CAPSTONE PROJECT #2

QUESTION-ANSWER BOT

BY KING WONG

JUNE 13th, 2019

1. INTRODUCTION

• Objective:

The objective of this project is to create a Question-Answer (QA) bot in Python that takes in a question related to a dataset and the bot, based on a trained model, would return the answer back to the user.

The two main components of this bot are as follow:

(i) The program, based on a trained model, analyzes the user's question (input) and predicts the question's **intent** and **entity**.

(ii) Based on the **intent** and the **entity**'s value, the program will extract the relevant information a dataset (database). The information would be returned to the user as an answer (output)

The framework of this bot is in a closed domain environment, meaning the questions and answers are always related to a particular dataset. The format of the answers is pre-defined, however, the information returned as answer is determined and based on the user's question.

Example 1:

<User's Input>
"who sang kiss kiss?"

<QA Bot's Output>
"chris brown featuring tpain" was the artist of the song "kiss kiss" released in year "2007"

Example 2:

<User's Input>
"what year did lollipop come out?"

<QA Bot's Output>
The song "lollipop" by "lil wayne featuring static major" came out in "2008"

As you can guess from the examples above, the dataset is about music. Details about the dataset will be described in the next section.

For this model, the user's can ask about the artist, year, billboard ranking and lyrics, given the song name. An interactive environment is created to receive inputs and feedback from the user. The program, based on the feedback from the user, can continuously train itself and improve the results. Details will be discussed in subsequent sections.

• Dataset:

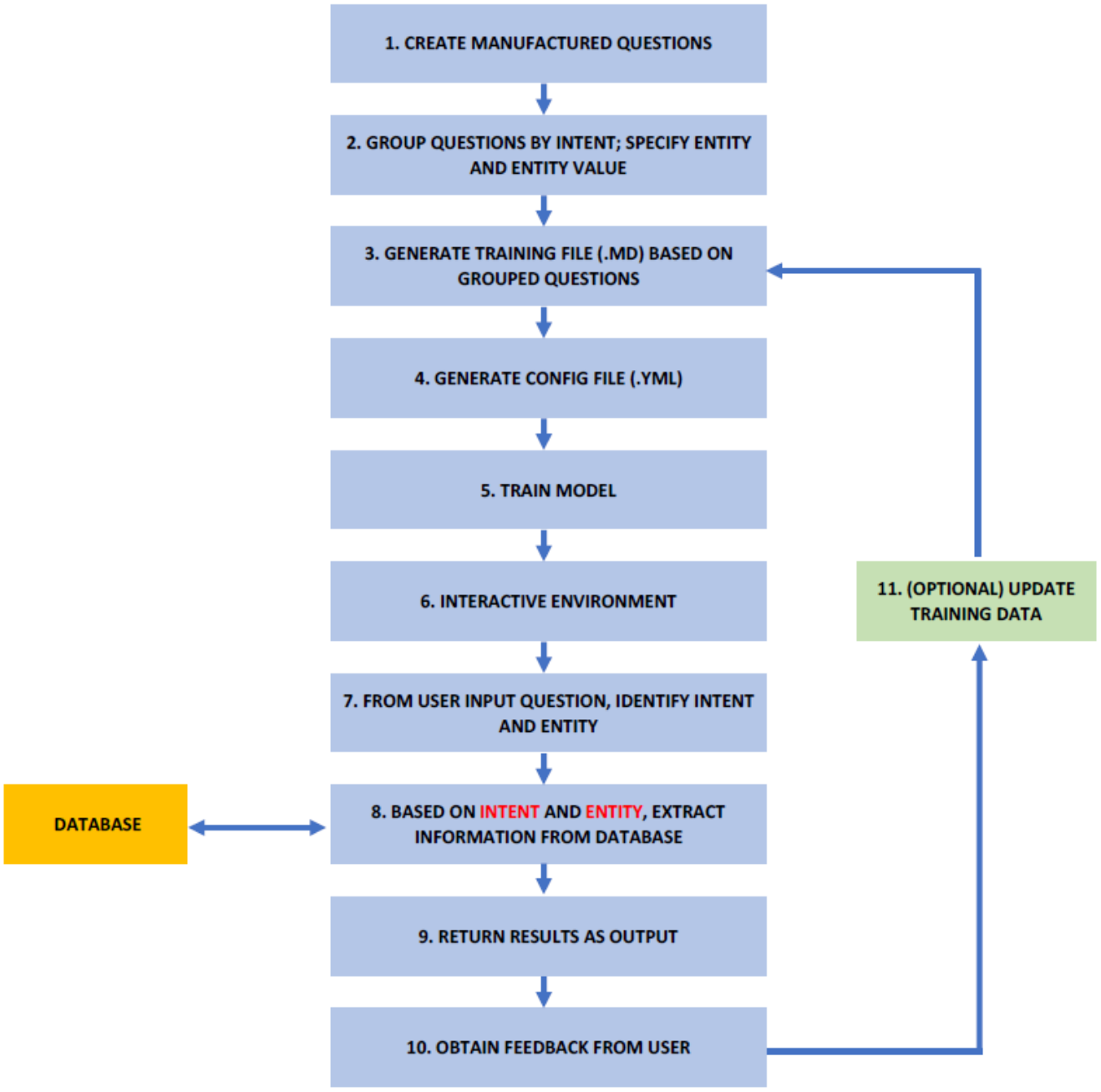
The original data set is called the "Billboard 1964-2015 Songs + Lyrics", which the top 100 billboard songs from year 1964 to 2015. < https://www.kaggle.com/rakannimer/billboard-lyrics >

