FINAL PROJRCT

Group 2

Supervised by: DR. Arya Rahgozar

Problem Formulation:

whenever, question up to our mind we run to Google or any browser and it give us many links with details, so our QA system give you specific information about your order without many links and a lot of details We are making our own QA system that was built on squad data set.

The system will take the questions and the context from the users as inputs and analyze them, then gives the user the needed answer based on how relevant the questions are to the answer.

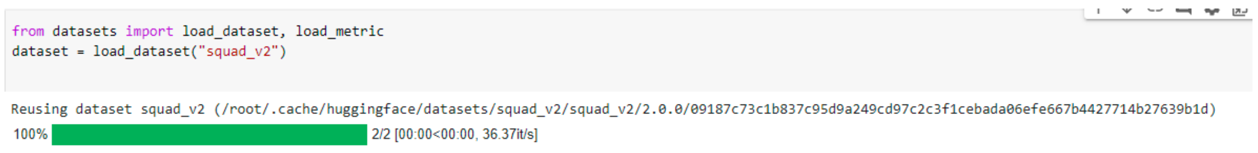
Our Innovation:

We have tried to do some clustering and classification techniques on the dataset and tried to make the models predict which title belongs to this context.So that we could gain more insights about the data .

We will present our work in parallel, what we have done in part I "clustering and classification using the pipeline", and part II "the QA system".

Data Preparation:

We have got the data from huggingface Datasets library



```
from datasets import load_dataset, load_metric
dataset = load_dataset("squad_v2")
```

```
Reusing dataset squad_v2 (/root/.cache/huggingface/datasets/squad_v2/squad_v2/2.0.0/09187c73c1b837c95d9a249cd97c2c3f1cebada06efe667b4427714b27639b1d)
100% 2/2 [00:00<00:00, 36.37it/s]
```

In Part I:

After loading the dataset we have only worked with a part of the dataset not the whole dataset because it needed a lot of computing power, and we had issue with the Google colab as it have been crashed a lot.

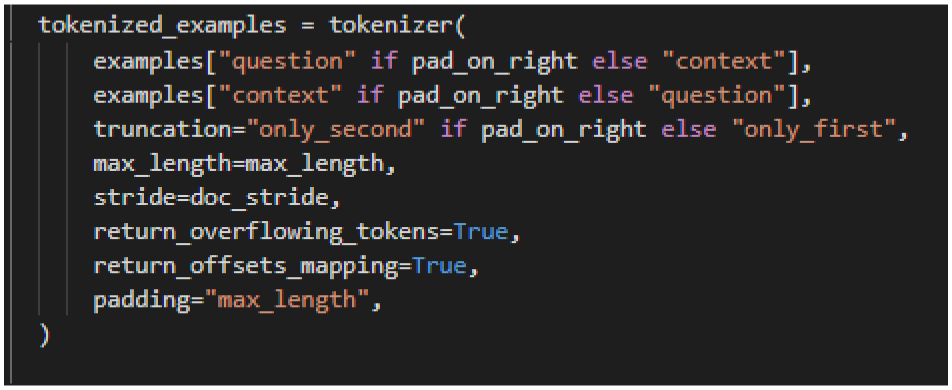
So we have cleaned dataset as we have dropped the extra useless columns for this part “ID”, “Questions”, “Answer” columns, removed the redundant contexts data, and we took the first 1,000 rows to do our clustering and classification on it.

Then we have separated our data into “X” being the context, and “Y” being the title, and we have 13 class.

In part II:

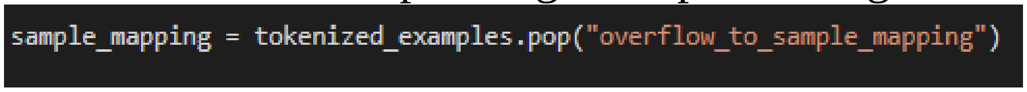
We have worked with training and validation datasets, and reduced the number of rows to 6,000 for training and 1,200 for validation.

In the data preparation part, we Tokenized our examples with truncation and padding, but keep the overflows using a stride.



