

Final Report QnA

1. Problem Statment

Question Answering (QA) system in information retrieval is a task of automatically answering a correct answer to the questions asked by humans in natural language using either pre-structured data or a collection of natural language documents. It presents only the requested information instead of searching full documents like a search engine.

As information in day-to-day life is increasing, so to retrieve the exact fragment of information even for a simple query requires large and expensive resources. This project aims to build the model for effective QA trained on multiple datasets. The system considers the context of the question and use background knowledge to generate the answer.

2. Different Methodologies in QnA:

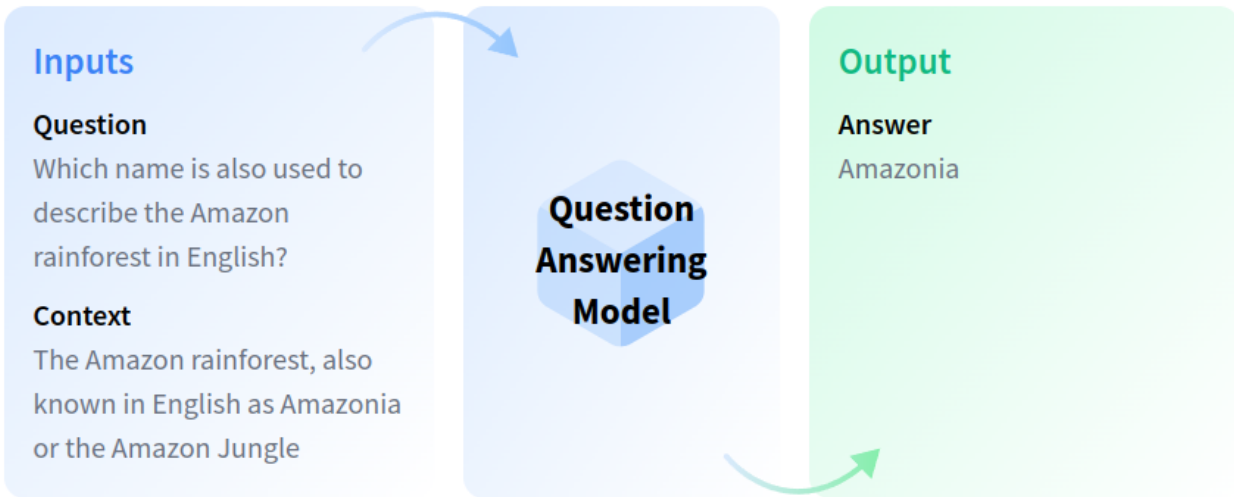
Diving the system on basis of **answer generation**:

- **Extractive QA**: The model **extracts** the answer from a context and provides it directly to the user.
- **Generative QA**: The model **generates** free text directly based on the context.

We are using SQUAD dataset so which is specifically designed for Extractive QA.

3. Dividing the system on basis of domains:

- **Closed-domain** question answering: It refers to specific domain related questions and can provide domain-specific knowledge. It has very high accuracy but limited to single domain. The example of such system is medicines or automotive maintenance.
- **Open-domain** question answering: It deals with the questions which are related to every domain. In this system, mainly have more data available from which the system extracts the answer. It can answer any question related to any domain but with very low accuracy as the domain is not specific.



4. Exploring Dataset- SQuAD

Dataset:

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions on a set of Wikipedia articles, where the answer to every question is a segment of text, from the corresponding reading passage, or the question might be unanswerable.

Format:

```
{
  "answers": {
    "answer_start": [1],
    "text": ["This is a test text"]
  },
  "context": "This is a test context.",
  "id": "1",
  "question": "Is this a test?",
  "title": "train test"
}
```

Evaluation Metric:

1. **Exact Match (EM):** score metric measures the percentage of questions for which the model produces an exact match with the correct answer.

2. **F1 score:**

$$precision * recall / (precision + recall)$$

$$precision = tp / (tp + fp)$$

$$recall = tp / (tp + fn)$$

tp : number of tokens* that are shared between the correct answer and the prediction.

fp : number of tokens that are in the prediction but not in the correct answer.

fn : number of tokens that are in the correct answer but not in the prediction.

Both EM and F1 scores are commonly used to evaluate the performance of machine learning models on the SQuAD dataset. In general, higher scores on both metrics indicate better performance.