However, for this project I have trimmed the dataset down to only containing song information from the 2000s (year 2000 to 2009). The approach, methodology and model would be exactly the same if the original dataset was used.

The dataset, which would act as the database also for information retrieval, contains song information including the billboard ranking (1 thru 100), song name, artist name, year and the song's lyrics.

• **Methodology:**

Below is a flowchart illustrating the pipeline of this project.



2. LOADING AND EXPLORING DATASET (DATABASE)

The dataset was downloaded as a .csv file from the source. See screenshots below upon loading the dataset into Pandas data frame.

From the first look, the text in the dataset is clean and only contains lower case (a-z) letters and (0-9) numbers. There is no special characters.

The dataset will act as the database, which the QA bot will retrieve information from.

Rar	Song	Artist	Yea	Lyrics
1	breathe	faith hill	2000	i can feel the magic floating in the air being with you gets me that
2	smooth	santana featuring rob thomas	2000	man its a hot one like seven inches from the midday sun well i hea
3	maria maria	santana featuring the product gb	2000	ladies and gents turn up your sound systems to the sound of carlos
4	i wanna know	joe	2000	yeah oh yeah alright oh oh ohits amazing how you knock me off m
5	everything you want	vertical horizon	2000	somewhere theres speaking its already coming in oh and its rising
...				
96	goodbye	kristinia debarge	2009	am i supposed to put my life on hold because you dont know how
97	say hey i love you	michael franti spearhead featuring cherine anderson	2009	this one goes out to you and yours worldwide i say hey i be gone to
98	pop champagne	jim jones featuring ron browz and juelz santana	2009	ether boy heyhow we ball in the club i know you hate it mami dan
99	pretty wings	maxwell	2009	time will bring the real end of our trial one day theyll be no remna
100	never say never	the fray	2009	some things we dont talk about rather do without and just hold the

3. CREATING THE QUESTION-ANSWER (QA) BOT MODEL

In this section, we will discuss the pipeline of creating this QA bot and go step-by-step in the process.

Step 1. Create manufactured questions

Since there is no dataset of questions pertaining to this specific dataset, simple questions were manually created (by me) for each type of intent. The manually created questions are made for asking about the four different kinds of intent --- artist, year, ranking and lyrics. The approach of this step is to create at least a minimum number and variation of questions as a starting point since there are many different ways to ask for the same intents.

Examples of these questions are shown below, using the song “bartender” by t pain in 2007.

- *“who sang bartender in year 2007?”*
- *“what are the lyrics for bartender by t pain?”*
- *“when did bartender by t pain hit the charts?”*
- *“what was the ranking of the song bartender in 2007?”*
- *“what year did bartender come out?”*
- *“where did bartender by t pain rank on the billboard charts?”*
- *“can you give me the lyrics of bartender?”*
- *“what is the name of the artist who sang bartender?”*

Step 2. Group questions, specify entity and entity value (as label)

After manually created some questions to start with, the questions are grouped by intent and the entity and the entity value (as label at this point) are inserted in a specific format. Since the RASA tool, which will be discussed in step 5, is used for model training and predicting, a specific format is followed to structure these questions. For example,

Intent = artist
- who sang [song name](song_name) in year
- what is the name of the artist who sang [song name](song_name)
…

Intent = year
- what year did [song name](song_name) come out
- when did [song name](song_name) by artist hit the charts
…

Intent = ranking
- what was the ranking of the song [song name](song_name) by artist in year
- where did [song name](song_name) by artist rank on the billboard charts
…

Intent = lyrics
- what are the lyrics for [song name](song_name) by artist
- can you give me the lyrics of [song name](song_name)
…

Step 3. Generate training file (.md) based on group questions

As mentioned in Step 2, the RASA tool takes in a markdown as input for training. A markdown file is basically a text file with a RASA-specific format and an extension of (.md).

