University of New Haven

Tagliatela college of Engineering

Master’s in Data Science

Course : Natural Language Processing

Project Title : BiDAF Model for Question Answering

Names : Arpitha Busireddy

Under the guidance of,

Prof. Vahid Behzadan.

vbehzadan@newhaven.edu

INTRODUCTION

Several businesses and academic organizations have invested large money in the development of question answering systems over the years due to their latent potential in deciphering unstructured text. These artificial intelligence systems attempt to capture the semantic and syntactic links between words in modern language. Several businesses and academic organizations have invested large money in the development of question answering systems over the years due to their latent potential in deciphering unstructured text. These artificial intelligence systems attempt to capture the semantic and syntactic links between words in modern language.

There are three different types of question-answering models

1) Open domain vs closed domain

2) Abstractive vs extractive

3) Ability to answer non factoid questions.

Open domain has an access to knowledge database, Closed domain has context to answer the question. An Extractive model just answers the question exactly from the context, whereas an Abstractive model paraphrases the answer and gives it as human readable format. Ability to answer non factoid questions like, giving explanations other than saying yes or no.

BiDAF model falls under the category of Closed domain, extractive, and can answer only factoid questions.

In this project, We want to identify answers in the SQuAd dataset, which contains tens of thousands of paragraph question pairs. To do so, we used the Bi-directional Attention Flow model (BiDAF), which is widely used in NLP to solve question answering difficulties. Using GloVe word embeddings, we encode the question and the accompanying context paragraph, compute an attention matrix, then decode the context paragraph to determine the answer to the question.

METHOD

BiDAF is an extractive Q&A model with a closed domain. This means that in order to respond to a Query, BiDAF must review an accompanying text containing the details required to respond to the Query. The Context is the name given to the corresponding text. BiDAF extracts a substring from the Context that best addresses the question which we call the Answer to the Query.

