uOttawa

ELG 5125:
Data Science Applications
Question Answering on SQuAD dataset

Prof. Arya Rahgozar

GROUP: 13

Table of Contents

Table of Figures

Abstract

In this project, three main approaches were applied. A question answering model was built using DistilBERT to answer the questions not by generating new text but by extracting substring from a paragraph using Stanford question answering dataset (SQuAD). Secondly, a clustering model was built using 4 pipelines with a gaussian mixture as its champion model. Lastly, a classification model was build based on the same data that clustering outputs with 2 pipelines resulting KNN and logistic regression as its champion models.

Introduction

Question answering is becoming essential as the amount of structured knowledge available on the web is growing steadily. Historically, building a system that can answer natural language questions in multiple contexts has been considered a very ambitious goal for many researchers. In this project, a partial sequential system was built to handle different tasks in this problem. It starts with the dataset which is Stanford question answering dataset (SQuAD), which is a reading comprehension dataset consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable [5]. Three main approaches will be showed later on with different or common techniques to achieve the highest possible goals in answering those questions using a transformer model based on the BERT architecture.

System Architecture

In this section, the model structure is discussed through this flow chart:

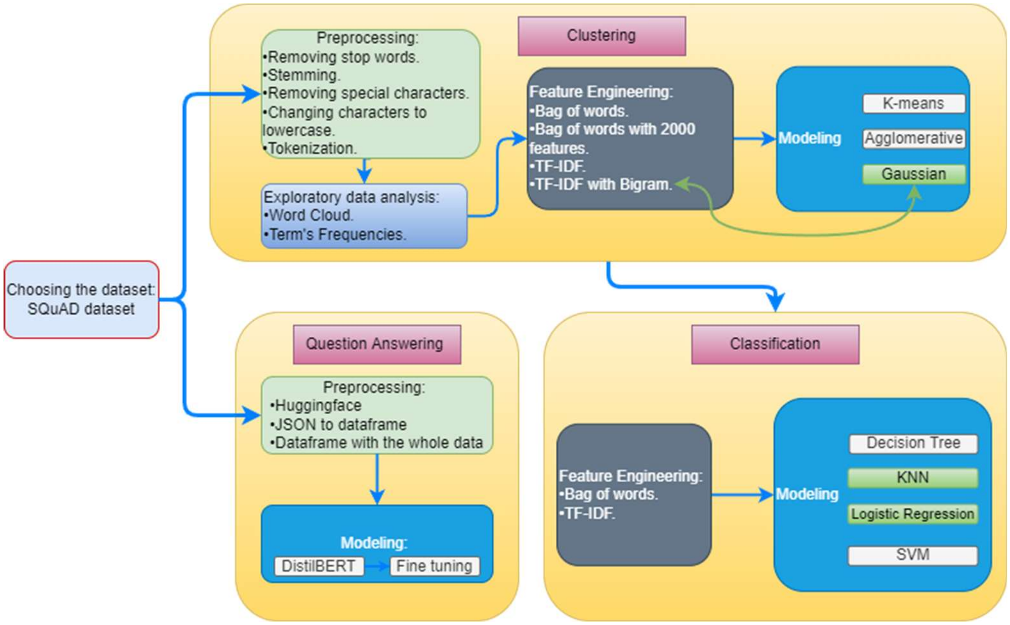
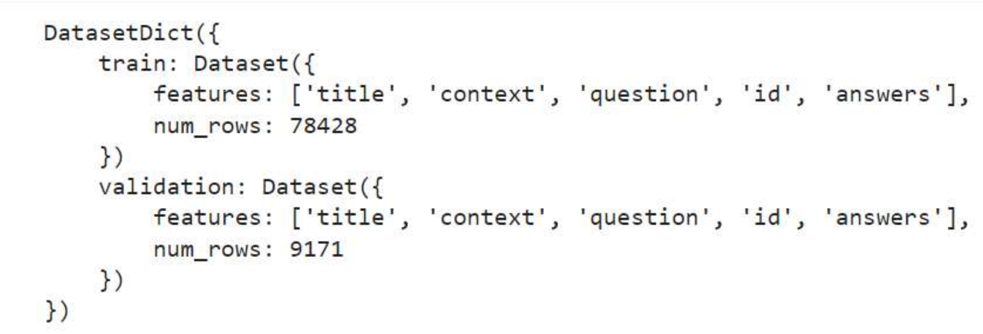


Figure 1 System Architecture

Methodology

Question Answering

First of all, we apply dataset split into training set and validation set with ratio 90% and 10% respectively. The following figure shows the output of dataset split:



```
DatasetDict({
    train: Dataset({
        features: ['title', 'context', 'question', 'id', 'answers'],
        num_rows: 78428
    })
    validation: Dataset({
        features: ['title', 'context', 'question', 'id', 'answers'],
        num_rows: 9171
    })
})
```

Figure 2 Dataset split

Preprocessing

Choosing the model

As previously mentioned, a transformer was used and has been pretrained on a generic task. Hence, in order to finetune it, it is important to faithfully repeat the preprocessing steps used during the pre-training phase. As such it's needed to define the model that it's going to be used straight from the preprocessing phase.

Since in this context it's required to answer the questions not by generating new text but by extracting substring from a paragraph, the ideal type of transformer to be used is the encoder kind.

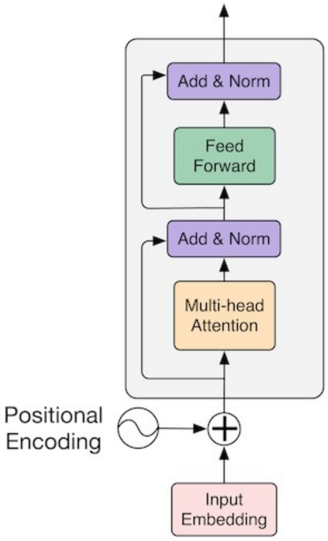


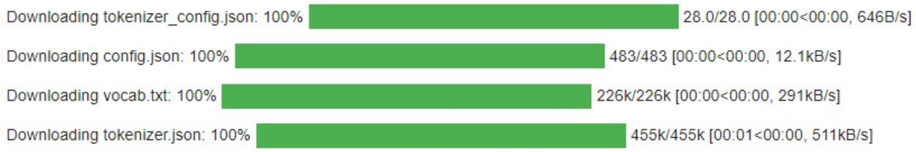
Figure 3 Typical structure of an encoder-based transformer.

```
model_checkpoint = "distilbert-base-uncased"
```

Loading the tokenizer

The preprocessing it's handled by HuggingFace's Tokenizer class. This class is able to handle the preprocessing of the dataset in conformity with the specification of each pre-trained model present in HuggingFace's model hub. In particular they hold the vocabulary built in the pre-training phase and the tokenization methodology used: it generally is word-based, character-based or sub word-based. DistilBERT uses the same as BERT, namely, end-to-end tokenization: punctuation splitting and word piece (sub word segmentation). The method Auto Tokenizer from pretrained will download the appropriate tokenizer.

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```



```
Downloading tokenizer_config.json: 100%  28.0/28.0 [00:00<00:00, 646B/s]
Downloading config.json: 100%  483/483 [00:00<00:00, 12.1kB/s]
Downloading vocab.txt: 100%  226k/226k [00:00<00:00, 291kB/s]
Downloading tokenizer.json: 100%  455k/455k [00:01<00:00, 511kB/s]
```

Figure 4 Loading the Tokenizer

Handling long sequences

The transformer models have a maximum number of tokens they are able to process with this quantity varying depending on the architecture.

