THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

Improving Evidence Retrieval for Open Domain Question Answering on Tables and Text

Qiyu Chen
SPRING 2022

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computer Science and Physics
with honors in Computer Science

Reviewed and approved* by the following:

Rui Zhang
Assistant Professor of Computer Science and Engineering
Thesis Supervisor

Danfeng Zhang
Associate Professor of Computer Science and Engineering
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Open Question Answering is a hot research topic in natural language processing. In a typical QA model, the answer to a question is produced by the model by retrieving relevant documents and analyzing these extracted materials. For a retriever-reader based QA model, the retriever model is responsible for retrieving documents that might contain imperative information for answering the question. The reader model then utilizes these most confident blocks to consider answer span for the question. However, past researches in Open Question Answering typically limit their information source in passage only. In real life, there are still a significant amount of question needs information stored in other forms such as video, table, and etc. In this paper, we are going to improve the evidence retrieval for an Open Domain Question Answering model on Tables and Text[1]. Compared to unstructured textual data, structured tabular data has its advantage in aggregating related information, which makes it a great choice for QA model. This paper first presents some background knowledge for understanding the baseline model. Later, detail information such as structure and training method for the model is discussed. As the code of model is not published, the baseline model is first implemented, and subsequent improvements are made. In the original paper, the ultimate model is comprised of fusion block retriever and cross-block reader. Here, we decide to implement one of the variants: fusion block retriever and single-block reader. Upon that, three improvement strategies: retain passages in the fusion procedure, use GPT-2 to augment search query, and use GENRE model to augment search query are made. Evaluation shows effectiveness and prospect of the first and third improvement strategies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Dr. Zhang for all of his help over the period of developing this paper. His patience on answering my questions and his support of server resource are undoubtedly crucial factors that help me to finish this research. Besides that, I would like to thank my friends and my parents for their financial and emotional supports on my Penn State life.

Chapter 1

Overview

Open Question Answering domain focuses on solving given question based on retrieval documents in corpus. Prior researches typically constrain their dataset in text form only. Even if some researches did consider tabular data as source of information, they are often in limited tabular form, or ground-truth tables for question are given without retrieval.

Following the requirement of model for Open Question Answering on Table and Text, a new large-scale dataset Open Table-and-Text Question Answering (OTT-QA)[1] was constructed. This dataset was built on the HybridQA dataset[2]. Questions in both of these two dataset are carefully designed as multi-hop questions, which requires inference jumping between tables and documents. However, in HybridQA, the ground-truth table and documents are already given for questions. Only reading comprehension is needed to answer the question. But in OTT-QA, both retrieval and reading process have to be handled. The baseline model in OTT-QA adopts a classical retriever-reader model, where the retriever is based on an iterative retrieval process, and the reader is a single-block reader that can handle only one block at a time.

The OTT-QA comes out a novel strategy named early fusion. This strategy attempts to fuse each table segment with multiple relevant passages into a fused block before retrieval process. The fused block is instead treated as the unit block for the retriever. Rather than using an iterative procedure to retrieve table and passages relevant to the query, the new design retrieves top-K fused blocks at once. In addition to the fusion retriever, OTT-QA replaces the

single-block reader with a cross-block reader based on the Extended transformer Construction (ETC) model[5], whose token limit is up to 4096 tokens. Instead of reading each candidate answer block one by one, the cross-block reader reads all top-K fused blocks at once.

As code of model in OTT-QA are not published, I have to first replicate the model in order to do the improvement research. After consideration, I pick fusion retriever and single-block reader instead of the ultimate model in OTT-QA as the baseline model for the research, which is based on two main reasons. First, the published code for training of cross-block reader which is based on ETC model is only available for TPU machine. Considering the available resource, recovering the single-block reader is a more reasonable choice. What's more, the emphasis of improvement is on the evidence retrieval. The effect of improvement strategies should be more apparent in the retrieval procedure, which means the replacement of reader model is acceptable.

Based on the baseline model, three improvement strategies are proposed. The first one tries to retain truncated passages in the fusion procedure to get them still involved in the retrieval procedure. The second and third strategies are using different models to augment the query in order to relieve the low lexical overlap issue in the retriever.

Chapter 2

Background

2.1 BM25 and TF-IDF

BM25 and TF-IDF are two famous retrieval functions used in information retrieval. In OTT-QA, TF-IDF is used as a candidate for the baseline retriever, which directly extracts confident blocks without training any neural network. Evaluation of HybridQA model on OTT-QA dataset adopts this method, where retrieval blocks from TF-IDF are treated as ground-truth table and passages and sent to the reader. The BM25 function is primarily used for the sparse retriever and the final entity linking step in query augmentation steps of our model.

The score of BM25 between a query and documents are computed by

$$Score(Q, d) = \sum_{i}^{n} W_i * R(q_i, d)$$

Where Q is the query, containing keywords q_1, q_2, …, q_i, d is the evaluated document, W_i represents weight for keyword q_i, and $R(q_i, d)$ is the relevant score between q_i and d.

Weight W_i and the relevant score are further calculated by

$$W_i = log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$$R(q_i, d) = \frac{f_i(k_1 + 1)}{f_i + K}$$

$$K = k_1(1 - b + b \frac{dl}{avgdl})$$

Where $n(q_i)$ is the number of documents contain keyword q_i, N is total number of documents in corpus, f_i is the occurrence of q_i in document d, dl is document length of document d, and k_1 = 1.5 and b = 0.75 in our model.

In TF-IDF, the score for a keyword q_i is computed as

$$TFIDF(q_i, d) = TF(q_i, d) * IDF(q_i, d)$$

Where

$$IDF(q_i, d) = W_i = log(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$$TF(q_i, d) = \log(1 + f_i)$$

Normally, when the query is given, the BM25 function could be directly applied to compute the score between query and documents and retrieve the one with largest BM25 score. However, the use of TF-IDF is typically combined with vectorization. That is, each document is represented by a vector in dimension of the vocabulary size. The value of the vector in n_{th}-dimension is exactly the TF-IDF value of keyword q_n. If we utilize vector to represent each document, it is expected that vectors with greater cosine similarity would be more relevant. Hence, the value of dot product between vector of query and documents becomes the principle of retrieval.

