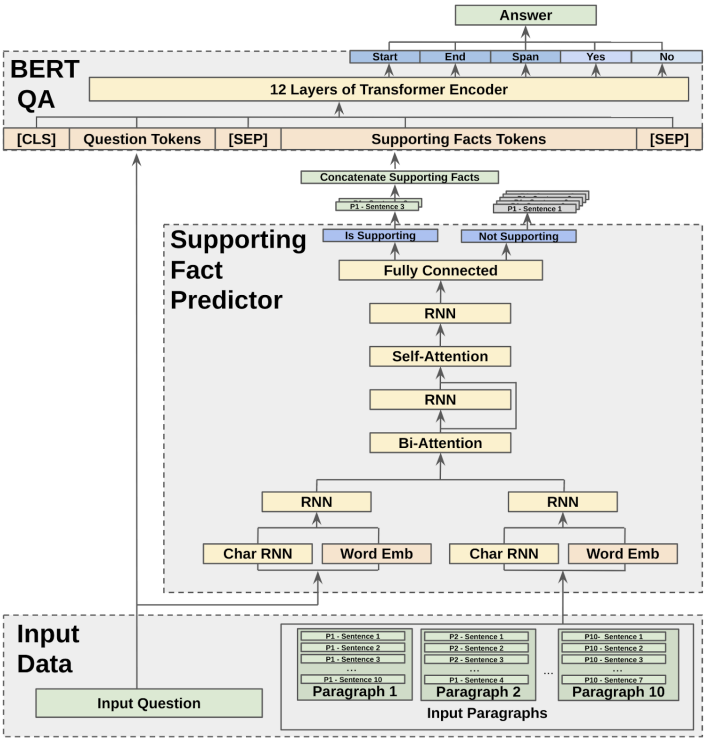


Figure 1: JAM Model for HotpotQA.

Context and Question Embeddings: Both the input text and the question are converted into a vector representations using both word and character level embeddings. Word embeddings correspond to 300-dimensional GloVe vectors and kept constant during training [Pennington et al., 2014]. The character embeddings are separately passed through 100 CNN filters and a max pooling layer, producing 100-dimensional character embeddings per word, updated during training. Word and character embeddings are concatenated and passed through the an encoder RNN, which is a single layer, bidirectional GRU with dropout.

$$q_{wordemb_i} = GloVe(q_i) \in \mathbb{R}^{300}$$

$$q_{charemb_i} = MaxPool(CharCNN(q_i)) \in \mathbb{R}^{100}$$

$$q_i = Dropout(BiRNN([q_{wordemb_i}; q_{charemb_i}])) \in \mathbb{R}^{400}$$

The same equations apply for $h_i \rightarrow h_i$. All BiRNNs are GRU in this model.

Bi-Attention: The outputs from the BiRNN are then input into a Bi-Attention module. This module is based on the attention flow layer from the BiDAF model [Seo et al., 2016]. The output is then passed through a linear layer with ReLU activations, and then a context-aware BiRNN with dropout. Equations adapted from Clark and Gardner [2017]. We compute attention between context word *i* and question word j as:

$$a_{ij} = \mathbf{w_1} \cdot \mathbf{h_i} + \mathbf{w_2} \cdot \mathbf{q_j} + \mathbf{w_3} \cdot (\mathbf{h_i} \odot \mathbf{q_j})$$

where $\mathbf{w_1}$, $\mathbf{w_2}$, and $\mathbf{w_3}$ are learned vectors and $\odot$ is element-wise multiplication. We then compute an attended vector c_i for each context token as:

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_q} e^{a_{ij}}} \qquad \mathbf{c_i} = \sum_{j=1}^{n_q} \mathbf{q_j} p_{ij}$$

Where *n_q* and *n_c* are the lengths of the question and context. We compute a query-to-context vector $\mathbf{q_c}$:

$$m_i = \max_{1 \leq j \leq n_q} a_{ij} \qquad p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}} \qquad \mathbf{q_c} = \sum_{i=1}^{n_c} \mathbf{h_i} p_i$$

the final vector computed for each token is built by concatenating $\mathbf{X_i} = [\mathbf{h_i}; \mathbf{c_i}; \mathbf{h_i} \odot \mathbf{c_i}; \mathbf{q_c} \odot \mathbf{c_i}]$. The above equations can be summarized as $\mathbf{X} = Bi{-}Attention(\mathbf{Q}, \mathbf{H})$. $\mathbf{X}$ is then passed through a Linear layer with ReLU activations, and then a BiRNN with Dropout: $\mathbf{X} = ReLU(Linear(\mathbf{X})), X = Dropout(BiRNN(\mathbf{X}))$. *Self-Attention*: The outputs from the Bi-Attention section are passed through a self-attention layer, which is the same bi-attention mechanism, only now between the question-aware context $\mathbf{X}$ and itself. In this case we do not use query-to-context attention, as our $\mathbf{X}$ is

already query-aware. Just like the Bi-Attention section, the results from this section are passed through a linear layer with ReLU activations, and then a BiRNN with dropout. The results from the Bi-Attention and Self-Attention sections are summed together residually.

