Translucent Answer Predictions in Multi-Hop Reading Comprehension

G P Shrivatsa Bhargav,[2*] Michael Glass,[1*] Dinesh Garg,[1] Shirish Shevade,[2]
Saswati Dana,[1] Dinesh Khandelwal,[1] L Venkata Subramaniam,[1] Alfio Gliozzo[1]
[*]Equal contribution
[1]IBM Research AI, [2]Dept. of CSA, IISc, Bangalore
{mrglass, gliozzo}@us.ibm.com, {bhargavs, shirish}@iisc.ac.in, {garg.dinesh, sadana04, dikhand1, lvsubram}@in.ibm.com

Abstract

Research on the task of Reading Comprehension style Question Answering (RCQA) has gained momentum in recent years due to the emergence of human annotated datasets and associated leaderboards, for example `CoQA`, `HotpotQA`, `SQuAD`, `TriviaQA`, etc. While state-of-the-art has advanced considerably, there is still ample opportunity to advance it further on some important variants of the RCQA task. In this paper, we propose a novel deep neural architecture, called TAP (Translucent Answer Prediction), to identify *answers* and *evidence* (in the form of supporting facts) in an RCQA task requiring multi-hop reasoning. TAP comprises two loosely coupled networks – *Local and Global Interaction eXtractor (LoGIX)* and *Answer Predictor (AP)*. LoGIX predicts supporting facts, whereas AP consumes these predicted supporting facts to predict the answer span. The novel design of LoGIX is inspired by two key design desiderata – *local context* and *global interaction*– that we identified by analyzing examples of multi-hop RCQA task. The loose coupling between LoGIX and the AP reveals the set of sentences used by the AP in predicting an answer. Therefore, answer predictions of TAP can be interpreted in a translucent manner. TAP offers state-of-the-art performance on the `HotpotQA` (Yang et al. 2018) dataset – an apt dataset for multi-hop RCQA task – as it occupies **Rank-1** on its leaderboard (`https://hotpotqa.github.io/`) at the time of submission.

1 Introduction

Natural language understanding has been one of the key drivers responsible for advancing the field of AI. To this end, automated Question Answering (QA) has served as an effective way of measuring the language understanding capabilities of AI systems. The field of QA is vast and can be classified along many different dimensions, including (i) knowledge based (Unger et al. 2014) vs. text based (reading comprehension) (Joshi et al. 2017), (ii) extractive (Rajpurkar et al. 2016) vs. abstractive (Lai et al. 2017), (iii) short (Rajpurkar et al. 2016) vs. long answers (Kwiatkowski et al. 2019), (iv) single-hop (Rajpurkar et al. 2016) vs. multi-hop reasoning (Yang et al. 2018), etc. Our focus in this paper is on Reading Comprehension style Question Answering (RCQA) task. Reading comprehension is the ability to answer questions over the supplied natural language passages, where a passage comprises one or more paragraphs. Specifically, we focus on *complex* questions that require multi-hop reasoning over facts spread across multiple passages.

Question: Brown State Fishing Lake is in a county that has a population of how many inhabitants ?
Context:
Passage 1: "Fishing Lake"
Fishing Lake is a lake in the Canadian province ...
...
Passage 3: "Brown State Fishing Lake"
Brown State Fishing Lake (sometimes also known as Brown State Fishing Lake And Wildlife Area) is a protected area in Brown County, Kansas in the United States.
The lake is 62 acres in area and up to 13 feet deep. The area was formerly known as Brown County State Park, and is 8 miles (13 km) east of Hiawatha, Kansas .
...
Passage 10: "Brown County, Kansas"
Brown County (county code BR) is a county located in the northeast portion of the U.S. state of Kansas .
As of the 2010 census, the county population was 9,984 .
Its county seat and most populous city is Hiawatha ...
Answer: 9,984

Figure 1: Output of the TAP model on a test example from `HotpotQA`, demonstrating its effectiveness in capturing **local context** as well as **global interaction** among sentences so as to identify supporting facts. The intensity of the highlight is proportional to the probability (predicted by our model) of the sentence being a supporting fact.

Recently, there has been a surge in the research activities surrounding RCQA task, primarily due to the emergence of large-scale public datasets such as `CoQA` (Reddy, Chen, and Manning 2019), `HotpotQA` (Yang et al. 2018), `SQuAD` (Rajpurkar et al. 2016), `TriviaQA` (Joshi et al.

2017), etc. Most of these datasets require a system to pick a short answer span from within the given context passage. For single-hop RCQA datasets (for example, SQuAD), majority of the proposed solutions are based on massively pre-trained Transformer-style models such as BERT (Devlin et al. 2019), XLNet (Yang et al. 2019), and RoBERTa (Liu et al. 2019). Some of these solutions have exhibited human level performance. Similar solutions have been proposed for the multi-hop RCQA datasets also and they have also improved the state-of-the-art. However, we believe that the core challenges involved in the multi-hop RCQA task have not been addressed effectively by existing solutions and hence there is an opportunity to advance the state-of-the-art.