Both BM25 and TF-IDF values could be pre-computed and stored in a word to document matrix. In running time, only the query is required to be tokenized and processed to compute its scores. To handle the case of unknown keywords, each keyword is first passed to a hash function, and the hashed value is indeed the index for each word in my implementation.

2.2 Sparse and Dense Retrieval

There are two types of retrieval function used in variants of model in OTT-QA: sparse retrieval and dense retrieval. The sparse retrieval uses a unigram-based BM-25 score to retrieve an evidence block from the candidate pool. The dense retrieval is built on a dual-encoder architecture[3], where query and candidate blocks are encoded with separate Transformers. In our model, both the query encoder and the block encoder are built upon Bert model. Specifically, each input to the Bert model starts with a [CLS] token, which is usually used as "pool" representation for the whole sequence. For instance, the encoded [CLS] token could be used in downstream task such as predicting whether two input sentences in the sequence are connected. The dual-encoder architecture aims to train both encoders to capture the semantics of the whole input sentences in [CLS] token. Then, the dot product between encoded [CLS] of query and document evaluates their similarity in vector space.

That is, for each query q and each block b in candidate pool B, the scoring function is

$$f(q, b) = cosine\_similarity(h_q, h_b)$$

$$h_q = W_q BERT_Q(q)[CLS]$$

$$h_b = W_b BERT_B(b)[CLS]$$

Where W_q and W_b are two projection matrices that map the encoded vector to dimension 128 in my implementation[4].

2.3 Transformer

Transformer is a deep neural network proposed in article “Attention Is All You Need”[7]. Even though the plain transformer model is not directly utilized in our model, but many subsequent models built upon it such as BERT and GPT-2 are used, which makes it important to understand some of its crucial mechanisms.

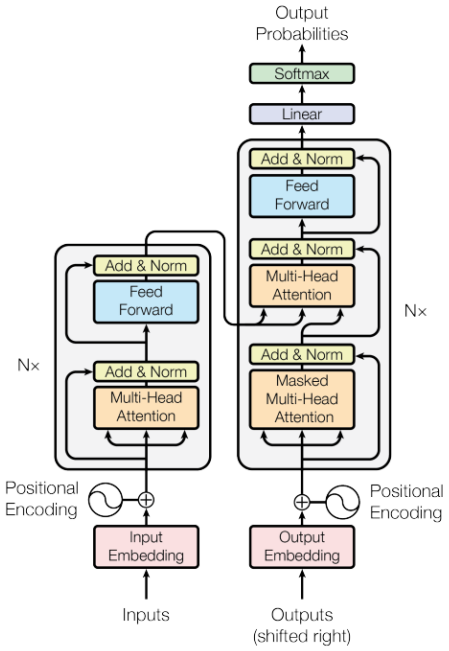

Figure 1 The Transformer architecture [7]

As Figure 1 indicated, the structure of Transformer is similar to many Seq2Seq model, which is comprised of encoder and decoder. But in fact, most Seq2Seq models prior to the Transformer are built upon RNN, while the Transformer is based solely on attention mechanisms.

The encoder component Transformer consists of six identical layers. In each layer, there are two sub-layers, which are separately Multi-Head Self-Attention layer and the Feed Forward

layer. Both outputs from these two sub-layers has to pass through the Add&Norm block, which performs residual connection and layer normalization.

Scaled dot-product attention

In Transformer, it proposes a new way of calculating attention, namely scaled dot-product attention. That is,

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

To understand the meaning of this attention formula, let's consider a token embedding x_i: By multiplying the embedding x_i with matrix Q, K, V, we can obtain three vectors query vector q_i, key vector k_i, and value vector v_i. If we want to know to what extend does other embedding x_j affects the encoding in x_i, we should use our query vector q_i to dot product with all other key vectors k_j. By diving each dot product result with $\sqrt{d_k}$ and performing SoftMax all over them, the result in each embedding is the extent of influence by this embedding when encoding current embedding. Then, the encoded embedding should be the weighted summation of all value vectors, where weight in each position is the result after SoftMax in this position, and this exactly matches the calculation of scaled dot-product attention.

Multi-Head Attention

Considering that different attention heads have different emphasis position in the sentence, the Transformer uses h different linear transformations to project matrix Q, K, and V to obtain different attention heads in order to allow the model to jointly attend to information from

different representation subspaces at different positions. Then, it simply concatenates these head results and multiply it with another projection matrix. That is,

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

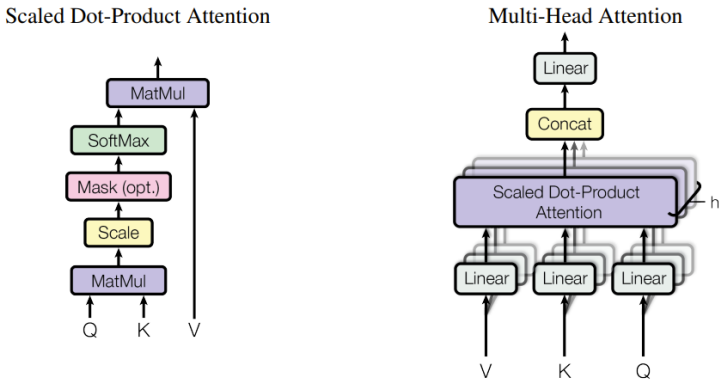

Figure 2 Attention in Transformer [7]

Similar to the encoder component, the decoder component is also comprised of six identical sub-layers. The first Masked Multi-Head Attention sub-layer aims to mask information from the future such that the model can only attend to tokens that have been outputted from decoder component in previous rounds. The second attention sub-layer is also called encoder-decoder attention layer. This is because only the query comes from the first sub-layer, but key and value all comes from the output of encoder component. In this way, each position in the decoder can attend to all positions in the input sequence.