Models

1. Unsupervised learning methods

- Here "unsupervised" means that we will not make use of the ground truth answer provided in the dataset (i.e., the columns `answer_start` , and `answer_end`). Instead, the sentence identification will be based only on the predefined heuristics.
- Below is implementation of 3 unsupervised learning models to identify the sentence that contains the answer to a given question:

A) Jaccard Similarity

- Jaccard similarity gives the similarity score between two sentences on the basis of common words appearing in the two.
- The Jaccard similarity coefficient $J(A, B)$ between sets A and B is calculated using the following formula.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Jaccard similarity ranges from 0 to 1, where 0 indicates no similarity between the sets and 1 indicates complete similarity. Hence sentence with the highest jaccard similarity is taken as the answer

B) TF-IDF Vectors

- TF-IDF (Term Frequency-Inverse Document Frequency) vectors are a numerical representation technique in NLP to quantify the importance of words in a document relative to a corpus.
- In the context of a sentence, TF-IDF vectors assign weights to each word based on two main factors:
 - $tf(t,d)$ = count of t in d / number of words in d
 - $idf(t)$: count of documents in which word t appears.

$$idf(t) = N/df(t) = N/N(t)$$

$$tf-idf(t,d) = tf(t,d) * idf(t)$$

- Similarity bw tf-idf sparse vector of two sentences is used to compare question and context paragraph.
- After performing experimentation with **cosine similarity** and **euclidean distance** metrics , in case of tf-idf vectors euclidean metric performs better.

C) Sent2vec Encoders

- Sent2Vec encoders are neural network models used for generating vector representations (embeddings) of sentences. These encoders aim to capture the semantic meaning and context of sentences in a continuous vector space..
- Cosine similarity bw encoded question and context paragraph sentences is used as comparison metric.

Results:

Similarity Method	Accuracy
Jaccard	63.2%

TF-IDF vectors	58.8%
Sen2vec Encoders	66.6%

Analysis:

We see a slight improvement in accuracy in case of **Sent2vec Encoders**. It is because of Sent2vec encoders captures the contextual and semantic meaning of a sentence in embeddings which leads to better representation of sentence.

2. Logistic Regression

2.1 Data preprocessing

We will first consider a binary classification setting, where we are given a question and a context sentence, and need to predict whether this context sentence contains the answer. In this setting, we can get several training data points from each row in the original dataset `df_squad`. In particular, if a row in `df_squad` looks like the following:

Question	Context Sentence	Answer
q	[s0,s1,s2,s3]	2

then it contributes four data points:

Question	Context Sentence	Answer
q	s0	0
q	s1	0
q	s2	1
q	s3	0

More generally, a row in `df_squad` where the `context_sentences` list has `n` sentences will be transformed into `n` rows, one for each context sentence. Among these new rows, only the row at index `answer_sent_index` gets assigned the label 1, while the others get the label 0. Implement the function `build_data_for_classification` that turns the original dataset `df_squad` into a dataframe with 3 columns -- `question`, `context_sentence` and `is_answer_sent` -- using the procedure specified above.

