CITS4012

NATURAL LANGUAGE PROCESSING ASSIGNMENT

Ronjon Kundu, Rahma A S and Jaya Motwani

University of Western Australia, Australia

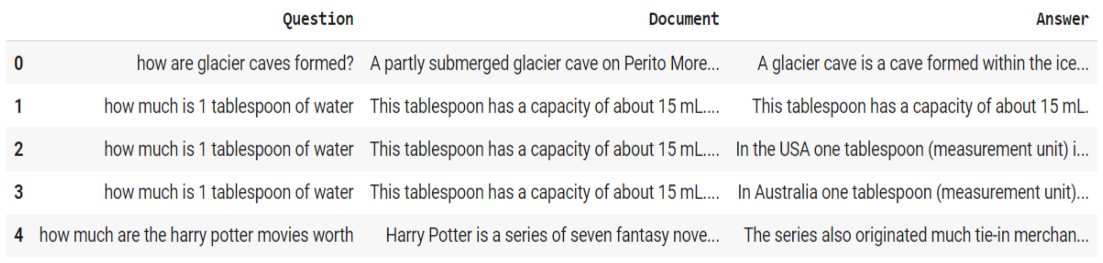
23215183@student.uwa.edu.au
23471149@student.uwa.edu.au
23990486@student.uwa.edu.au

1 Dataset

STEPS FOR DATA WRANGLING. In this implementation, Data Wrangling has been used to convert the raw data into a format that is convenient for the consumption of data for feeding in the model. In our method, we have followed the following steps for Data Wrangling on both training and testing data sets: -

1) **Extraction** of data from the CSV files.

2) **Decomposing the data into the following structure** by merging all the sentences in the same document having the same document ID and extracting the answer sentence with a sentence having label=1. Thereafter, resetting the index of the entire data frame:

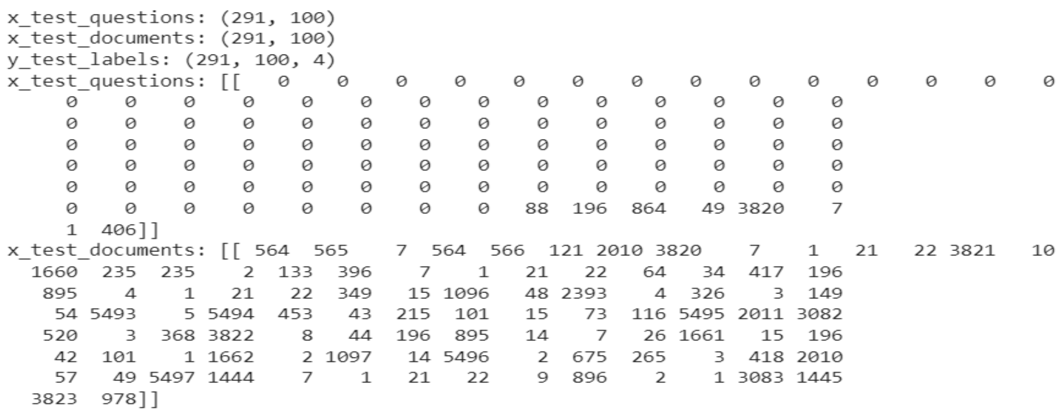


	Question	Document	Answer
0	how are glacier caves formed?	A partly submerged glacier cave on Perito More...	A glacier cave is a cave formed within the ice...
1	how much is 1 tablespoon of water	This tablespoon has a capacity of about 15 mL....	This tablespoon has a capacity of about 15 mL.
2	how much is 1 tablespoon of water	This tablespoon has a capacity of about 15 mL....	In the USA one tablespoon (measurement unit) i...
3	how much is 1 tablespoon of water	This tablespoon has a capacity of about 15 mL....	In Australia one tablespoon (measurement unit)...
4	how much are the harry potter movies worth	Harry Potter is a series of seven fantasy nove...	The series also originated much tie-in merchan...

3) **Tokenization** of the document and the answer using the NLTK library including padding. The reasons for using the NLTK package are as follows:

a) NLTK provides a wide range of tokenization methods that cater for basic word tokenization, sentence tokenization, and regular expression-based tokenization.
b) It includes modules for stemming, lemmatization, stop-word removal, part-of-speech tagging, and more. These preprocessing capabilities complement tokenization and enabled us to perform comprehensive text cleaning.
c) NLTK integrates well with other libraries and tools in the NLP ecosystem. It can be used in conjunction with libraries like spaCy, sci-kit-learn, and TensorFlow to build end-to-end NLP pipelines.