$$\mathbf{Y} = Bi - Attention(\mathbf{X}, \mathbf{X})$$
$$\mathbf{Y} = ReLU(Linear(\mathbf{Y})), \mathbf{Y} = Dropout(BiRNN(\mathbf{Y}))$$
$$\mathbf{Z} = \mathbf{X} + \mathbf{Y}$$

Output Predictions Finally, another BiRNN is used to produce the final predictions for supporting facts. A linear layer is used to map this output into logits for every sentence among the ten paragraphs. The loss function used was an average cross entropy loss between the logits and a one-hot encoding of the supporting fact sentences in the paragraphs.

$$\mathbf{Z} = BiRNN(\mathbf{Z})$$
$$PredictedLogits = Linear(\mathbf{Z})$$
$$\texttt{Loss} = \frac{\text{CrossEntropyLoss(PredictedLogits,OneHotLabels)}}{\text{Number of Sentences}}$$

3.3 Question Answering Module

The second component of our network architecture is a reading comprehension and question answering model. Similar to models designed for the SQuAD dataset, this component takes in a paragraph and a question as an input, and outputs predictions on the start and end index of the answer span in the input paragraph. Additionally, our model has the yes/no/span output channel to accommodate with the wide variety of questions types hotpot QA has to offer. The input data is a concatenation of all the predicted supporting facts from our supporting evidence identifier.

The core of our QA model is BERT, which is a general purpose pre-trained language representation on a large text corpus [Devlin et al., 2018]. Similar to computer vision models trained on ImageNet, the presentations learned by BERT can be transferred to different NLP tasks. The effectiveness of BERT is not only attributed to the large corpus that it is trained on (Wikipedia + BookCorpus) and the computational resources used (4 days on 16 TPUs), but also due it's model architecture. The 24 layers of bi-directional Transformers allow BERT to learn contextual representations that achieve states-of-the-art results on a wide array of NLP tasks, including SQuAD question answering.

Since a BERT model has already shown state-of-the-art predictions for SQuAD (87.4 EM and 93.2 F1), we pretrained our model using the SQuAD dataset and fine tuned it using the HotPotQA training data. We modified the BERT implementation in Pytorch by Huggingface by adding an additional linear layer to predict start index, end index, yes, no and span. The span output layer is to indicate if our answer should be a span from the input text, or should be a yes or no. For example, if the span output is 1, then our final prediction should be words between the start and end index of the model's input. If the span output is 0, then our final prediction is yes or no. The loss function is a simple average of the cross entropy between the start logits, the end logits and the yes/no/span classification.

$$\text{L} = \frac{1}{3} * (CEL(StartPred, StartLabels) + CEL(EndPred, EndLabels) + CEL(YesNoSpan, YNSLabels))$$

4 Experiments

4.1 Datasets and Preprocessing

Supporting Facts Model: For this module, we trained on the HotpotQA dataset. HotpotQA is novel dataset of 112,779 examples that is crowdsourced from the first paragraph of diverse Wikipedia articles, where crowd workers were shown pairs of supporting documents and asked to explicitly come up with questions requiring reasoning about both of the documents, as well as label supporting parts of the text that are relevant to answering the question published by Yang et al. [2018]. The model features two main types of questions: bridge and comparison questions. Bridge questions are labeled so because they involve reasoning over a bridge entity that connects two paragraphs of text. For example, "What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?", involves reasoning about one fact relating "Shirley Temple" to her role in "Kiss and Tell", as well as an additional one relating "Shirley Temple" to her government role to produce the answer. The second main type of question involves a comparison between two entities from the same category, for example "Who died first, George Archainbaud or Ralph Murphy?". Within the latter there is a subset of yes/no questions, that comprise roughly 6% of all of the questions in the data. Roughly, 42% of the questions are of the bridge questions. 27% compare two entities, 15% involve locating the answer entity by checking more than one property and 8% require other types of reasoning. In addition, about 6% of

questions involve single-hop reasoning and 2% are not answerable. The original dataset includes two main modes of model evaluation: the distractor and fullwiki. We have focused on the distractor setting for our project. Each training example consists of a question, a list of paragraphs, a list of sentences in the paragraphs that support answering the question, and the answer (either yes/no or a span in the paragraphs). Each testing example is a question and a list of paragraphs, and the model must produce the answer and the supporting sentences. During preprocessing for the Supporting Facts Classifier, all words in both the question and the context are mapped to indices, and embedded with pretrained 300-dimensional GloVe vectors. The vectorized text is stored prior to start of training in order to save time.