In this case, there are 100 billboard songs in each year, and from 2000 to 2009 there are 1000 songs. To generate the markdown file, 1000 training questions are generated for each intent. This is done using a script written to randomly select from the manufactured questions from Step 1 and fill in the entity value with a different song name in each question. Therefore, in the 1000 training questions for the intent "artist", the questions could repeat but the entity values (the song name) are unique in each question.

In total, since there are 4 intents – artist, year, ranking, lyrics --- 4000 training questions total were generated.

A snippet of the markdown file showing intent "artist" is as follows:

```
## intent: artist
- who sings [breathe](song_name) in [year](year)
- which artist sang [smooth](song_name) in [year](year)
- what is the artist of the song [maria maria](song_name) in [year](year)
- who was the girl that sings [i wanna know](song_name) in [year](year)
- who was the guy that sings [everything you want](song_name) in [year](year)
- who sang [say my name](song_name)
- who was the guy that sang [i knew i loved you](song_name) in [year](year)
- who sings [amazed](song_name)
- what is the name of the singer that sang [bent](song_name) in [year](year)
- what is the name of the singer that sang [he wasnt man enough](song_name) in [year](year)
- who sang [higher](song_name) in [year](year)
- what is the name of the artist who sang [try again](song_name) in [year](year)
- who was the guy that sang [jumpin jumpin](song_name) in [year](year)
- who was the girl that sang [thong song](song_name) in [year](year)
- which band played [kryptonite](song_name) in [year](year)
- who was the band that sang [there you go](song_name) in [year](year)
- which artist sang [music](song_name) in [year](year)
- which singer sang [doesnt really matter](song_name)
- who is that song [what a girl wants](song_name) in [year](year) by
- who was the guy that sang [back at one](song_name) in [year](year)
- who was the guy that sang [bye bye bye](song_name) in [year](year)
```

The song name 'breathe' is enclosed in brackets [ ] to indicate that it is the entity value associated with the entity (song_name).

For the purpose of this QA Bot, only entity value related song name is needed to return an answer. Therefore no value in [year] (year) was filled in. The intention to include [year] (year) in the training questions is to better help the trained model to know which position the song name would end. For example, for question *"who sang because of you in year 2005?",* the model would be better identifying "because of you" as the entity value (song name) and not "because of you in year 2005".

Step 4. Generate configuration file (.yml)

Another important process besides generating the training questions is setting up the NLU pipeline in a configuration file.

RASA offers two types of pipeline --- “pretrained_embeddings_spacy” and “supervised_embeddings” --- for training. The difference is that the “pretrained_embeddings_spacy” pipeline uses pre-trained word vectors from spaCy, while the “supervised_embeddings” pipeline does not use any pre-trained word vectors, but instead fits them specifically for the dataset. For this project, the “pretrained_embeddings_spacy” pipeline is used since the words in questions are general in nature and not domain specific. The following configuration definitions are implemented.

```
language: "en_core_web_sm"
pipeline: "pretrained_embeddings_spacy"
```

Based on the RASA documentation, the following pipeline definition is equivalent to the above. Let’s walk through the definitions and try to understand the various components one-by-one.

```
language: "en_core_web_sm"

pipeline:
- name: "SpacyNLP"
- name: "SpacyTokenizer"
- name: "SpacyFeaturizer"
- name: "RegexFeaturizer"
- name: "CRFEntityExtractor"
- name: "EntitySynonymMapper"
- name: "SklearnIntentClassifier"
```

First a language parameter is set to English and the definition “en_core_web_sm" is specified for spaCy.

Secondly a sequence of components are defined and they will be executed sequentially for the machine learning model. Each component processes the input and creates an output. The output can be used by the following component that comes after in the pipeline.

"SpacyNLP"
This component initializes the spaCy model and it is defined at the beginning of the pipeline that uses any spacy components. The spaCy model contains pre-trained word vectors and other functions such as part-of-speech tagging, dependency parsing and named entity recognition.

"SpacyTokenizer"
This component uses spaCy tokenizer to split sentences into word tokens.