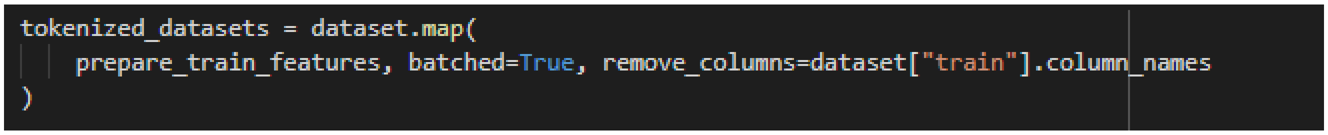
```
tokenized_examples = tokenizer(
    examples["question" if pad_on_right else "context"],
    examples["context" if pad_on_right else "question"],
    truncation="only_second" if pad_on_right else "only_first",
    max_length=max_length,
    stride=doc_stride,
    return_overflowing_tokens=True,
    return_offsets_mapping=True,
    padding="max_length",
)
```

This resulted in one example having the possibility of giving several features when a context is long, each of those features having a context that overlaps a bit the context of the previous feature. So to solve this problem we needed to map from the features to its corresponding example. Using :



```
sample_mapping = tokenized_examples.pop("overflow_to_sample_mapping")
```

Then the context will be labeled with a CLS token at the start and of the answer and if their was no possible answer the index of end of text will be the same as the CLS at the beginning. This processes will be applied on the training and validation data set using the map Function   :



```
tokenized_datasets = dataset.map(
    prepare_train_features, batched=True, remove_columns=dataset["train"].column_names
)
```

Classification:

We have separated the data into training and testing datasets, then we have applied 3 machine learning models with 3 different feature engineering models.

The first one was “multinomial naïve Bayes” with “count vectorizer, N-gram, and TF-ID” as feature engineering.

The second was “decision tree” with “BOW”.

And finally “KNN” with “BOW”.

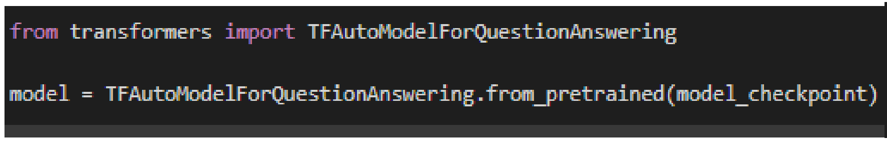
Clustering:

We have used the whole data which is 1,000 records, then we used the K-means as our algorithm and “BOW & TF-IDF” as feature engineering.

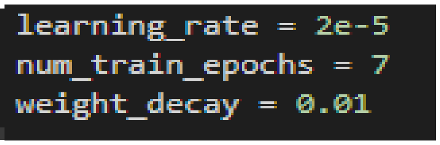
The EL-BOW method suggested that we use 11 clusters even though we have 13 class.

QA system model:

By using the **TFAutoModelForQuestionAnswering**, we have used the check point of **Bert-base-uncased,** and using **below weight**

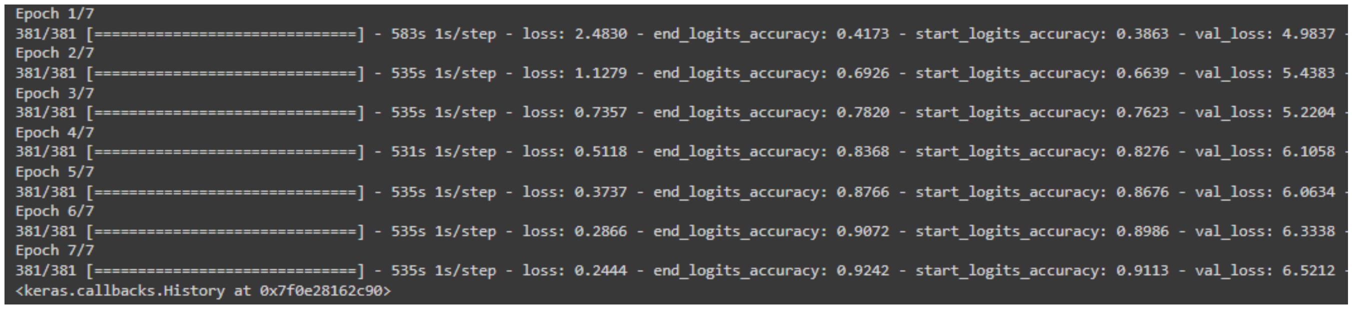


```
from transformers import TFAutoModelForQuestionAnswering

model = TFAutoModelForQuestionAnswering.from_pretrained(model_checkpoint)
```