Lastly, as the architecture of Transformer is not in traditional RNN form, the model lacks information about the order of the sequence. The Transformer utilizes “positional encodings” to inject relative position information by adding it with input embeddings at the bottoms of the encoder and decoder stacks.

2.4 GPT and GPT-2

The Generative Pre-Training (GPT) model is a language model primarily built on the stack of decoder component in Transformer. Unlike the decoder component of Transformer that has a Multi-Head Attention sub-layer in the middle receives output from the encoder component, each decoder layer of GPT model consists of only one Masked Multi Self Attention and Feed Forward sub-layer, where layer normalizations are added around their outputs[8].

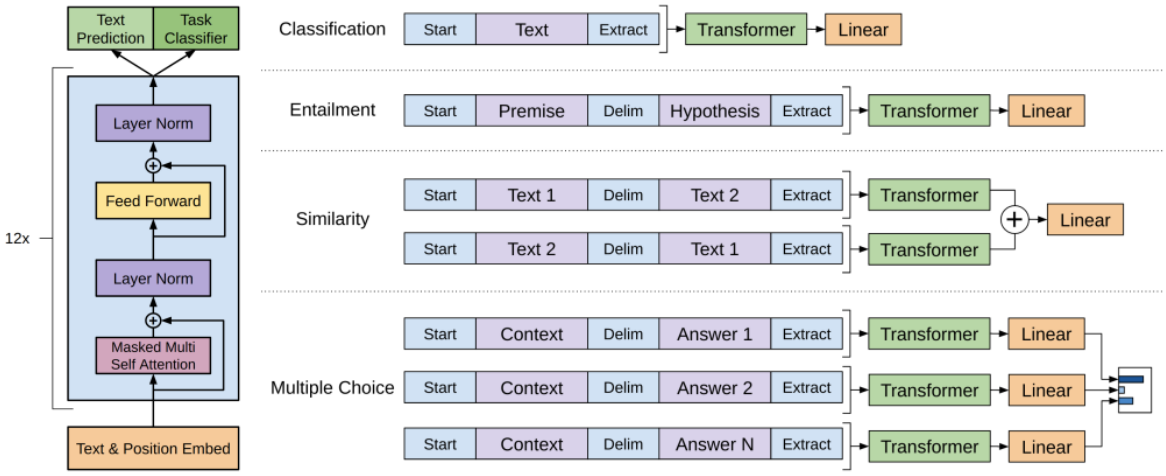

Figure 3 (left) Architecture of GPT model. (right) Fine-tune on different tasks [8]

The training on GPT model is based on semi-supervised learning. Strictly speaking, the training is split into two parts: pre-train and fine-tune.

In the pre-training procedure, the GPT model is trained on an extremely large unsupervised corpus of tokens, where a standard language modeling objective is used. After pre-training, the GPT model is supposed to learn more universal and practical representation for sentences. Then, it is passed to different supervised learning tasks such as Classification and Entailment. All structured inputs are first preprocessed based on the specific task and then processed by the pre-trained model. Vector representation of tokens outputted from model is further passed through a Linear Layer to do specific fine-tune training.

One of the drawbacks of GPT model is that the fine-tuned model is only applicable to specific task. In order to train a language model that can solve any natural language problem, the GPT-2 model is proposed[9].

GPT-2 model retains most parts of architecture of GPT model but makes several improvements. First, the fine-tune layer is removed. The GPT-2 model is supposed to automatically identify the type of problem that needed to be handled. Also, the layer normalization is now put before each sub-layer, and one more layer normalization is added to the end of final self-attention sub-layer. In addition to these difference in architecture, more hidden layers and larger hidden size are adopted. The size of vocabulary and the scale of dataset are also increased to handle richer input context.

2.5 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is another model that used pervasively in our baseline model. As the reader is not implemented as cross-block reader but single-block reader based on BERT. Both retriever and reader in our baseline models are built on BERT model, which indirectly proves the great performance of BERT in NLP tasks[10].

Even though BERT is also Transformer-based, but unlike the GPT-2 model which depends on the decoder component of Transformer, BERT is utilizing the encoder component of Transformer. There are two pre-training tasks in BERT: Masked Language Model and Next Sentence Prediction.

Masked Language Model

To get rid of the limitation that traditional language model can only exploit information in one direction, such as the GPT-2 model, BERT model decides to train on a masked language model rather than traditional language model. In comparison to traditional language model, the masked language model randomly mask a given percentage of tokens in the input and require BERT model to predict those masked tokens. In the implementation of BERT, 15% of tokens are chosen to be the masked tokens. By doing so, BERT forces encoding of a token to attend its context in both directions.

However, as the [MASK] token only appears in the pre-train procedure, this may result in a mismatch between the pre-train and fine-tune process. To resolve this issue, BERT adopts below strategy: For tokens selected to be masked tokens, they have 80% of chance to be replaced as a [MASK] token, 10% of chance to be replaced as a random token, and 10% of chance to

remain unchanged. By this approach, BERT could not figure out whether a token is masked or not, and this further restricts BERT model to attend a token's context rather than the token itself when encoding it.

Next Sentence Prediction

As many downstream NLP tasks are based on the understanding between two sentences, BERT model designs the Next Sentence Prediction task in pre-training procedure. To understand this pre-train task, it is required to know the input representation of BERT.