QA Model: For the QA Module, we pre-trained our model using the SQuAD 1.1 training data and fine tuned our model using HotPotQA's distractor training set. We used HotPotQA's dev set to evaluate the performance of our model. The Squad1.1 dataset has more than 100k question-answer pairs, while HotpotQA's training set consists of 90k distractor sets. Since the json file format is different between HotpotQA and SQuAD, we transform HotpotQA's dataset into SQuAD format before training and testing. During training, the gold-standard supporting facts from each distractor sets are concatenated into a single paragraph, and paired with the original question and answer. Several processing steps arre taken before data is inputted into the QA Module. First, the data that we receive from the Supporting Facts Classifier is a list of lists, with each sub-list consisting of 2 items, the title of the paragraph in which the sentence belongs to, and the index of the sentence in the paragraph. We perform a lookup into the original Hotpot data to retrieve the actual sentences, and concatenate them together to form a new paragraph in the SQuAD format, paired with the question text. Then, various processing steps are performed as a part of BERT; each word in both the context and the question text are split into three representations:

- A WordPiece [Wu et al., 2016] token representation, which is similar to GloVe vectors, except BERT operates with a much reduced vocabulary 30,522 tokens by breaking up more complicated words into their components (e.g. "strawberries" : "straw" and "berries").
- A segment representation. BERT concatenates the question and context text together with separator tokens such as "<SEP>". Thus, in order to distinguish between question words and context words, there is a segment representation that is a vector of zeros for all question words, and a vector of ones for all context words.
- A position embedding, which for each word, represents their position in the sentence. Hence all words at the first position will have the embedding, and so on. These are introduced because Transformers (which make up the bulk of BERT) are position invariant, and would otherwise generate the same embedding for the same word in diverse positions.

4.2 Evaluation Method Metrics: EM and F1

We report our model performance based on EM and F1. EM is the percentage of exact match of the output predictions with the gold-standards. For the supporting fact classifier, we adds up the number correctly predicted supporting facts and divide it by the total number of correct supporting facts. For the QA model, we count the number of instance where our model's predicted answer is exactly the same as the gold standard answer, then divide the count by the total number of questions. The F1 score of the model is calculated using (2 x precision x recall) / (precision + recall). The precision is defined by the number overlaps between the prediction and ground truth divided by the total number of predictions, while the recall is calculated by the overlaps divided by the number of ground truth labels. For the QA model, the overlaps are defined by the number of overlapping words between the prediction and target. For example, if the predicted answer is "Robert Downey" but the actual answer is "Robert Downey Jr", the number of overlapping words will be 2. For the supporting fact model, the overlap is defined by the number of same paragraph and sentence index between prediction and label.

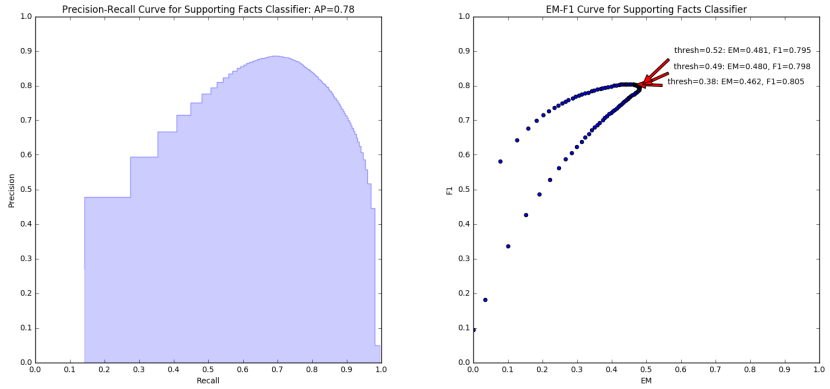
4.3 Experimental Details

Supporting Facts Classifier Module Experimental Details: The Supporting Facts model predicts the probability of every sentence in the 10 paragraphs in being supporting facts. During training, we use an average cross entropy loss (between sentences) of the logits and the one-hot labels (vector of length *num_sentences*, 1 for sentence which is a supporting fact, 0 otherwise). The loss was optimized via Pytorch's included Adam optimizer. During testing, we use a threshold, *sp_threshold*, a value between 0 and 1, to classify each sentence as a supporting fact. A higher threshold will result in less supporting facts being predicted with more confidence. Thus, in the context of our pipelined model, we have two goals: to maximize supporting fact accuracy, as well as to include as many correct supporting facts as possible to pass on to the QA module. Thus, we have decided to use two thresholds: *sp_threshold* will be optimized to produce the best supporting facts results and will be higher, and *qa_sp_threshold* will be lower in order to pass on the correct supporting facts to the QA module, even with added noise from non-supporting facts (we decided on a

qa_sp_threshold=0.2). In order to choose the optimal *sp_threshold*, we plotted a Precision-Recall curve and a "EM-F1" curve for the supporting facts category and found that *sp_threshold*=0.49 yields the best overall results.

Figure 2A shows precision and recall values associated with *sp_threshold*s from 0 to 1 in 0.01 increments. It is possible to get arbitrarily large recall by decreasing the threshold, at the expense of diminishing precision. Precision and recall for one threshold is calculated by averaging all precisions and recalls from all example data points, each of which is a collection of around 100 sentences (10 sentences per paragraph). Hence we cannot draw a horizontal line for the definite proportion of "positive examples". However, a good heuristic is 2-3 positive supporting fact sentences out of the 100 sentences, or a 3% random classification accuracy. **Figure 2B** shows EM and F1 values (for supporting facts classification) associated with *sp_threshold*s from 0 to 1 in 0.01 increments. The arrows point to points with best EM, best F1, and best EM+F1. Along with choosing an optimal *sp_threshold*, experiments were done to choose the best learning rate and dropout probabilities.