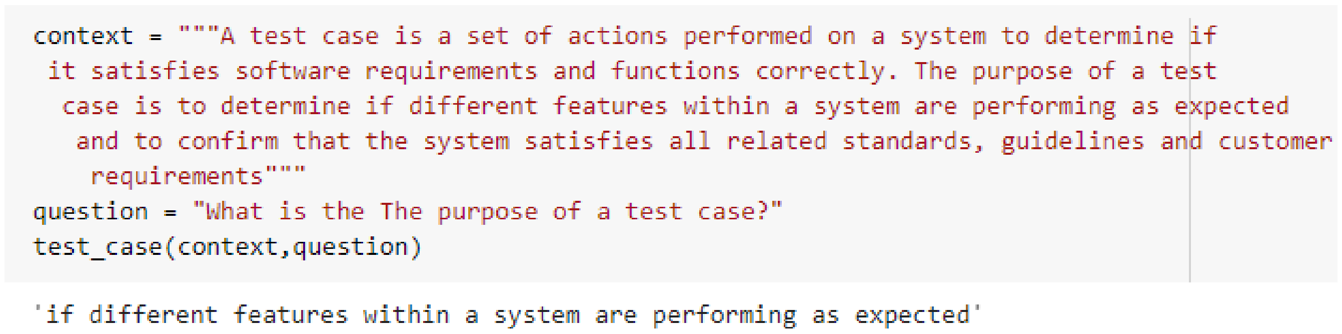
```
learning_rate = 2e-5
num_train_epochs = 7
weight_decay = 0.01
```

We built our model with 16 as patch size and it ran over 7 epoch.



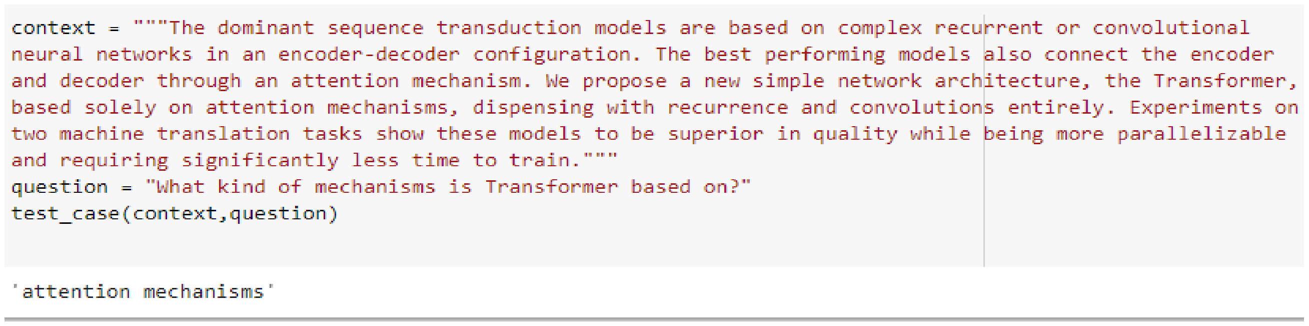
```
Epoch 1/7
381/381 [==============================] - 583s 1s/step - loss: 2.4830 - end_logits_accuracy: 0.4173 - start_logits_accuracy: 0.3863 - val_loss: 4.9837 -
Epoch 2/7
381/381 [==============================] - 535s 1s/step - loss: 1.1279 - end_logits_accuracy: 0.6926 - start_logits_accuracy: 0.6639 - val_loss: 5.4383 -
Epoch 3/7
381/381 [==============================] - 535s 1s/step - loss: 0.7357 - end_logits_accuracy: 0.7820 - start_logits_accuracy: 0.7623 - val_loss: 5.2204 -
Epoch 4/7
381/381 [==============================] - 531s 1s/step - loss: 0.5118 - end_logits_accuracy: 0.8368 - start_logits_accuracy: 0.8276 - val_loss: 6.1058 -
Epoch 5/7
381/381 [==============================] - 535s 1s/step - loss: 0.3737 - end_logits_accuracy: 0.8766 - start_logits_accuracy: 0.8676 - val_loss: 6.0634 -
Epoch 6/7
381/381 [==============================] - 535s 1s/step - loss: 0.2866 - end_logits_accuracy: 0.9072 - start_logits_accuracy: 0.8986 - val_loss: 6.3338 -
Epoch 7/7
381/381 [==============================] - 535s 1s/step - loss: 0.2444 - end_logits_accuracy: 0.9242 - start_logits_accuracy: 0.9113 - val_loss: 6.5212 -
<keras.callbacks.History at 0x7f0e28162c90>
```