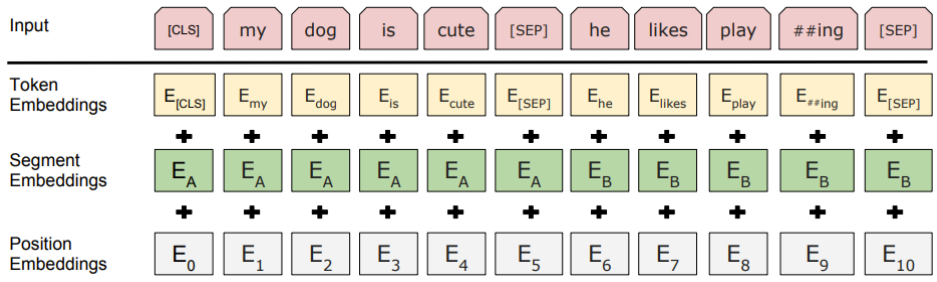

Figure 4 Input representation of BERT [10]

Considering there is a task to predict whether two sentences are connected in semantics, let's say "My dog is cute" and "He likes playing", as illustrated in Figure 4. Firstly, there is a [CLS] token added to front of the first sentence, indicating the start of input. Each sentence is tokenized by the BERT tokenizer following WordPiece embedding. Simply speaking, the WordPiece embedding can split a single word to combination of different words in the vocabulary. For instance, "playing" is split into two tokens: "play" and "##ing". There are several benefits brought by this embedding strategy, such as decreasing vocabulary size and handling unknown word. After tokenizing input sentences, special [SEP] token is added to the end of each sentence. As shown in Figure 4, there are also segment embeddings and position embeddings for the input sentence. The segment embedding is used to indicate the sentence that

current token belongs to, and the position embedding is needed to inject positional information in BERT.

Besides the use of indicating start of the input sentence, the [CLS] token is supposed to capture the semantics of entire sentence after encoding. It is selected to be encoded in this way because this token does not have apparent semantics meaning, which allows it to fuse semantics information from other tokens fairly. Therefore, in the pretraining task of Next Sentence Prediction, the encoding of [CLS] token is passed to two-category classifier to predict whether or not two sentences in the input are connected in semantics.

Based on above two pre-training procedures, the pre-trained BERT model is fine-tuned in our subsequent models based on given dataset and task.

2.6 GENRE Model

Generative Entity Retrieval (GENRE) model is an autoregressive entity retrieval model used in one of the improvement strategies[11]. In GENRE model, it believes that entity name could also be treated as a unique entity identifier besides the unique atomic label binding to the entity. For instance, the identifier for entity Tool could be either an atomic label like 4713 or a unique entity name Tool(band). In the paper, it points out some use of entity name as the identifier for entity such as the title in Wikipedia. Compared to the integer atomic label, an entity name provides more semantic information for the entity and is highly structured and compositional. Most importantly, there are indeed predictable patterns between unambiguous entity name and mention context, where mention is reference or representation of entity in texts.

The article summarizes six patterns matching entity name and mention context as illustrated in Figure 5.

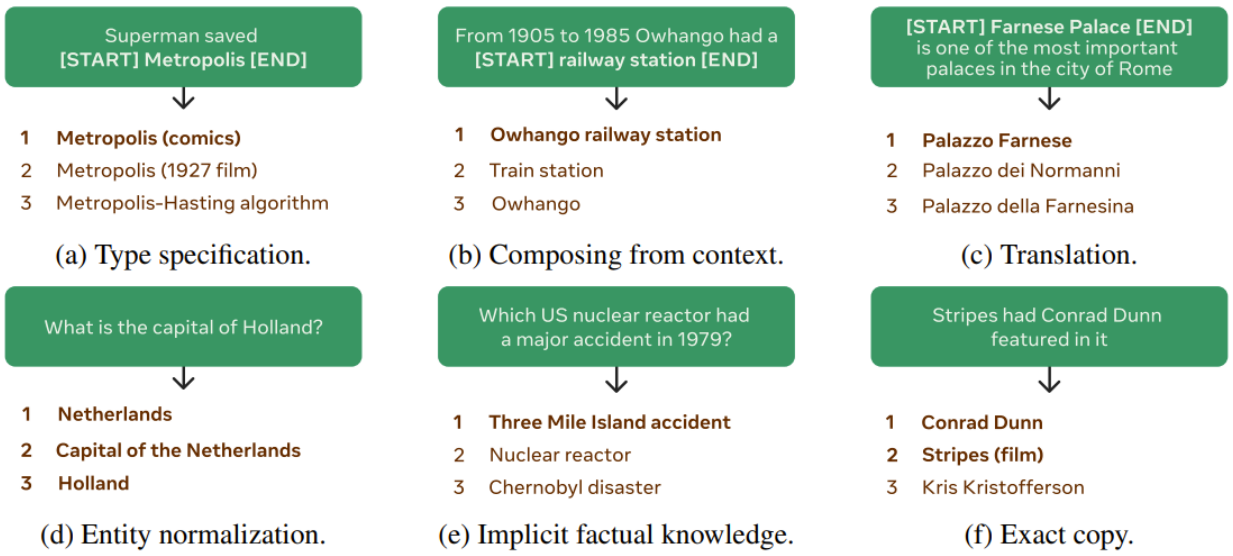

Figure 5 Patterns between entity name and mention context [11]

In other words, there exists a mapping relationship between the entity name and the input with mention. It is possible to generate the unique entity name from the input.

Based on above concept, the GENRE model adopts a sequence-to-sequence model. The GENRE model uses BART model[12] as the pre-training model, which is also a transformer-based architecture. Then, the GENRE model fine-tunes to generate the entity name by computing the log-likelihood score between mention input and each candidate entity.

To restrict that the generation of entity names merely comes from the knowledge base, the GENRE model utilizes a constrained beam search in decoding procedure. The constrain is achieved by using a prefix-Trie data structure. Each node in the tree represents a token in vocabulary, and children of the node indicates all possible tokens followed by.