Figure 3A (Appendix) shows the loss curve with different learning rates while dropout probability is fixed at 0.2. Due to time limitations, three initial learning rates for the Adam Optimizer were tested. Notably, 0.001 resulted in a divergence in loss during training, accompanied with a classification accuracy drop to zero. **Figure 3B** (Appendix) shows the loss curve with different dropout probabilities while initial learning rate is fixed at 0.0005. Due to time limitations, two dropout probabilities during training were tested. It is obvious that reducing dropout helps training but may lead to overfitting. We decided to reduce dropout from the default 0.2 to 0.1 as the 0.2 dropout model did even better on the dev set than the training set, leading us to believe that the model did not have high variance. Indeed, even with a dropout of 0.1 the dev results are still very comparable to the training results. The final parameters for the Supporting Facts Classifier Module that were used for prediction are: the Adam Optimizer (with initial learning rate 0.0005), a batch size of 24 QA-pairs, an epoch of 10000 steps, dropout probability of 0.1, and a sp_threshold of 0.49.



(a) Precision-Recall Curve (b) EM-F1 Curve

Figure 2: Experiments done to determine *sp_threshold*

Question Answering Module Experimental Details: Our QA model consists of a BERT model with one linear layer with three output neurons; one to predict the question type yes/no/span, and the other two predict the start index and the end index of a potential span answer. During test, if the "yes" probability was highest the output is "yes", and similarly for "no", the start and end indices are ignored. If "span" had the highest probability, then we take the maximum probability start and end indices, and out the corresponding text span in the data. The original pre-trained BERT weights were loaded to jump start the training process and all of the weights were left unfrozen during training. During training, all weights except for the batch normalization layer and the biases have weight decay of 0.01. The loss function is the average of three Cross Entropy Losses (yes/no/span, start, end). The BertAdam optimizer was used during training with a learning rate of 3e-5. Due to computational power constraint, for most experiments we ran two epochs of mini-batch gradient descent with a batch size of 6.

The experiments performed on this module focus on two dimensions: 1) Whether or not the model supported yes/no questions (our model at the checkpoint did not support this), and 2) Whether the model was trained on SQuAD, HotpotQA, or both. For the models that don't support yes and no questions, the model is only trained to output start and end logits, since it is a simpler model and only 6% of all questions are of type yes/no. Therefore, we wanted to experiment if the additional output neurons truly improves the model's overall performance. The training loss from each can be found in Figure 4 in the Apprendix. Note that another important parameter that was kept constant during these experiments is the *qa_sp_threshold*=0.2, which for the Supporting Facts Classifier outputs, how confidence threshold in which to output a sentence as a supporting fact to the QA module. We chose a lower threshold here as we prefer to have a higher recall than specificity - the QA networks can learn to ignore false positives, but we needed to reduce false

Table 1: Performance of pipelined versions of JAM on dev set compared to baseline

Metric	Training Data Source	**Answer**		**Supporting Facts**		**Joint**	
		EM	F1	EM	F1	EM	F1
JAM	**SQuAD+HotpotQA**	**54.68**	**68.64**	**47.95**	**79.79**	**29.39**	**57.40**
JAM	HotpotQA	53.64	67.53	47.95	79.79	28.71	56.45
JAM†	SQuAD+HotpotQA	51.32	64.85	47.95	79.79	27.46	53.63
JAM†	HotpotQA	50.32	63.54	47.95	79.79	20.93	52.94
Baseline‡	HotpotQA	44.44	58.28	21.95	66.66	11.56	40.86

†Model without prediction of yes/no questions
‡As implemented by Yang et al. [2018]

negatives to as low as possible, as the QA model cannot get a correct answer if the corresponding supporting fact is not provided.

4.4 Results

Supporting Facts Classifier Module Results: Table 3 (Appendix) shows the results of our final Supporting Facts Classifier compared to our checkpoint model and HotpotQA's baseline. The final result of EM 47.95 and F1 79.79 represent 15 points improvements over the baseline. In all cases, the training converged and stopped at between 5 to 8 epochs. For our chosen *sp_threshold*=0.49, our classification precision is 85.79 and recall is 77.86. The recall can be increased arbitrarily by lowering the *sp_threshold*.

Our results are expected, as we are using a stripped down version of the baseline model in order to only predict supporting facts. The improvement over the baseline could be attributed to increased specialization of the network in predicting only supporting facts. The baseline model's loss function was a combination of three factors, while ours is focused solely on supporting facts. However, it should be noted that most of the improvement came from switching the optimizer from SGD to Adam and tuning parameters, as our checkpoint model has the same architecture as the final. At the project checkpoint we were still using the default SGD Optimizer, which resulted in the loss converging at 0.08, compared to 0.05 for Adam with tuned hyperparameters in the final.