<u>Sample Tokenization of Test Data Set Questions and Documents (Post Padding)</u>



```
x_test_questions: (291, 100)
x_test_documents: (291, 100)
y_test_labels: (291, 100, 4)
x_test_questions: [[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0   88  196  864   49 3820    7
     1  406]]
x_test_documents: [[ 564  565    7  564  566  121 2010 3820    7    1   21   22 3821   10
  1660  235  235    2  133  396    7    1   21   22   64   34  417  196
   895    4    1   21   22  349   15 1096   48 2393    4  326    3  149
    54 5493    5 5494  453   43  215  101   15   73  116 5495 2011 3082
   520    3  368 3822    8   44  196  895   14    7   26 1661   15  196
    42  101    1 1662    2 1097   14 5496    2  675  265    3  418 2010
    57   49 5497 1444    7    1   21   22    9  896    2    1 3083 1445
  3823  978]]
```

4) **Labelling** the entire training data frame with four types of labels – ‘START’, ‘END’,’INNER’,’ANSWER’.

2 ***<u>Sequence QA model</u>***

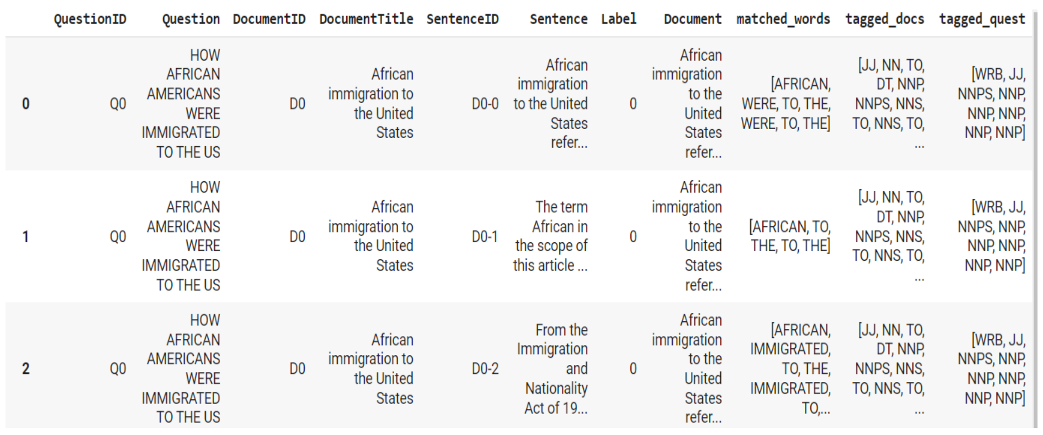
This section has been sub divided into 2 parts:

A. <u>Input Embedding.</u> We have generated word vector by using the pretrained word embedding model and extracting three different types of features.

1) **<u>Word Embedding</u>**. We have used ***glove-wiki-gigaword-100*** **pretrained model** for preparing embeddings. *The GloVe embeddings are trained on an English Wikipedia dump and English Gigaword 5th Edition dataset. Its dimensionality is 100 and has 6B tokens (uncased). The original source of the embeddings can be found here: https://nlp.stanford.edu/projects/glove/. We have imported this as an api through genism downloader. The vocab size of Glove is 400001, which is large enough for our data frame.*

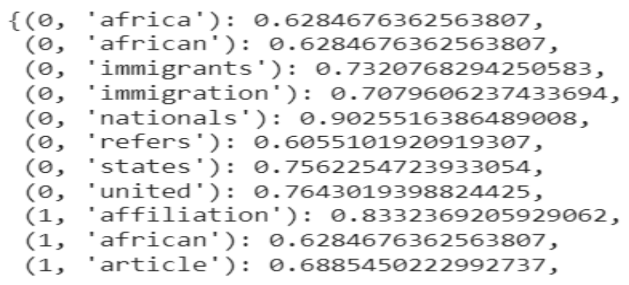
2) **<u>Feature Extraction</u>.** We have extracted three different types of features from the data set, which are as follows:

a) **<u>PoS Tags</u>**. We have pre-processed the data set by removing punctuations, tokenization and then used NLTK POS tagger.