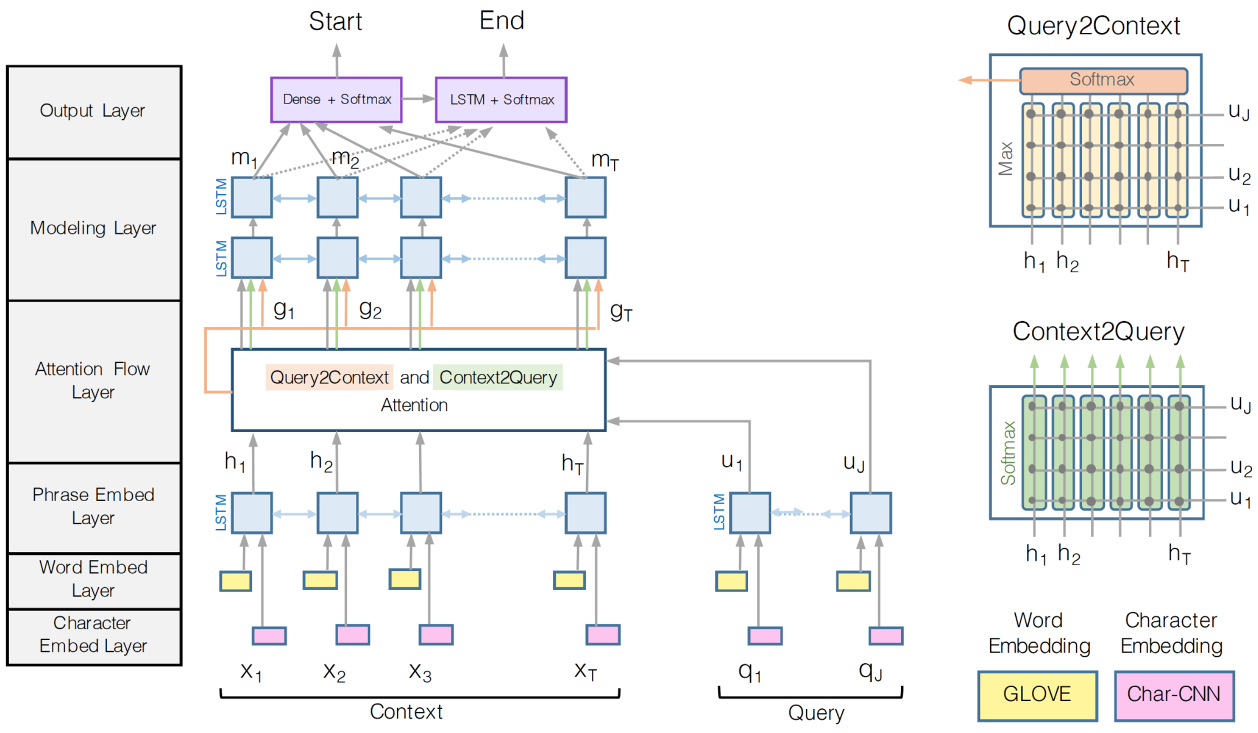


Fig: BiDAF Model

BiDAF comprises different layers as shown in the figure above.

Word Embedding Layer:

To begin, we use pre-trained word vectors GLoVe as the model's initial word representations. This layer transforms each word from a discrete to a continuous high-dimensional vector. The GloVe representations were pre-trained; their values were locked and will not change throughout training. As a result, the word embedding stage of BiDAF can be thought of as a basic dictionary lookup step in which we replace words (the GloVe 'dictionary's' keys) with vectors (the 'dictionary's' values). The word embedding process yields two matrices: one for the Context and one for the Query.

e(ci) = f([GloVe(ci);charEmb(ci)])

e(qi) = f([GloVe(qi);charEmb(qi)])

Contextual Layer:

The word embeddings are then passed into two bi-directional LSTMs, one for meaning and one for inquiry. This allows you to see how the terms in the meaning or question are related. BiDAF employs a bidirectional LSTM (bi-LSTM) that consists of forward and backward LSTM sequences. The forward and backward LSTM output embeddings combine to encode information from both past (backwards) and future (ahead) states.

$$\overrightarrow{\mathbf{c}}_i = \text{LSTM}(\overrightarrow{\mathbf{c}}_{i-1}, e(c_i)) \in \mathbb{R}^H$$
$$\overleftarrow{\mathbf{c}}_i = \text{LSTM}(\overleftarrow{\mathbf{c}}_{i+1}, e(c_i)) \in \mathbb{R}^H$$
$$\mathbf{c}_i = [\overrightarrow{\mathbf{c}}_i; \overleftarrow{\mathbf{c}}_i] \in \mathbb{R}^{2H}$$

$$\overrightarrow{\mathbf{q}}_i = \text{LSTM}(\overrightarrow{\mathbf{q}}_{i-1}, e(q_i)) \in \mathbb{R}^H$$
$$\overleftarrow{\mathbf{q}}_i = \text{LSTM}(\overleftarrow{\mathbf{q}}_{i+1}, e(q_i)) \in \mathbb{R}^H$$
$$\mathbf{q}_i = [\overrightarrow{\mathbf{q}}_i; \overleftarrow{\mathbf{q}}_i] \in \mathbb{R}^{2H}$$

Attention Layer:

During this layer, we assess the bi-directional importance of each context word on each question word, as well as each question word on each context word. The idea behind this is that we want knowledge to flow in both directions when we are paying attention. To begin, we'll create a matrix and calculate a similarity score for each context-query word pair. The next stage is to figure out what context to query attention in. The background focus question is then computed. Finally, the bi-directional layer attention output is created by combining the two computed attention outputs.