Question Answering Module Results: Table 1 shows the results these various experiments. Supporting the yes/no question type in HotpotQA led to a 3% absolute gain in EM and F1, which suggests that around half of the yes/no questions were answered correctly. Furthermore, starting from only training on SQuAD, we achieve a consistent increase in performance when going from only training on HotpotQA, to training on both. Our best performing model, which involves supporting yes/no questions and training first on SQuAD then on HotpotQA, achieved F1 54.68 and EM 68.64, represents a F1 9.08 and EM 9.62 absolute increase from the baseline in producing answers, and an even greater increase in the joint performance.

5 Analysis

Table 2: Performance of Supporting Facts Classifier by Question Type

Model	**Bridge**		**Comparison**	
	EM	F1	EM	F1
JAM	43.09	77.52	67.31	88.79
Baseline	43.41	59.09	48.55	55.05

Table 2 shows the performance of our supporting facts classifier by question type. Contrary to the baseline model, which has similar performance for both question types our model performs significantly better in comparison questions compared to bridge. We suspected that one factor that could contribute to this is the length of each question, considering that bridge entity questions tend to be longer (Figure 5 (Appendix)), yet are not associated to an increased number of supporting facts (Figure 6(Appendix)) and are less likely to include the named entity within the text. Figure 7 (Appendix) shows the performance metrics on the dev set by question type and relative length of question. Although in the bridge questions there is a relatively lower performance compared to comparison questions, the performance is similar across question lengths. However, the EM on shorter as opposed to longer questions is markedly different in the case of comparison questions. Comparing the first and fifth quintile, fespite the fact that our model recognizes the correct paragraphs in a similar proportion (97.5% vs 95.4% of cases, respectively), the proportion of cases where our model gathers all of the correct facts (ignoring false positiveS) is 86.9% vs 71.8% respectively. We have implemented dot product bi-attention, which grows in size with similarity

of context and query. The longer queries are less similar to the text, even though they are more likely to contain the named entities, and thus may receive lower attention scores. Thus, a slightly more complex attention model would be helpful in this regard, such as multiplicative attention. Furthermore, the relatively low improvement on bridge question types compared to the baseline model likely reflects that our model architecture is not significantly different enough to model complex reasoning over an unnamed entity. A more complex architecture is likely to be necessary to successfully perform on this type of reasoning.

A random selection of 50 examples with incorrect matches were chosen the dev set and compared with our best performing model's predictions. An example of each error subtype can be seen in Table 4 (Appendix). Nearly half of these errors correspond to nearly precise but yet innacurate span answers. This may be because of our lack of weighting mechanism in the calculation of the loss for the fully implemented model, and may improve with further fine tuning such as implementing a weighted average of the loss. Another important source of error corresponds to innacurate calculations on numerical values. This may be because only about 20% of the data corresponds to comparison questions, thus, incrementing the amount of training questions that involve comparing quantities may increase performance in future implementations. Also, the tokenization used by our model could be better designed to account for numerical values and vector build representations that conserve their arithmetic properties. A final important source of error was lacking the key supporting fact that changed the answer of a comparison question. We believe an implementation of current well performing models, including ensemble methods and/or modifications of BERT to identify supporting facts could be implemented and show improved results in this regard.

For the QA model, we achieved the best results when we first trained our model with the SQuAD dataset then finetuned it with the Hotpot dataset. Since the yes/no questions only take up a tiny proportion of the entire Hotpot dataset, we forced our model to focus it's attention to only the span questions by using the SQuAD dataset. After the training loss for the span questions from SQuAD has coverged, we then fine tune our model with the Hotpot dataset by introducing the yes/no questions. As shown in 4 (Appendix), training for SQuAD dataset converges at a lower loss as compared to training with Hotpot. Using the SQuAD pretrained weight to continue training out model on the hotpot dataset gave us the lowest loss, which translates to our model's overall performance as well.

6 Conclusion

Our pipelined model approach to the HotpotQA dataset shows a marked improvement in all metrics in HotpotQA over the baseline model (see Table 1). This success could be explained by three main benefits of a pipelined model: **1)** Ability for the two models to specialize on separate tasks with separate loss functions **2)** Increased interpretability of results - we can visualize the supporting facts intermediate results before feeding it on to the QA model. The two models could be optimized separately and brought together without conflict. **3)** The ability to use a large and complex model for the QA Module. The Supporting Facts Classifier heavily filters the input, so that the QA module needs to process 2-3 sentences compared to dozens of sentences. This opens up the possibility of using models such as BERT with a large number of parameters and would have been difficult to train on the full ten paragraphs. Not all improvements were due to pipelining - experimenting with optimizers, hyperparameters, and the introduction of BERT all significantly boosted our results. However, without pipelining it would have been both computationally intractable to experiment with all of these variations.

On top of the improvement in model performance and interpretability, we demonstrated that operating two large models in parallel at test time, although requiring careful partitioning of GPU resources, was computationally feasible (making this work was in fact, a highly time-consuming task). However, the requirement of two large GPUs at the same time is a limitation of this approach.

