Intro To NLP

NLP BOIS

Nikunj Garg	2021101021
Vansh Marda	2021101089
Manav Shah	2021101090

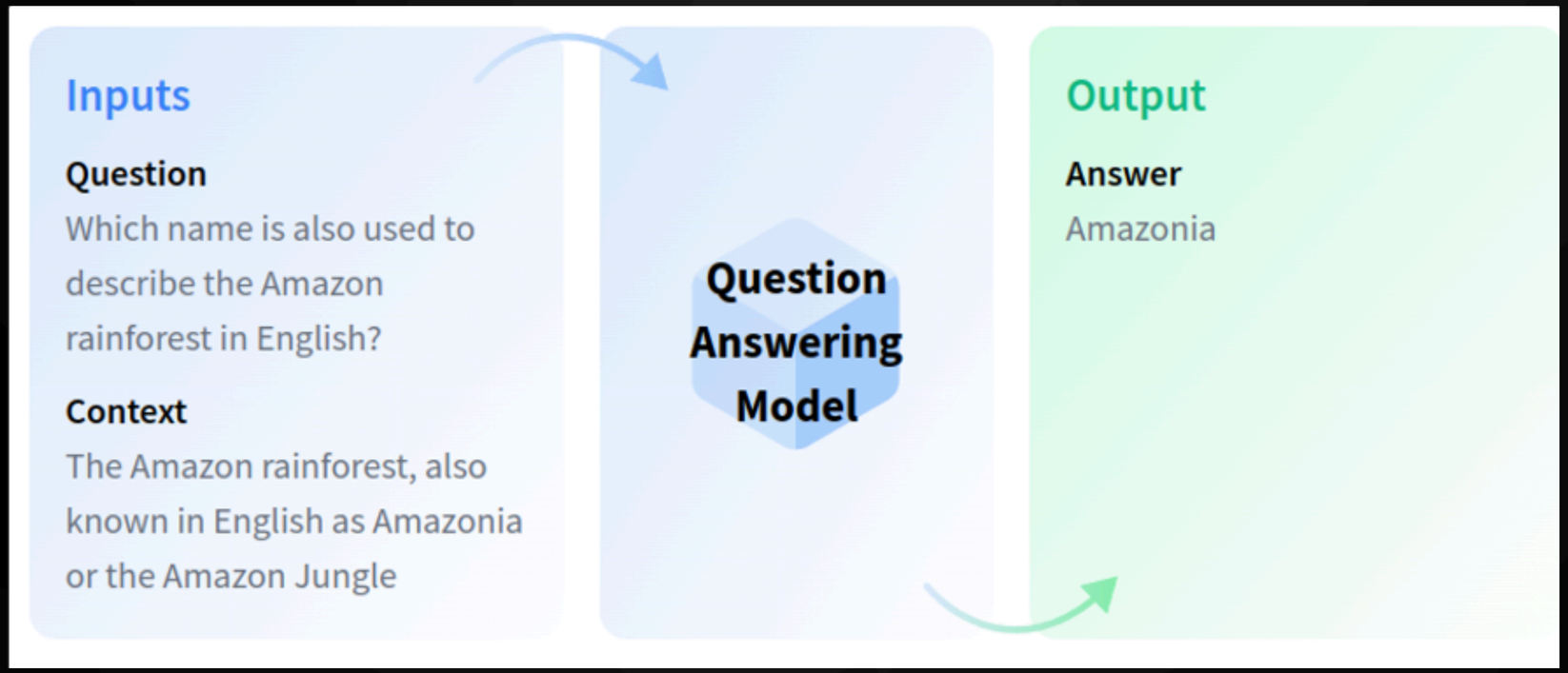
Content

- Problem Statement
- Different Methodologies in QnA
- Exploring Dataset
- Evaluation Metric
- Models
  - Unsupervised Learning
    - Jaccard Similarity
    - TF-IDF Vectors
    - Sent2vec Encoders
  - Supervised Learning
    - Logistic Regression
    - SeqtoSeq Model
    - BIDAF
    - DrQA
    - BERT
- Comparision

Problem Statement

- Develop a Question Answering (QA) system for efficient retrieval of information from structured data or natural language documents.

- Address the challenge of providing precise answers to user queries amidst the growing volume of information.



Different Methodologies in QnA

- **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.
- **Closed-domain QA:** Limited to a single domain.
- **Open-domain QA:** Provides answers for any domain but with lower accuracy due to lack of specificity.

Exploring Dataset

- **SQUAD** - 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. This dataset is for extractive QA.


```
{
    "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

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

- F1 score:

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

$$\text{precision} = tp/(tp + fp)$$

$$\text{recall} = tp/(tp + fn)$$

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.

Unsupervised Learning Methods

A) Jaccard Similarity

- similarity score between two sentences on the basis of common words appearing in them.

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

B) Tf-idf vectors

- 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
  - tf(t,d) = count of t in d / number of words in d
- We use similarity score as the euclidean distance between 2 tf-idf vectors.

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

C) Sent2vec Encoders

- Sent2Vec encoders are transformer based models used for generating vector embeddings of sentences.
- Cosine similarity bw encoded question and context paragraph sentences is used as comparison metric

$$similarity \propto e(q)^T e(p)$$

Results:

Similarity Method	Accuracy
Jaccard	63.2%
TF-IDF vectors	58.8%
Sen2vec Encoders	66.6%

Supervised Learning Methods

Logistic Regression

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

Data Preprocessing

Model

- Set up dataset for binary classification.