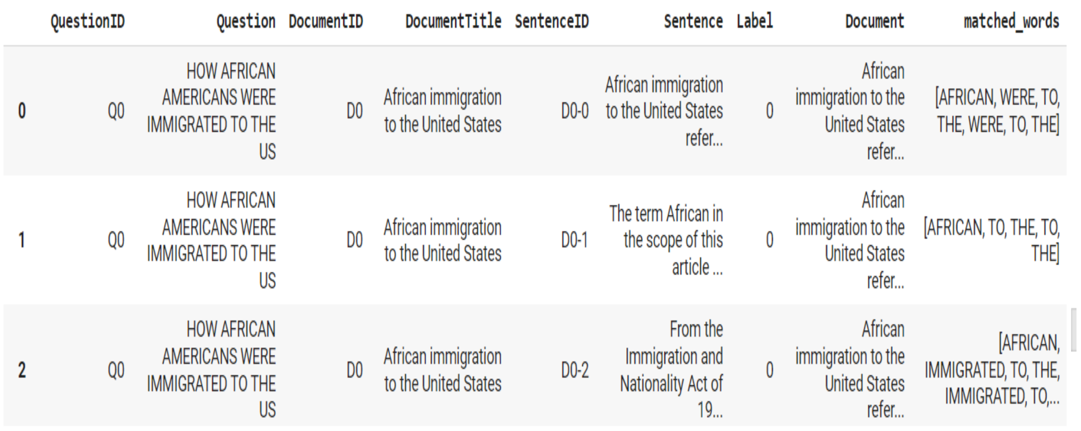
	QuestionID	Question	DocumentID	DocumentTitle	SentenceID	Sentence	Label	Document	matched_words	tagged_docs	tagged_quest
0	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-0	African immigration to the United States refer...	0	African immigration to the United States refer...	[AFRICAN, WERE, TO, THE, WERE, TO, THE]	[JJ, NN, TO, DT, NNP, NNPS, NNS, TO, NNS, TO, ...	[WRB, JJ, NNPS, NNP, NNP, NNP, NNP, NNP]
1	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-1	The term African in the scope of this article ...	0	African immigration to the United States refer...	[AFRICAN, TO, THE, TO, THE]	[JJ, NN, TO, DT, NNP, NNPS, NNS, TO, NNS, TO, ...	[WRB, JJ, NNPS, NNP, NNP, NNP, NNP, NNP]
2	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-2	From the Immigration and Nationality Act of 19...	0	African immigration to the United States refer...	[AFRICAN, IMMIGRATED, TO, THE, IMMIGRATED, TO,...	[JJ, NN, TO, DT, NNP, NNPS, NNS, TO, NNS, TO, ...	[WRB, JJ, NNPS, NNP, NNP, NNP, NNP, NNP]

b) **TF-IDF.** We have imported the NLTK module and downloaded the PUNKT package for calculating TF-IDF for our data frame. So, we have done tokenization, removal of punctations, lower casing of tokens and then removal of stop words for calculation of TF-IDF. A sample is shown below:



```
{(0, 'africa'): 0.6284676362563807,
 (0, 'african'): 0.6284676362563807,
 (0, 'immigrants'): 0.7320768294250583,
 (0, 'immigration'): 0.7079606237433694,
 (0, 'nationals'): 0.9025516386489008,
 (0, 'refers'): 0.6055101920919307,
 (0, 'states'): 0.7562254723933054,
 (0, 'united'): 0.7643019398824425,
 (1, 'affiliation'): 0.8332369205929062,
 (1, 'african'): 0.6284676362563807,
 (1, 'article'): 0.6885450222992737,
```

c) **Word Match Feature**. After decapitalization and lemmatization, we have used **Python Package Spacy** for extracting word matching feature.