2.2 Model

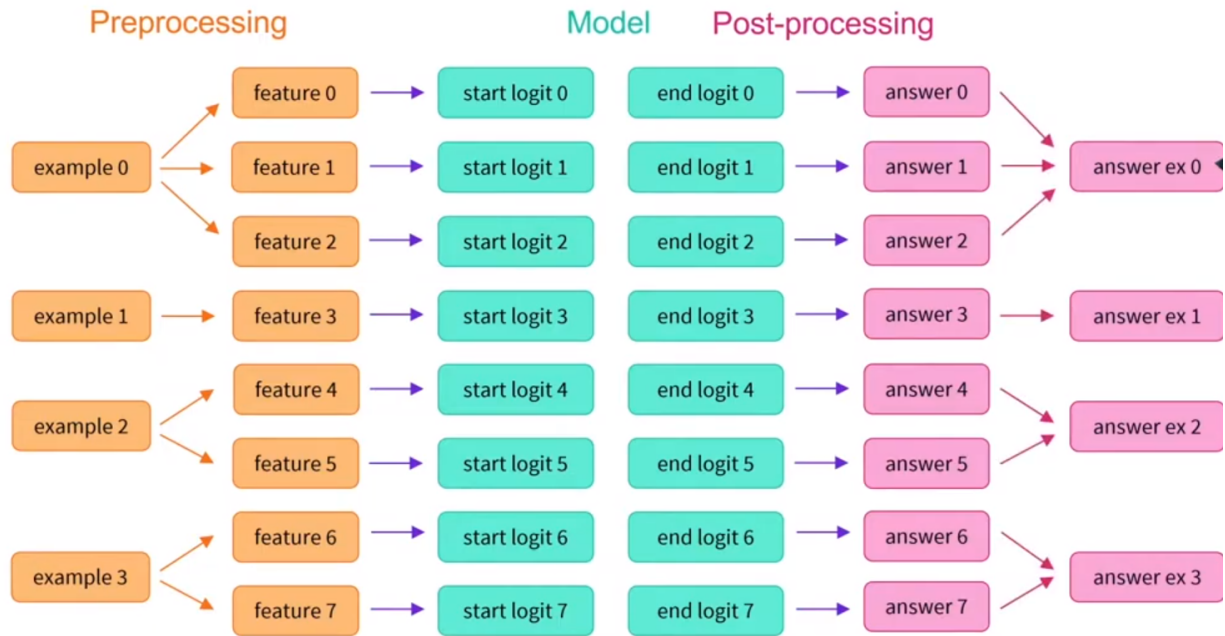
Having set up the dataset for binary classification, we can now train a logistic regression model. While the binary labels are already set up, we still need to construct the feature vectors as follows. For every question `q` and context sentence `s`:

1. Convert the question to its word embedding
2. Convert the question to its word embedding
3. Concatenate these two vectors to get the input vector to the logistic regression model:

$$x_{q,s} = (x_{q1} \quad x_{q2} \quad \dots \quad x_{qk} \quad x_{s1} \quad x_{s2} \quad \dots \quad x_{sl})^T \in \mathbb{R}^{k+l}.$$

Implement the function `get_lr_prediction` that performs the following steps:

1. Construct the feature vector for each row of the train set `df_train_formatted`, using the above formula.
2. Use an Sklearn `StandardScaler` (with default parameters) to fit and transform the train set `df_train_formatted`.
3. Train an Sklearn `LogisticRegression` model on the train set `df_train_formatted`.
4. Use this model to perform prediction on the test set `df_test_formatted`.
5. Return the trained LR model and its accuracy on the test set (i.e., the number of correct predictions divided by the test set size).



2.3 Results and Analysis

Accuracy: 71%

We obtain about 70% accuracy in this binary classification task. It's important to note that this accuracy cannot be compared with those from Supervised learning methods, because it is evaluated in a different setting (`df_test_formatted`). If we were only interested in whether the ground truth answer sentences are correctly detected, we would evaluate the accuracy on the original test set `df_squad_test`, instead of the formatted one.

While logistic regression on Sent2vec representation performs relatively well, it still relies on a unidirectional representation of words. In this setting, the same word is always mapped to the same vector, even though it may have different meanings in different contexts (e.g., the word 'bank' in 'bank account' is not the same as in 'river bank'). The final model we will explore in this project, which also addresses the above issue, is called BERT (Bi-directional Encoder Representations from Transformers).

3. Seq2seq-Attention Question Answering Model :

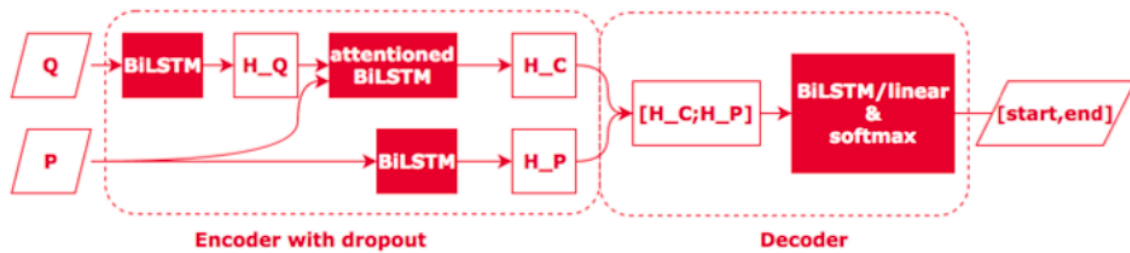


Fig.2 model architecture

This model is implemented according to a standford paper.