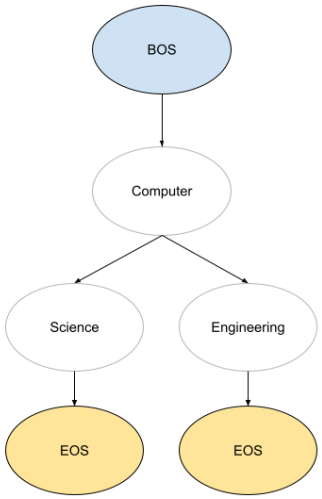

Figure 6 Prefix-Trie example

Based on the fine-tune tasks, there are several different functionalities accomplished by the GENRE model: Entity Disambiguation, Document Retrieval, and End-to-End Entity Linking. The Entity Disambiguation tries to generate entity name for an input including mention, where the mention is manually enclosed by special tokens [START_ENT] and [END_ENT]. The Document Retrieval strives to return a Wikipedia article title based on a given query. The End-to-End Entity Linking is the functionality that used in our experiment. Given an input sentence, it detects entity mentions in it and returns entity names corresponding to them.

Chapter 3

Task and Dataset

3.1 Task

OTT-QA dataset originates from the HybridQA dataset[2]. In HybridQA dataset, each question is produced by crowd workers using a table and hyperlinked passages in the cell of table. At inference time, the question and ground-truth table and its linked passages are given. The task in HybridQA is simply performing reading comprehension using given information. However, in OTT-QA, all inherited questions were re-annotated by decontextualizing them. Re-annotated questions were made to be suitable for open-domain setting such that unique answer for each question can be determined without assuming any provided context from tables or passages. Now, the retrieval corpus of OTT-QA consists of set of table candidates B_T and set of passage candidates B_P. Given a query q, the task of OTT-QA is first retrieving set of relevant blocks b from $b \in B_T \cup B_P$. Then, retrieval blocks are used as evidence blocks for the reader to answer the question by extracting answer span inside them.

However, it is extremely easy to exceed the 512 input tokens limit of the BERT model if we want to represent even a single complete table. To handle this problem, OTT-QA splits each table into several table segments. Each table segment represents a single row in the original table. Besides information of each cell in the row of table segment, metadata, headers, and global max/min information are also stored in the table segment. In this way, each table segment is still meaningful enough as unit block. The task of OTT-QA becomes retrieving blocks in the combination set of tables segments and passages and aggregating them to answer the question.

3.2 Dataset

As mentioned above, all questions from HybridQA underwent a decontextualization step before putting into use. Basically, the decontextualization step is composed of two phases. In the first phase, workers were only allowed to add minimum words or phases into the question based on meta information of the ground-truth table. In the second phase, annotators manually selected questions that are complicated to make them concise and natural enough.

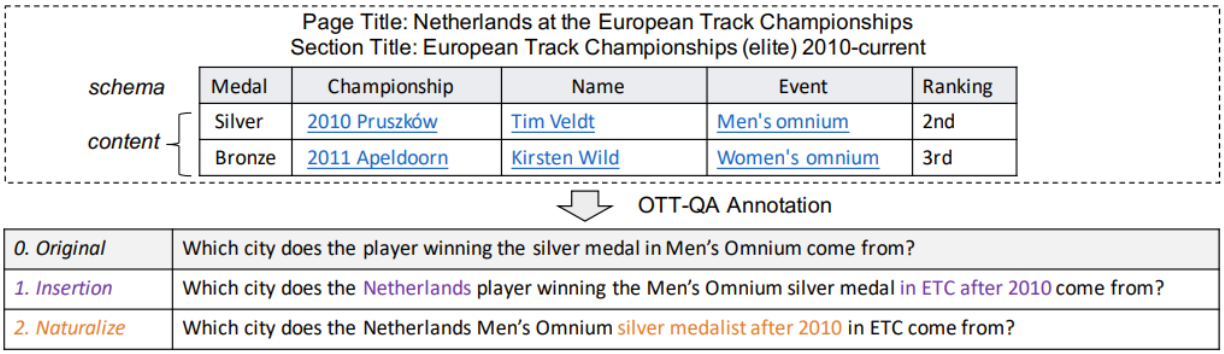

Figure 7 Example of decontextualization [1]

The HybridQA dataset contributed around 13k tables for OTT-QA. In addition to that, OTT-QA crawled 400k new tables and followed the same question generated procedure as HybridQA does. Eventually, The OTT-QA has 41,469 questions in the training set, 2,214 dataset in the dev set, and 2,158 questions in the test set.

Chapter 4

Replication of baseline model

4.1 Overview of the architecture

As mentioned in prior sections, even though the model achieves best performance in OTT-QA is the combination of fusion retriever and cross-block reader, the baseline model picked in this research utilizes the fusion retriever and the single-block reader.

Before the retrieval procedure, the fusion procedure is needed to combine each table segment with its most relevant passages to form a fused block, where the fused block is the unit retrieval block for the fusion retriever. As noted before, in HybridQA, the ground-truth table and hyperlinked passages are given for the question. In other words, the fusion procedure attempts to find the hyperlinked passages appeared in the cell of table segments. To do so, the GPT-2 model is utilized to augment the table segment by generating queries token by token. Each query then searches for the closest passage title using BM25 as the criterion. That is, each generated query contributes one linked passage to the fused block. Finally, entity linking is performed to produce the fused block.

The fused block becomes the unit block for retrieval. In order to have a better performance on the fusion retriever, the dual-encoder is first pre-trained using a technique named Inverse Cloze Task (ICT). ICT helps dual-encoder to develop greater ability on extracting lexically matched blocks. Then, the dual-encoder is fine-tuned to retrieve top-K fused blocks and pass them to the single-block reader. In our experiment, the value of K is chosen as 15.

Unlike the cross-block reader which reads all evidence blocks at once, the single-block reader reads each retrieved block separately and find the answer span that has greatest probability for that block. Assuming the score for retrieving each evidence block b is S_{b_i}, scoring function for the answer span is

$$f_{read}(a) = S_{b_i} * f_{read}(a|b_i)$$

Where a is an answer span, $f_{read}(a|b_i)$ is the probability of treating a as the answer span in block b_i.

Multiple answer spans are ranked using this scoring function, and the highest score answer span will be returned as the answer for the question.