A solution usually adopted in case of sequences longer than the limited amount (other than choosing a model that can handle longer sequences) is to truncate the sentence. While this approach may be effective for some tasks in this case it's not a valid solution since there would be the risk of truncating out from the context the answer to the question.

In order to overcome this limitation, what was done was sliding the input sentence over the model with a certain stride allowing a certain degree of overlap. The overlap is necessary as to avoid the truncation of a sentence in a point where an answer lies.

HuggingFace's tokenizer allow to perform this kind of operation by passing to the tokenizer the argument `return_overflowing_tokens=True` and by specifying the stride through the argument `stride`.

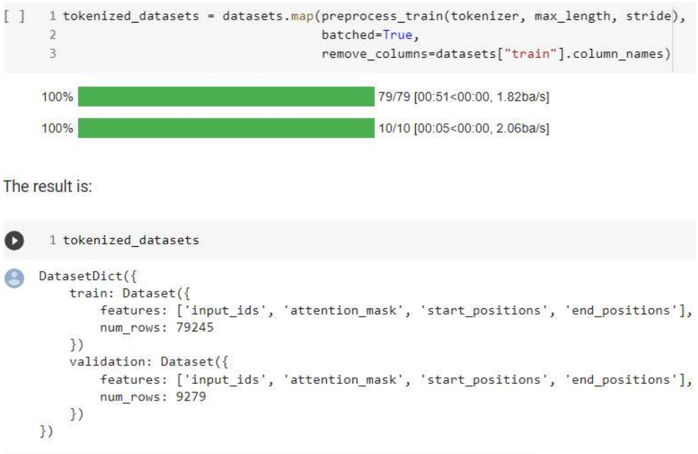
The division of a context in numerous truncated context create some issues regarding the detection of the answer inside the context since a pair of question-context may generate multiple pairs question-truncated context. This implies that using `answers["answer_start"]` is not sufficient anymore. As such, an ulterior preprocessing steps needs to be integrated in the preprocessing pipeline: the detection of the answers in the truncated contexts. The first step is to retrieve the answer position in the original context.

Since the tokenized input sequence encodes both the question and the context it is necessary to indentify which part of the sequence match the context. In order to complete this task the method `sequence_ids()` come into aid. In particular `sequence_ids()` tags the input tokens as 0 if they belong to the quesiton and 1 if they belong to the context (the reverse is instead true in the case the model pad the sequence to the left); `None` is for special tokens.

In order to properly tag the position of an answer in a truncated context the answer itself needs to be fully included inside the truncated context, since partial answers may not be fully explicative, nor have grammatical consistence, etc. Having the start and end answer's indexes inside the original context and the position of the truncated context inside the tokenized input sequence (which is composed by the question and the context), what's left it to identify the position of the answer in the tokenized and truncated context. This is done through the aid of the tokenized sequence attribute `offset_mapping` (obtained using the argument `return_offsets_mapping=True` to call the tokenizer) which indicates for each tokenized word its starting and ending index in the original sequence.

Calling the preprocessing method

The `map` method of the DatasetDict apply a given function to each row of the dataset (to each dataset's split).



```
[ ]  tokenized_datasets = datasets.map(preprocess_train(tokenizer, max_length, stride),
                                       batched=True,
                                       remove_columns=datasets["train"].column_names)
```

```
100% 79/79 [00:51<00:00, 1.82ba/s]
100% 10/10 [00:05<00:00, 2.06ba/s]
```

The result is:

```
tokenized_datasets
```

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'attention_mask', 'start_positions', 'end_positions'],
        num_rows: 79245
    })
    validation: Dataset({
        features: ['input_ids', 'attention_mask', 'start_positions', 'end_positions'],
        num_rows: 9279
    })
})
```

Figure 5 Tokenized datasets

Training

As previously mentioned, a pretrained model was used and then finetuned on the task at hand. In particular DistilBERT, just like BERT, is trained to be used mainly on masked language modeling and next sentence prediction tasks. Since the model has already been defined during the preprocessing phase, it's now possible to direcly download it for HuggingFace Model Hub using the `from_pretrained` method. `AutoModel` is the class that instantiate the correct architecture based on the model downloaded from the hub. `AutoModelForQuestionAnswering` in addition attaches to the pretrained backbone the head needed to perform this kind of task (which is not pretrained).

Trainer Class Definition

The pretraining of the model will be handled by the class `Trainer`. Still, some things need to be defined before being able to use the Trainer class. The first thing is the `TrainingArguments` which specify the saving folder, batch's size, learning rate, etc.

```
batch_size = 16
args = TrainingArguments(
    "squad",
    evaluation_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    weight_decay=0.01
)
```

The second and last thing to define is the data collator, which is used to batch together sequences having different length.

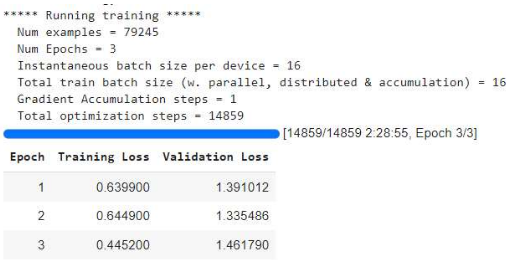
```
from transformers import default_data_collator
data_collator = default_data_collator
```

Now it's finally possible to define the Trainer class.

```
trainer = Trainer(
    model,
    args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    tokenizer=tokenizer,
)
```

Fine Tuning

The method `train` of the `Trainer` class is used to trigger the finetuning process.