Figure 1 illustrates, by example, the key challenges involved in solving the multi-hop RCQA task. First, we observe that only a small subset of sentences, $\mathcal{S}$, from the given passage set (context) is needed to answer the question. The sentences in such a subset are typically called *Supporting Facts (SF)* (Yang et al. 2018). Furthermore, we observe that whether a given sentence $\langle s \rangle$ is a supporting fact or not depends on two factors: (i) how much the information present in the sentence $\langle s \rangle$ is relevant to the given question $\langle q \rangle$, (ii) the presence of other sentences $\langle s' \rangle$ in the context *related* to the sentence $\langle s \rangle$; and whether these related sentences $\langle s' \rangle$ are supporting facts or not. The first aspect is relatively easier to handle but the second can be quite intimidating when it comes to designing the right solution approach. For example, in Figure 1, the second sentence of passage 10 seems irrelevant to the question when it is examined in isolation. But, in the presence of the first sentences of passages 3 and 10, it becomes extremely relevant.

1.1 Design Desiderata

In light of the challenges discussed above, we believe any effective solution for the multi-hop RCQA task needs to address the following two *design desiderata.*

(1) Local Context: Each sentence should be understood in the context of its neighboring sentences and the question.

(2) Global Interaction: A global (aka long-range or inter-passage) interaction among sentences must be identified and used in predicting the set $\mathcal{S}$ of supporting facts.

To meet these design desiderata, we propose a deep neural architecture, called TAP (Translucent Answer Prediction). TAP comprises two loosely coupled networks – Local and Global Interaction eXtractor (LoGIX) and Answer Predictor (AP). LoGIX predicts a set of supporting facts, say $\hat{\mathcal{S}}$, whereas AP takes these predicted supporting facts as input and predicts the answer span. This two stage pipeline reveals the subset of sentences $\hat{\mathcal{S}}$ that were used to predict the answer and therefore, allows us to *interpret* the predictions made by TAP in a *translucent* manner[1]. LoGIX comprises a novel 2-level hierarchical architecture to address the design desiderata identified before. The first layer, called the Local Layer, captures the *local context* whereas, the second layer, called the Global Layer, captures the *global interaction* among sentences.

Key contributions of this paper are follows – (1) We propose an architecture called TAP to solve multi-hop RCQA task. (2) LoGIX of TAP is a novel hierarchical architecture that effectively captures the local context and the global interactions between the sentences. (3) The proposed solution achieves state-of-the-art performance on the `HotpotQA` dataset. Our submission, called TAP 2 (ensemble) and TAP 2 (single model), achieved **Rank-1** and **Rank-2**, respectively, on the leaderboard of `HotpotQA` at the time of submission.

2 Related work

Datasets: In recent years, many high-quality large scale public datasets have emerged. Datasets like SQuAD (Rajpurkar et al. 2016) provide single passage as the context to each question whereas, datasets like MS MARCO (Bajaj et al. 2017) and TriviaQA (Joshi et al. 2017) provide multiple large documents as the context. A question in these datasets can usually be answered using a single passage. A more challenging setting, popularly known as *multi-hop* RCQA, is one where the information required to answer the question is spread over multiple, disconnected passages. The `HotpotQA` (Yang et al. 2018) dataset is designed precisely for the multi-hop RCQA task. Similarly, in the QAngaroo (Welbl, Stenetorp, and Riedel 2018) dataset, the questions are such that multiple documents are required to answer the questions. We evaluate our model on `HotpotQA` because it is the only one that has annotations for supporting facts.

Pre-trained Transformers: Transformer based models such as BERT (Devlin et al. 2019), XLNet (Yang et al. 2019), GPT-2 (Radford et al. 2019) and RoBERTa (Liu et al. 2019) have dominated various RCQA task leaderboards (Rajpurkar et al. 2016; Wang et al. 2019), especially where the context size is relatively small.

Cascade Models: When the context size becomes larger than what can be taken as input by many of these Transformer based models, a well known alternative approach is to use cascade style models. For end-to-end question answering, Chen et al. (2017), Wang et al. (2018a), Lin et al. (2018), Yan et al. (2019), Clark and Gardner (2018) use an IR technique to first shortlist N documents. Next, one or more steps of passage selection is performed to select K passages. Then a neural model is used to predict an answer candidate from each of these K passages independently. The final answer is selected from these candidate answers. Wang et al. (2018b) propose a method to rank the answer candidates according to the evidence supporting them. Swayamdipta, Parikh, and Kwiatkowski (2018) present a cascade model consisting of light-weight submodules which can be trivially parallelized to speed up training and inference. These models encode passages independently. As a consequence, any dependencies that exist between sentences of different passages is not captured.

[1]The reason behind calling our model TAP.

Choi et al. (2017) encode the context into a single vector by using hard attention over the sentence representations. Hewlett et al. (2017) summarize the context document into a single vector by using hierarchical attention over the passage representations. This vector is used by a sequence model to generate the answer. While Choi et al. (2017) ignores inter-sentence dependencies, the design of Hewlett et al. (2017) can capture long and short term dependencies in theory. In our work, we prioritize facilitating this. At a macro level, the cascade model which is most similar to ours is described in Min et al. (2018). Like TAP, they too have a sentence selector and an answer predictor. The key difference lies in the sentence selector. They encode each sentence in isolation and predict whether the sentence is a supporting fact. While this strategy was shown to be effective for some factoid questions, it is not sufficient for multi-hop reading comprehension. The intuition behind this was described in the introduction and an ablation study (Table 3) supports it.