Distant Supervision Signals

In HybridQA, only the answer for question but not the position of answer span is given. Therefore, OTT-QA traverses table and hyperlinked passages to find all exact matches as candidate answer source for each question. However, newly crawled tables in OTT-QA do not have hyperlinks. To resolve this problem, it uses BM25 on each cell of a table to find closest passages for that table and treat them as ground-truth hyperlinks.

4.2 Details of GPT-2 model

The GPT-2 model in my implementation starts from the GPT2LMHeadModel, which is a GPT-2 model with a language modelling head on top. By giving input, this model can directly return logits for each vocabulary tokens. At inference time, the model directly outputs the token with greatest logit value as the next predicted token.

The GPT-2 model is used in the query augmentation step for table segments. In order to let the GPT-2 model capture meta information of the table segment, we preprocess each input table segment following below format

Figure 8 GPT-2 preprocess format

New special tokens [SEP], [ENT], [START], and [EOS] are added to the vocabulary of the model. The [SEP] token separates different meta information, [ENT] tokens enclose the cell content of the table segment, and the [START] and [EOS] tokens indicate the start and ending position for generated tokens. In training, the [START] and [EOS] tokens enclose the ground-truth hyperlink title as predicted output. But in inference time, only the [START] token is required, as it denotes the start of output tokens.

As we see, even though we are processing table segment as a unit in the query augmentation step, each time GPT-2 model is only generating query for one cell of the table segment. After prediction have finished for all cells in a table segment, predicted queries are aggregated and treated as queries for the single table segment.

The GPT-2 model has already been pre-trained. Based on that, the model is fine-tuned on training pairs (table segment, titles of hyperlinked passages). As mentioned in distant supervision signal section, hyperlink passages for each table are given only in HybridQA. To compensate that, BM25-generated titles of passages are treated as ground-truth hyperlinked passages used in the fine-tune procedure of the GPT-2 model.

4.3 Details of dual-encoder retriever

The dual-encoder retriever trains two separate BERT based model: One for encoding the query and one for encoding the fused block. The architecture of both encoders are identical, which are all based on BERT model followed by a projection layer, where the projection layer simply projects the vector of [CLS] token to lower dimension as [4] does. Both encoders are trained simultaneously with a cross entropy loss objective to maximize the probability of retrieving correct fused block. Weights of neural networks in query encoder and block encoder get deviated during the training procedure.

Inverse Cloze Task

Before training of the dual-encoder, an unsupervised pre-training technique named Inverse Cloze Task (ICT) are performed. In text only corpus, ICT treats context of a sentence as the pseudo-question and the sentence as the pseudo-evidence. ICT requires the retriever to select the correct pseudo-evidence for the pseudo-question. The pre-training of ICT provides a strong initialization for text-based dense retrieval. However, as OTT-QA also considers tabular data, ICT in OTT-QA is different. Given a fused block, the table segment will be corrupted by randomly dropping half of words from the table metadata and cells to become a partial table segment. Then, one passage will be sampled from linked passages of the fused table segment. The combination of partial table segment and the sampled passage are treated as unit for the pseudo-query, and the original fused block becomes the pseudo-evidence. In this way, ICT is still effective in the setting of OTT-QA.

In-batch negatives

In my implementation, the in-batch negatives trick is used in both the pre-train and fine-tune procedure for the retriever[3]. That is, considering we have B pairs of training instance (query, ground-truth block), when we are retrieving blocks for certain query, we only treat the ground-truth block as the positive case and all other blocks as negative. In other words, for each pair of training instance, there are B-1 negative samples for them. By this approach, the model can reuse previous computation and train on B^2 pairs of training instance in each batch.

Decision of ground-truth blocks in fine-tune process

The location of answer span of each question is not given. But by using the distant supervision signal, we can still trace possible source of the answer using exact match. That is, if the content or the hyperlinked passages of a cell in table contains the answer, it is treated as answer node according to distant supervision signal strategy. In my implementation, considering each answer node has the identical chance of being the ground-truth answer node, I select the table segment with most answer nodes as the ground-truth table segment for fine-tune procedure.

4.4 Details of BERT based single-block reader

As the dual-encoder requires the BERT to capture the vector representation of whole sequence in the [CLS] token, which is not it originally designed to do, it requires large training instance for BERT to understand both its purpose and the structure of the fused block. Unlike them, the BERT is originally designed for downstream tasks such as question answering. There is no need for pre-training tasks for the model. In my implementation, the

BertForQuestionAnswering model is adopted for the single-block reader, which is a Bert model with a span classification head on top for extractive question-answering tasks. The model is fine-tuned on supervised pairs (query, answer_start_index_in_block, answer_end_index_in_block). For the choice of ground-truth answer span, I traverse all fused blocks corresponding to rows in the ground-truth table and treat the first fused block containing the answer as the ground-truth fused block, where the first exact match location of answer span provides start and end indices for the ground-truth answer.

Chapter 5

Improvement Strategies

5.1 Retain remaining passages in the fusion procedure

In the fusion procedure, each table segment is combined with multiple relevant passages to form a fused block. However, if the size of fused block reaches the input token limit of the BERT model, the fused block has to be truncated by default in order to fit the model, and remaining passages will not be fused. This results in the error in fusion error type in the retriever error analysis of OTT-QA. That is, if the ground-truth table for the question are linked to too many passages, it is hard to associate all relevant passages into the fused block. The lack of information in linked passages contributes to the error of retriever.

To relieve this type of error, I come up with a strategy that tries to retain remaining linked passages in the fused block. Specifically, if there are still remaining linked passages when the size of fused block reaches the token limit, it will try to fuse remaining linked passages to the original table segment to form another fused blocks. This procedure continues until all linked passages have been fused into blocks. In the case where the size of table segment already reaches the token limit before fusion, it stops the fusion procedure immediately.

5.2 Use GPT-2 model to augment the query

Another type of error greatly affects the performance of retriever is low lexical overlap. This category of error primarily comes from abbreviation and rephrasing of the table metadata. For instance, the “New York University” in the question is often shorten as “NYU”. Even though human reader will not get hindered by these abbreviation as our “knowledge base” already incorporates these concepts, the retriever does not capture this relationship during training.