<https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761153.pdf>


- The basic structure of the model is a network of two encoders and a decoder, all implemented in bidirectional LSTM's.
- First, question Q and context paragraph P are encoded in two independent BiLSTM's which produce corresponding hidden states at each word position as H_P and H_Q . Then the encoded question and paragraph matrices H_Q and H_P are put in to another encoder with sequence to sequence attention.
- For each hidden state vector h_Q in H_Q , we calculated its attention score over all hidden states in H_P , which we used simple dot product here. Then the score matrix of each question position over the whole paragraph is multiplied by H_P . The product is sent into another layer of LSTM to generate the weighted context H_C under this specific question. H_C is concatenated with H_P into a larger state matrix that contain information about which parts are strongly focused and which are not.
- The next step is to feed the matrix $[H_C; H_P]$ into decoder, which consists of two separate bidirectional LSTM networks, one for start index and the other for end index. These gives two output vectors a_s and a_e , where largest element of each is the predicted index. Finally, use softmax activation and cross-entropy loss to arrive at the terminal.

Result —

These results are on validation data.

EM = 0.4432

F1 = 0.5125

Vairation with dropout rates 

Dropout_Rate	EM	F1

0.15	0.392	0.462
0.2	0.443	0.512
0.3	0.402	0.478

4. BIDAf-

In this approach, we implemented a bidirectional attention flow neural network to answer questions in the SQUAD dataset. We encode the question and corresponding context paragraph using GloVe word embeddings, compute an attention matrix, and find the start and end vertices from the context.

Model Architecture

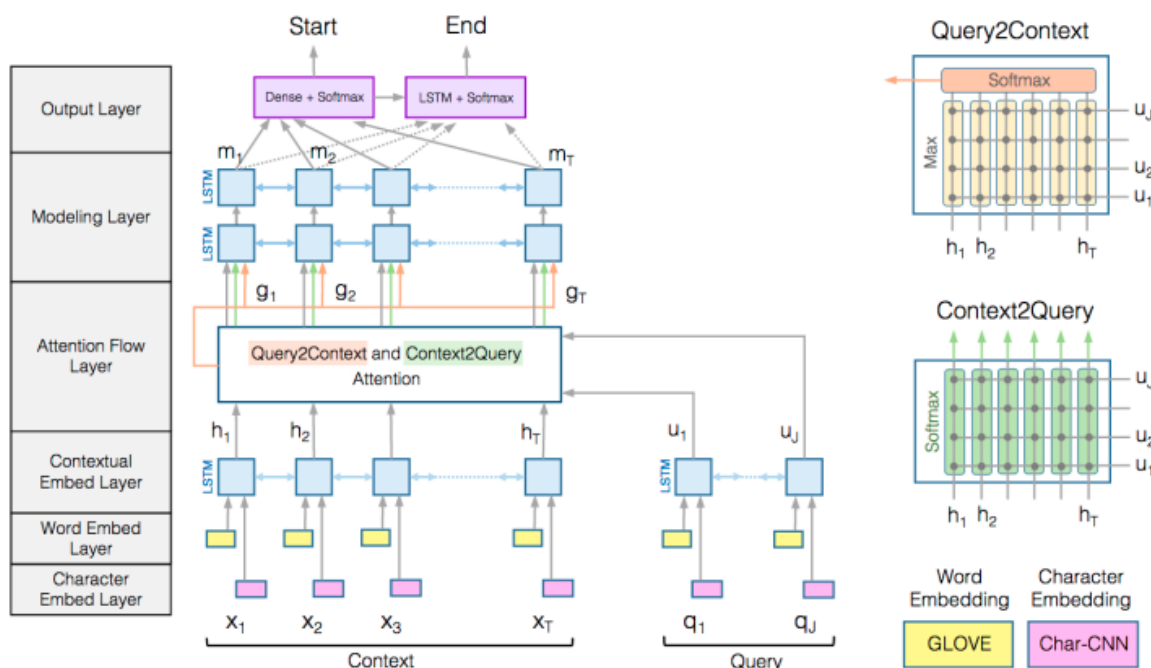


Diagram of the BiDAF model, taken from the paper by Minjoon Seo et al. Note that our implementation omits the character embedding layer, due to memory constraints.