Multi-hop Reasoning: Graph based approaches are widely used for the purpose of multi-hop reasoning. DFGN (Qiu et al. 2019) constructs a graph out of the information given in the context and employs a combination of reasoning over the graph and text to answer the question. Cognitive Graph QA (Ding et al. 2019) is built for end-to-end question answering. It consists of a reader and a reasoner which iteratively fetch and read new information and reason over them until the answer is found.

Talmor and Berant (2018) and Min et al. (2019) follow the approach of decomposing a complex question into multiple single-hop questions. The final answer is composed from the answers to these simple questions.

Nishida et al. (2019), Qiu et al. (2019), and Feldman and El-Yaniv (2019) follow a multi-task learning approach to simultaneously extract the supporting facts and answer the question. There is no guarantee that any of the predicted supporting facts are even used by the model to arrive at the answer. In TAP we create a hard dependency between the predicted supporting facts and answer prediction.

3 Problem Statement

Suppose we are given a set of training data $\{Q_i, C_i, A_i\}_{i=1}^{n}$, where Q_i is a question (in natural language text), C_i is a set of possibly relevant passages called the context, and A_i is the answer span within C_i. The answer span A_i is typically expressed via start and end indexes $< start_i,\ end_i >$.

Each context C_i is composed of multiple passages denoted by $\langle P_{i1}, P_{i2}, \ldots \rangle$. Each of these passages P_{ij} is a sequence of sentences S_{ijk}, each of which may or may not express a supporting fact indicated by a binary label F_{ijk}. For training data, we assume the labels F_{ijk} are made available.

Our goal is to train a deep-net model that can predict supporting facts (i.e. labels $F_{test,jk}$ of individual sentences) as well as (start, end) indexes $\langle start_{test},\ end_{test} \rangle$ of the answer A_{test} for a test example $\{Q_{test}, C_{test}\}$.

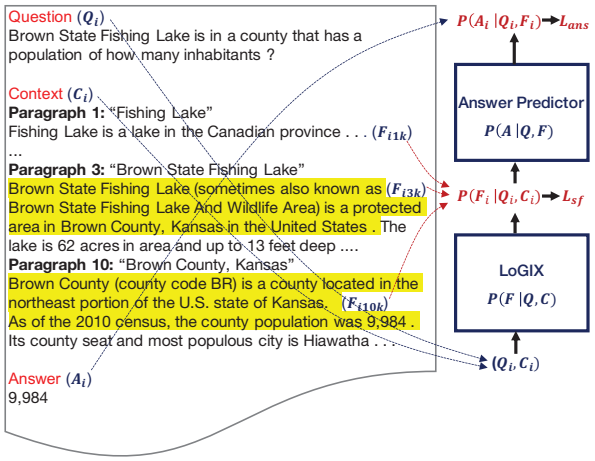

Figure 2: A high level architecture of the TAP network

4 Solution Approach

The idea behind our solution approach is shown in Figure 2 which depicts the architecture of the proposed TAP network at a high level. TAP system is composed of two networks: *Local and Global Interaction eXtractor (LoGIX)*, and *Answer Predictor (AP)*. Input to LoGIX is a pair (Q_i, C_i) of question and context and the output of LoGIX comprises, for each sentence S_{ijk}, a score $p(F_{ijk} = 1)$ denoting the probability of it being a supporting fact. The input to the AP is a *pseudo-passage* formed by concatenating the supporting facts selected by a threshold on LoGIX's output. The output of the AP is a predicted answer A_i (given by indexes $\langle start, end \rangle$) or one of the special answer types: yes or no.

4.1 Achieving Translucency via Loose Coupling

In the proposed model, it is guaranteed that the prediction of the answer is influenced only by the supporting facts that LoGIX predicts. Therefore, we can easily understand the accurate (and inaccurate) predictions made by the AP. Specifically, we can argue whether the prediction was correct because the supplied supporting facts were having high precision and recall or because of some spurious correlation that the model has learned during training. Studies like Jia and Liang (2017) have shown that neural reading comprehension models often exploit spurious correlations in the data to predict the answer. By predicting the supporting facts as the first step, our approach limits the possibility of such correlations. It also becomes easier to detect spurious correlations during model development time since the Answer Predictor is fed only a few sentences rather than the entire context.

5 Architecture and Training of TAP

Figure 3 captures the finer details of the Answer Predictor and LoGIX in the TAP network.