As the GPT-2 model is used for the cell of table segment to generate extended query searching for hyperlinked passages, I’m wondering if this generation procedure could be also applied to the query before retrieval. Ideally, the hyperlink title could be treated as entity name. The process of extending the cell content using context from table segment is similar to the process of extending abbreviations of mentions.

In this consideration, I think the fine-tuning of the GPT-2 model on fusion procedure should be enough for the augment of input query. The key is that I should preprocess the input query to the model in order to fit the structure of table segment that it learnt in the fine-tune process. In my implementation, the input query is preprocessed in below format

Figure 9 Preprocess format for input query

Intention behind this representation is that when we are generating output for the cell in the table segment, tokens enclosed by [ENT] tokens are the part where we think the mention exists, while remaining tokens are context of it. As we have no idea where the mentions in input query are, we should put the whole query between [ENT] tokens, and there is no context for it.

5.3 Use GENRE model to augment the query

Similar to the idea of using GPT-2 model to relive the low lexical overlap problem, the GENRE model is also utilized to augment the input query in order to resolve the problem related to abbreviations.

Unlike the GPT-2 model which needs fine-tune to fit our dataset, there are already pre-trained GENRE model which is based on Wikipedia. Specifically, the End-to-End entity linking functionality is what makes me think GENRE prospective in this task.

Giving an input sentence, the End-to-End entity linking can detect entity mentions and bind entity name in knowledge base to them. For instance, input of "Einstein received a Nobel Prize" would return "{Einstein} [Albert Einstein] received a {Nobel Prize} [Nobel Prize in Physiology or Medicine]". In this example, Einstein and Nobel Prize are evaluated to be entity mentions, and their unique entity names are bound to them.

In my implementation, entity mentions in the input query are replaced by their unique entity names. Then, the new query is passed to the QA system. As GENRE model explained, the entity name typically contains more information of the entity than the mention as they combine context information. In my viewpoint, this enrichment of entity information should enhance the performance of the retriever and the reader.

Chapter 6

Evaluation on Performance

The evaluation of the retriever performance is conducted on hit rate@15. That is, if the retriever successfully retrieves the ground-truth fused block in top-15 retrieval blocks, there is a hit for the retriever. The ground-truth fused block in my experiment is defined as the fused block containing the table segment that is in the ground-truth table of the question. Initially, ground-truth fused block are fused blocks containing the table segment that is in the row of answer cell for the question. However, as questions in OTT-QA are mainly multi-hop questions, jumping between table segment is often needed for the inference of answer. It is not certain that the table segment with the answer cell is enough for answer comprehension. Hence, I expand the scope for the ground-truth fused block as all fused block containing table segment in the ground-truth table.

The evaluation of the reader model is not conducted alone. Instead, the joint evaluation of the retriever and the reader are conducted. The evaluation metrics is the exact match score between predicted answer and true answer.

Table 1 Evaluation on models (all based on fusion retriever + single-block reader)

Evaluated architecture	Hit rate@15	EM score (Test-Best)
OTT-QA	No independent evaluation for retriever	13.4
My baseline implementation	32.2 %	5.5
Improvement strategy 1	34.6 %	5.7
Improvement strategy 2	32.3 %	5.3
Improvement strategy 3	35.2 %	7.2

As illustrated in table 1, there are indeed gap of performance between my implementation and OTT-QA's implementation on the fusion retriever and single block reader architecture. This is acceptable to me as the code of OTT-QA is not published. There might be misunderstanding to its architecture. Besides that, there are still many factors such as the implementation details, choice of values for hyper-parameters and etc. which can affect the learning procedure of model. As the retriever model requires the BERT model to study for the new structure based on combination of tabular and textual data, if the study for the new pattern failed, the lose in performance is very predictable. But even though the performance of my implementation doesn't match the expected value, it could still be a criterion on judging the effectiveness of each improvement strategy.

The improvement strategy of retaining remaining passages in the fusion procedure enhances the performance of retriever while does little impact to the joint performance. I think the little impact to the joint performance could be explained based on the choice of single-block reader. As the single-block reader only outputs answer span from the fused block with most

confident score, the information between retrieval blocks are not fully exploited. Even though these hyper-linked passages are retained with these strategy, if the question still requires multi-hop between fused blocks, then the single-block reader still cannot utilize these added information. Besides that, if the original table segment is large in size, such that each time only small portion of the passage are kept and most parts are truncated, then the remaining incomplete information of passage may even be a distraction to the reader.

Now considering the improvement strategy of using GPT-2 model to augment the query, both the independent retriever performance and the joint performance are nearly unaffected. In my implementation to the second strategy, the generated tokens are appended to the end of original query by enclosing with parentheses. However, I've analyzed some of the augmented input queries and figured out that most of the generated tokens are indeed copy of the mention in original sentence. Even though there are cases where generated tokens are extension to the mention, they are either not providing any information enriching the context or just misleading the input query. Therefore, the little impact of this improvement strategy is apparent.

The last improvement strategy of using GENRE model instead of GPT-2 model to augment the query did achieve greater performance on both retriever and joint models. The richer information for the mention did provide support for the whole architecture. Even surprisingly, I found that the GENRE model has a nice ability on extending abbreviation of mentions. For instance, the "NYU" abbreviation mentioned early could be handled by the GENRE model and replaced with "New York University". Considering that, it is expected to observe an improvement on overall performance.

Chapter 7

Conclusion

In this paper, I'm mainly focusing on different improvement strategies that might enhance the evidence retrieval for OTT-QA. Even though the replication of model didn't achieve a satisfactory level, it is still enough to examine effectiveness and prospect of each proposed strategies.

During the implementation procedure of models, I also figured out some potential drawbacks that are currently not handled by OTT-QA. As mentioned in previous section, the content of table is not processed manually in OTT-QA when they were crawled from the web. There are many icon or special characters that are based on specific encoding rules. When they are viewed from Wikipedia, they displays and functions normally. However, when they are crawled from the web, they becomes meaningless combination of characters for both human and language model. This is a question that should deserve some attention especially when crawling of information has no limitation on them.