"SpacyFeaturizer"
This component transform the tokens into word vectors to feed into the RASA machine learning algorithm because the model only understands numerical information. A doc vector, in this case question-vectors, is used and is based on averaging the word vectors before being fed to the SVM algorithm as described below.

"RegexFeaturizer"
Regex features provide patterns to help the classifier recognize intent and entity classification. During training, the regex intent featurizer creates a list of regular expressions defined in the training data format. For each regex, a feature will be set marking whether this expression was found in the input, which will later be fed into intent classifier / entity extractor to simplify classification (assuming the classifier has learned during the training phase, that this set feature indicates a certain intent).

"CRFEntityExtractor"

This component implements conditional random fields to perform named entity recognition. Features of the words (capitalization, POS tagging, etc.) provide probabilities to certain entity classes, as are transitions between neighboring entity tags: the most likely set of tags is then calculated and returned.

"EntitySynonymMapper"
This component maps the synonymous entities together, if a value is specified. In this project, this is not used.

"SklearnIntentClassifier"
The sklearn intent classifier trains a linear SVM which gets optimized using a grid search. It also provides rankings of the labels that did not gets placed first. The spacy intent classifier needs to be preceded by a featurizer in the pipeline. This featurizer creates the features used for the classification.

Step 5. Train model

As mentioned in Step 3, RASA NLU (Natural Language Understanding) tool is used for this project to train questions for intent classification and entity extraction. In Step 4, we specified intent_classifier_sklearn, therefore the intent classifier trains a linear SVM and gets optimized using grid search.

To train the model, some RASA-specific steps have to be followed. First, the generated markdown file described in Step 3 gets loaded and assigned to variable training_data. Then the configuration file gets passed to the Trainer and assigned to variable trainer. The model gets trained by calling trainer.train(training_data). Once the training is complete, the information is saved to a local directory for use.

```
# loading the nlu training samples
training_data = load_data("nlu.md")

# load configuration file
trainer = Trainer(rasa_config.load("config.yml"))

# train the model
interpreter = trainer.train(training_data)

# store it for future use
model_directory = trainer.persist("./models/nlu", fixed_model_name="current")
```

Once the model is trained and saved, the trained results can be called out using RASA's interpreter function. The interpreter function is embedded in the QA Bot program shown in the next section.

```
# loading the interpreter
interpreter = Interpreter.load('./models/nlu/default/current')
```

4. QUESTION-ANSWER (QA) BOT IN ACTION

Step 6. Interactive Environment

An question-answer environment was created to take user's inputs and return answer as output. It can be executed in Jupyter Notebook or in the command prompt.

```
**WELCOME TO BILLBOARDS TOP 100 (2000s EDITION)**

ASK A QUESTION ABOUT ARTIST, YEAR, RANKING OR LYRICS GIVEN A SONG NAME.

EXAMPLES

ARTIST
  • QUESTION: who sang kiss kiss?
  • ANSWER: "chris brown featuring tpain" was the artist of the song "kiss kiss" released in year "2007"

YEAR
  • QUESTION: what year did lollipop come out?
  • ANSWER: The song "lollipop" by "lil wayne featuring static major" came out in "2008"

RANKING
  • QUESTION: what was the billboard ranking of the song womanizer?
  • ANSWER: The song "womanizer" by "britney spears" ranked "39" in year "2009"

LYRICS
  • QUESTION: what are the lyrics for what ive done by linkin park?
  • ANSWER: Here are the lyrics of the song "what ive done by "linkin park" [...in this farewell theres]

--------------------------------------------------------------------------------------------------------------"

Type your question here:
```

Here user can type his/her question.

Step 7. Identify Intent and Entity

```
--------------------------------------------------------------------------------------------------------------"

Type your question here:
which artist sang summer love?


"justin timberlake" was the artist of the song "summer love" released in year "2007"
```

Once the user enters his/her question, the program will clean the text of the question by converting all the characters to lower case and removing all apostrophe. The trained model will then process the question and tries to identify the intent and the entity. Internally, the trained model, returns the following dictionary full of information.

```
{'intent': {'name': 'artist', 'confidence': 0.9996413469739005},
 'entities': [{'start': 18,
   'end': 30,
   'value': 'summer love ?',
   'entity': 'song_name',
   'confidence': 0.9998216944834029,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'artist', 'confidence': 0.9996413469739005},
  {'name': 'year', 'confidence': 0.0003570809161967702},
  {'name': 'ranking', 'confidence': 1.4963489678030372e-06},
  {'name': 'lyrics', 'confidence': 7.576093502715249e-08}],
 'text': 'which artist sang summer love?'}
```

As you can see, the model was able to process the question and classify the intent as shown in 'name': 'artist' with confidence of 0.9996. The associated 'entity': 'song_name' is identified and its value is correctly identified as 'summer love' as well. Note that the confidence of the other intents are displayed as well as part of the information.