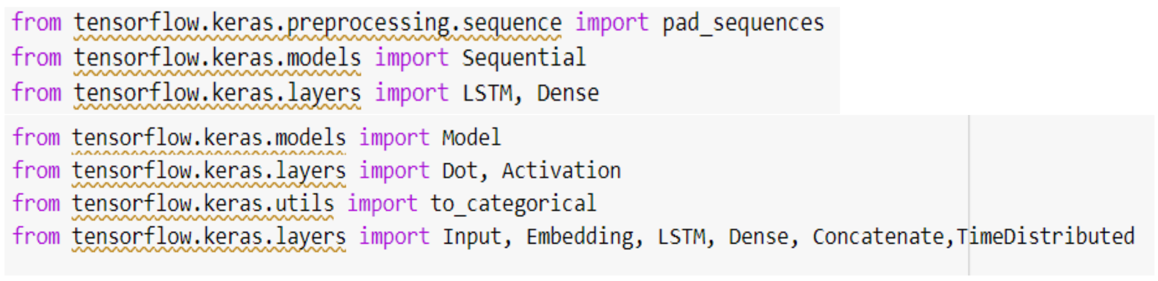
	QuestionID	Question	DocumentID	DocumentTitle	SentenceID	Sentence	Label	Document	matched_words
0	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-0	African immigration to the United States refer...	0	African immigration to the United States refer...	[AFRICAN, WERE, TO, THE, WERE, TO, THE]
1	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-1	The term African in the scope of this article ...	0	African immigration to the United States refer...	[AFRICAN, TO, THE, TO, THE]
2	Q0	HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US	D0	African immigration to the United States	D0-2	From the Immigration and Nationality Act of 19...	0	African immigration to the United States refer...	[AFRICAN, IMMIGRATED, TO, THE, IMMIGRATED, TO,...

B. **Implementation of Recurrent Model with Attention**. We have used **TensorFlow Framework and Keras API** for implementation of our Model.

1) **Justification for using TensorFlow Framework**:

a) It is relatively easy to learn and work with since it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes.
b) This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly. Keras also allows to switch between different back ends.
c) Since TensorFlow has adopted Keras as its official high-level API, Keras is embedded in TensorFlow and can be used to perform deep learning fast as it provides inbuilt modules for all neural network computations.
d) At the same time, computation involving tensors, computation graphs, sessions, etc can be custom made using the TensorFlow Core API, which gave us total flexibility and control over our application and helped us implement the model in a relatively short time.

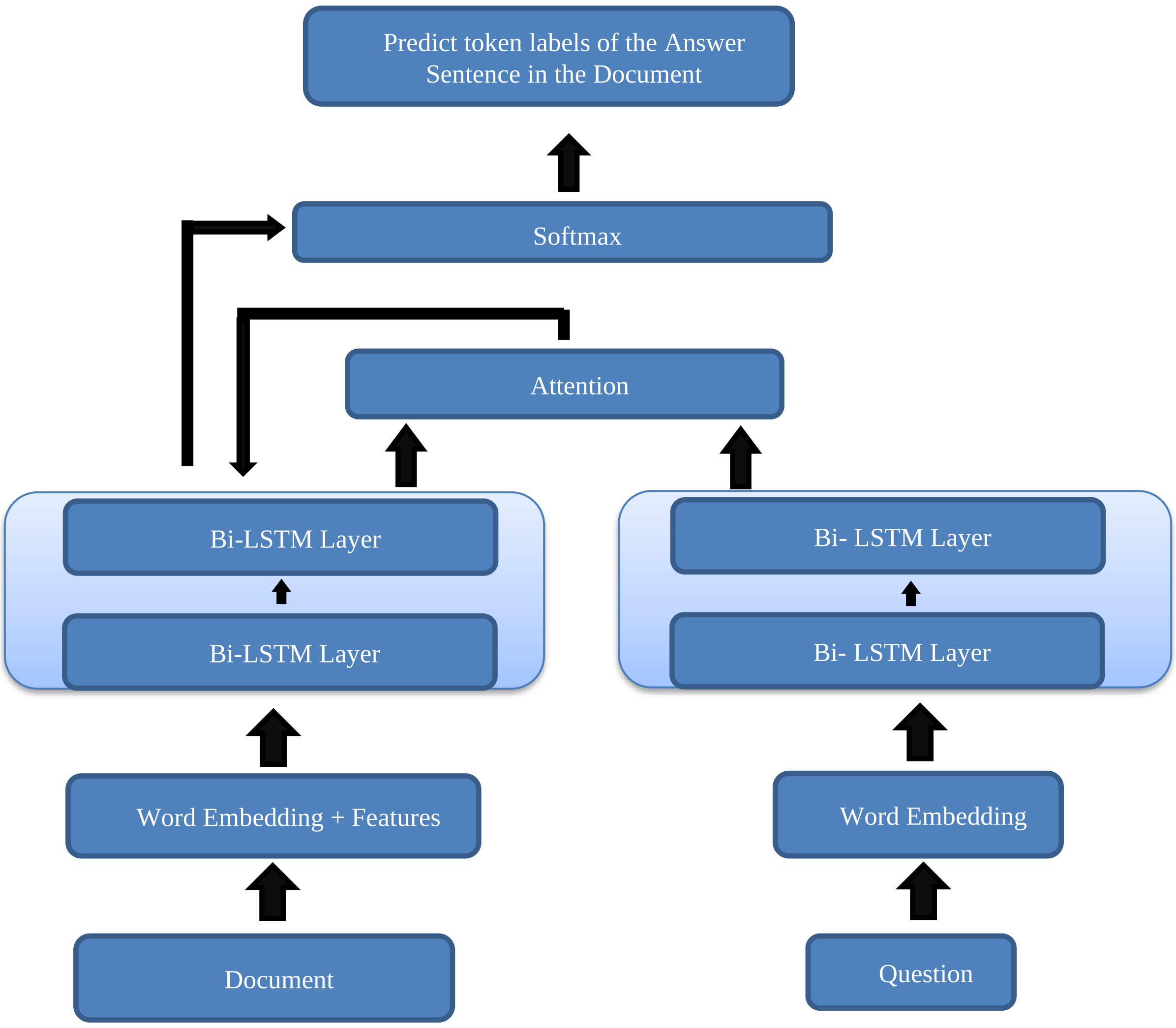
2) **Modules imported from TensorFlow.keras**:



```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dot, Activation
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Concatenate,TimeDistributed
```