5.1 LoGIX

LoGIX has three key layers – (1) **Local Layer:** Responsible for capturing the local context (intra-passage sentence dependencies). It is replicated over k super-passages (defined

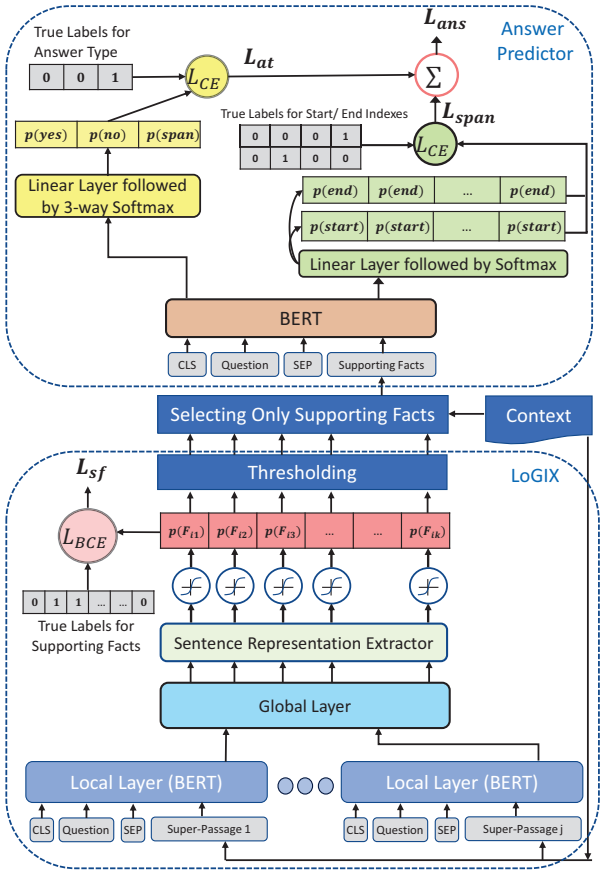

Figure 3: Detailed architecture of the proposed TAP network

later), (2) **Global Layer:** Responsible for capturing global interactions (inter-passage or long range dependencies between sentences), (3) **Supporting Facts Prediction Layer:** Responsible for extracting sentence representations and predicting the probability that a sentence is a supporting fact.

Relative to capturing only local context, the Global Layer provides an important performance boost as we demonstrate in the Experiments section. Relative to using only the Global Layer, splitting the context into m_1 super-passages enables a wider and deeper transformer architecture since its time complexity is quadratic in the length of the input sequence.

Input Data Shaping For training LoGIX, we are required to process the given training data $\{Q_i, C_i\}_{i=1}^{n}$ and shape them into a suitable form. The available GPU memory size becomes a key determining factor while shaping these inputs. The limited position embeddings of BERT ($E_p \in \mathbb{R}^{p \times d}, p = 512$) also factors into the shape of the input, as well as the computational cost of applying large transformer networks on long sequences.

For each training instance $\{Q_i, C_i\}$, the question Q_i is first tokenized. The set of passages in the context C_i is split on passage boundaries into $m_1 = 4$ disjoint pieces such that the length of each piece is equal to $m_0 = p - |Q_i|$ (pad or trim if necessary). Each such piece is called a *super-passage*. The super-passages are constructed to fit the context into minimal number of copies of the Local Layer and they capture a bit of cross-passage interactions wherever possible. The super-passages can be thought of as bins with a fixed capacity (512 tokens) and the passages are selected such that each bin is filled to maximum capacity and no passage is split across super-passages. Each of these super-passages is prefixed with the special $[CLS]$ token, followed by the question sequence, and the special token $[SEP]$ (as shown in Figure 3) so as to obtain a sequence ϕ_{ij} (of length $p = 512$), where $j = 1 \rightarrow m_1$. This $\{\phi_{ij}\}_{j=1}^{m_1}$ is the input to the LoGIX for the training instance $\{Q_i, C_i\}$.

Local Layer The Local Layer of LoGIX is composed of a Transformer (Vaswani et al. 2017) that is capable of generating the vector representations of the input token sequence ϕ_{ij} by capturing semantics across the sequence. In our specific model, we used BERT. That is, each ϕ_{ij} constructed previously is passed through a pre-trained BERT (Devlin et al. 2019) to obtain a sequence of token vectors each of d dimensions. By using the self attention mechanism, BERT captures the intra-passage dependencies among context sentences as well as the question. Because these super-passages typically contain multiple passages, this phase also captures some inter-passages dependencies. Out of this sequence of token vectors, we take the trailing sub-sequence corresponding to only the passage tokens (ignoring the question and special $[CLS], [SEP]$ tokens). This sub-sequence of vectors is denoted by $\{\boldsymbol{r}_q\}_{q=1}^{m_0}$.

Global Layer The sequence of vectors $\{\boldsymbol{r}_q\}_{q=1}^{m_0}$ obtained from encoding each ϕ_{ij} are concatenated to obtain $\{\boldsymbol{r}_q\}_{q=1}^{m_0 \times m_1}$. This sequence is fed to the Global Layer, composed of ℓ layers of Transformers (Vaswani et al. 2017) to obtain another sequence of vectors $\{\boldsymbol{u}_q\}_{q=1}^{m_0 \times m_1}$, where ℓ is a hyperparameter. Since the entire context is visible to the transformer at once, the self-attention mechanism in the transformer captures inter-passage dependencies between the sentences.