- **Embedding Layer:**

1. **Word Embeddings**

Uses pre-trained GloVe vectors to convert words into embeddings that capture semantic meanings.

2. **Character Embeddings**

- **Usage:** Character embeddings are used to augment word embeddings by providing a more granular representation of words based on their constituent characters. This helps the model handle out-of-vocabulary words better.
- **Benefits:** Character embeddings capture morphological information, which can be beneficial for understanding the meaning of words based on their prefixes, suffixes, and roots. They also help in handling misspellings or uncommon words not present in the pre-trained word embeddings.

3. **Contextual Embedding Layer:**

- After embedding both questions and contexts at the word and (optionally) character level, each sequence is passed through a bidirectional LSTM (BiLSTM). This layer captures the context of each word by considering both left and right surrounding contexts, effectively creating a rich representation that takes into account the entire sequence.

4. **Attention Flow Layer:**

- This layer is the core of the BiDAF model, focusing on linking and fusing information from the question and the context.
- **Types of Attention:**
 - **Query-to-Context Attention:** This attention mechanism identifies which context words are most relevant to each question word. It helps highlight parts of the context that are likely to contain the answer to the question.
 - **Context-to-Query Attention:** This form of attention highlights which question words are most relevant to each word in the context. It effectively allows the model to weigh the importance of question words when interpreting each word of the context.
- **Significance:** These dual attention mechanisms allow the model to dynamically focus on different parts of the question and context as it decodes the answer. By computing both types of attention, the model captures a comprehensive understanding of how each word in the question is related to each word in the context and vice versa, enhancing its ability to pinpoint the correct answer span.