3) **The Architecture of Model (Diagram)**



4) **Description of the Architecture** (**Justification for usage of Bi- LSTM Model)**.

a) We have compared LSTM neural network with RNN neural networks and found out that RNNs are particularly suited for tasks that involve sequences, for example, machine translation, where the sequences are sentences or words.

b) The main difference between an LSTM unit and a standard RNN unit is that the LSTM unit **is *more sophisticated***. More precisely, it is composed of the so-called **gates** that supposedly regulate better the flow of information through the unit.

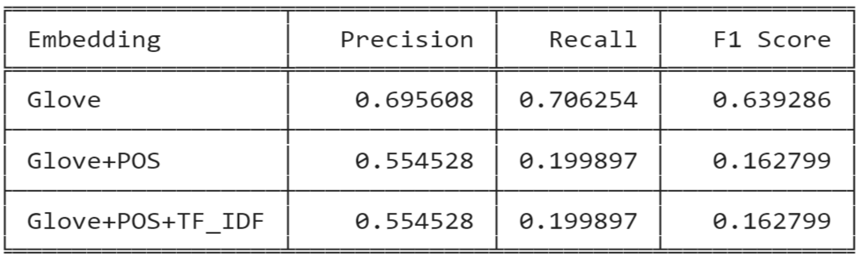
c) In practice, an LSTM has been used, as opposed to a vanilla (or standard) RNN, because it is **more computationally effective**. Also, the LSTM solves a problem that standard RNNs suffer from, i.e., the **vanishing gradient problem**.

d) LSTM has the ability to learn long-term dependencies and capture complex patterns in sequential data. The LSTM cell adds **long-term memory in an even more performant way** because it allows even more parameters to be learned. This makes it the most powerful as compared to RNN to do prediction, especially when we have a longer-term trend in our data set.

e) And we have used **Bi-LSTM** to leverage future context chunks to learn better representations of single words, just by reading a sentence **forward and backward to capture more information**.

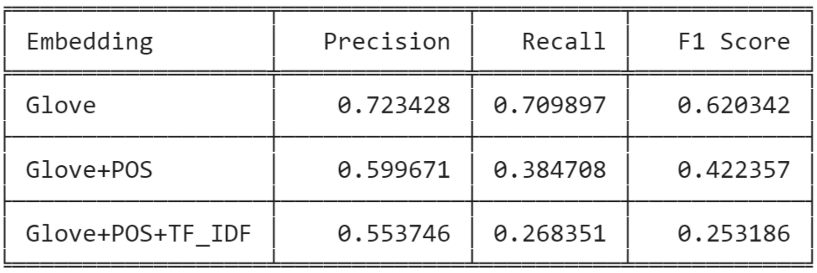
5) **Single vs Double layer Bi-LSTM.** We have tested the Model for Single Layer Bi-LSTM and dual layer Bi-LSTM design. The performance of the model for both combinations has been compared and **based on the improved performance of the model with Dual Layer Bi-LSTM Model, we have chosen number of layers to be two in the final implementation, even at the cost of slightly increased computational load**. The comparison is shown as below:

a) **Single Layer Bi-LSTM**



Embedding	Precision	Recall	F1 Score
Glove	0.695608	0.706254	0.639286
Glove+POS	0.554528	0.199897	0.162799
Glove+POS+TF_IDF	0.554528	0.199897	0.162799

b) **Dual Layer Bi-LSTM**



Embedding	Precision	Recall	F1 Score
Glove	0.723428	0.709897	0.620342
Glove+POS	0.599671	0.384708	0.422357
Glove+POS+TF_IDF	0.553746	0.268351	0.253186