Step 8. Extract information from database

Based on the intent and entity (its value) identified in the last step, the program takes these clues and extracts the song information --- artist, year, ranking, lyrics --- from the billboard database. Basic equality operator and column indexing were used.

```
song_artist = str(df[df.Song == song_name].Artist.values[0])
song_year = str(df[df.Song == song_name].Year.values[0])
song_rank = str(df[df.Song == song_name].Rank.values[0])
song_lyrics = str(df[df.Song == song_name].Lyrics.values[0])
```

Step 9. Return results as output

Based on the information extracted from the database, an answer is generated and returned to the user using basic pre-defined response formats and filling in the values of the variables from Step 8.

For example,

if the intent is "artist", the response would look like this:
"{song_artist}" was the artist of the song "{song_name}" released in year "{song_year}"

if the intent is "year", the response would look like this:
The song "{song_name}" by "{song_artist}" came out in "{song_year}"

if the intent is "ranking", the response would look like this:
The song "{song_name}" by "{song_artist}" ranked "{song_rank}" in year "{song_year}"

if the intent is "lyrics", the response would look like this:
Here are the lyrics of the song "{song_name}" by "{song_artist}"
(Output lyrics…)

Here is what it looks like on screen.

```
"justin timberlake" was the artist of the song "summer love" released in year "2007"
```

Afterwards, user is invited to continue to ask another question or end the bot.
If user types yes, the program restarts.

```
----------------------------------------------------------------------------------------------------------------------"

Type your question here:
```

If user types no, the program terminates.

```
Would you like to ask another question? (Enter Yes or No)
no
Have a good day. Program Terminate.
```

5. EXPLORE THE QA BOT

Let's try some more questions and compare to what we have generated in the training data.

Example 1:

```
------------------------------------------------------------------------------------------------------------"

Type your question here:
what band produced that song stop and stare

Answer:
"onerepublic" was the artist of the song "stop and stare" released in year "2008"
```

Let's output the results from processing the question.

```
{'intent': {'name': 'artist', 'confidence': 0.9882493935248448},
 'entities': [{'start': 29,
   'end': 43,
   'value': 'stop and stare',
   'entity': 'song_name',
   'confidence': 0.9998695383087648,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'artist', 'confidence': 0.9882493935248448},
  {'name': 'ranking', 'confidence': 0.010643872607134555},
  {'name': 'year', 'confidence': 0.0008816441447821887},
  {'name': 'lyrics', 'confidence': 0.00022508972323848382}],
 'text': 'what band produced that song stop and stare'}
```

We asked a question of 'which band produced the song…'. The program was able to classify the intent with high confidence and return the correct answer. Interestingly, if we investigate the training data, the model actually has not seen this exact question before. The closest training question was *'which band played [song name](song_name) in year'*. The model has probably picked up the word 'band', which looking in more detail, does not exist in the training questions in other intents.

Example 2:

```
------------------------------------------------------------------------------------------------------------"

Type your question here:
in the early 2000s, who was the artist that sang he wasn't man enough

Answer:
The song "he wasnt man enough" by "toni braxton" ranked "10" in year "2000"
```

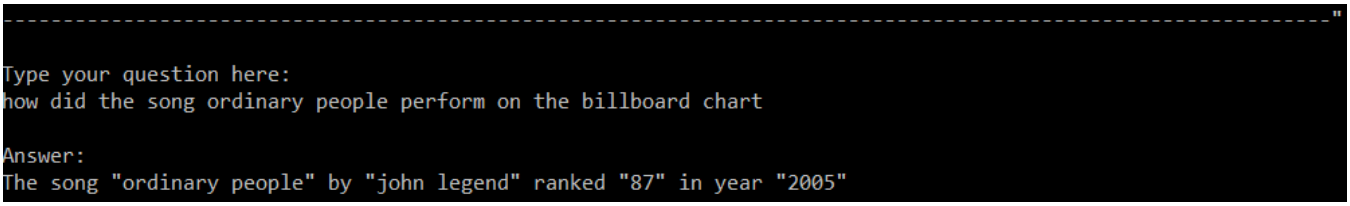
Let's output the results from processing the question.

```
{'intent': {'name': 'artist', 'confidence': 0.702924820997266},
 'entities': [{'start': 48,
   'end': 67,
   'value': 'he wanst man enough',
   'entity': 'song_name',
   'confidence': 0.9996492487577134,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'artist', 'confidence': 0.702924820997266},
  {'name': 'ranking', 'confidence': 0.2932055462224076},
  {'name': 'lyrics', 'confidence': 0.00310101207443414},
  {'name': 'year', 'confidence': 0.0007686207058921094}],
 'text': 'in the early 2000s who was the artist that sang he wanst man enough'}
```