First, compute a similarity score for every pair of $(\mathbf{c}_i, \mathbf{q}_j)$:

$$S_{i,j} = \mathbf{w}_{\text{sim}}^{\mathsf{T}}[\mathbf{c}_i; \mathbf{q}_j; \mathbf{c}_i \odot \mathbf{q}_j] \in \mathbb{R} \qquad \mathbf{w}_{\text{sim}} \in \mathbb{R}^{6H}$$

Context-to-query attention (which question words are more relevant to c_i):

$$\alpha_{i,j} = \text{softmax}_j(S_{i,j}) \in \mathbb{R} \qquad \mathbf{a}_i = \sum_{j=1}^{M} \alpha_{i,j}\mathbf{q}_j \in \mathbb{R}^{2H}$$

Query-to-context attention (which context words are relevant to some question words):

$$\beta_i = \text{softmax}_i(\max_{j=1}^{M}(S_{i,j})) \in \mathbb{R}^N \qquad \mathbf{b} = \sum_{i=1}^{N} \beta_i\mathbf{c}_i \in \mathbb{R}^{2H}$$

Modeling layer:

Before moving on to the final output layer, the model can use the modeling layer to undertake extra processing. The model acquires contextual knowledge between the context words based on what it's learnt so far about the query and its link to the context in both directions. We do this by sending the Attention layer's output into a totally connected layer that first down samples the dimensionality of each representation. After that, the information is transferred via two layers of bidirectional LSTMs. After that, the data is sent through a fully connected layer. Using entirely linked layers, we can shrink the model without sacrificing learning capability.

$$\text{mi} = \text{BiLSTM(gi)} \in \text{R2H}$$

Output Layer:

We create a prediction about where the span should start and utilize the result of that prediction to learn where the span should stop in the output layer. The hidden states that specify the start location are created by using the output from the modeling layer. We use a linear layer and a softmax to obtain the start pointer. The start pointer's hidden layer is then supplied into a second LSTM, which produces the end pointer distribution by passing its hidden states through a linear layer into a softmax. The greatest join likelihood of the start and end points was then determined, under the assumption that the start pointer comes first.

$$p_{\text{start}} = \text{softmax}(\mathbf{w}_{\text{start}}^{\mathsf{T}}[\mathbf{g}_i; \mathbf{m}_i]) \qquad p_{\text{end}} = \text{softmax}(\mathbf{w}_{\text{end}}^{\mathsf{T}}[\mathbf{g}_i; \mathbf{m}'_i])$$

$$\mathbf{m}'_i = \text{BiLSTM}(\mathbf{m}_i) \in \mathbb{R}^{2H} \quad \mathbf{w}_{\text{start}}, \mathbf{w}_{\text{end}} \in \mathbb{R}^{10H}$$

EXPERIMENTS

Dataset:

This dataset is two separate json files with dev and training data, The data consists of context, its title, several related questions and answers. The dev data consists of 10570 questions and answers.

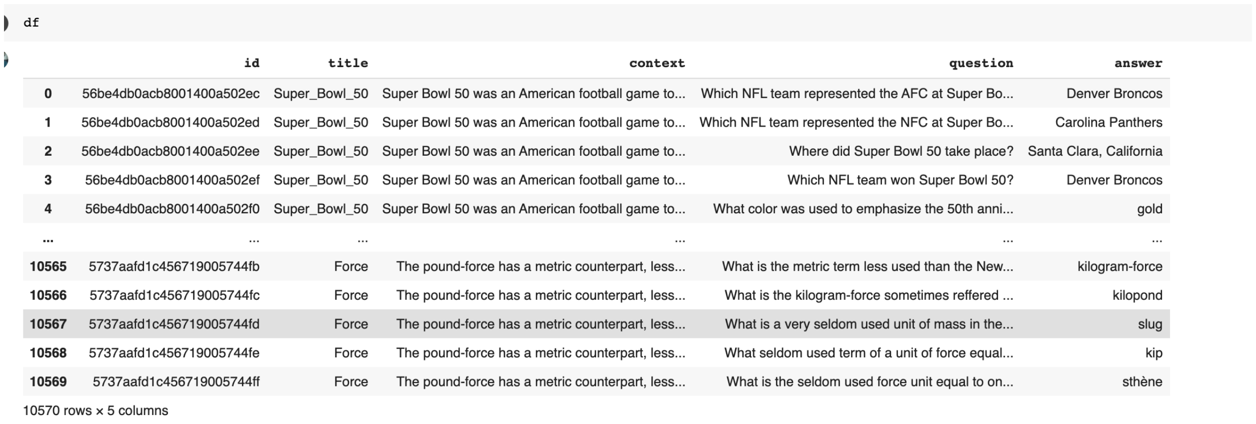
Data Pre-processing:

Steps followed in data preprocessing,

- Uploaded your dataset into google drive.
- Loaded json data into pandas data frame for both dev and train datasets one by one for context, query and answers.
- After loading the datasets, we need to transform the data
- Took all the unique context from the data frame and split each word in each context
- Took all the questions from data frame and split each words into string for each questions
- Took all the answers from data frame and split each words into string for each answers
- Created a vocabulary for each context, each questions and each answers
- Added pad and unk to each context, question and answers
- Converted each word in context, query and answer into index using word2idx function.

- After converting, we need to load data using data loader

- Created a batch of 16 and created train loader and test loader which will be the input to the model.



```
df
```

	id	title	context	question	answer
0	56be4db0acb8001400a502ec	Super_Bowl_50	Super Bowl 50 was an American football game to...	Which NFL team represented the AFC at Super Bo...	Denver Broncos
1	56be4db0acb8001400a502ed	Super_Bowl_50	Super Bowl 50 was an American football game to...	Which NFL team represented the NFC at Super Bo...	Carolina Panthers
2	56be4db0acb8001400a502ee	Super_Bowl_50	Super Bowl 50 was an American football game to...	Where did Super Bowl 50 take place?	Santa Clara, California
3	56be4db0acb8001400a502ef	Super_Bowl_50	Super Bowl 50 was an American football game to...	Which NFL team won Super Bowl 50?	Denver Broncos
4	56be4db0acb8001400a502f0	Super_Bowl_50	Super Bowl 50 was an American football game to...	What color was used to emphasize the 50th anni...	gold
...	...	...	...	...	...
10565	5737aafd1c456719005744fb	Force	The pound-force has a metric counterpart, less...	What is the metric term less used than the New...	kilogram-force
10566	5737aafd1c456719005744fc	Force	The pound-force has a metric counterpart, less...	What is the kilogram-force sometimes reffered ...	kilopond
10567	5737aafd1c456719005744fd	Force	The pound-force has a metric counterpart, less...	What is a very seldom used unit of mass in the...	slug
10568	5737aafd1c456719005744fe	Force	The pound-force has a metric counterpart, less...	What seldom used term of a unit of force equal...	kip
10569	5737aafd1c456719005744ff	Force	The pound-force has a metric counterpart, less...	What is the seldom used force unit equal to on...	sthène