6) **Position of Attention Layer**. The attention layer has been inserted between Output of LSTM Layer and SoftMax.

   a) This attention mechanism enhances the model's ability to understand and extract meaningful patterns from the input sequences with respect to the question sentence.
   b) By using the scaled dot product attention mechanism, the model can identify the relevance or importance of different parts of the question and document sequences.
   c) This allows the model to focus more on important words or phrases and down weight less relevant information, effectively capturing the key elements needed for accurate predictions.
   d) **Method used for Attention Calculation**
      i. attention = scaled_dot_product_attention (output_question_lstm, output_document_lstm).
      ii. Apply attention weights to LSTM output.
      iii. Concatenate attention-weighted representation with LSTM output.
      iv. Use this concatenated representation Output layer for token prediction.

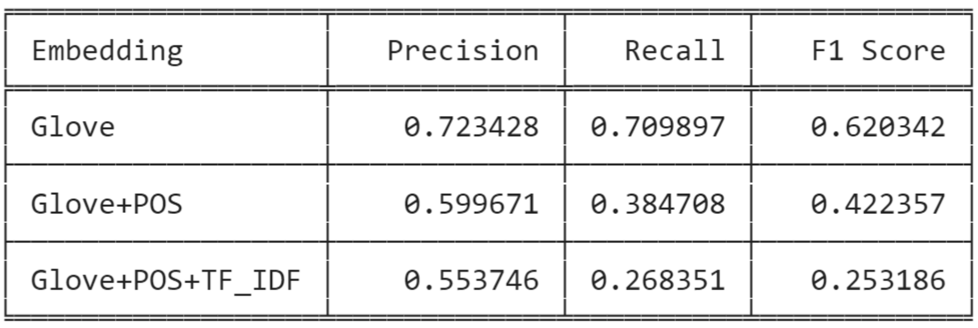
3 ***Model Testing*.**

The model has been tested with different permutations and combinations of various types of inputs, different variants of attention and different number of epochs while training the model. The performance of the model is described as follows:

3.1 **Input Embedding Ablation Study.**

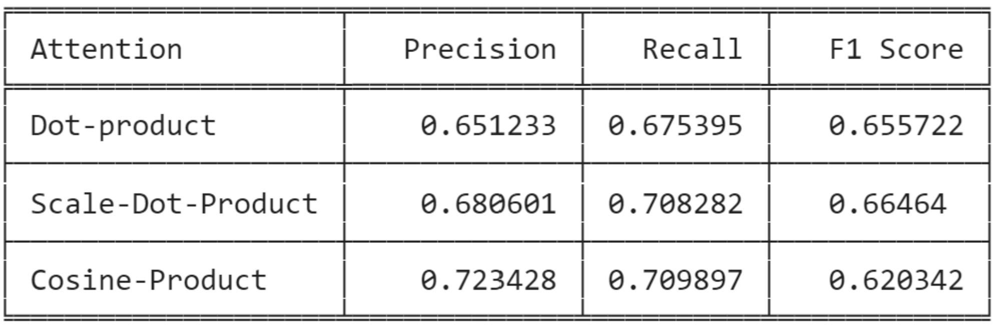
a) The performance of the model is highest in case of basic Word Embedding (Pretrained Glove Wiki Giga 100 Model). However, the performance of the model decreases while we combined Word Embedding with PoS Tagging and Word Embedding with both PoS tagging as well as TF-IDF.

b) The reason might be that the features are not relevant as per the model architecture design. Since we are predicting token labels, not generating the answer, so the PoS tags as well as TF-IDF seem irrelevant in this case. They might just be adding noise and hence decreasing the performance of the model.