Nonetheless, the OTT-QA provides a great starting point on combining data of different modalities on Open Domain Question Answering. I believe the modality of data could be more various in the future, such as in form of images or audios.

Appendix A

More Details on Model

A.1 Representation of fused block in BERT

The representation of fused block in my implementation is based on OTT-QA, which is illustrated in Figure 10.

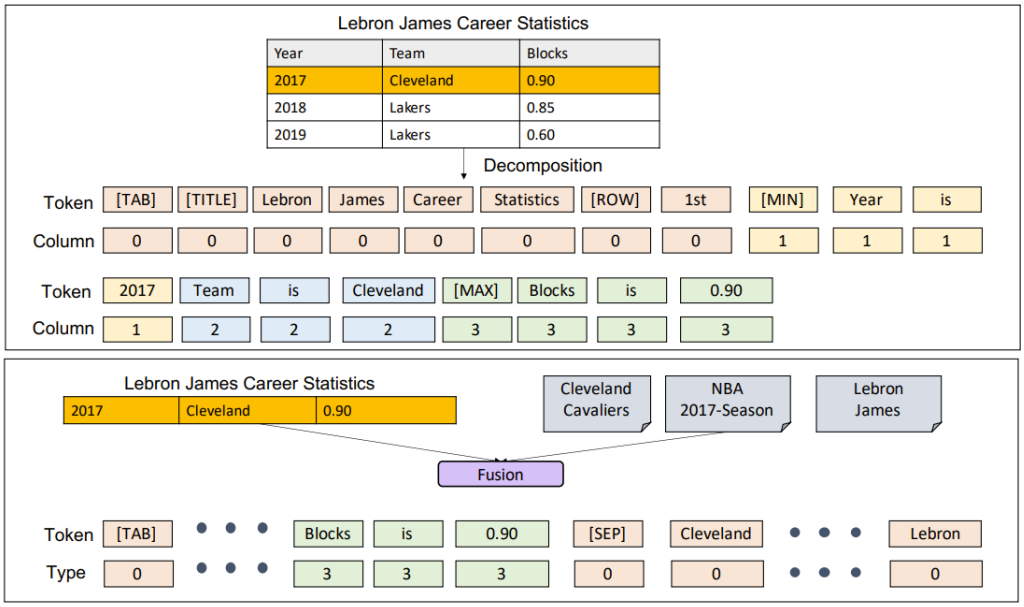

Figure 10 Fused block representation in OTT-QA [1]

Tokens [TAB], [TITLE], [ROW], [MIN], [MAX], [EAR], [LAT] are added to the vocabulary of block encoder for the representation of table segment structure. [TAB] token indicates the start of table segment, [TITLE] tokes indicates that subsequent tokens are title of the table segment, and [ROW] token denotes the end of meta information and start of table segment information. In addition to these special tokens indicating the structure of table segment, [MAX], [MIN], [EAR], [LAT] tokens are added to incorporate global max/min, earliest/latest

column information for the table segment. The [SEP] token is then used to separate each linked passage from others. However, in my implementation, the column embedding is not added as another vector to the representation but included in the token type ids as input to BERT model. For places where column ids changes, I simply flip the token type id since that token. For instance, from 0 to 1 or from 1 to 0. My reasoning for this simplicity is that the column position of each cell is not crucial like positional information of tokens in sentence. In most cases, different columns are just different attributes of the same object (in same row). The switch between columns doesn't affect understanding to the table. That's why I don't strictly follow the representation of the fused block in OTT-QA.

BIBLIOGRAPHY

[1] Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, William W. Cohen. Open Question Answering over Tables and Text.

[2] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *Proceedings of Findings of EMNLP 2020, 2020*.

[3] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wentau Yih. Dense passage retrieval for open-domain question answering. *EMNLP 2020, 2020.*

[4] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, 2019.

[5] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. *Proceedings of EMNLP 2020*, 2020.

[6] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[7] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia (2017-06-12). "Attention Is All You Need".

[8] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[11] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

QIYU CHEN

qfc5019@psu.edu

Education

Pennsylvania State University – Schreyer Honor College 09/2017-05/2022
Bachelor's Degree in Computer Science and Physics

Experience

Improvement on a state-of-the-art Open Question Answering Model 05/2021-04/2022
Researcher

- Did independent researches on a recently published paper describing an Open Question Answering Model, whose information source is not only limited to paragraphs, but also the table data.
- Developed and replicated the ultimate model based on description of the model described in the paper, which only published the code of the baseline model.

Eshine Cloud (Shenzhen) Technology Co., Ltd 11/2021-01/2022
Software Engineer Intern

- Participated in developing a LCDP (Low-Code Development Platform) for Enterprise Resource Planning management system.
- Understood the structure of KingDee BOS and the principles behind the interaction between plugins (such as form plugin and service plugin), graphic user interface.
- Implemented back-end functionalities such as automatic generation of smart forms.

Trend on Text-to-Video Generation Research 11/2020-12/2020
Researcher

- Discussed some common challenges that typically cumbered the progress in text-to-video area and used the viewpoints in several articles to illustrate the main interest and application in this area.
- Compared the difference in their performance for different methodologies based on the specific task.

Development of a Web-based Course Information Management System 03/2020-05/2020
Developer

- Analyzed customer needs and requirement based on the project proposal describing the developing scenario.
- Finished the project documentation which includes Requirement Analysis, Conceptual Database Design, Technology Survey, and Logical Database Design and Normalization.
- Developed the web application following specifications listed in the documentation, including coding, debugging, and testing of the product.
- Applied Flask for Python-based Web development, with SQLite for the database, and HTML and CSS languages for web design.

Programming Skills

C, C++, Java, Python, MATLAB, JavaServer Pages, JavaScript, MySQL, HTML and CSS