- Train logistic regression model.
- Construct feature vectors for each question-context pair by converting them to word embeddings and concatenating them.

Analysis

- Accuracy: 71%
- Cannot compare with supervised methods
- Same word always mapped to same vector

Seq2seq-Attention Question Answering Model

Data Preprocessing

- Auto-tokenizer is used from transformers from tokenization.
- Appropriate padding is done to construct a tokenized dataset so that we can perform batch wise learning and to construct dataloader.
- We have used Glove Embeddings as our embedding matrix. So, if a word is not in glove embeddings, it is initialized to a random value.

Model Architecture

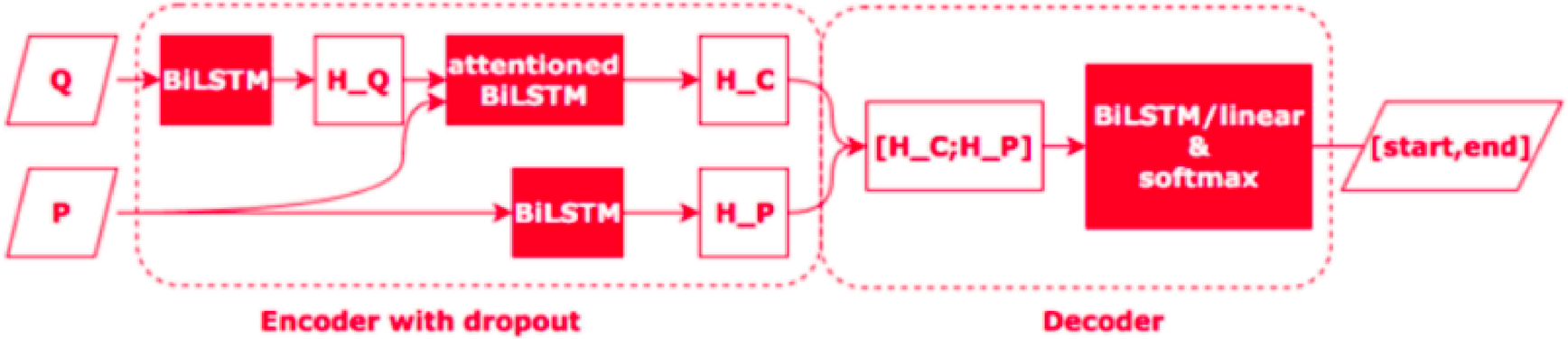
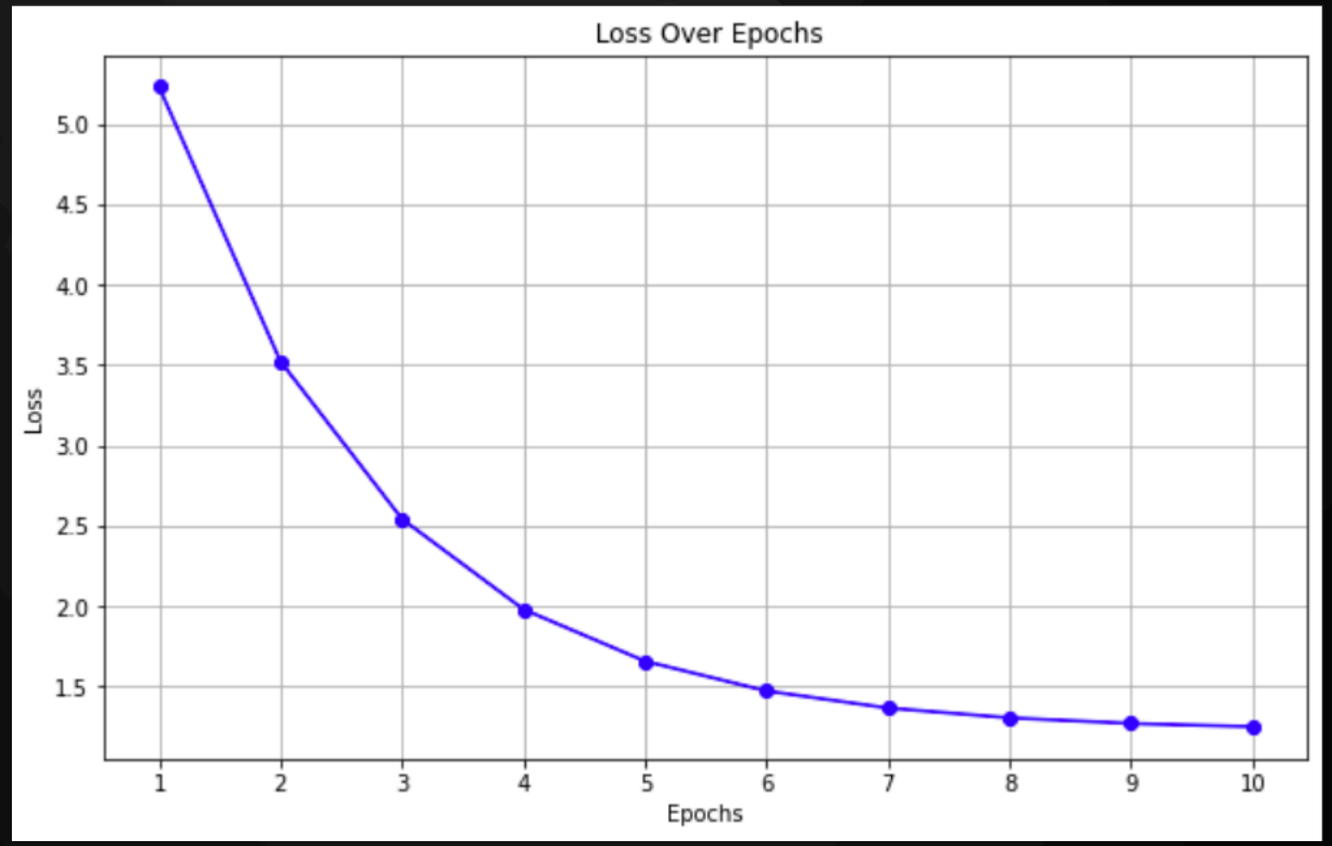


Fig.2 model architecture

- Encoder Decoder pair both containing Bi-directional LSTMs.