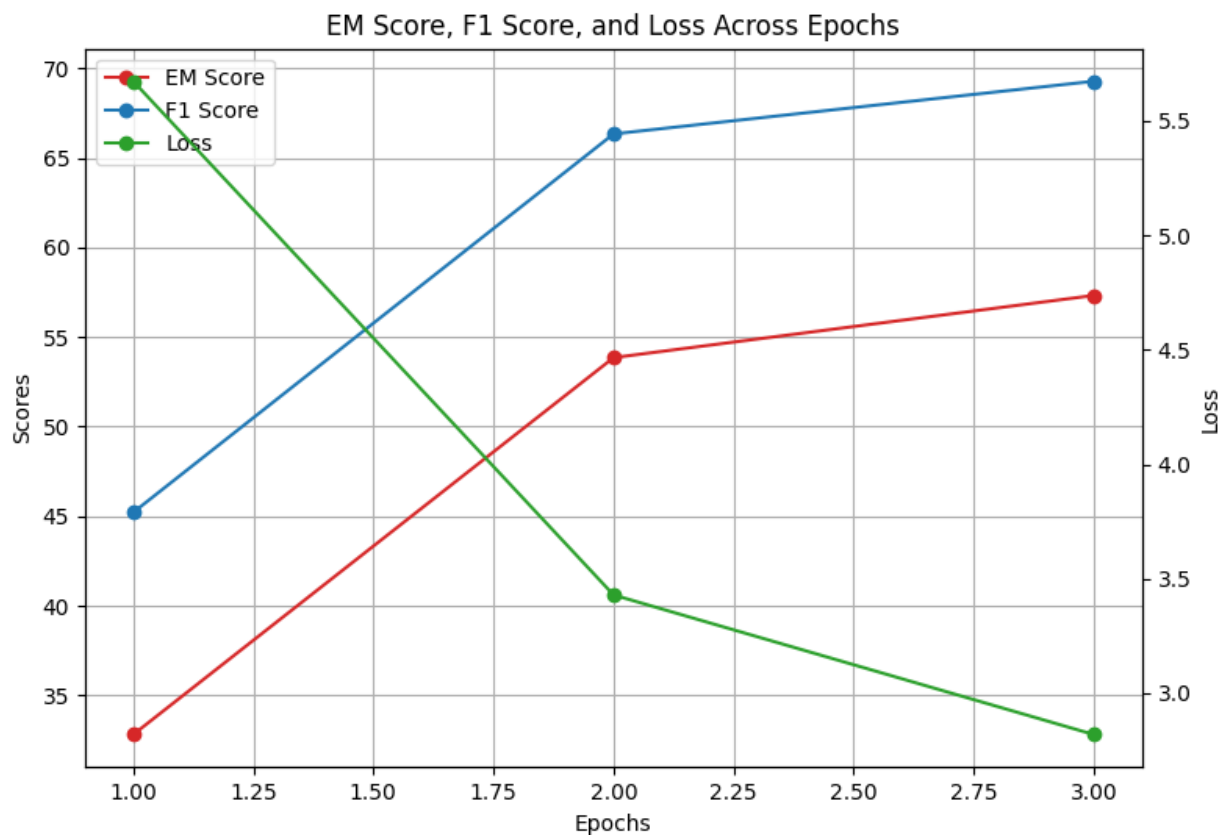
5. **Modeling Layer:**

- After the attention flow layer, the output is passed through another set of BiLSTM layers. This stage further processes the attention-enhanced features.

6. **Output Layer:**

- Finally, the model uses the outputs from the modeling layer to predict the start and end positions of the answer in the context. This typically involves a simple classifier and a softmax layer, that outputs probabilities for each position in the context being the start or the end of the answer span.

Results



Results On Validation Dataset (Unknown) After 3 epochs on whole

EM = 57.32678299169913

F1 = 69.2827315503881

Running Time

- The training time was significant, with each epoch of the BiDAF model taking approximately 80 minutes on GPU and 1.5 hour on preprocessing the dataset. This long training time made it difficult to experiment with different hyperparameters and also number of epochs are only 3.
- Parallelization by GPU didn't help much as the model is much more complex and it is not designed to perform parallelization. Presence of multiple layers and multiple LSTMs reduces parallelization.

5. DrQA:

5.1 Model proposed in **Reading Wikipedia to Answer Open-Domain Questions:**

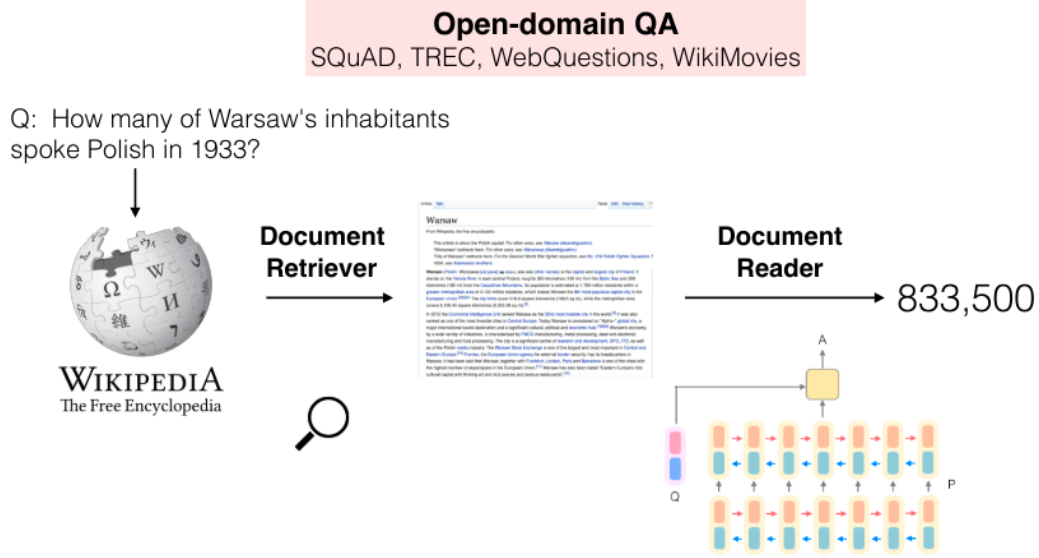


Figure 1: An overview of our question answering system DrQA.

- Our system DrQA for QnA consists of two components: (1) Document Retriever module for finding relevant articles and (2) a machine comprehension model, Document Reader, for extracting answers.
- Document retriever for QnA uses tf-idf vectors for retrieving relevant context. For our project we have trained and inference only the retriever part on Squad dataset.

5.2 Model Components:

a. Paragraph Encoding:

- In this step, each paragraph of the document is encoded into a representation that captures contextual information.
- $p_1, \dots, p_m = RNN(p_1, \dots, p_m)$, where the feature vector p_i is comprised of the following parts
 - *Word Embeddings* : $=E(pi)$ which represents the 300-dimensional Glove word embeddings.