Supporting Facts Prediction Layer The vectors $\{\boldsymbol{u}_q\}_{q=1}^{m_0 \times m_1}$ are fed to a module called the *Sentence Representation Extractor (SRE)*. From the input sequence $\{\boldsymbol{u}_q\}_{q=1}^{m_0 \times m_1}$, SRE picks the vectors corresponding to the first token $\langle start \rangle_{ik}$ and the last token $\langle end \rangle_{ik}$ of each sentence S_{ik} in the input context. SRE uses these vectors to predict the probability that the sentence is a supporting fact. The probability of a sentence being a supporting fact is given by $p(F_{ik} = 1) = sigmoid(\boldsymbol{w}^\top [\boldsymbol{u}_{\langle start \rangle_{ik}}; \boldsymbol{u}_{\langle end \rangle_{ik}}] + b)$, where $[\cdot \ ; \ \cdot]$ denotes the concatenation operation and $\boldsymbol{w} \in \mathbb{R}^{2d}, \boldsymbol{b} \in \mathbb{R}$ are trainable parameters.

LoGIX is trained using the binary cross entropy loss between the true supporting fact labels F_{ik} and the predicted $p(F_{ik} = 1)$ as $L_{BCE}(ik) = F_{ik} \cdot p(F_{ik=1}) + (1 - F_{ik}) \cdot (1 - p(F_{ik} = 1))$ for each sentence S_{ik}. The total loss for LoGIX is $L_{sf} = \sum_{S_{ik}} L_{BCE}(ik)$.

5.2 Answer Predictor

The role of the Answer Predictor (AP) is to answer the question by reasoning over the facts received from LoGIX. For

this purpose, we train the AP on a subset of the context which is known to contain all the gold supporting facts.

Input Data Shaping For the i^{th} training instance, we first prepare a pseudo-passage P_i' by concatenating all the supporting facts in the context. Each of these pseudo-passage P_i' is prefixed with the special $[CLS]$ token, followed by the question sequence, and the special token $[SEP]$ (see Figure 3) to obtain a sequence ϕ_i' (padded to length p).

Context Encoding The sequence ϕ_i' is fed to another instance of pre-trained BERT to obtain a sequence $\{\boldsymbol{v}_q\}_{q=1}^{p}$ of d dimensional vectors.

Answer Type Predictor One part of the Answer Predictor (yellow colored part in Figure 3) deals with identifying the answer type for the input question. It classifies the given input question into one of the three classes, namely $c = \{yes, no, span\}$, depending on the nature of its answer. For each input data instance (Q_i, P_i'), it uses the vector $\boldsymbol{v}_{CLS}$ in $\{\boldsymbol{v}_q\}_{q=1}^{p}$ corresponding to the $[CLS]$ token to assign the probability $p(c_i|Q_i, P_i')$ for each of the three classes using a single linear-layer followed by 3-way softmax. That is $p(c_i|Q_i, P_i') = softmax(\mathbf{W}_{CLS}^{\top}\boldsymbol{v}_{CLS})$, where $\mathbf{W}_{CLS} \in \mathbb{R}^{3\times d}$ is a learnable parameter. The predicted probability vector $p(c_i|Q_i, P_i')$ and true answer type of the input question are used to compute a Cross-Entropy loss L_{at} for answer type.

Answer Span Predictor The sequence $\{\boldsymbol{v}_q\}_{q=1}^{p}$ is passed into two separate softmax functions each inducing a probability distribution across these tokens. One of these distributions, $p(start_q)$, corresponds to the start index probability and the other, $p(end_q)$, corresponds to the end index probability. At training time, using the true start and end index labels supplied by the input training example, we compute the cross entropy loss separately for both indexes and sum them to get the total cross entropy loss for the answer span.

5.3 Pre-training

We use two types of pre-training for the Local Layer of LoGIX and the transformer in the AP. First, we use the pre-trained models distributed by Devlin et al. (2019) which are trained using the Masked Language Model and Next Sentence prediction tasks. We extend the pre-training of these models with the Span Selection Pre-Training (SSPT) task developed by Glass et al. (2019).

5.4 Joint Training

We experimented with three styles of training for the Answer Predictor. First and most simply, the Answer Predictor can be trained independently of LoGIX, using the provided gold standard supporting facts. Alternatively, the Answer Predictor can be trained with the predicted supporting facts from LoGIX. This requires first training five folds of LoGIX, each trained on 80% of the training data. Then supporting fact predictions are made over the training set, with each model making predictions for the instances absent in its training set. The Answer Predictor is then trained to identify answers from these predicted supporting facts. We call this second style of training *Joint Training*. In the third style, the predicted supporting fact training set can be concatenated with the gold standard training set. We found the Joint Training to be the most effective. Training LoGIX took approximately 24 hours on 8 P100 GPUs. In the joint setting this training was done for each of the five folds. The Answer Predictor takes under 10 hours to train on 4 P100 GPUs.

5.5 Ensemble

We ensemble the best performing variations of our models for LoGIX and the Answer Predictor. In LoGIX, the ensemble's confidence for supporting facts is an average of the confidences predicted by each LoGIX model. In the Answer Predictor ensemble, each constituent model first ranks its answers, then each distinct answer is scored by the sum of the inverse of the rank each model assigns it. The ensemble predicts the answer with the highest sum-inverse-rank.