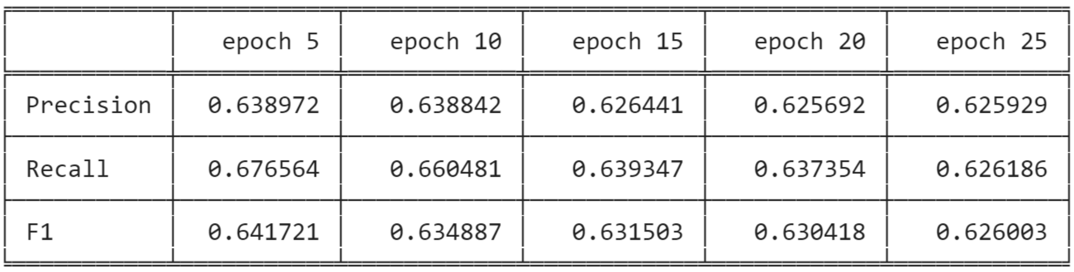
Embedding	Precision	Recall	F1 Score
Glove	0.723428	0.709897	0.620342
Glove+POS	0.599671	0.384708	0.422357
Glove+POS+TF_IDF	0.553746	0.268351	0.253186

3.2 Attention Ablation Study. The attention calculation has been done for three different types of variations and we got the highest Precision and recall in Cosine-Product and the highest F1 Score in Dot Product. The table of comparison is as follows:



Attention	Precision	Recall	F1 Score
Dot-product	0.651233	0.675395	0.655722
Scale-Dot-Product	0.680601	0.708282	0.66464
Cosine-Product	0.723428	0.709897	0.620342

3.3 **Hyper Parameter Testing.** Five different epochs have been performed on our model. From the provided table, it can be seen epoch 5 has the highest score. However, it seems that the model's precision, recall, and F1 score have relatively consistent values across the different epochs, especially with epochs greater than or equal to 15.



	epoch 5	epoch 10	epoch 15	epoch 20	epoch 25
Precision	0.638972	0.638842	0.626441	0.625692	0.625929
Recall	0.676564	0.660481	0.639347	0.637354	0.626186
F1	0.641721	0.634887	0.631503	0.630418	0.626003

References

1) **NLP CITS4012** - Lecture 2, Lecture 4, Lecture 6, Lecture 7

2) **NLP CITS4012** - Lab 2, Lab 4, Lab 6, Lab 8

3) **Author** : Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, **Article title** : “Attention Is All You Need”, **Journal** : Published in the journal "Advances in Neural Information Processing Systems" (NeurIPS) in 2017 **URL**: *http://arxiv.org/abs/1706.03762*

4) **Author** : Pengfei Li, Jungang Xu, Xinhua Lai, **Article title** : “LSTM Neural Network with Attention Mechanism for Stock Timing in Quantitative Trading Field”, **Journal** : 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), **URL**: http://ieeex-plore.ieee.org/document/7836709/

5) **Author** : Yujin Tang, Jianfeng Xu, Kazunori Matsumoto, Chihiro Ono, **Article title** : “Se-quence-to-Sequence Model with Attention for Time Series Classification”, **Journal** : 2022 International Conference on Intelligent Technologies (CONIT), **URL**: *https://ieeex-plore.ieee.org/document/9848232/*

6) **Blog post**: "Attention and Memory in Deep Learning and NLP", **Author** : Denny Britz, This blog post provides a comprehensive overview of attention mechanisms, including their motivation, types, and applications in deep learning and NLP tasks. It covers the attention mechanism in bi-LSTM models as well. URL : *http://www.wildml.com/2016/01/atten-tion-and-memory-in-deep-learning-and-nlp/*

7) **TensorFlow official website**: The TensorFlow website is a great resource for learning about various aspects of TensorFlow, including LSTM and bi-LSTM models. The website offers comprehensive documentation, tutorials, and code examples. **URL** : *https://www.tensorflow.org/guide/keras/sequential_model*

8) **TensorFlow Tutorials**: The TensorFlow website provides a dedicated section for tutorials, which includes tutorials on LSTM and bi-LSTM models. These tutorials provide step-by-step explanations and code examples to understand and implement bi-LSTM using Ten-sorFlow. **URL** : *https://www.tensorflow.org/tutorials*

9) **Machine Learning Mastery**: Machine Learning Mastery is a website by Jason Brownlee that offers practical tutorials and resources on machine learning and deep learning. The website provides tutorials on various topics, including LSTM and bi-LSTM models using TensorFlow. **URL**: *https://machinelearningmastery.com/*