- *Exact Match*: $=I(p_i \in q)$, where $I(\cdot)$ is the indicator function.
- *Token Features*: $token(p_i) = (POS(p_i), NER(p_i), TF(p_i))$, where POS represents part-of-speech, NER represents named entity recognition, and TF represents term frequency.
- *Aligned Question Embedding*: $align(p_i) = \sum a_{i,j} E(q_j)$ where $a_{i,j}$ is the attention score between p_i and q_j where:

$$a_{i,j} = \frac{\exp(\alpha(E(p_i)) \cdot \alpha(E(q_j)))}{\sum_{j'} \exp(\alpha(E(p_i)) \cdot \alpha(E(q_{j'})))}$$

b. Question Encoding:

- *Word Embeddings*: $q = \sum_j b_j q_j$, where q_j is the embedding of the individual token and b_j encodes the importance of each question word.

$$b_j = \frac{\exp(w \cdot q_j)}{\sum_{j'} \exp(w \cdot q_{j'})}$$

c. Start and End token prediction:

- At the paragraph level, the model aims to predict the most likely span of tokens that answers the question.
- We take the the paragraph vectors p_1, \dots, p_m and the question vector q as input and use attention based architecture particularly **bilinear attention** to encode likelihood of word as a start and end token.
- *Probability Computation*:

$$P_{start}(i) \propto \exp(p_i W_s q)$$

$$P_{end}(i) \propto \exp(p_i W_e q)$$

- The training objective is the sum of the log-likelihoods of the correct start and end positions, in which we backtrack over matrices W_s and W_e .
- During prediction, we choose the best span from token i to token i' such that $i \leq i' \leq i + 15$ and $P_{start}(i) \times P_{end}(i')$ is maximized.

5.3 Results and Analysis:

EM Score: 69.5 (as proposed in paper with tuned hyperparameters)
F1 Score: 78.8

Features	F1
Full	78.8
No f_{token}	78.0 (-0.8)
No f_{exact_match}	77.3 (-1.5)
No $f_{aligned}$	77.3 (-1.5)
No $f_{aligned}$ and f_{exact_match}	59.4 (-19.4)

Table 5: Feature ablation analysis of the paragraph representations of our Document Reader. Results are reported on the SQuAD development set.

- DrQA system (single model) can achieve 70.0% exact match and 79.0% F1 scores on the test set due to attention based mechanisms. It is not a transformer based architecture and surpasses top performers on the SQuAD leaderboard at the time of writing.
- Additionally, DrQA model is conceptually simpler than most of the existing system (BiDAF, Dynamic Coattention Networks, etc.)
- More interestingly, if we remove both $f_{aligned}$ and $f_{exactmatch}$ match from paragraph encoding the performance drops dramatically by $\sim 20\%$, so we conclude that both play a similar but complementary role in the feature representation related to the paraphrased nature of a question vs. the context around an answer.

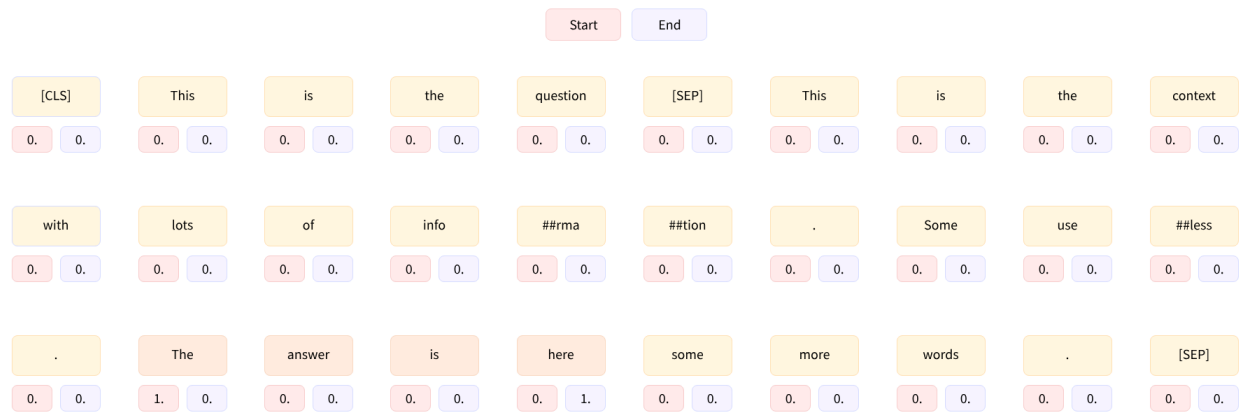
6. BERT

6.1 Preprocessing

1. Tokenization: Combine question and context, add special tokens ([CLS] and [SEP]).

[CLS] question [SEP] context [SEP]