The models in the LoGIX ensembles are those derived by using only the pre-trained BERT models and the BERT models with pre-training extended by span selection. For the Answer Predictor ensemble, the models trained with the three variations of joint training are used.

5.6 Evaluation

For any given test example (Q_{test}, C_{test}), LoGIX predicts the probabilities $p(F_{ik})$ at the sentence level. By using an appropriate threshold τ, LoGIX converts these probabilities into binary labels for the sentences. The sentences with positive predicted labels are used by the AP.

The question is concatenated with the predicted supporting facts and fed to the AP. The AP first predicts the answer type by selecting the answer type class having maximum $p(c_i|Q_i, P_i')$. If the predicted answer type is yes or no then the answer is simply the class name itself. Otherwise, the AP predicts the answer span as follows. For all spans $[q_s, q_e]$ of the input token sequence, the corresponding answer span probability is $\log(p(start_{q_s})) + \log(p(end_{q_t}))$. The AP outputs the span for which this probability is maximum.

6 Experiments

`HotpotQA` is a large scale QA dataset focusing on *explainability* and *multi-hop reasoning*. This dataset comes with human annotated sentence level binary labels indicating which sentences are supporting facts for answering a given question. The *distractor setting* of `HotpotQA` provides ten passages as context for each question. Out of these ten, two passages have facts relevant to the question. However, to ensure the generality of our model, TAP does not exploit this fact. The remaining eight passages are included to *distract* the model. Table 1 shows some statistics on the training and development sets. The accuracy of the predicted answers and the supporting facts are measured using *Exact Match (EM)* and *F1* scores. The metrics *Joint EM* and *Joint F1* are used to jointly evaluate the performance on answers and supporting facts. We evaluate TAP on the hidden test set for the distractor setting of `HotpotQA` by submitting our system for evaluation. We also use the publicly available development set to explore the impact of decisions in our architecture.

Statistic	Training set	Dev. Set
Number of questions	90447	7405
Avg. question length (words)	17.81	15.72
Avg. answer length (words)	2.22	2.46
Avg. context length (words)	886.23	896.96
Avg. number of sentences	40.94	41.38
Avg. number of sup. facts	2.38	2.43

Table 1: HotpotQA dataset statistics

6.1 Experimental Setup

We use pre-trained BERT$_{LARGE}$ models. In the Global Layer of LoGIX, there are two transformer layers. For both networks we use the ADAM (Kingma and Ba 2015) optimizer with a maximum learning rate of 3×10^{-5} and a triangular learning schedule, warming up over the first 10% of training instances. Questions are truncated to 35 tokens and passages are truncated to 512 tokens. The total length of the passage set is limited to 2048 tokens, with the longest passages truncated to fit. We trained LoGIX for 4 epochs with a batch size of 8 and the Answer Predictor also for 4 epochs with a batch size of 16. PyTorch was used to develop TAP. The TAP code repository can be found at https://github.com/IBM/translucent-answer-prediction.

6.2 Results and Model Behavior Analysis

Method	ANSWER		SUP. FACTS		JOINT	
	EM	F1	EM	F1	EM	F1
TAP 2 ensemble	**66.64**	**79.82**	57.21	**86.69**	41.21	**70.65**
TAP 2 single model	64.99	78.59	55.47	85.57	39.77	69.12
EPS+ BERT	63.29	76.36	**58.25**	85.60	**41.39**	67.92
P-BERT	61.18	74.16	51.38	82.76	35.42	63.79
⋮						
DFGN	56.31	69.69	51.50	81.62	33.62	59.82

Table 2: Performance of TAP (ours) in comparison with the next closest and closest published models on the HotpotQA leader board. The results of the unpublished models have been borrowed from the HotpotQA leader board.

Main results In Table 2 we compare the results of our model TAP with other top models from the HotpotQA leaderboard. We perform the best on four of the six metrics.

Effect of Context Encoder in LoGIX We experimented with different architectures for the context encoder[2] of the LoGIX (Figure 3) in order to study its effect on selecting supporting facts. For this study, we used the pre-trained BERT$_{BASE}$. Each architecture captured intra and inter-passage sentence dependencies to different extents. As a starting point, we used just the Local Layer and supplied only the *(question, sentence)* pairs to it. The sentence representations, obtained from BERT, were used to predict if the sentence was a supporting fact. This is referred to as *Sentence Level BERT* in Table 3. A similar experiment was carried out for *(question, passage)* pairs and is referred to as *Passage Level BERT* in Table 3.

[2]All the layers of LoGIX before the sigmoid layer.

Context Encoding	SUP. FACTS	
	EM	F1
LoGIX	**53.11**	**83.62**
Passage Level BERT	44.07	79.95
Sentence Level BERT	27.70	70.75

Table 3: Impact of different *context encoding schemes* on supporting fact prediction. All evaluations are on *development set* with BERT$_{BASE}$.