- Query to Context Attention is used in Encoder.
- Gradient Clipping was also used to avoid overshooting of gradients as multiple BiLSTMs are involved in the architecture.

Loss v/s Epoch Curve



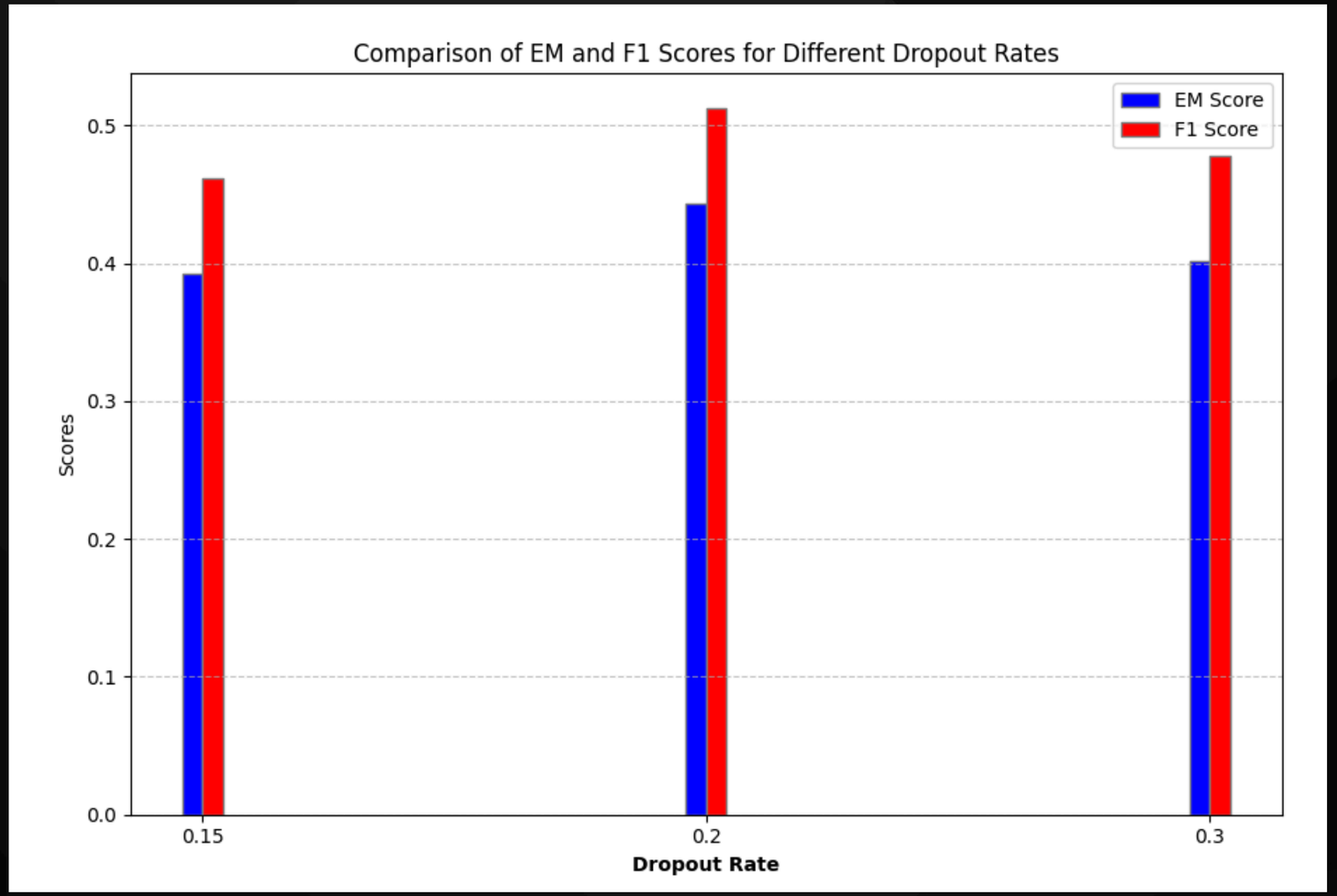
- We ran the code for 10 epochs. Each epoch approximately took 15 mins with GPU.

Results

- Training Data
  - EM Score = 0.532
  - F1 score = 0.653
- Validation Data
  - EM Score = 0.4432
  - F1 score = 0.5125

- Hyper-Parameters Used
  - Embedding Dimension = 200
  - Learning Rate = 0.001
  - Optimizer = Adam
  - Cross Entropy Loss for start and end logits.

Variation in Dropout Rates



BIDAF (Bidirectional Attention Flow) Model

Data Preprocessing

- Data Preprocessing is done in a similar manner as in the previous model.

Character Embedding Layer.