Further limitations of this work include the more abstract concept of the purpose of artificial intelligence. Are we doing a disservice to "true language understanding" by pursuing a pipelined approach? In theory shouldn't a single model that truly understands language be able to solve both the supporting facts classification and QA task at the same time? With unlimited computational resources and data, an end-to-end approach would perhaps still prevail and be preferable.

A future direction is to modify our system to operate with the full-wiki data from HotpotQA, which would be a perfect use case of our pipeline model, since the supporting facts classifier will filter out most of the noise, and the QA model can still operate with a tractable few sentences. We believe techniques like locality sensitive hashing [Satuluri and Parthasarathy, 2012] could prove useful to quickly initially screen for a subset of text that our model could then evaluate. We would also like to brainstorm other avenues and datasets where JAM could be useful - for example, first summarizing and then finding key words in a text. Any natural language processing task that requires a "filtering" step and then a more focused "answer generation" step could benefit from our general approach.

7 Additional Information

Our Mentor is Peng Qi.

References

Quac. URL https://quac.ai.

Squad2.0. URL https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac : Question answering in context. *CoRR*, abs/1808.07036, 2018. URL http://arxiv.org/abs/1808.07036.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551, 2017. URL http://arxiv.org/abs/1705.03551.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015. URL http://arxiv.org/abs/1508.06615.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107, 2017. URL http://arxiv.org/abs/1708.00107.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Venu Satuluri and Srinivasan Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *Proc. VLDB Endow.*, 5(5): 430–441, January 2012. ISSN 2150-8097. doi: 10.14778/2140436.2140440. URL http://dx.doi.org/10.14778/2140436.2140440.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *CoRR*, abs/1710.06481, 2017. URL http://arxiv.org/abs/1710.06481.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

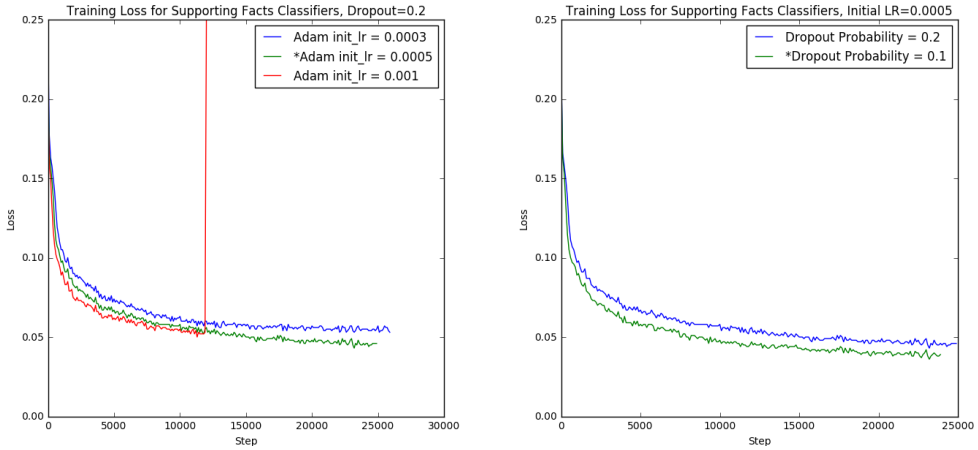
Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL http://arxiv.org/abs/1804.09541.

8 Appendix

Table 3: Pogress on Supporting Facts Classifier Module Results

Metric	**Supporting Facts**	
	EM	F1
Our Final	**47.95**	**79.79**
Our Checkpoint	22.72	67.73
†Baseline	20.32	64.49

†As implemented by Yang et al. [2018]



(a) Different Initial Learning Rates (b) Different Dropout Probabilities

Figure 3: Experiments done to determine Adam initial learning rate and dropout probability.

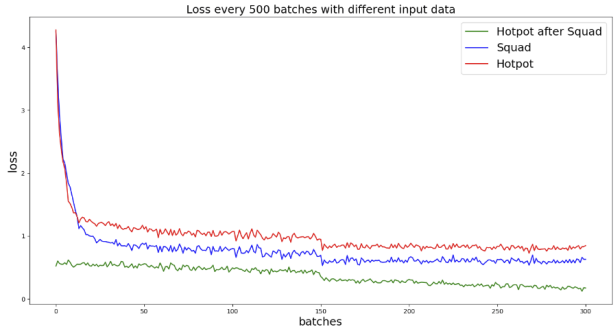


Figure 4: Question answering module training loss every 500 batches for different input datasets