The program was able to classify the intent with decent confidence and return the correct answer. Interestingly, if we investigate the training data, all of the questions for the intent : artist start with 'who',

'what', 'which', but never "in the year [year]". It shows that not only slight variation of the question with the same intent does not affect classifier, the order does not matter as well. This is some good results. Let's continue.

Example 3:

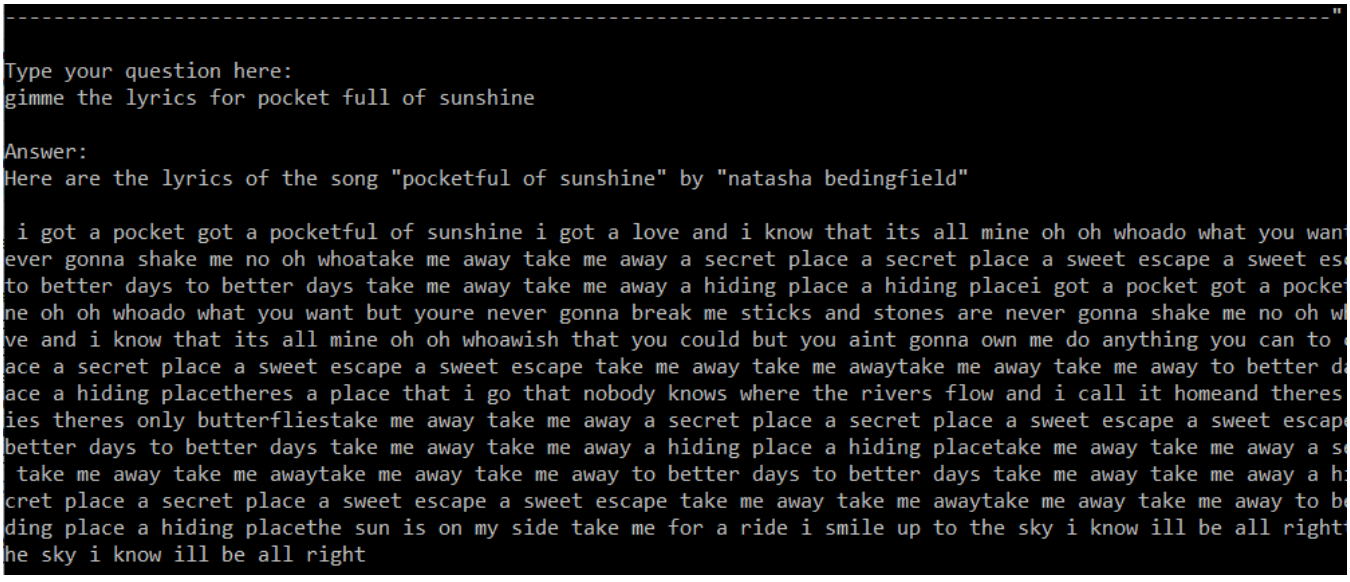


Let's output the results from processing the question.

```
{'intent': {'name': 'ranking', 'confidence': 0.9783501778539209},
 'entities': [{'start': 17,
   'end': 40,
   'value': 'ordinary people perform',
   'entity': 'song_name',
   'confidence': 0.9991784118184697,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'ranking', 'confidence': 0.9783501778539209},
  {'name': 'year', 'confidence': 0.021041049867031394},
  {'name': 'lyrics', 'confidence': 0.0003052349602616012},
  {'name': 'artist', 'confidence': 0.0003035373187860483}],
 'text': 'how did the song ordinary people perform on the billboard chart'}
```

First of all, this is one of my favorite songs and I am upset that it was ranked so low at 87. Anyways, the program was able to classify the intent with high confidence and return the correct answer. Again the program has not seen this exact question. The closest training question was *'how did [song name](song_name) by artist do on the billboard'*. The program may have picked up the word 'how' and 'billboard' that contributed to the correct prediction. Notice in this case the entity value was identified to be 'ordinary people perform'. It included one more word that it should have. In the next section, we will discuss how extracting information based on the wrong song name 'ordinary people perform' did not produce an error, and also how adding a feedback feature would help this case.