```
***** Running training *****
  Num examples = 79245
  Num Epochs = 3
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 14859
                                        [14859/14859 2:28:55, Epoch 3/3]
 Epoch  Training Loss  Validation Loss
     1       0.639900         1.391012
     2       0.644900         1.335486
     3       0.445200         1.461790
```

Figure 6 Fine Tuning of QA

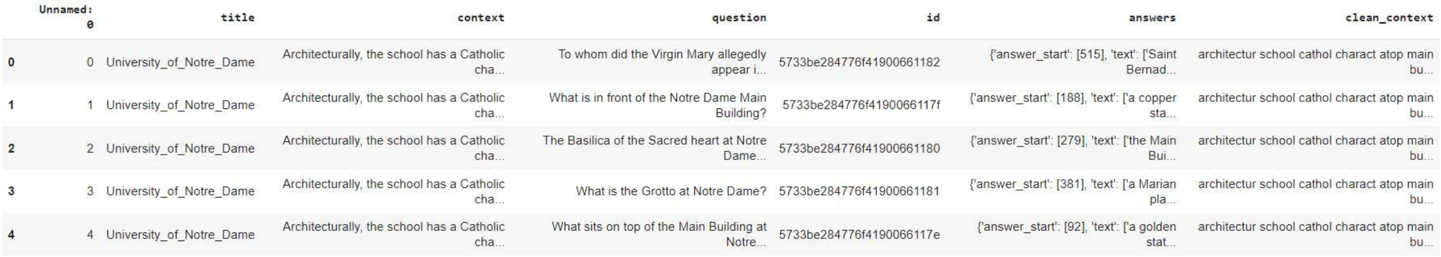
Clustering

Preprocessing (Text Cleaning)

The following techniques were applied:

- Removing stop words
- Stemming
- Removing special characters
- Changing characters to lowercase
- Tokenization

And the following figure shows a sample of the cleaned data:



	Unnamed: 0	title	context	question	id	answers	clean_context
0	0	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	To whom did the Virgin Mary allegedly appear i...	5733be284776f41900661182	{'answer_start': [515], 'text': ['Saint Bernad...	architectur school cathol charact atop main bu...
1	1	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	What is in front of the Notre Dame Main Building?	5733be284776f4190066117f	{'answer_start': [188], 'text': ['a copper sta...	architectur school cathol charact atop main bu...
2	2	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	The Basilica of the Sacred heart at Notre Dame...	5733be284776f41900661180	{'answer_start': [279], 'text': ['the Main Bui...	architectur school cathol charact atop main bu...
3	3	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	What is the Grotto at Notre Dame?	5733be284776f41900661181	{'answer_start': [381], 'text': ['a Marian pla...	architectur school cathol charact atop main bu...
4	4	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	What sits on top of the Main Building at Notre...	5733be284776f4190066117e	{'answer_start': [92], 'text': ['a golden stat...	architectur school cathol charact atop main bu...

Figure 7 Cleaned data

Exploratory Data Analysis

In this part, two EDA methods were used which are:

1-Terms' frequencies:

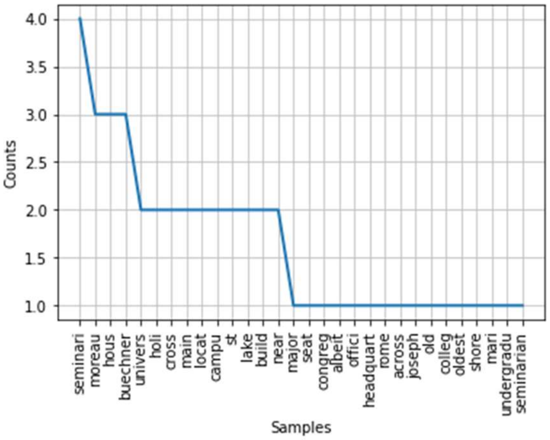


Figure 8 Term's Frequencies

2-Word cloud:

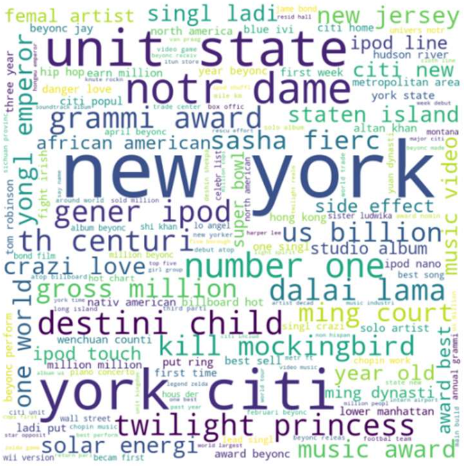


Figure 9 Word Cloud

Training

In feature engineering, four methods were used which are:

- Bag of words
- TF-IDF
- Bag of words with 2000 features
- TF-IDF with Bi-gram

Three algorithms were used with each feature engineering technique which are K-means clustering, agglomerative clustering and gaussian mixture

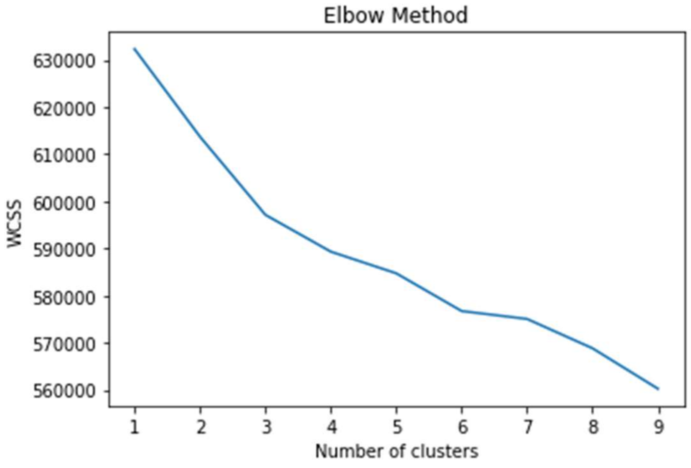


Figure 10 WCSS (BOW)

Bag of words (BOW)

First of all, bag of words was applied on the dataset then sum of squared distance (WCSS) was calculated using K-means which resulted figure 10. Silhouette score, T-SNE projection and dendrogram for were calculated for <u>K-means</u> as follows:

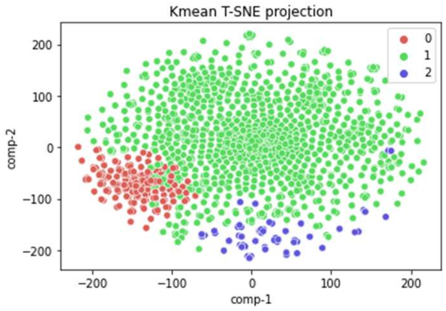


Figure 12 K-means (BOW) T-SNE

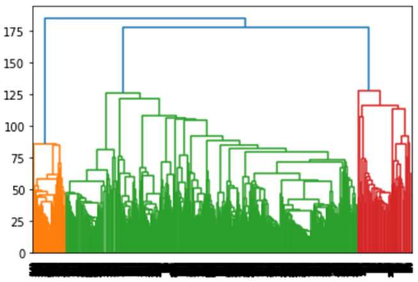


Figure 11 K-means (BOW) dendrogram

Silhouette Score:
0.1053076123324918