These are the prediction of the model :



```
context = """A test case is a set of actions performed on a system to determine if
 it satisfies software requirements and functions correctly. The purpose of a test
  case is to determine if different features within a system are performing as expected
   and to confirm that the system satisfies all related standards, guidelines and customer
    requirements"""
question = "What is the The purpose of a test case?"
test_case(context,question)
```

```
'if different features within a system are performing as expected'
```



```
context = """The dominant sequence transduction models are based on complex recurrent or convolutional
neural networks in an encoder-decoder configuration. The best performing models also connect the encoder
and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,
based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on
two machine translation tasks show these models to be superior in quality while being more parallelizable
and requiring significantly less time to train."""
question = "What kind of mechanisms is Transformer based on?"
test_case(context,question)
```

```
'attention mechanisms'
```

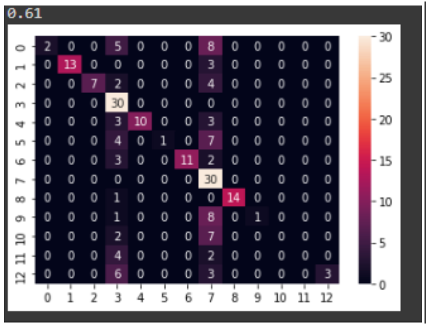
Evaluation of ML Results:

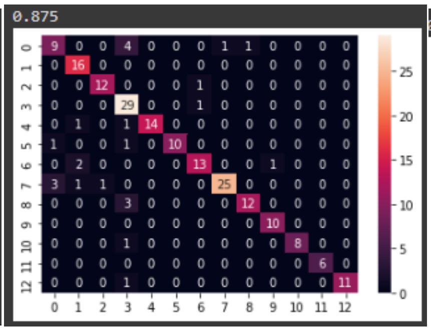
In part I:

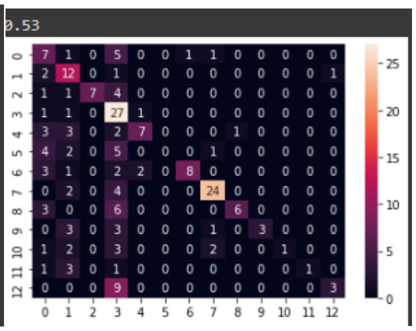
The classification part:

We have compared 3 models and we have found that the champion model was **Decision Tree** that have the biggest accuracy compared to the other 2 model, since its accuracy was 87% compared to Multinomial naïve Bayes which its accuracy was 61%, and the KNN was 53%.

MultinomialNB Decision Tree KNN

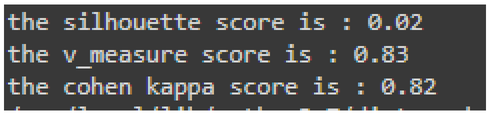






The clustering part:

The K-means had 0.02 silhouette score and 83% V-measure, and 82% Cohen Kappa score.