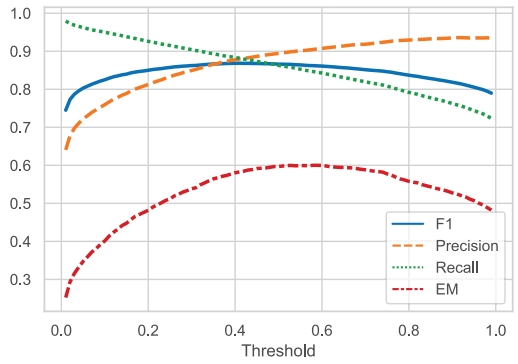

Figure 4: Variation in Precision, Recall, F1, and EM scores of the predicted supporting facts as a function of threshold

Compared to the *Sentence Level BERT*, *Passage Level BERT* is better equipped to capture the intra-passage sentences dependencies, providing a better encoding of the local context. Finally, we evaluated LoGIX as shown in Figure 3. From Table 3, it is evident that LoGIX used in TAP outperforms the other approaches due to its ability to capture both local and global dependencies between sentences.

Effect of the Threshold τ An important hyperparameter in TAP is the threshold τ used by LoGIX to assign $\{0, 1\}$ labels to the facts. If a sentence has predicted probability $p(F_{ik}) \geq \tau$, it will be a predicted as supporting fact. Figure 4 shows how the supporting fact metrics of LoGIX vary as threshold τ is increased from 0.1 to 0.9. Since these facts will be used by the Answer Predictor, omitting a supporting fact will be costlier than including a non-supporting fact because it will make the question *un-answerable*.

Table 4 shows the effect of threshold τ on the accuracy of answers given by the Answer Predictor. The second column of this table shows the percentage of questions in the development set for which LoGIX predicts the supporting facts with 100% recall. We call such questions as *answerable* question. It is evident that as the threshold τ increases,

the number of *answerable* questions decreases and hence the accuracy of the Answer Predictor also drops.

Threshold	Relative size of $D_{r=1}$	ANSWER	
		EM	F1
0.1	87.95%	**65.01**	**78.57**
0.2	82.26%	64.61	78.24
0.3	77.50%	64.47	78.05
0.4	73.18%	64.32	77.73

Table 4: D denotes the development set and is divided into two parts $D = D_{r=1} \cup D_{r<1}$, where $D_{r=1}$ denotes that part of the development set for which the predicted supporting facts have full recall. The set $D_{r=1}$ is called *answerable* set.

	EM(Ans)	
	$= 1$	< 1
$Recall(SF) = 1$	60.55%	27.40%
$Recall(SF) < 1$	5.90%	6.14%

Table 5: Error analysis of the AP where entire development set is classified into four categories. Rows(columns) correspond to recall of supporting facts (exact match of answers).

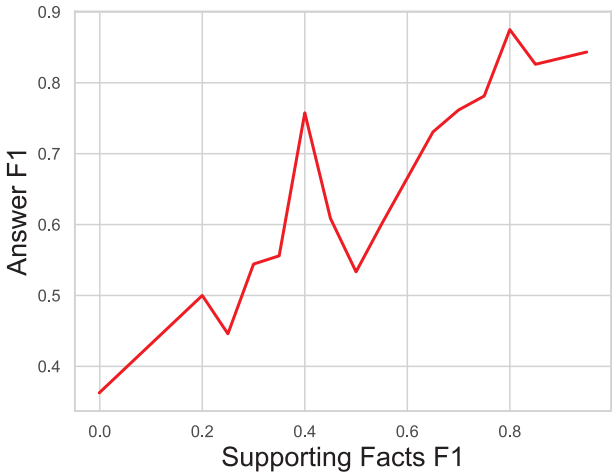

Figure 5: Behavior of the Answer Predictor

Answer Analysis The translucent nature of TAP makes it possible for us to obtain a set of context sentences such that TAP considered nothing other than these sentences to arrive at the answer. This allows us to analyze the performance of the Answer Predictor as function of the performance of LoGIX. We assigned each question in the evaluation set into one of four categories as shown in Table 5. The entries in each cell denote the percentage of the questions in the evaluation set falling in that category. Ideally, we would like all the questions to fall in the Recall(SF)=1, EM(Ans)=1 category. The category Recall(SF)<1, EM(Ans)=1 shows that

	Yes	No	Span
Yes	199	26	0
No	42	191	0
Span	6	6	6935

Table 6: Confusion matrix for the answer type classifier. The rows (columns) indicate the true (predicted) labels.

in only 5.90% of the questions, the model has *guessed* the right answer even though not all the supporting facts were available. A high number in this category would indicate that the model is less *trustworthy* as it frequently *guesses* the answers rather than *reasoning* over the facts. Furthermore, in Figure 5, we have also plotted the average $F1$ scores of the predicted answers against $F1$ scores of the predicted supporting facts across questions in the evaluation set.

Answer Type Classification Table 6 shows the confusion matrix for the answer type classifier. The answer type classifier almost never confuses between yes/no type answers and span type answers. It has an accuracy of 98.92%.

	LoGIX		AP	
	F1	EM	F1	EM
TAP (single model)	86.17	57.57	79.39	65.87
-SSPT	85.27	55.99	75.48	61.62
-BERT$_{LARGE}$	85.13	56.58	77.25	63.31
-Joint Training	Not applicable		78.63	65.16

Table 7: Ablation analysis.