For agglomerative clustering the following plot shows its T-SNE projection and silhouette score:

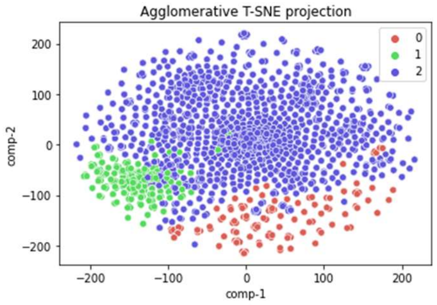


Figure 13 Agglomerative (BOW) T-SNE

Silhouette Score:
0.08737506799310384

For gaussian mixture the following plot shows its T-SNE projection and silhouette score:

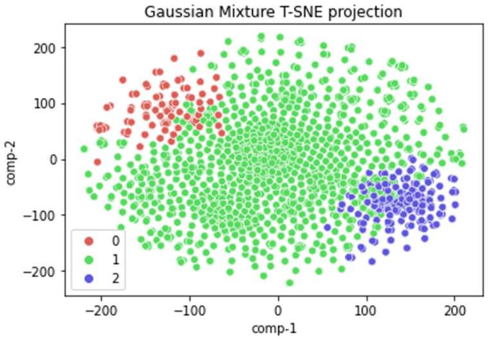


Figure 14 Gaussian (BOW) T-SNE

Silhouette Score: -
0.0025406249613993672

TF-IDF

Just like in BOW, TF-IDF was applied on the dataset then sum of squared distance (WCSS) was calculated using K-means which resulted figure 15. Silhouette score, T-SNE projection and dendrogr am for were calculated for K-means as follows:

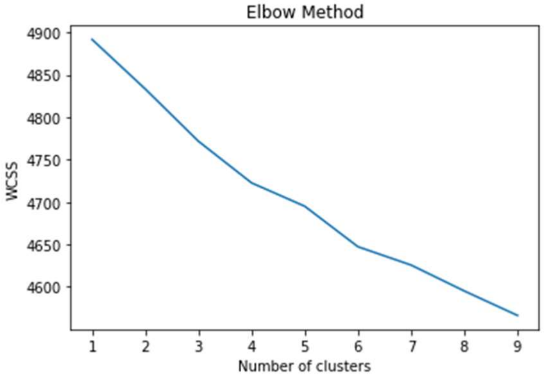


Figure 15 WCSS (TF-IDF)

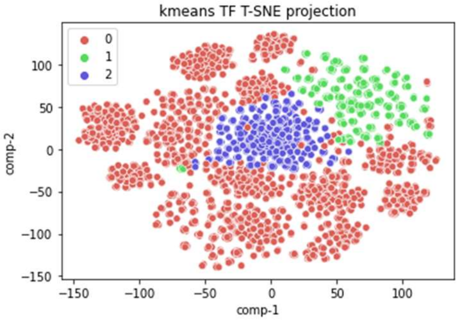


Figure 17 K-means (TF-IDF) T-SNE

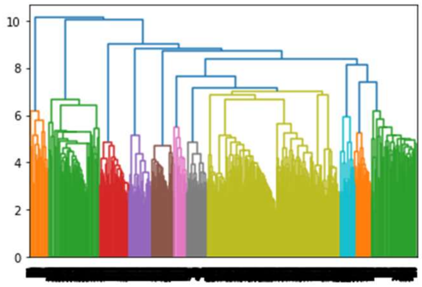


Figure 16 K-means (TF-IDF) dendrogram

Silhouette Score:
0.015994630853594243

For agglomerative clustering the following plot shows its T-SNE projection and silhouette score:

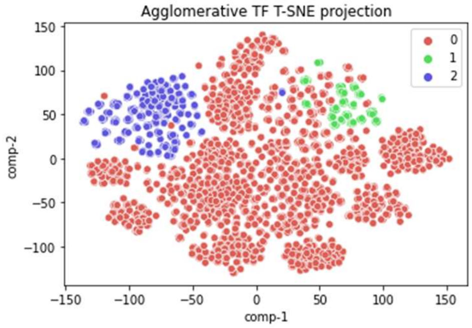
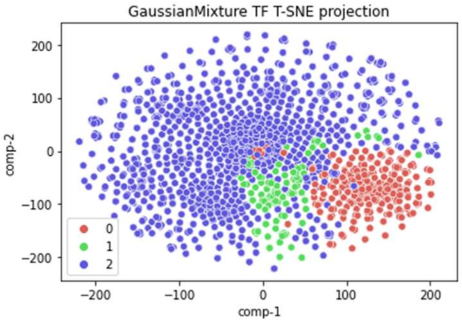


Figure 18 Agglomerative (TF-IDF) T-SNE

Silhouette Score:
0.010759753564862572

For gaussian mixture the following plot shows its T-SNE projection and silhouette score:



Silhouette Score:
0.013475419112979916

Figure 19 Gaussian (TF-IDF) T-SNE

Bag of words (BOW) with 2000 features

By applying the same steps of BOW but with 2000 features, the sum of squared distance (WCSS) was calculated using K-means which resulted figure 20. Silhouette score, T-SNE projection and dendrogram for were calculated for K-means as follows:

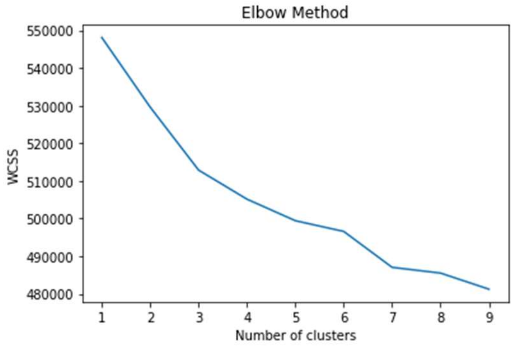


Figure 20 WCSS (BOW 2000)

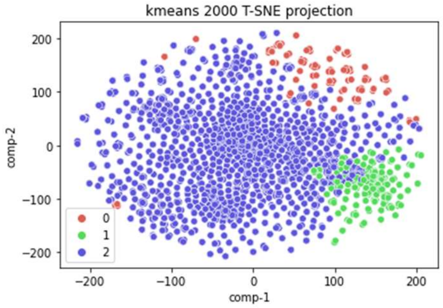


Figure 22 K-means (BOW 2000) T-SNE

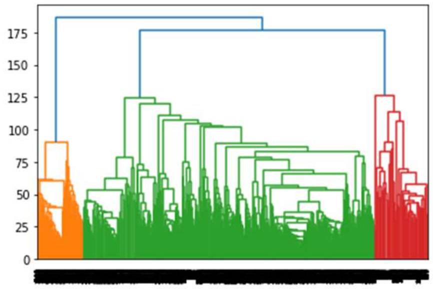
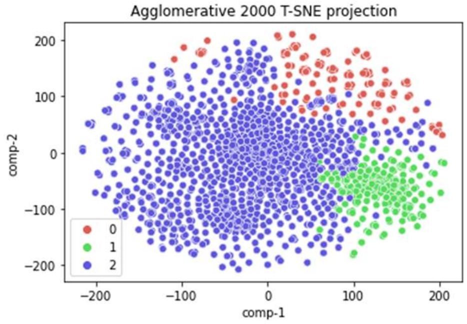


Figure 21 K-means (BOW 2000) dendrogram

Silhouette Score:
0.12911411050392638

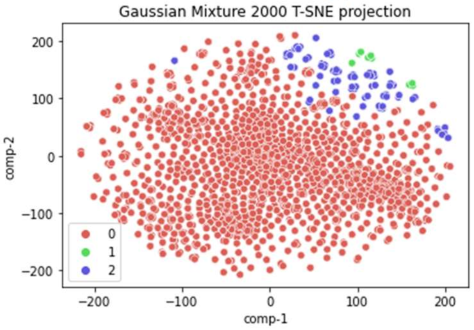
For agglomerative clustering the following plot shows its T-SNE projection and silhouette score:



Silhouette Score:
0.0771750999898270

Figure 23 Agglomerative (BOW 2000) T-SNE

For gaussian mixture the following plot shows its T-SNE projection and silhouette score:



Silhouette Score:
0.16078049439336548

Figure 24 Gaussian (BOW 2000) T-SNE

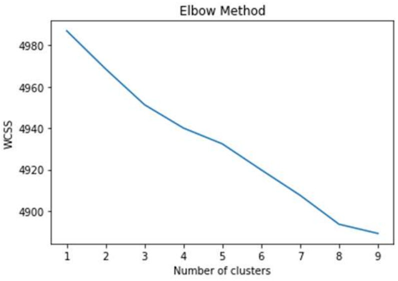


Figure 25 WCSS (TF-IDF 2000)

TF-IDF with Bi-gram

By applying the same steps of TF-IDF but with bi-grams , the sum of squared distance (WCSS) was calculated using K-means which resulted figure 25. Silhouette score, T-SNE projection and dendrogram for were calculated for K-means as follows:

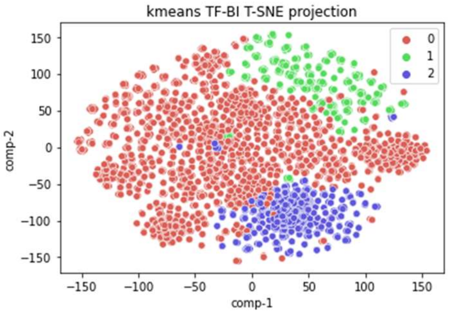


Figure 26 K-Means (TF-IDF Bi) T-SNE

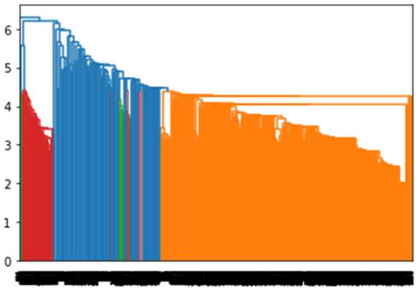


Figure 27 K-Means (TF-IDF Bi) dendrogram

Silhouette Score:

0.00635291178747055

For agglomerative clustering the following plot shows its T-SNE projection and silhouette score:

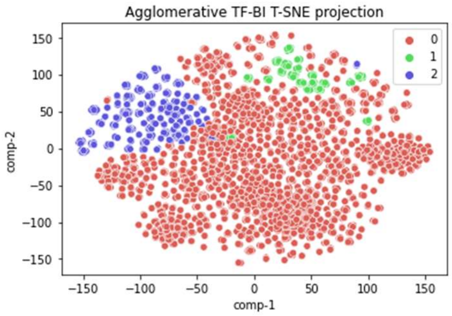


Figure 28 Agglomerative (TF-IDF Bi) T-SNE

Silhouette Score:

0.006400279131565316

For gaussian mixture the following plot shows its T-SNE projection and silhouette score:

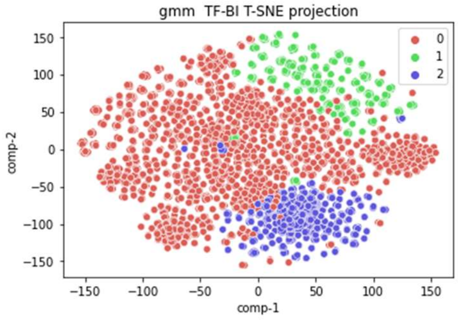


Figure 29 Gaussian (TF-IDF Bi) T-SNE

Silhouette Score:

0.12911411050392638

Classification

Preprocessing

The preprocessing in classification is completely dependent on that of clustering so no additional processes or approaches were done. There are "context" and" clean_context" columns, obtained clean_context column after we cleaned context column in clustering stage.

Training

In feature engineering, four methods were used which are:

- Bag of words
- TF-IDF

Four algorithms were used with each feature engineering technique which are decision tree, K-nearest neighbor, logistic regression and support vector machine (SVM). Confusion matrix and word cloud were plotted in each, also, kappa scores were calculated

Bag of words

Decision Tree

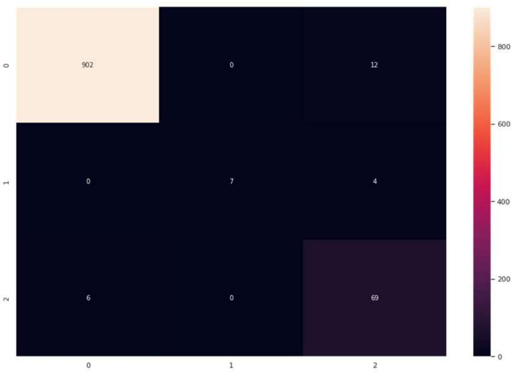


Figure 30 DT BOW CM

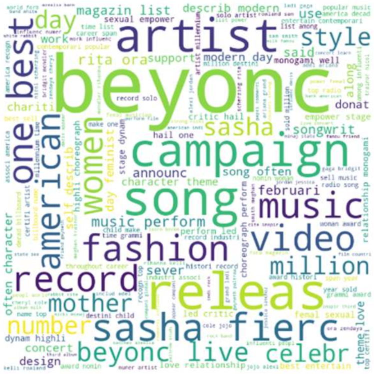


Figure 31 DT BOW Word cloud

Kappa Score:
0.8655552567894595

KNN

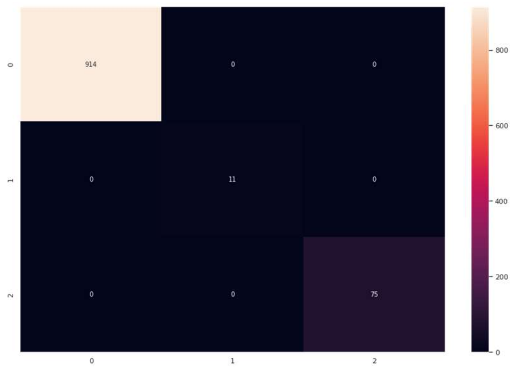


Figure 32 KNN BOW CM

We cannot plot word cloud for because there's no error, Kappa score = 1.0

Kappa Score:
1.0

Logistic regression

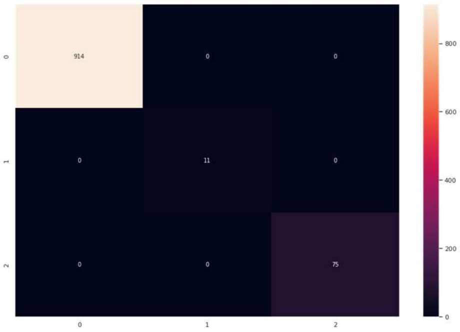


Figure 33 LR BOW CM

We cannot plot word cloud for because there’s no error, Kappa score = 1.0

Kappa Score:
1.0

SVM

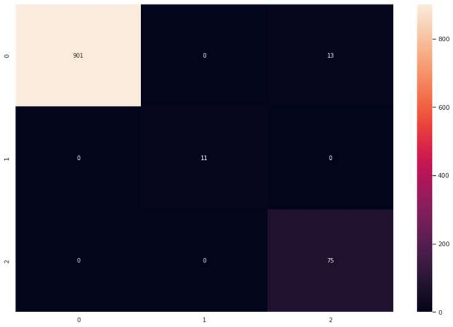


Figure 35 SVM BOW CM

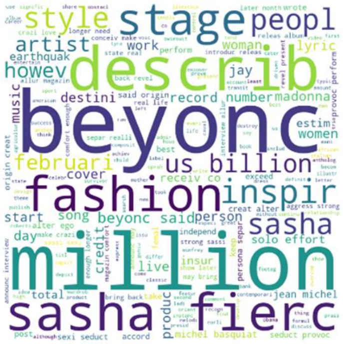


Figure 34 SVM BOW Word cloud

Kappa Score:
0.9234235560922451

TF-IDF

Decision Tree

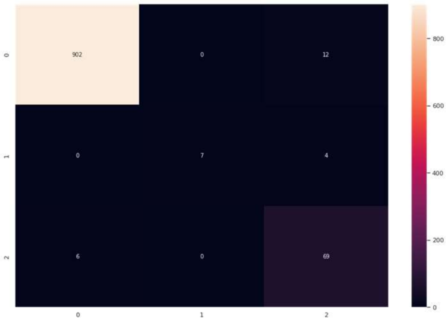


Figure 36 DT TF-IDF CM

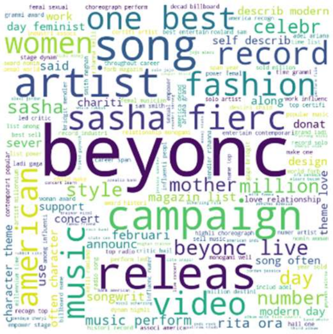


Figure 37 DT TF-IDF Word cloud

Kappa Score:
0.8655552567894595

KNN

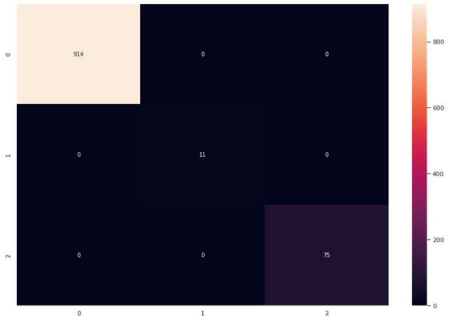


Figure 38 KNN TF-IDF CM

We cannot plot word cloud for because there’s no error, Kappa score = 1.0

Kappa Score:
1.0

Logistic Regression

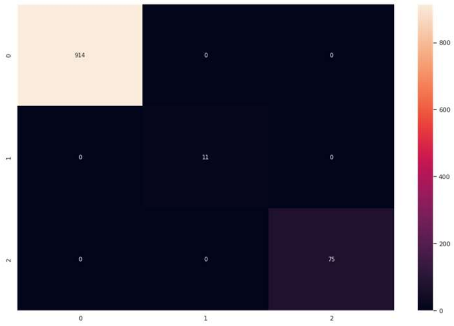


Figure 39 LR TF-IDF CM

We cannot plot word cloud for because there’s no error, Kappa score = 1.0

Kappa Score:
1.0

SVM

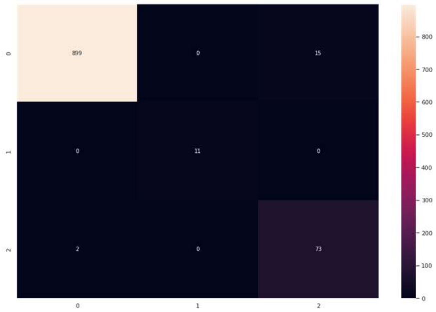


Figure 41 SVM TF-IDF CM

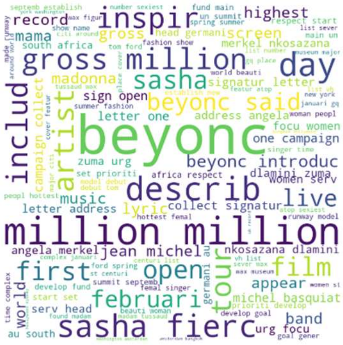


Figure 40 SVM TF-IDF Word Cloud

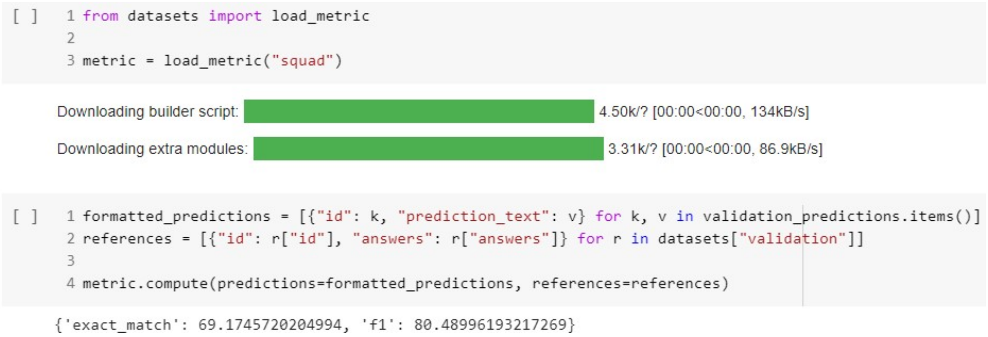
Kappa Score:
0.8998615733513975

Performance Evaluation

Question Answering

The evaluation phase it's not straightforward and requires some additional steps in order to perform it. In particular the output of the model are the loss and two scores indicating the likelihood of a token being the start and end of the answer. Simply taking the argmax of both will not do since it may create unfeasible situations: start position greater than end position and/or start position at question (remember that the input sequence is composed by the union of the tokenized answer and tokenized context).

These are the metrics that are provided by HuggingFace for the squad dataset: exact match and f1 score and their results:



```
from datasets import load_metric

metric = load_metric("squad")
```

```
Downloading builder script: 4.50k/? [00:00<00:00, 134kB/s]
Downloading extra modules: 3.31k/? [00:00<00:00, 86.9kB/s]
```

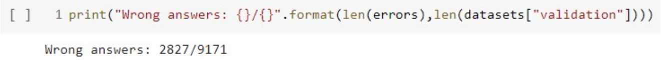
```
formatted_predictions = [{"id": k, "prediction_text": v} for k, v in validation_predictions.items()]
references = [{"id": r["id"], "answers": r["answers"]} for r in datasets["validation"]]

metric.compute(predictions=formatted_predictions, references=references)
```

```
{'exact_match': 69.1745720204994, 'f1': 80.48996193217269}
```

Figure 42 Exact match and F1 score

In error analysis, in order to analyze what kind of errors the model made, the mistaken predictions should first be retrieved. With "mistaken predictions" are intended those predictions that do not exactly match with the ground truth.

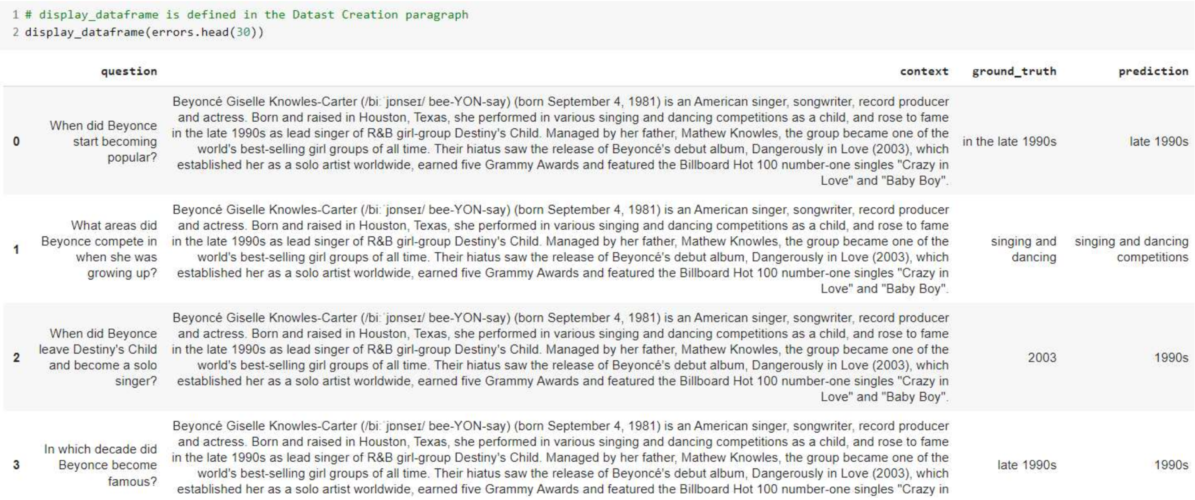
Total number of mistaken predictions:



```
[ ]  print("Wrong answers: {}/{}".format(len(errors),len(datasets["validation"])))

     Wrong answers: 2827/9171
```

Figure 43 Mistaken predictions

In order to check what kind of mistakes the model made, some of the errors will be displayed. First 30 errors:

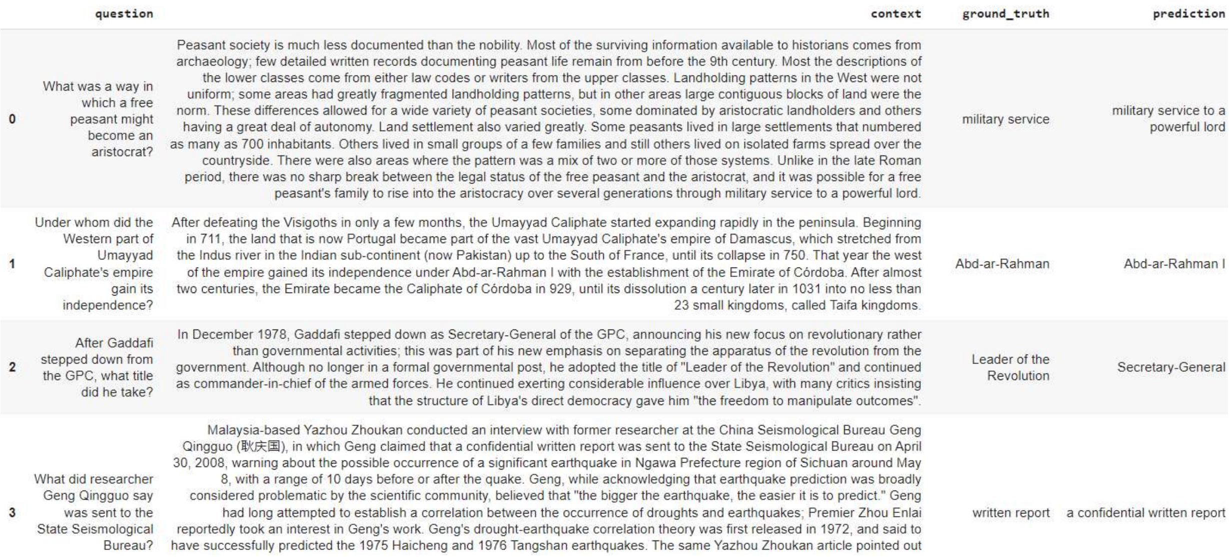