Table 4: Error causes of 50 randomly selected errors from dev set by subtype

Error type/Cause	Amount	Example
Innacurate (but close) span length	23	**Question:** John T. Mather Memorial Hospital is located in a village in what Suffolk County, New York town? **Supporting Facts:** John T. Mather Memorial Hospital is a general hospital located in Port Jefferson, New York. Port Jefferson (informally known as Port Jeff) is an incorporated village in the Town of Brookhaven in Suffolk County, New York on the North Shore of Long Island. **Gold/Model Answer:** Town of Brookhaven /Brookhaven
Incorrect quantitative comparison	9	**Question:** Which came out first, Dinosaur or McFarland, USA **Supporting Facts:** Dinosaur is a 2000 American CGI animated adventure film produced by Walt Disney Feature Animation and The Secret Lab and released by Walt Disney Pictures. McFarland, USA (also known as McFarland) is a 2015 American sports drama film directed by Niki Caro, produced by Mark Ciardi and Gordon Gray, written by Christopher Cleveland, Bettina Gilois and Grant Thompson with music composed by Antônio Pinto. **Gold/Model Answer:** Dinosaur/McFarland, USA
No agreement because of conflicting evidence/More than one right answer	6	**Question:**Swiss music duo Double released their best known single "The Captain of Her Heart" in what year? **Supporting Facts:** Double (pronounced "doo-blay") was a Swiss music duo best known for their hit single "The Captain of Her Heart". In addition to containing updated versions of two of the band's earlier singles ("Woman of the World" and "Rangoon Moon"), the album included the international smash hit, "The Captain of Her Heart", a plaintive, atmospheric, piano-led ballad which was an immediate success throughout Europe upon its 1986 single release. **Additional supporting fact from model:**"The Captain of Her Heart" is a single by the Swiss duo Double in 1985. **Gold/Model Answer:** 1986/1985
External knowledge of cultural representation of text required	3	**Question:** Who died last Vladimir Arnold or Georg Cantor? **Supporting Facts:** Georg Ferdinand Ludwig Philipp Cantor (March 3 [O.S. February 19] 1845 – January 6, 1918) was a German mathematician. Vladimir Igorevich Arnold (12 June 1937 – 3 June 2010) was a Soviet and Russian mathematician. **Gold/Model Answer:** Vladimir Igorevich Arnold/Georg Ferdinand Ludwig Philipp Cantor
Missing necessary supporting facts for prediction	7	**Question:**What party was the secretary of commerce and housing under the 42 Governor of Kansas? **Supporting Facts:** Robert G. "Bob" Knight (born July 31, 1941) was the Republican mayor of Wichita, Kansas for seven terms. He also served under Democratic Governor Joan Finney as Kansas Secretary of Commerce and Housing. Joan Finney (February 12, 1925 – July 28, 2001), served as the 42nd Governor of Kansas from 1991 to 1995. **Predicted Supporting Facts:**Joan Finney (February 12, 1925 – July 28, 2001), served as the 42nd Governor of Kansas from 1991 to 1995. He also served under Democratic Governor Joan Finney as Kansas Secretary of Commerce and Housing. **Gold/Model Answer:** Republican/ Democratic
Question not answerable with gold supporting facts	2	**Question:**How does this series represent a function, which is also used in the Homotopy analysis method? **Supporting Facts:** In mathematics, a Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. This is enabled by utilizing a homotopy-Maclaurin series to deal with the nonlinearities in the system. **Gold/Model Answer:** infinite sum of terms/Taylor series
Model failed to reduce length of supporting fact	1	**Question:**Which former New York senator did Marc Elias help run for office? **Supporting Facts:** Marc E. Elias was the general counsel for Hillary Clinton's 2016 presidential campaign. The 2016 presidential campaign of Hillary Rodham Clinton was announced in a YouTube video, on April 12, 2015. She was previously the United States Senator from New York from 2001 to 2009, and is the wife of former President Bill Clinton. **Gold/Model Answer:** Rodham Clinton/Marc E. Elias was the general counsel for Hillary Clinton's 2016 presidential campaign.

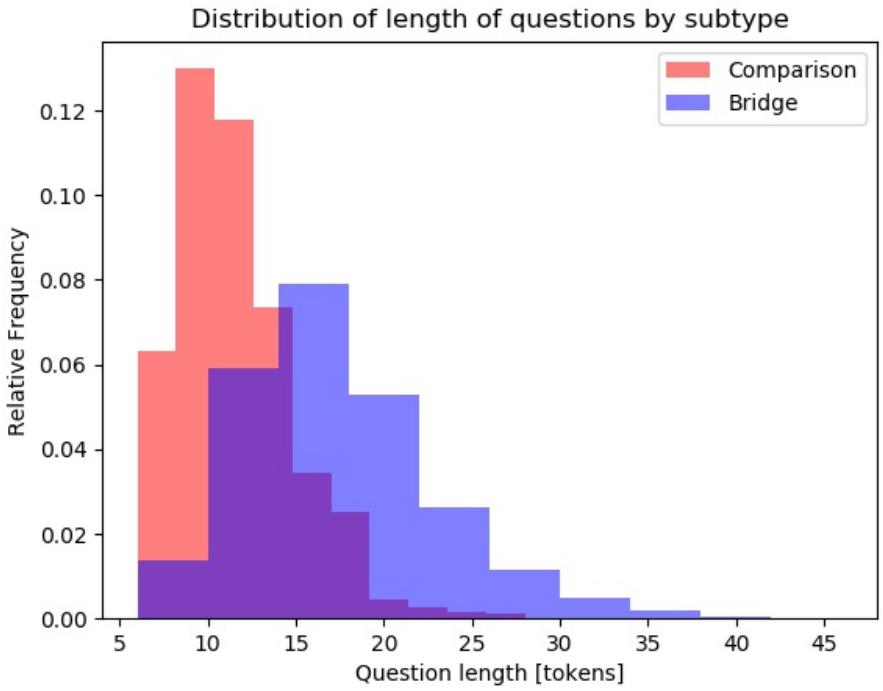


Figure 5: Length of question on dev set in quantity of tokens differs by question type