10570 rows × 5 columns

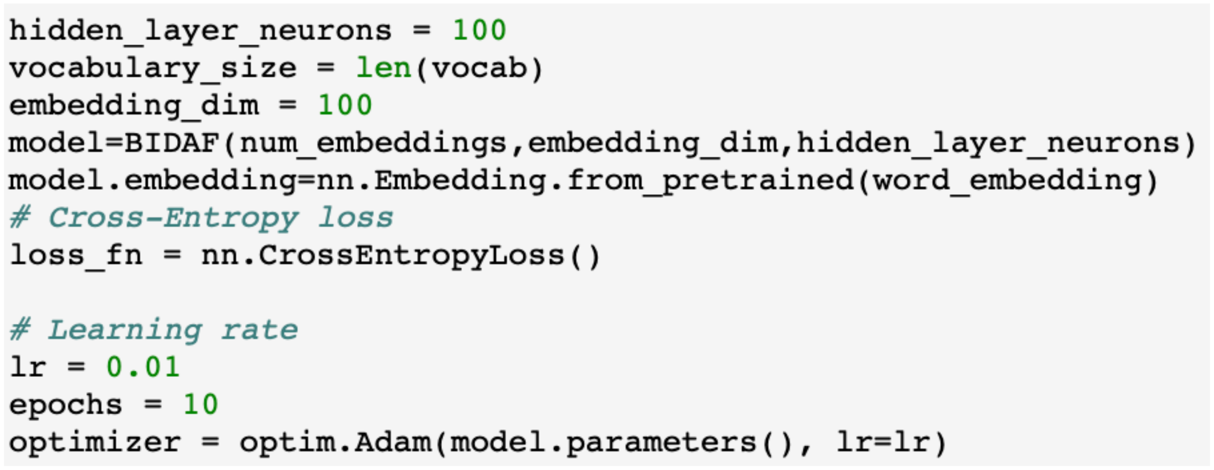
IMPLEMENTATION

Building a Network:

Now, after preprocessing the data, we are ready to build a model. For both the query and the background paragraph, we start with a word embedding layer that maps each word to an embedding vector (using pre-trained GloVe embeddings to represent words). We then use a BiLSTM to encode the query and context paragraph embeddings, concatenating the outputs of the forward and backward LSTMs. After that, we added an attention flow layer, in which we compute a similarity matrix that captures the similarity between each word in the paragraph and each word in the query. We may derive context-to-query and query-to-context matrices from this similarity matrix, which reflect the relevance of each word in the question to each word in the paragraph and the relevance of each word in the paragraph to each word in the question, respectively. We combine these two "attention" matrices with the original paragraph embedding to produce G, a matrix "where each column vector can be considered as the query-aware representation of each background term." G is fed into a two-layer BiLSTM, which produces M by concatenating the outputs of each path. Finally, we apply a linear layer and softmax to G and M to obtain probability distributions on the answer's start term. We run a second BiLSTM on M to get N, which is then used to measure the probability distribution of the answer's last term.

Parameter Settings:

Next, we assign a value of 100 to hidden layer neurons and 100 to embedding dimension. We used cross entropy loss to calculate the training loss and test loss. Finally, we use Adam optimizers with a learning rate of 0.01.



```
hidden_layer_neurons = 100
vocabulary_size = len(vocab)
embedding_dim = 100
model=BIDAF(num_embeddings,embedding_dim,hidden_layer_neurons)
model.embedding=nn.Embedding.from_pretrained(word_embedding)
# Cross-Entropy loss
loss_fn = nn.CrossEntropyLoss()

# Learning rate
lr = 0.01
epochs = 10
optimizer = optim.Adam(model.parameters(), lr=lr)
```

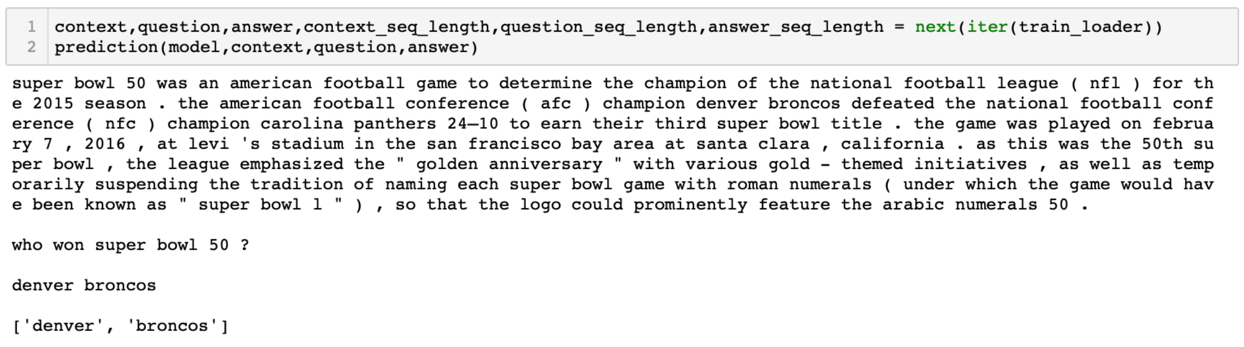
Fig 7: Parameter Setting

Training & Testing:

The number of epochs and the input data were the two key variables that were modified during the training phase. We trained our data to get train loss and start index and end index. Using the indexes to predict the start position and end position of the answer we then converted the index to word and finally predicted the answer for the given question.