```
# display_dataframe is defined in the Datast Creation paragraph
display_dataframe(errors.head(30))
```

	question	context	ground_truth	prediction
0	When did Beyonce start becoming popular?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".	in the late 1990s	late 1990s
1	What areas did Beyonce compete in when she was growing up?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".	singing and dancing	singing and dancing competitions
2	When did Beyonce leave Destiny's Child and become a solo singer?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".	2003	1990s
3	In which decade did Beyonce become famous?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in	late 1990s	1990s

Figure 44 First 30 errors

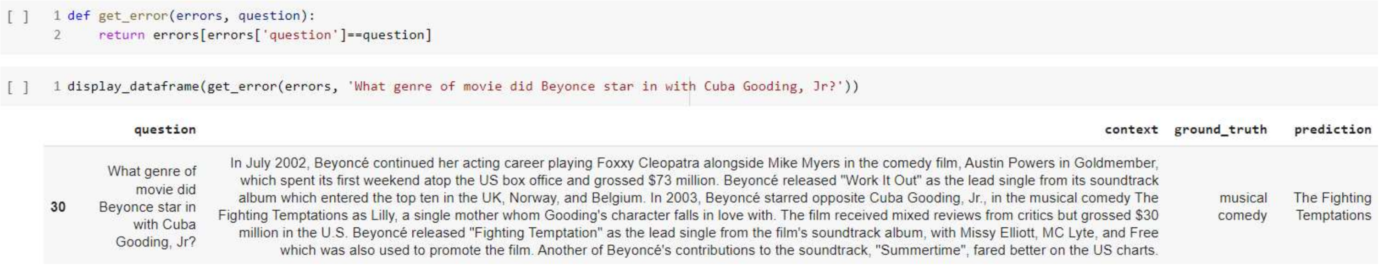
Random 30 errors:



	question	context	ground_truth	prediction
0	What was a way in which a free peasant might become an aristocrat?	Peasant society is much less documented than the nobility. Most of the surviving information available to historians comes from archaeology; few detailed written records documenting peasant life remain from before the 9th century. Most the descriptions of the lower classes come from either law codes or writers from the upper classes. Landholding patterns in the West were not uniform; some areas had greatly fragmented landholding patterns, but in other areas large contiguous blocks of land were the norm. These differences allowed for a wide variety of peasant societies, some dominated by aristocratic landholders and others having a great deal of autonomy. Land settlement also varied greatly. Some peasants lived in large settlements that numbered as many as 700 inhabitants. Others lived in small groups of a few families and still others lived on isolated farms spread over the countryside. There were also areas where the pattern was a mix of two or more of those systems. Unlike in the late Roman period, there was no sharp break between the legal status of the free peasant and the aristocrat, and it was possible for a free peasant's family to rise into the aristocracy over several generations through military service to a powerful lord.	military service	military service to a powerful lord
1	Under whom did the Western part of Umayyad Caliphate's empire gain its independence?	After defeating the Visigoths in only a few months, the Umayyad Caliphate started expanding rapidly in the peninsula. Beginning in 711, the land that is now Portugal became part of the vast Umayyad Caliphate's empire of Damascus, which stretched from the Indus river in the Indian sub-continent (now Pakistan) up to the South of France, until its collapse in 750. That year the west of the empire gained its independence under Abd-ar-Rahman I with the establishment of the Emirate of Córdoba. After almost two centuries, the Emirate became the Caliphate of Córdoba in 929, until its dissolution a century later in 1031 into no less than 23 small kingdoms, called Taifa kingdoms.	Abd-ar-Rahman	Abd-ar-Rahman I
2	After Gaddafi stepped down from the GPC, what title did he take?	In December 1978, Gaddafi stepped down as Secretary-General of the GPC, announcing his new focus on revolutionary rather than governmental activities; this was part of his new emphasis on separating the apparatus of the revolution from the government. Although no longer in a formal governmental post, he adopted the title of "Leader of the Revolution" and continued as commander-in-chief of the armed forces. He continued exerting considerable influence over Libya, with many critics insisting that the structure of Libya's direct democracy gave him "the freedom to manipulate outcomes".	Leader of the Revolution	Secretary-General
3	What did researcher Geng Qingguo say was sent to the State Seismological Bureau?	Malaysia-based Yazhou Zhoukan conducted an interview with former researcher at the China Seismological Bureau Geng Qingguo (耿庆国), in which Geng claimed that a confidential written report was sent to the State Seismological Bureau on April 30, 2008, warning about the possible occurrence of a significant earthquake in Ngawa Prefecture region of Sichuan around May 8, with a range of 10 days before or after the quake. Geng, while acknowledging that earthquake prediction was broadly considered problematic by the scientific community, believed that "the bigger the earthquake, the easier it is to predict." Geng had long attempted to establish a correlation between the occurrence of droughts and earthquakes; Premier Zhou Enlai reportedly took an interest in Geng's work. Geng's drought-earthquake correlation theory was first released in 1972, and said to have successfully predicted the 1975 Haicheng and 1976 Tangshan earthquakes. The same Yazhou Zhoukan article pointed out	written report	a confidential written report

Figure 45 Random 30 errors

And finally, an error can be reteived by querying a question:



```
[ ]  def get_error(errors, question):
         return errors[errors['question']==question]
```