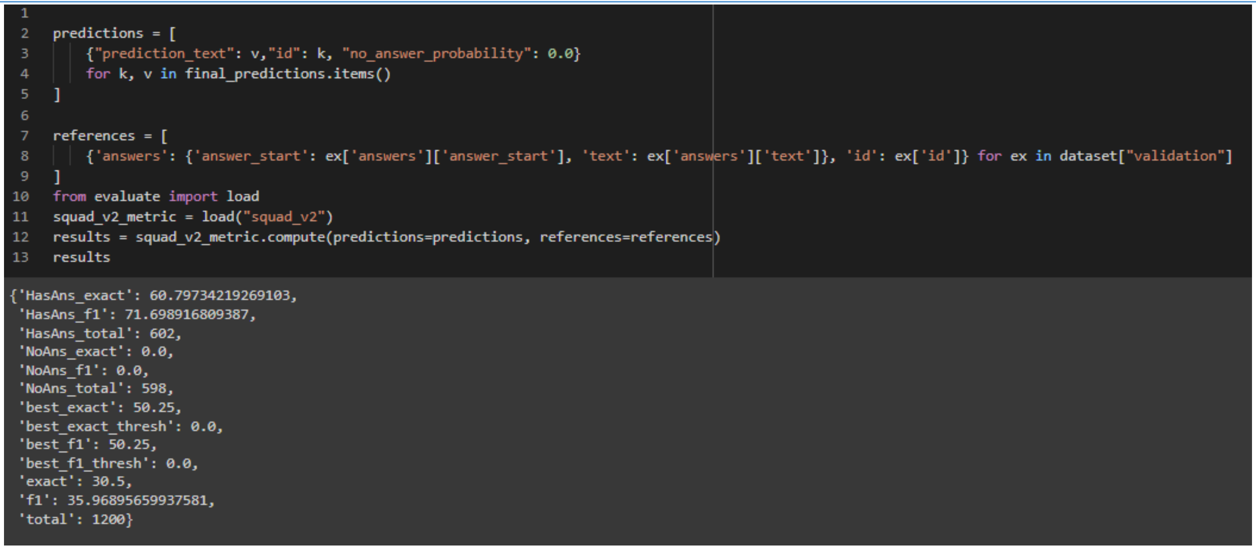


```
the silhouette score is : 0.02
the v_measure score is : 0.83
the cohen kappa score is : 0.82
```

In part II:

When we evaluated our model using huggingface built in matric that was made for our dataset SQAUD.2 .

It had F1-score 35.9%.



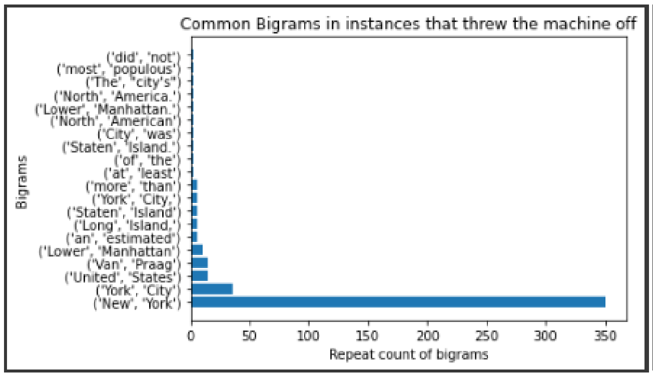
```python
predictions = [
    {"prediction_text": v,"id": k, "no_answer_probability": 0.0}
    for k, v in final_predictions.items()
]

references = [
    {'answers': {'answer_start': ex['answers']['answer_start'], 'text': ex['answers']['text']}, 'id': ex['id']} for ex in dataset["validation"]
]
from evaluate import load
squad_v2_metric = load("squad_v2")
results = squad_v2_metric.compute(predictions=predictions, references=references)
results
```

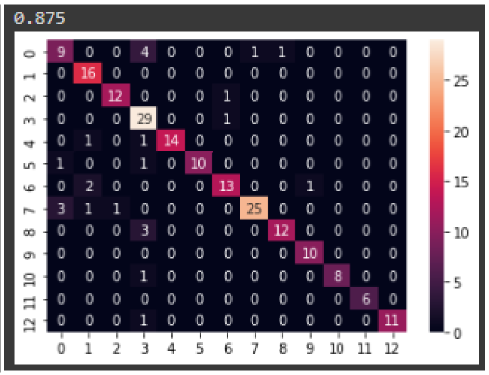
```
{'HasAns_exact': 60.79734219269103,
 'HasAns_f1': 71.698916809387,
 'HasAns_total': 602,
 'NoAns_exact': 0.0,
 'NoAns_f1': 0.0,
 'NoAns_total': 598,
 'best_exact': 50.25,
 'best_exact_thresh': 0.0,
 'best_f1': 50.25,
 'best_f1_thresh': 0.0,
 'exact': 30.5,
 'f1': 35.96895659937581,
 'total': 1200}
```

Error Analysis:

In part I:

From the classification and clustering we have found that the some words in the context caused confusion to the model as it have been referred to the wrong title.