Ablation Analysis We examine the impact of model size and span selection pre-training (SSPT) with an ablation analysis. Table 7 shows the impact of model size and pre-training style on the development set. In LoGIX the gains for both span selection pre-training and moving from BERT$_{BASE}$ to BERT$_{LARGE}$ are approximately one percent absolute. In the Answer Predictor we find the largest gains coming from span selection pre-training (4%), while increased model size has half the impact (2%). The gain from using joint training is less than one percent absolute.

7 Conclusions

TAP is a novel architecture for the multi-hop reasoning based RCQA task. Core to this system is LoGIX, a new approach that effectively addresses the challenges of local context and global interactions present in multi-passage, multi-hop QA. We have shown that TAP advances the state-of-the-art on `HotpotQA` dataset, reaching Rank-1 and Rank-2 in its ensemble and single model variants at the time of submission. Finally, by restricting the input of the AP to LoGIX's selected supporting facts, TAP admits interpretability that can be used to debug the model for performance enhancement purposes prior to its deployment.

References

Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; Rosenberg, M.; Song, X.; Stoica, A.; Tiwary, S.; and Wang, T. 2017. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of ICLR*.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of ACL*, 1870–1879.

Choi, E.; Hewlett, D.; Uszkoreit, J.; Polosukhin, I.; Lacoste, A.; and Berant, J. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of ACL*, 209–220.

Clark, C., and Gardner, M. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of ACL*, 845–855.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, 4171–4186.

Ding, M.; Zhou, C.; Chen, Q.; Yang, H.; and Tang, J. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of ACL*, 2694–2703.

Feldman, Y., and El-Yaniv, R. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of ACL*, 2296–2309.

Glass, M.; Gliozzo, A.; Chakravarti, R.; Ferritto, A.; Pan, L.; Bhargav, G. P. S.; Garg, D.; and Sil, A. 2019. Span selection pre-training for question answering. arXiv preprint.

Hewlett, D.; Jones, L.; Lacoste, A.; and Gur, I. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of EMNLP*, 2011–2020.

Jia, R., and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*, 2021–2031.

Joshi, M.; Choi, E.; Weld, D.; and Zettlemoyer, L. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*, 1601–1611.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Kelcey, M.; Devlin, J.; Lee, K.; Toutanova, K. N.; Jones, L.; Chang, M.-W.; Dai, A.; Uszkoreit, J.; Le, Q.; and Petrov, S. 2019. Natural Questions: A benchmark for question answering research. *TACL*.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of EMNLP*, 785–794.

Lin, Y.; Ji, H.; Liu, Z.; and Sun, M. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of ACL*, 1736–1745.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A robustly optimized bert pretraining approach. arXiv preprint.

Min, S.; Zhong, V.; Socher, R.; and Xiong, C. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of ACL*, 1725–1735.

Min, S.; Zhong, V.; Zettlemoyer, L.; and Hajishirzi, H. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of ACL*, 6097–6109.

Nishida, K.; Nishida, K.; Nagata, M.; Otsuka, A.; Saito, I.; Asano, H.; and Tomita, J. 2019. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *Proceedings of ACL*, 2335–2345.

Qiu, L.; Xiao, Y.; Qu, Y.; Zhou, H.; Li, L.; Zhang, W.; and Yu, Y. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of ACL*, 6140–6150.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. arXiv preprint.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2383–2392.

Reddy, S.; Chen, D.; and Manning, C. D. 2019. CoQA: A conversational question answering challenge. *TACL* 7:249–266.

Swayamdipta, S.; Parikh, A. P.; and Kwiatkowski, T. 2018. Multi-mention learning for reading comprehension with neural cascades. In *Proceedings of ICLR*.

Talmor, A., and Berant, J. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of NAACL*, 641–651.

Unger, C.; Forascu, C.; López, V.; Ngomo, A. C. N.; Cabrio, E.; Cimiano, P.; and Walter, S. 2014. Question answering over linked data QALD-4. In *Proceedings of CLEF*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of NIPS*, 5998–6008.

Wang, S.; Yu, M.; Guo, X.; Wang, Z.; Klinger, T.; Zhang, W.; Chang, S.; Tesauro, G.; Zhou, B.; and Jiang, J. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of AAAI*.

Wang, S.; Yu, M.; Jiang, J.; Zhang, W.; Guo, X.; Chang, S.; Wang, Z.; Klinger, T.; Tesauro, G.; and Campbell, M. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of ICLR*.

Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Welbl, J.; Stenetorp, P.; and Riedel, S. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL* 6:287–302.

Yan, M.; Xia, J.; Wu, C.; Bi, B.; Zhao, Z.; Zhang, J.; Si, L.; Wang, R.; Wang, W.; and Chen, H. 2019. A deep cascade model for multi-document reading comprehension. In *Proceedings of AAAI*, 7354–7361.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2369–2380.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; and Le, Q. V. 2019. XLNet: Generalized autoregressive pretraining for language understanding. arXiv preprint.