Example 4:



Let's output the results from processing the question.

```
{'intent': {'name': 'lyrics', 'confidence': 0.9805672687066653},
 'entities': [{'start': 21,
   'end': 44,
   'value': 'pocket full of sunshine',
   'entity': 'song_name',
   'confidence': 0.9994784886650014,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'lyrics', 'confidence': 0.9805672687066653},
  {'name': 'artist', 'confidence': 0.015366584208348518},
  {'name': 'ranking', 'confidence': 0.002125786881503364},
  {'name': 'year', 'confidence': 0.001940360203482783}],
 'text': 'gimme the lyrics for pocket full of sunshine'}
```


The program was able to classify the intent with high confidence and return the correct answer. Again the program has not seen this exact question. The closest training question was *'give me the lyrics for [song name](song_name)'*. The song name was entered incorrectly, similar to Example 2, but the program was still able to extract information from the database and return the right answer.

Overall, the QA Bot performed as expected. It was interesting that the program was able to go beyond its training data and classify the correct intent from questions that were slightly, but not exactly, different.

In the next section, we will discuss examples that go beyond the QA Bot's limit and how it was handled or improved on.

Model Testing

But before, let's look deeper into the model and try to quantify how well the QA Bot does at predicting the correct intent based on the manually created questions.

	index	True_Label
0	what is the name of the artist who sang here without you in 2003	artist
1	What are the lyrics in sugar were goin down	lyrics
2	What are the lyrics in i need you	lyrics
3	where did beautiful by christina aguilera rank on the billboard charts	ranking
4	who is that song same girl in 2007 by	artist
5	When did walked outta heaven hit the charts	year

…

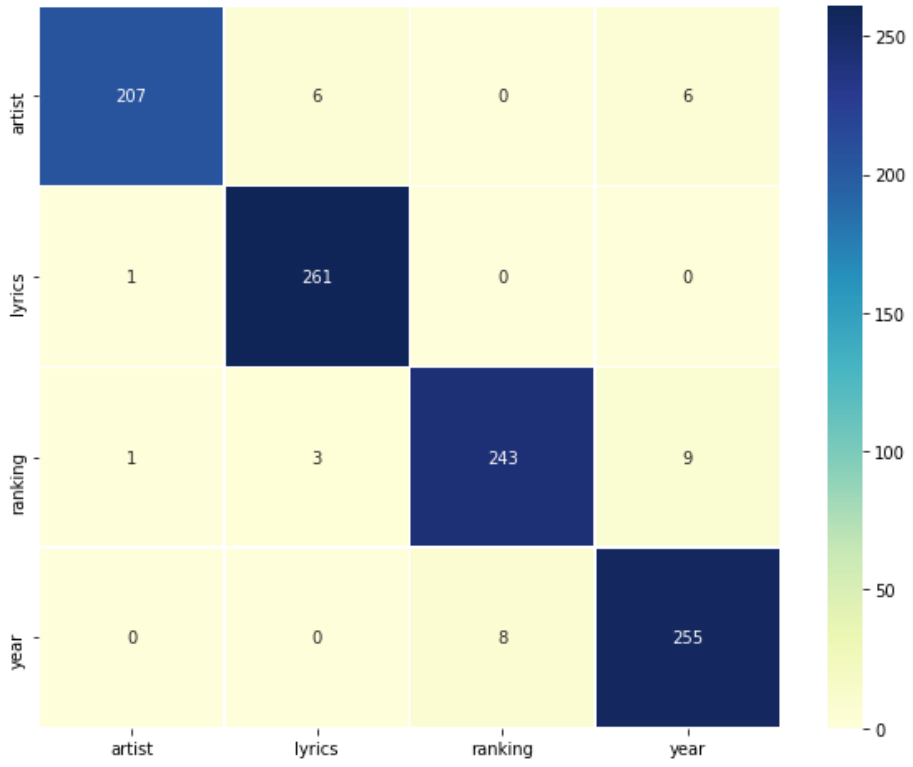
A set of 1000 test questions, 25% the size of the training questions, are generated randomly from the manually created questions. A script was written to randomly select an intent and then select a question within that intent each time. Afterwards, other variables such as the song names, year and artist are randomly filled in. Here are the breakdown of the distribution of the questions per intent:

Intent	# of Questions
Artist	219
Year	263
Ranking	256

Lyrics	262
	Total = 1000

Each question is fed into the model and the predicted intents are compared to the true intents.