- char_idx is also calculated and each character is assigned a vector whose length is called character embedding dimension.
- The problem of unseen words can be solved by using character embeddings layer.
- This layer can be visualized as a CNN which helps to find the patterns at the character layer.

Model Architecture

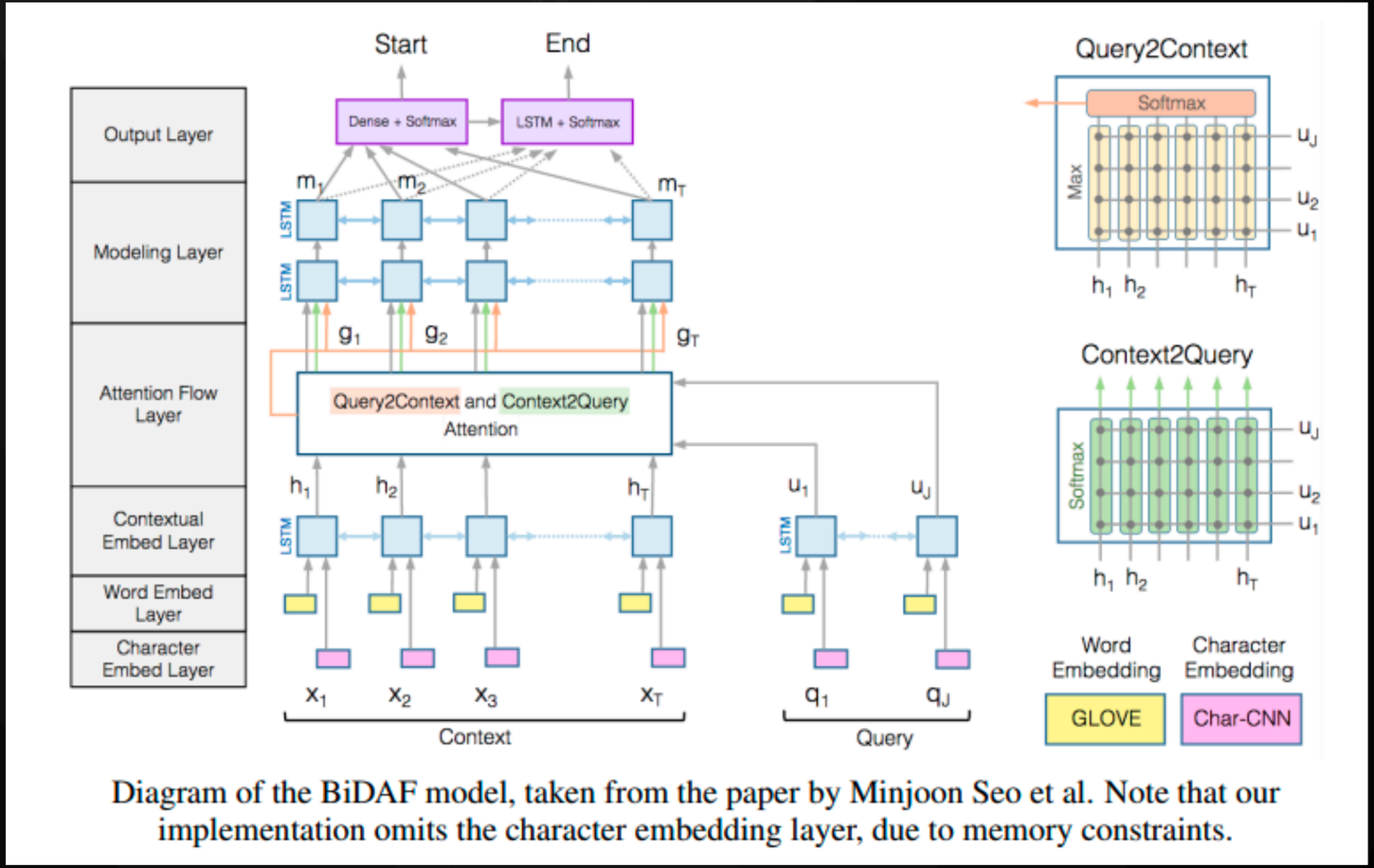


Diagram of the BiDAF model, taken from the paper by Minjoon Seo et al. Note that our implementation omits the character embedding layer, due to memory constraints.

Description of the Layers Used

- **Embeddings**: Three types of embedding layers are used. Word Embeddings Layer (Glove) , Character Embedding Layer and Contexutal Embedding Layer is used.