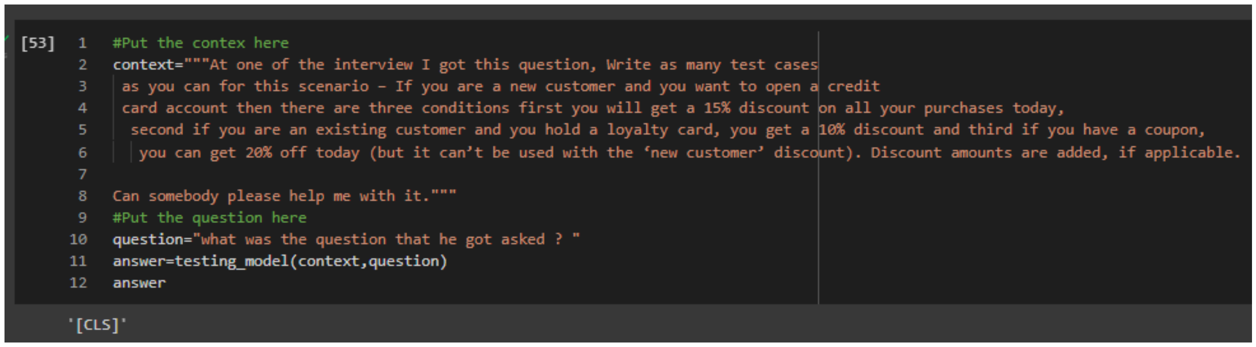
```
0.875
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	9	0	0	4	0	0	0	1	1	0	0	0	0
1	0	16	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	1	0	0	0	0	0	0
3	0	0	0	29	0	0	1	0	0	0	0	0	0
4	0	1	0	1	14	0	0	0	0	0	0	0	0
5	1	0	0	1	0	10	0	0	0	0	0	0	0
6	0	2	0	0	0	0	13	0	0	1	0	0	0
7	3	1	1	0	0	0	0	25	0	0	0	0	0
8	0	0	0	3	0	0	0	0	12	0	0	0	0
9	0	0	0	0	0	0	0	0	0	10	0	0	0
10	0	0	0	1	0	0	0	0	0	0	8	0	0
11	0	0	0	0	0	0	0	0	0	0	0	6	0
12	0	0	0	1	0	0	0	0	0	0	0	0	11

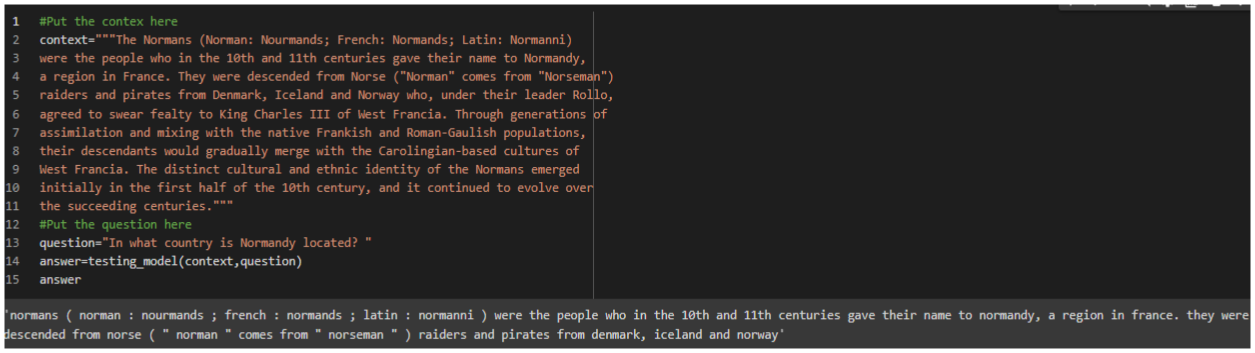
In part II:

We have noticed that when we asked the model a certain question coupled with context, it had a large variance, sometime it gives the exact answer that we are looking for and sometime it gives us irrelevant answer or totally wrong answer and that was expected from our side, because we have trained the model with only 6,000 records from the initial 130319 because as we have mentioned before that the Google colab notebook crashed a lot because of the lack of the computing power and the large data size

And these are some of the QA failed cases :



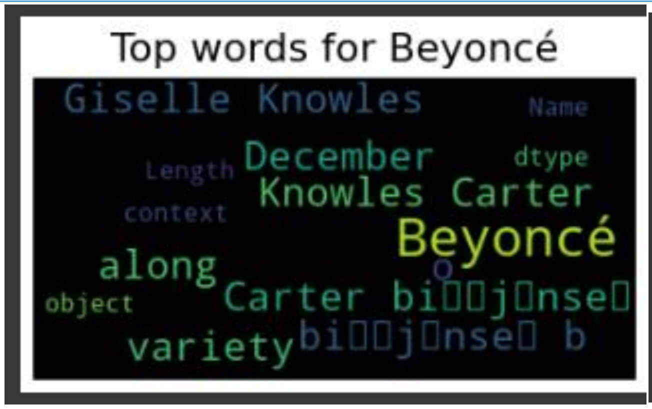
```
[53]
#Put the contex here
context="""At one of the interview I got this question, Write as many test cases
 as you can for this scenario – If you are a new customer and you want to open a credit
 card account then there are three conditions first you will get a 15% discount on all your purchases today,
  second if you are an existing customer and you hold a loyalty card, you get a 10% discount and third if you have a coupon,
   you can get 20% off today (but it can't be used with the 'new customer' discount). Discount amounts are added, if applicable.