2. Labeling: Set start and end indices for answers. For long contexts, create multiple inputs with sliding window.



In this case the context is not too long, but some of the examples in the dataset have very long contexts that will exceed the maximum length we set (which is 384 in this case). We will deal with long contexts by creating several training features from one sample of our dataset, with a sliding window between them.

For example:

```
[CLS] To whom did the Virgin Mary allegedly appear in 1858 in L
[CLS] To whom did the Virgin Mary allegedly appear in 1858 in L
[CLS] To whom did the Virgin Mary allegedly appear in 1858 in L
[CLS] To whom did the Virgin Mary allegedly appear in 1858 in L
```

1. Handling long contexts: Split examples into multiple inputs. For answers not fully included, set `start_position = end_position = 0`.
2. Padding: Pad inputs to maximum length. No dynamic padding as most contexts are long.
3. Validation data: No need for labels. Store offset mappings and match features to original examples using ID column.

6.2 Fine-Tuning

Since we padded all the samples to the maximum length we set, there is no data collator to define. The difficult part is to post-process the model predictions into spans of text in the original

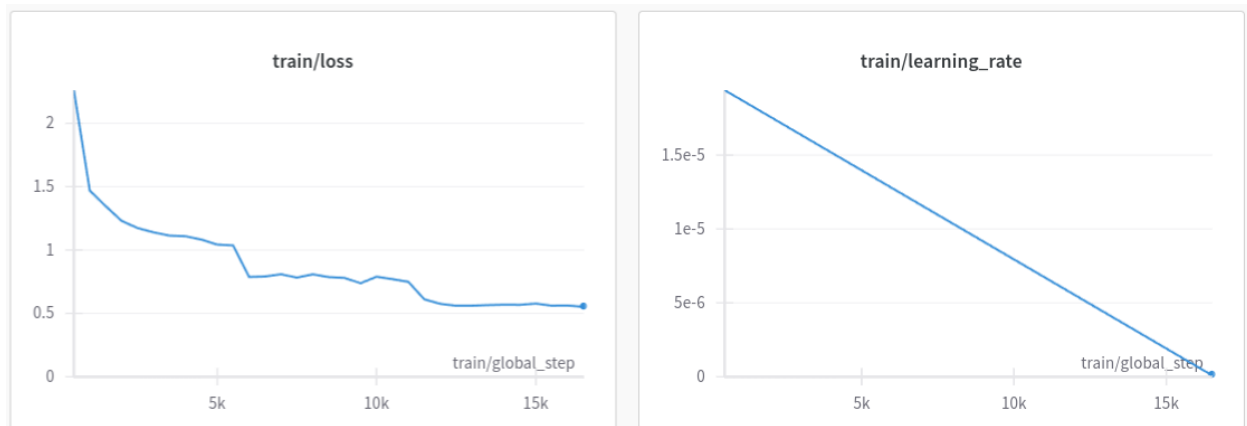
examples, once we have done that, the metric from the hugging face Datasets library will help us evaluate the model.

Model that we use is : **“Distilbert-base-cased-distilled-squad”**

6.3 Post Processing

1. Compute probabilities for start and end logits using softmax.
2. Calculate score for each (start_token, end_token) pair by multiplying corresponding probabilities.
3. Find pair with maximum score for valid answer (start_token < end_token).
4. Change process: Skip softmax step and score only the highest n_best logits (n_best=20). Use logit scores obtained by summing start and end logits.

6.4 Results and Analysis



EM Score: 80.92

F1 Score: 88.38

Comparison across various models

<u>Models</u>	<u>EM score</u>	<u>F1 Score</u>
Seq2Seq Model	0.4432	0.5125
BIDAF	0.5732	0.6928
DrQA	0.695	0.788
BERT	0.8092	0.8838