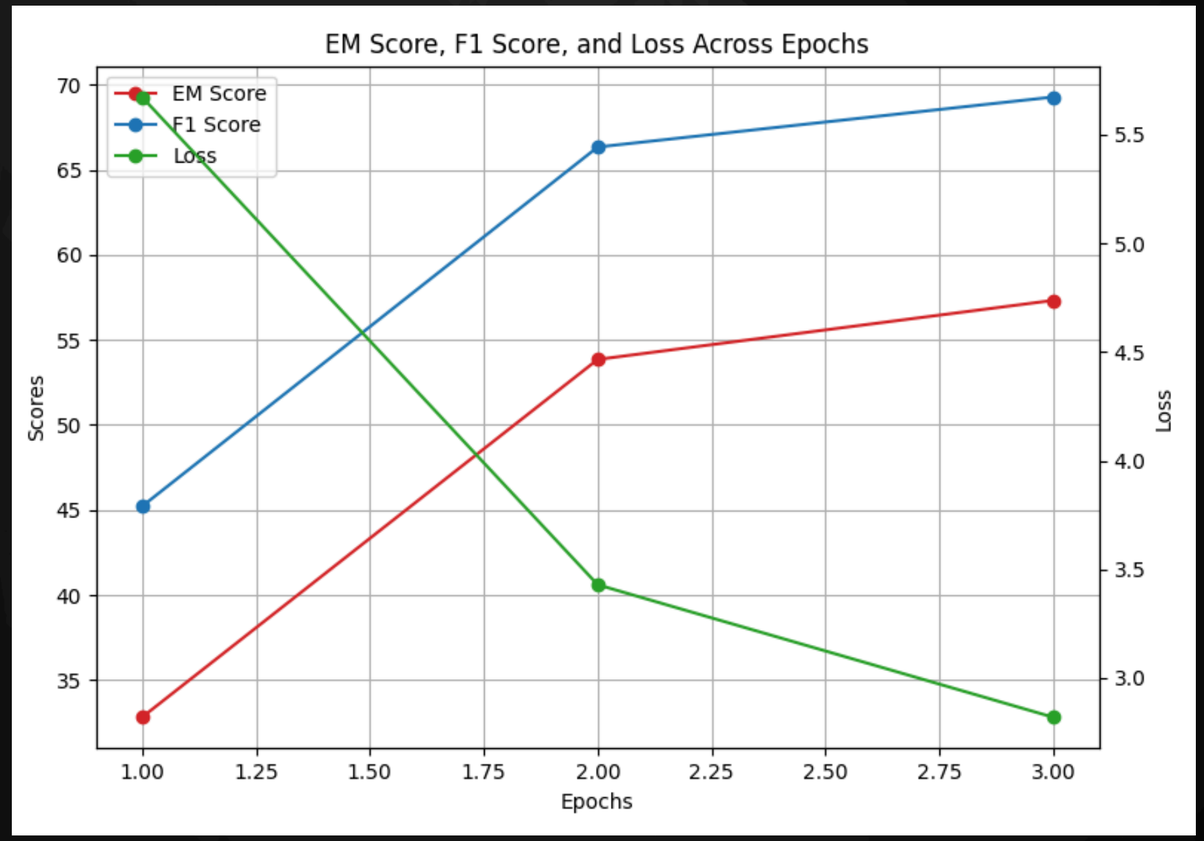
- **Attention Flow Layer**: This layer is the core of the BiDAF model, focusing on linking and fusing information from the question and the context. Two types of attention namely query to context and context to query attention is used. That’s why this model is called BIDAF.

Description of the Layers Used

- **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.

- **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



- EM and F1 score is on the validation data and Loss is on the training data.

- **Final Results :** After three epochs value of EM is 57.326 and F1 is 69.282

- **Run Time :**
  - The training time was significantly high, 1.5 hours to preprocess the dataset and then for each epoch, model taking approximately 80 minutes on GPU.
  - Parallelization by GPU didn’t help much as the model is much more complex and it is not designed to perform parallelizaton.
  - Presence of multiple layers and multiple LSTMs reduces parallelization.

Sample Test Example

- **Context :** beyonce became popular in 1990s
- **Question :** when did beyonce became popular
- **Start Logits : [-2.3235, -1.6142, -1.0650, 4.4662, 8.0298]**
- **End Logits : [-4.2915, -3.2854, -3.2325, -1.1319, 7.5230]**
- So, the two answers with highest probability are "**1990s**" and "**in 1990s**"
- So, we can analyze that model does give very short and precise answer for the question. The model is not much open ended.

DrQA (Reading Wikipedia to Answer Open-Domain Questions)

DrQA Model

- Our system DrQA for QnA consists of two components:
  - Document Retriever module for finding relevant articles.
  - A machine comprehension model, Document Reader, for extracting answers.

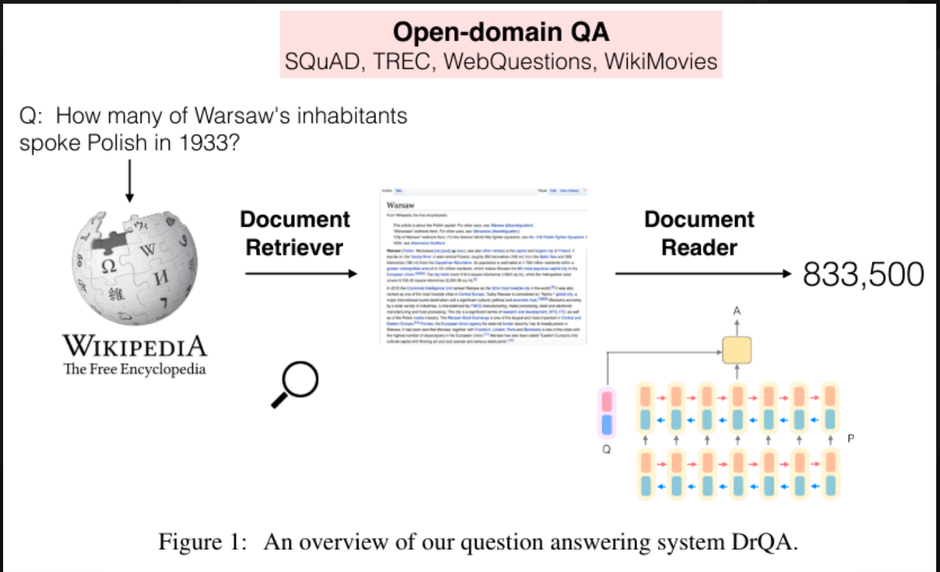


Figure 1: An overview of our question answering system DrQA.