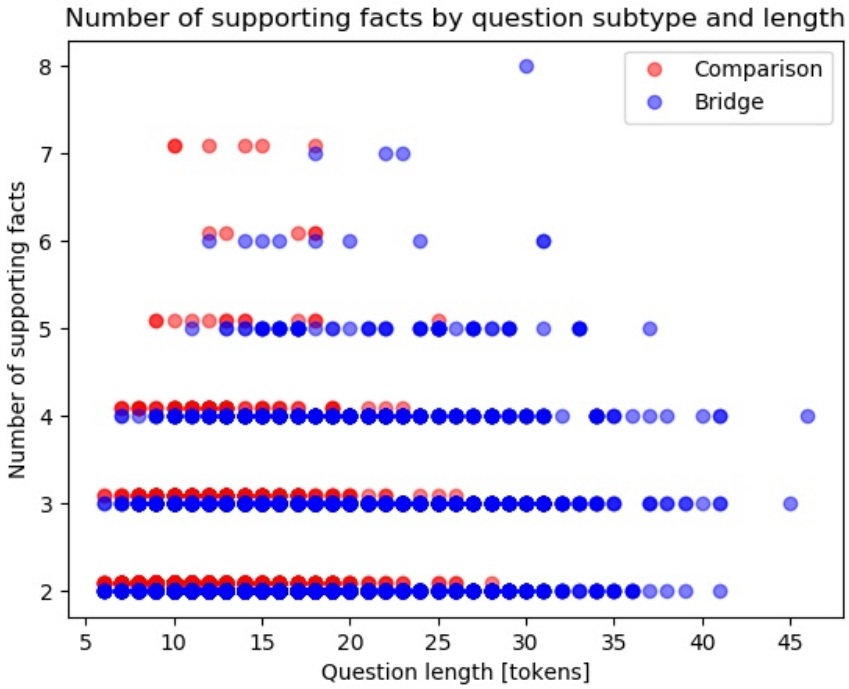


Figure 6: Amount of supporting facts by question length in dev set

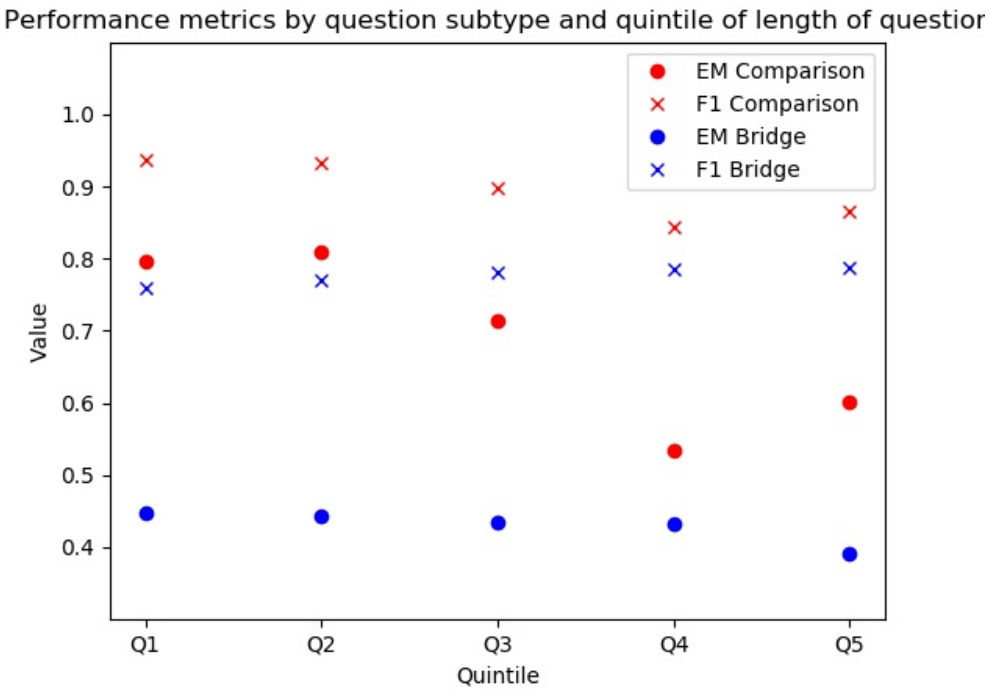


Figure 7: Performance metrics by quintiles of question lengths by subtype