Can somebody please help me with it."""
#Put the question here
question="what was the question that he got asked ? "
answer=testing_model(context,question)
answer

'[CLS]'
```

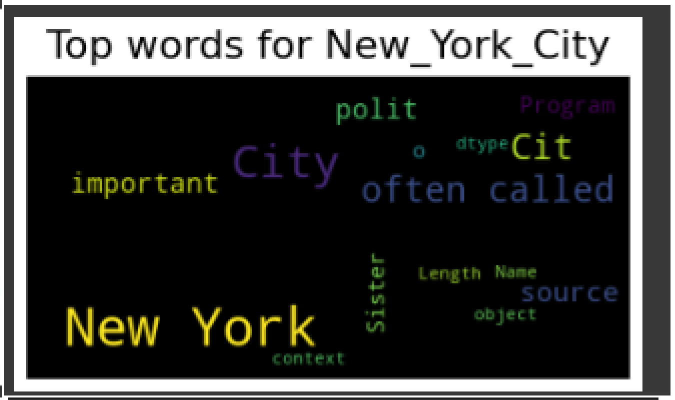


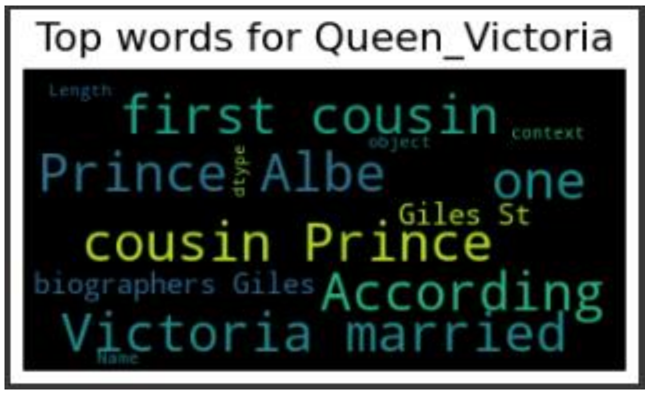
```
#Put the contex here
context="""The Normans (Norman: Nourmands; French: Normands; Latin: Normanni)
were the people who in the 10th and 11th centuries gave their name to Normandy,
a region in France. They were descended from Norse ("Norman" comes from "Norseman")
raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo,
agreed to swear fealty to King Charles III of West Francia. Through generations of
assimilation and mixing with the native Frankish and Roman-Gaulish populations,
their descendants would gradually merge with the Carolingian-based cultures of
West Francia. The distinct cultural and ethnic identity of the Normans emerged
initially in the first half of the 10th century, and it continued to evolve over
the succeeding centuries."""
#Put the question here
question="In what country is Normandy located? "
answer=testing_model(context,question)
answer