- 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.

DrQA Model Components

- Paragraph Encoding: feature vector pi comprised of the following parts:
  - *Word Embeddings: $E(pi)$ which represents the 300-dimensional Glove word embeddings.*
  - *Exact Match: I(pi∈q), where $I(\cdot)$ is the indicator function*
  - *Token Features: token(pi) = (POS(pi), NER(pi) ,TF(pi))*
  - *Aligned Question Embedding: align(p_i)=∑a_{i,j} E(q_j) where $a_{i,j}$ is the attention score between p_i and q_j:*

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

- RNN Model:
  - Final embedding is the concatenation of embedding of multi layers.

$$p_1, \ldots, p_m = RNN(p_1, \ldots, p_m)$$

DrQA Model Components

- **Question Encoding: W**ord Embeddings : q=∑_jb_jq_j, where q_j is the embedding of the questionl token and b_j encodes the importance of each question word:

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

- **Start and End token prediction:** We take the the paragraph vectors {p_1,...,p_m} and the question vector q as input and use **bilinear** attention to encode likelihood of word as a start and end token.

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

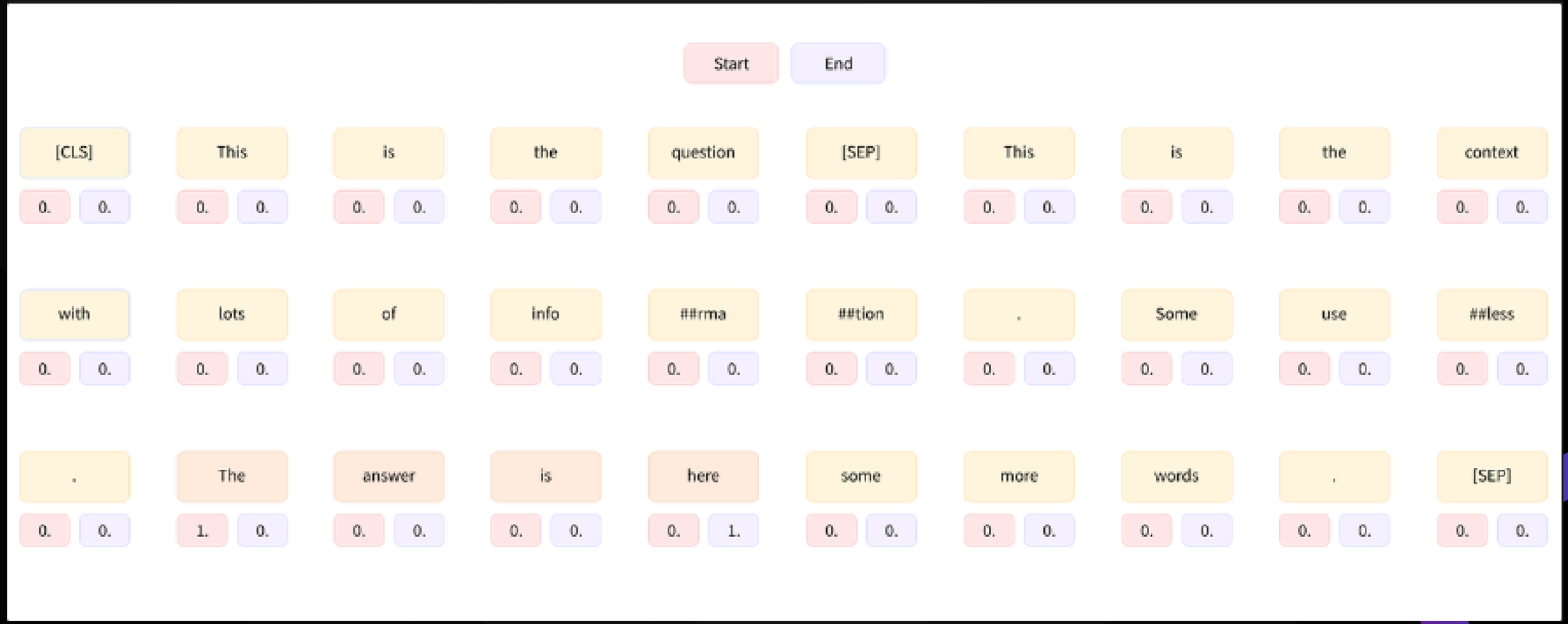
$$P_{end}(i) \propto exp(p_i W_e q)$$

DrQA Model Inference

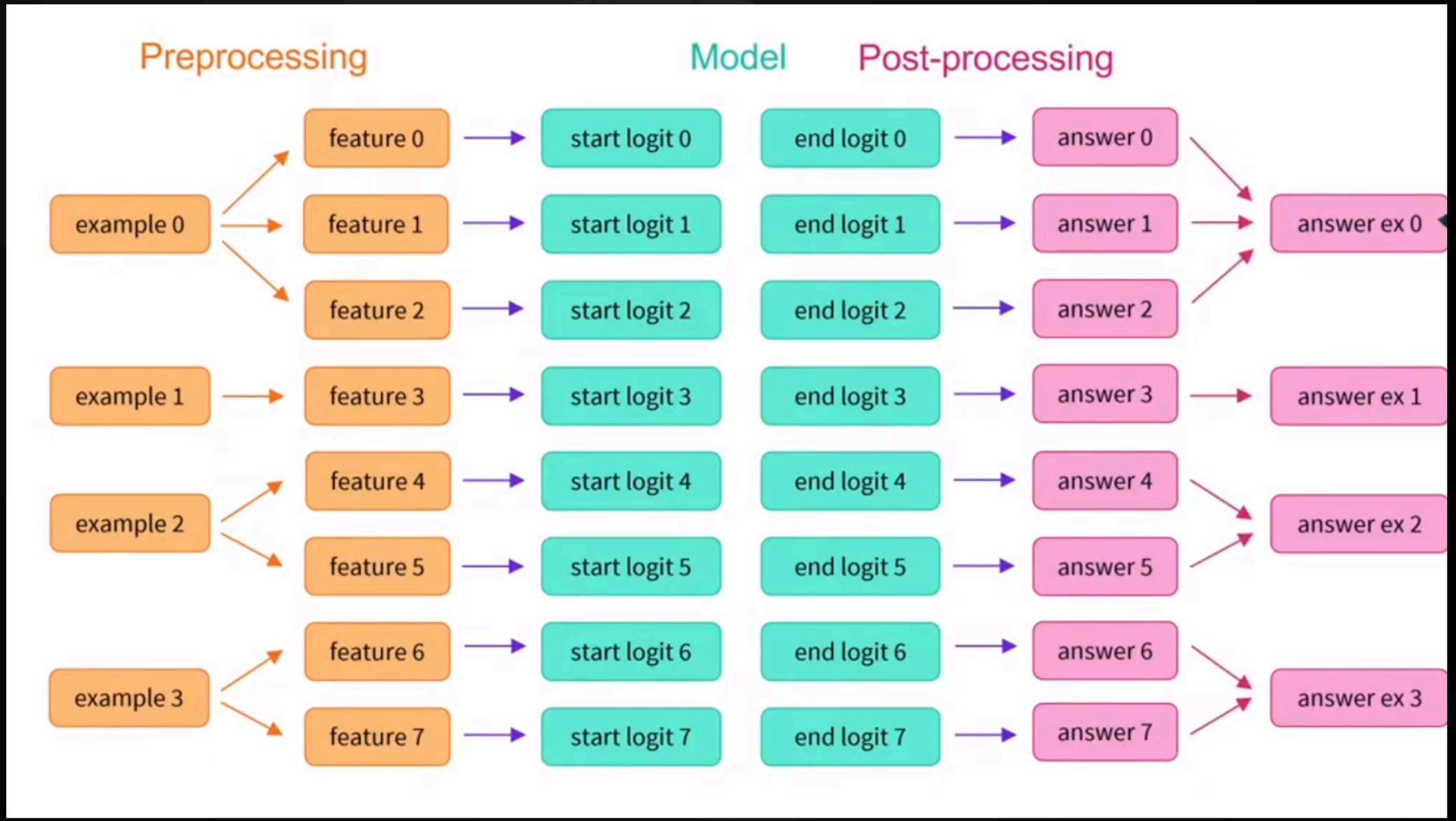
- During prediction, we choose the best span from token i to token i^{'} such that i ≤ i^{'} ≤ i + 15 and Pstart(i)×Pend(i^{'}) is maximized .
- The training objective is the sum of the log-likelihoods of the correct start and end tokens.
- EM Score: 69.5 (as proposed in paper with tuned hyperparameters), F1 Score: 78.8.
- EM Score: 57.34% (our implementation), F1 Score: 69.3014438187849
- Interestingly, if we remove both faligned and exactmatch from paragraph encoding the performance drops dramatically by ~ 20%