The performance metrics like loss and accuracy were identified across the both training and testing datasets.

RESULTS



```
context,question,answer,context_seq_length,question_seq_length,answer_seq_length = next(iter(train_loader))
prediction(model,context,question,answer)
```

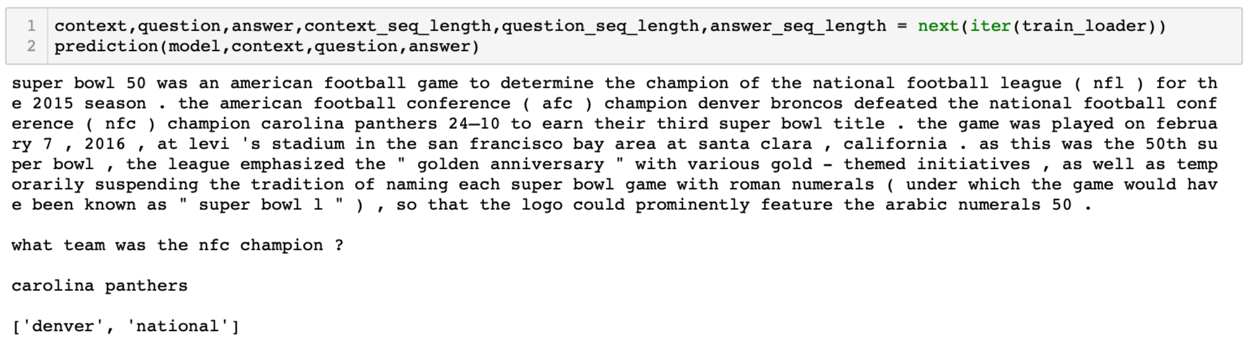
```
super bowl 50 was an american football game to determine the champion of the national football league ( nfl ) for th
e 2015 season . the american football conference ( afc ) champion denver broncos defeated the national football conf
erence ( nfc ) champion carolina panthers 24–10 to earn their third super bowl title . the game was played on februa
ry 7 , 2016 , at levi 's stadium in the san francisco bay area at santa clara , california . as this was the 50th su
per bowl , the league emphasized the " golden anniversary " with various gold - themed initiatives , as well as temp
orarily suspending the tradition of naming each super bowl game with roman numerals ( under which the game would hav
e been known as " super bowl l " ) , so that the logo could prominently feature the arabic numerals 50 .

who won super bowl 50 ?

denver broncos

['denver', 'broncos']
```

Output 1



```
context,question,answer,context_seq_length,question_seq_length,answer_seq_length = next(iter(train_loader))
prediction(model,context,question,answer)
```