```
[ ]  display_dataframe(get_error(errors, 'What genre of movie did Beyonce star in with Cuba Gooding, Jr?'))
```

	question	context	ground_truth	prediction
30	What genre of movie did Beyonce star in with Cuba Gooding, Jr?	In July 2002, Beyoncé continued her acting career playing Foxxy Cleopatra alongside Mike Myers in the comedy film, Austin Powers in Goldmember, which spent its first weekend atop the US box office and grossed $73 million. Beyoncé released "Work It Out" as the lead single from its soundtrack album which entered the top ten in the UK, Norway, and Belgium. In 2003, Beyoncé starred opposite Cuba Gooding, Jr., in the musical comedy The Fighting Temptations as Lilly, a single mother whom Gooding's character falls in love with. The film received mixed reviews from critics but grossed $30 million in the U.S. Beyoncé released "Fighting Temptation" as the lead single from the film's soundtrack album, with Missy Elliott, MC Lyte, and Free which was also used to promote the film. Another of Beyoncé's contributions to the soundtrack, "Summertime", fared better on the US charts.	musical comedy	The Fighting Temptations

Figure 46 Error retreival

Clustering

A comparison between the used models and pipelines was plotted regarding the silhouette scores and the champion model is Gaussian mixture with bag of words using 2000 features:

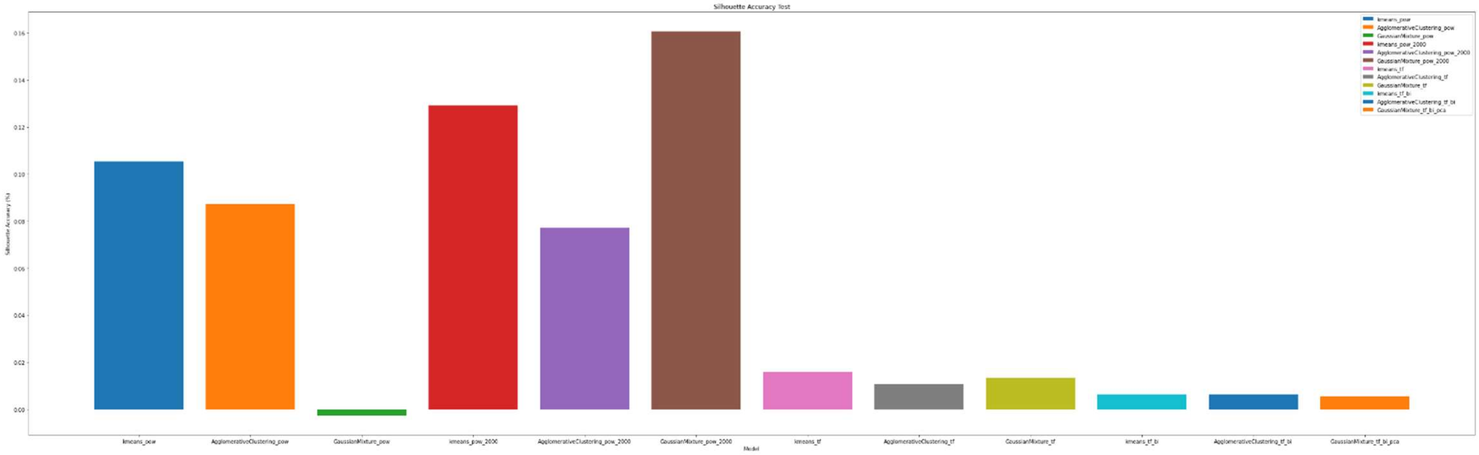


Figure 47 Clustering Evaluation (Silhouette Scores)

Classification

Comparisons between F1 scores in bag of words and TF-IDF were done as follows:

BOW

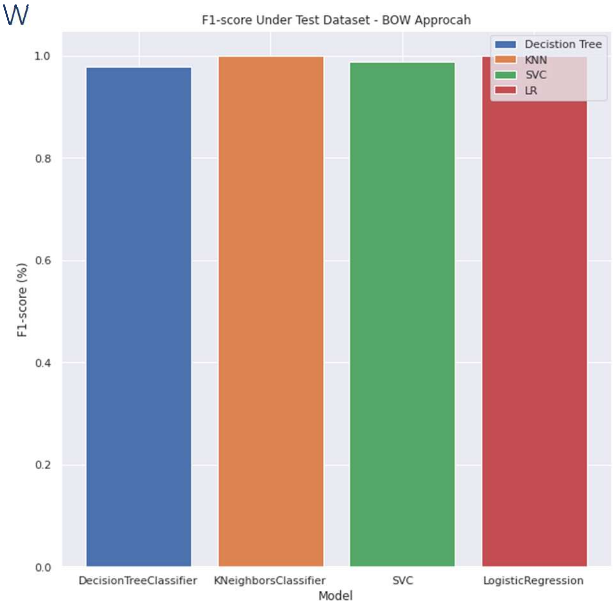
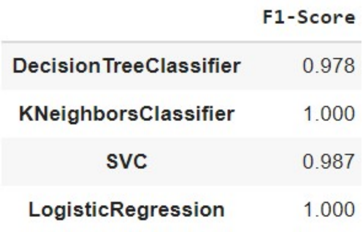


Figure 49 BOW comparison



	F1-Score
DecisionTreeClassifier	0.978
KNeighborsClassifier	1.000
SVC	0.987
LogisticRegression	1.000

Figure 48 F1 Scores (BOW)

TF-IDF

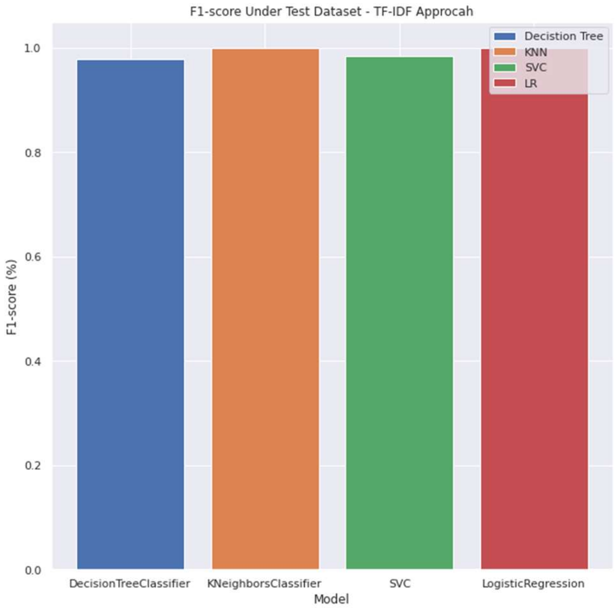
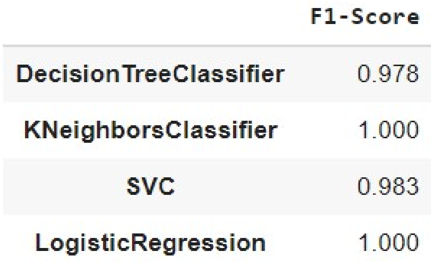


Figure 51 TF-IDF comparison



	F1-Score
DecisionTreeClassifier	0.978
KNeighborsClassifier	1.000
SVC	0.983
LogisticRegression	1.000

Figure 50 F1 Scores TF-IDF

As shown previously in BOW and TF-IDF, KNN and logistic regression showed F1 score of 1 so these are the champion models. When different feature engineering methods were applied on the models, there was no difference in the outcome in decision tree, K-Nearest neighbors and logistic regression.

Conclusion

Generally, Stanford question answering dataset (SQuAD) was used in this project. A question answering model was built based on DistilBERT then a fine tuning was done and it showed 2827 mistaken predictions out of 9171 which is an exact match of 69.17% with F1-Score of 80.4899% (0.804899). Secondly, the clustering pipeline was done using 4 feature engineering techniques with 3 classifiers each resulting Gaussian mixture with BOW using 2000 features as the champion model with silhouette score of 0.16078. Nevertheless, Classification pipeline was done using 2 feature engineering techniques with 4 classifiers each which showed the same champion models in both of BOW and TF-IDF which are Logistic regression and KNN.

References

[1] “The Stanford Question Answering Dataset,” *rajpurkar.github.io*. https://rajpurkar.github.io/SQuAD-explorer/#:~:text=What%20is%20SQuAD%3F.

[2] “b’SQuAD Dataset’,” *DeepAI*. https://deepai.org/dataset/squad. **(Main Dataset)**

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” *arXiv.org*, 2019. https://arxiv.org/abs/1910.01108

[4] B. J. McCaffrey and 11/16/2021, “How to Fine-Tune a Transformer Architecture NLP Model -,” *Visual Studio Magazine*. https://visualstudiomagazine.com/articles/2021/11/16/fine-tune-nlp-model.aspx .

[5] S. P. Lende and M. M. Raghuwanshi, “Question answering system on education acts using NLP techniques,” *IEEE Xplore*, Feb. 01, 2016. https://ieeexplore.ieee.org/abstract/document/7583963.