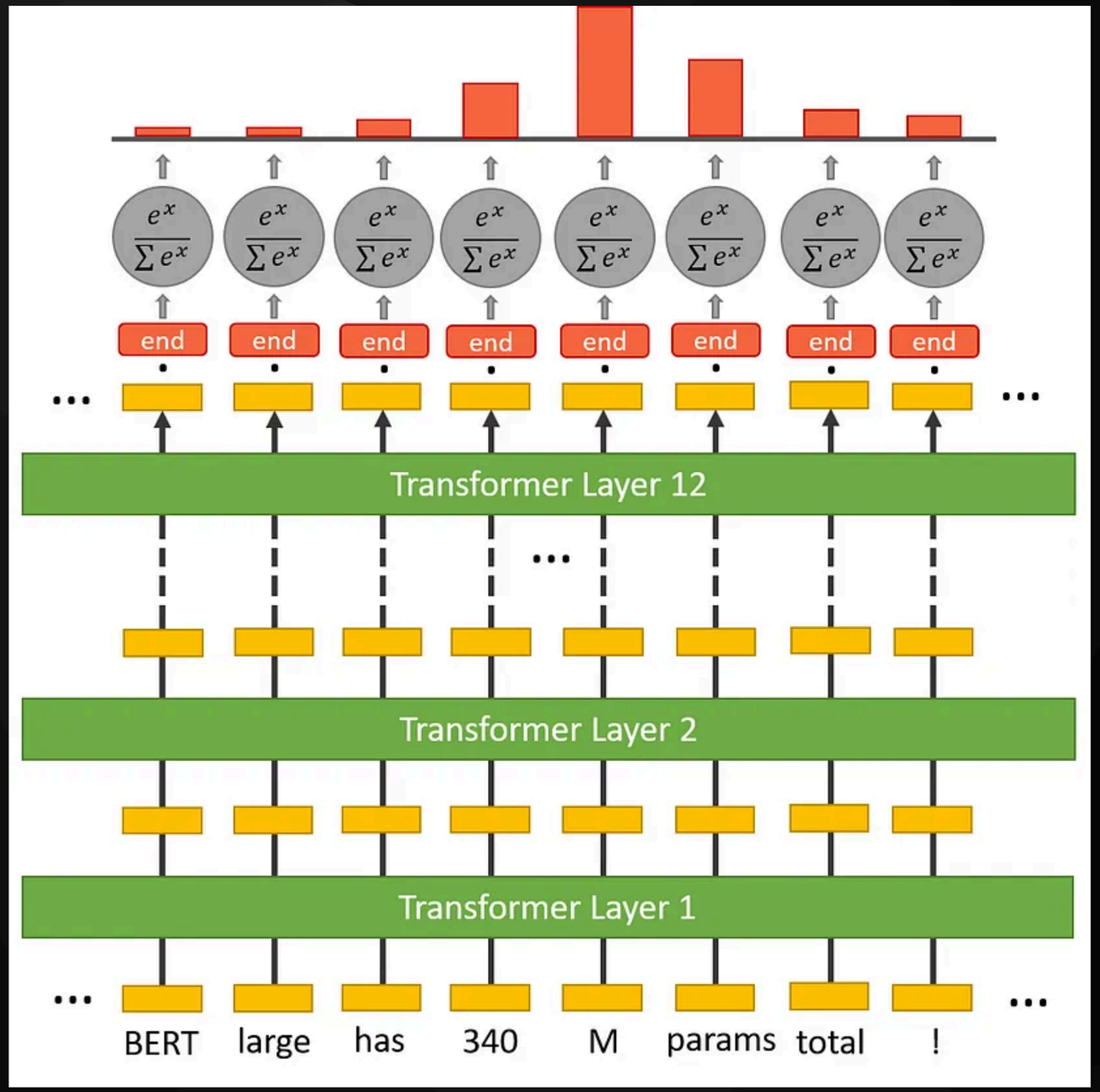
BERT

Data Preprocessing



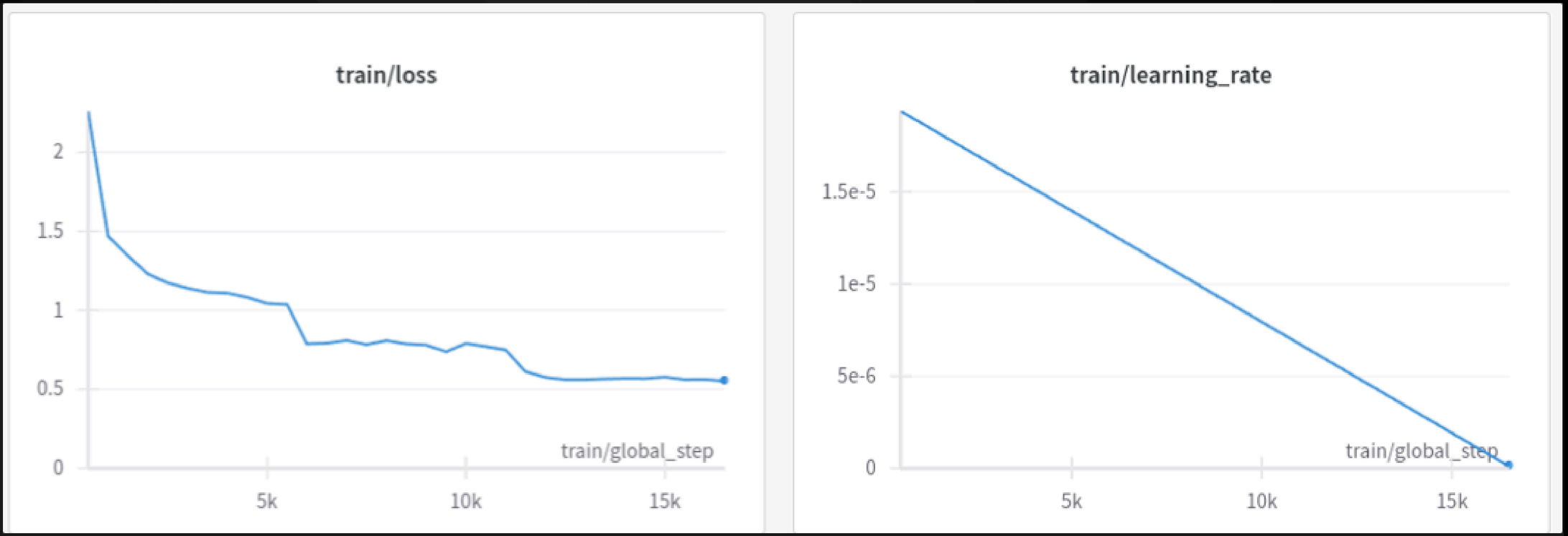
Model


Preprocessing
Model
Post-processing
example 0
example 1
example 2
example 3
feature 0
feature 1
feature 2
feature 3
feature 4
feature 5
feature 6
feature 7
start logit 0
start logit 1
start logit 2
start logit 3
start logit 4
start logit 5
start logit 6
start logit 7
end logit 0
end logit 1
end logit 2
end logit 3
end logit 4
end logit 5
end logit 6
end logit 7
answer 0
answer 1
answer 2
answer 3
answer 4
answer 5
answer 6
answer 7
answer ex 0
answer ex 1
answer ex 2
answer ex 3


$\frac{e^x}{\sum e^x}$
end
Transformer Layer 12
Transformer Layer 2
Transformer Layer 1
BERT
large
has
340
M
params
total
!

Analysis

- EM Score: 80.92
- F1 Score: 88.38



Comparison Across Vaious Models

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