'normans ( norman : nourmands ; french : normands ; latin : normanni ) were the people who in the 10th and 11th centuries gave their name to normandy, a region in france. they were
descended from norse ( " norman " comes from " norseman " ) raiders and pirates from denmark, iceland and norway'
```

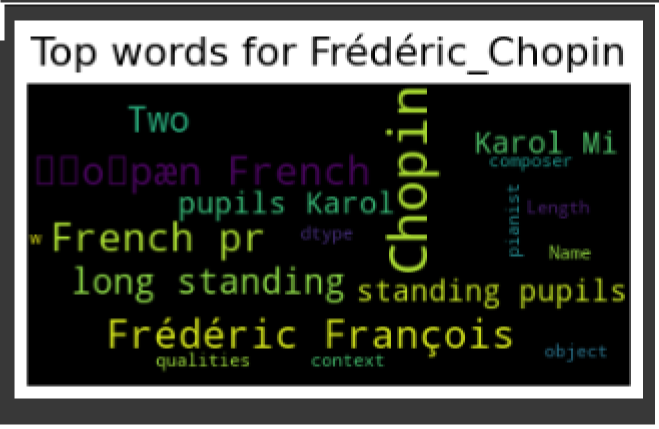
Visualization of results:

This is some of the EDA That was done on the Dataset

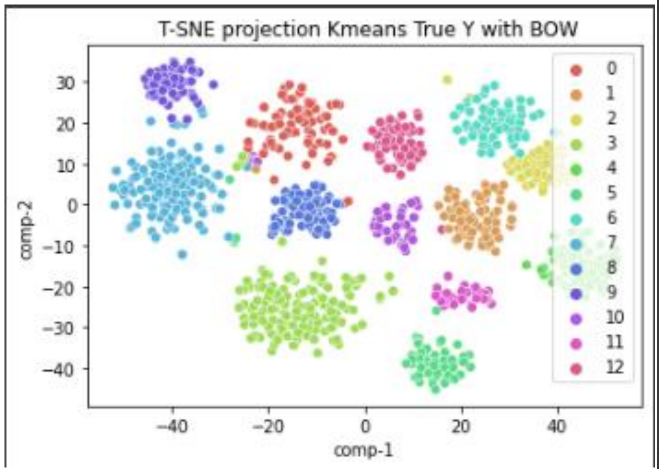


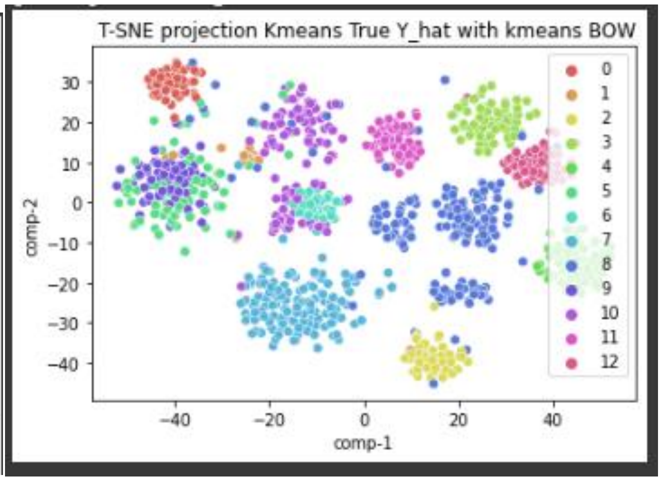






This the difference between the actual data and the predicted Cluster from the Kmeans





Future work

we will try to merge the predilection from the classification and QA System so it give us more accurate and related answer