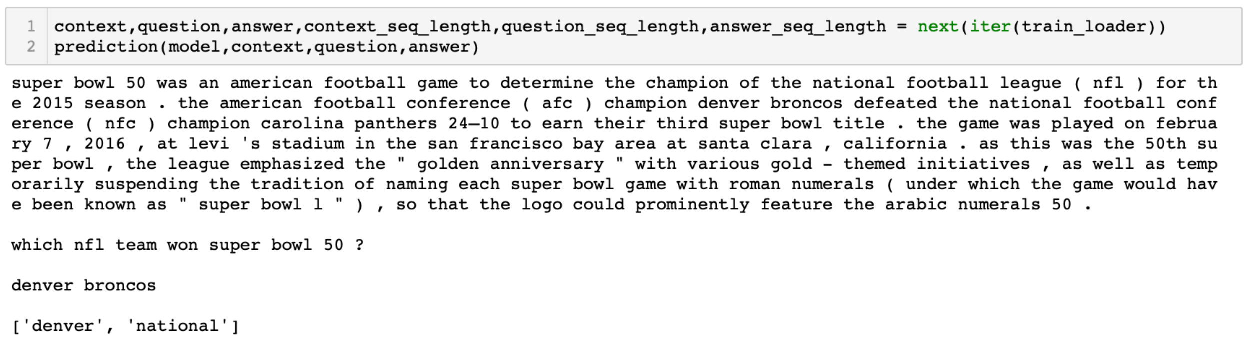
```
super bowl 50 was an american football game to determine the champion of the national football league ( nfl ) for th
e 2015 season . the american football conference ( afc ) champion denver broncos defeated the national football conf
erence ( nfc ) champion carolina panthers 24–10 to earn their third super bowl title . the game was played on februa
ry 7 , 2016 , at levi 's stadium in the san francisco bay area at santa clara , california . as this was the 50th su
per bowl , the league emphasized the " golden anniversary " with various gold - themed initiatives , as well as temp
orarily suspending the tradition of naming each super bowl game with roman numerals ( under which the game would hav
e been known as " super bowl l " ) , so that the logo could prominently feature the arabic numerals 50 .

what team was the nfc champion ?

carolina panthers

['denver', 'national']
```

Output 2



```
context,question,answer,context_seq_length,question_seq_length,answer_seq_length = next(iter(train_loader))
prediction(model,context,question,answer)
```

```
super bowl 50 was an american football game to determine the champion of the national football league ( nfl ) for th
e 2015 season . the american football conference ( afc ) champion denver broncos defeated the national football conf
erence ( nfc ) champion carolina panthers 24–10 to earn their third super bowl title . the game was played on februa
ry 7 , 2016 , at levi 's stadium in the san francisco bay area at santa clara , california . as this was the 50th su
per bowl , the league emphasized the " golden anniversary " with various gold - themed initiatives , as well as temp
orarily suspending the tradition of naming each super bowl game with roman numerals ( under which the game would hav
e been known as " super bowl l " ) , so that the logo could prominently feature the arabic numerals 50 .

which nfl team won super bowl 50 ?

denver broncos

['denver', 'national']
```

Output3

The above image depicts the results of our model, we have passed three different contexts, questions to our models and these were the results we have achieved.

As the squad data set is huge, to train the entire data set is taking nearly 5 hours, because arranging the training data into data frames has a complexity of O(n^3). This is the best optimistic logic we could achieve. Furthermore, we have used google colab to train our data to reduce the toll of time complexity on the processor. This improved a little bit to our relief but not much useful.

We have curtailed some of the training data and we have trained my model. We have done only 5 epochs, because each is taking an average of 76 minutes. Despite having loss epochs, we managed to achieve a stable system. If we could observe the results, for the context-question, our model did a great job by answering accurately. For the second question-content model, our model didn't do well for this input, which is bad. For the third question-context model, It answered partially.

Our model was giving 40% of accurate answers. 30% of partial answers, and 30% of inaccurate answers.

With all the Limited processors, if we could achieve a 70% of accuracy, with the best processors from AWS or google cloud, we could run more epochs and could increase the prediction of answers accurately.

CONCLUSION

Our initial goal for this research was to use Neural Networks to correctly anticipate replies from the SQUAD dataset, which included context and query, and we feel we succeeded. We modified our epochs to examine how they affected the results, as well as to save time and compute complexity, and discovered that raising the epochs improved the results. We can conclude that we learnt a lot about neural networks and BiDAF, and that the material we utilized was quite useful. We discovered that debugging neural models is incredibly challenging. This was a fantastic learning experience for us.

REFERENCES

- ***https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760988.pdf***
- ***https://towardsdatascience.com/the-definitive-guide-to-bi-directional-attention-flow-d0e96e9e666b***
- ***https://web.stanford.edu/class/cs224n/materials/CS224N_PyTorch_Tutorial.html***