As illustrated in the confusion matrix below, the model predicted correctly 966 times over 1000 questions. Therefore, the overall accuracy is 96.6% and shows great performance.



Looking into this further, the results for the individual intents are as follow:

Intent	# of Questions
Artist	94.5%
Year	99.2%
Ranking	94.9%
Lyrics	97.3%

Based on this information, it looks like the questions corresponding to intent = year has the highest accuracy rate, while questions corresponding to intent = artist has the lowest accuracy rate.

This is perhaps the questions in the intent = year have consistently similar words such as what year, which year, when, etc., meaning there are less variation in the manually created questions. This may change if someone else has created these questions manually or if more variation of questions has been exposed to the model.

Let's take a look at the questions that their intents are incorrectly predicted.

index	True_Label	Pred_Label
which band played one step at a time in 2008	artist	year
show me the lyrics of the way you love me by faith hill	lyrics	artist
what was stronger by kanye west rank	ranking	artist
when did lost without u by robin thicke come out	year	ranking

Unfortunately, it is not immediately obvious why the model has mis-classified these questions' intents. There is potentially "gray area" in the feature space where the question can be mis-classified. Perhaps a different model and/or more training data would be helpful to mitigate these errors. In addition, the feedback feature discussed in the next section will provide a solution to improve the model.

6. LIMITATIONS AND IMPROVEMENTS TO THE QUESTION-ANSWER (QA) BOT

In the previous Section 5, we discussed two issues:

(i) What if the user types in an incorrect song name that does not match exactly the song name in the database?

Let's take Example 4 from Section 5 for example. The incorrect song name was entered "pocket full of sunshine" and program identified the entity value as such. To try to help the program to find the right song, a CountVectorizer was used to make a vocabulary of all the unique words. (TfidfVectorizer could have been used also, but CountVectorizer was deemed sufficient for the use here). The incorrect song name "pocket full of sunshine" would be represented as a count vector and have a count of 2 in this vocabulary, 1 for "of" and 1 for "sunshine", assuming the words "pocket" and "full" are not in the vocabulary. When taking the dot product of this vector to each of every count vector representation of the song names, the highest sum would equal the closest match in terms of similarity in words. This would give a good approximation of the closest song name to the incorrect song name entered. For this case,

Incorrect song name, *"pocket full of sunshine"*

Correct song names in the database:

1. *"pocketful of sunshine"* = 2 matching words: "of", "sunshine"
2. *"the game of love"* = 1 matching word: "of",
3. *"full moon"* = 1 matching words "full"
4. *"absolutely story of a girl"* = 1 matching words "of"
5. *"empire state of mind"* = 1 matching words "of"

Based on the above, the program selected the top match *"pocketful of sunshine"* as the correct song name.

(ii) There are many ways to ask about the same intent. What if user types in a question that is totally unseen by the model?

For example, this situation could happen:

```
----------------------------------------------------------------------------------------------------"


Type your question here:
justin timberlake sang cry me a river in what year
```

```
Answer:
"justin timberlake" was the artist of the song "cry me a river" released in year "2003"
```

In this case, the program was not able to classify the intent to be "year", but instead classified it as "artist". It was able to identify the entity and its value "cry me a river" as the song name.

```
{'intent': {'name': 'artist', 'confidence': 0.9986450362263157},
 'entities': [{'start': 23,
   'end': 37,
   'value': 'cry me a river',
   'entity': 'song_name',
   'confidence': 0.9993206600781767,
   'extractor': 'CRFEntityExtractor'},
  {'start': 46,
   'end': 50,
   'value': 'year',
   'entity': 'year',
   'confidence': 0.4861276133189548,
   'extractor': 'CRFEntityExtractor'}],
 'intent_ranking': [{'name': 'artist', 'confidence': 0.9986450362263157},
  {'name': 'year', 'confidence': 0.0013152864998786443},
  {'name': 'ranking', 'confidence': 3.5839617977531866e-05},
  {'name': 'lyrics', 'confidence': 3.837655828149319e-06}],
 'text': 'justin timberlake sang cry me a river in what year'}
```

```
Did we provide you with the correct result? (Enter Yes or No)
no
Sorry. We will update our training data to improve the results. Please help us here...

Are you asking about an "artist", "year", "ranking" or "lyrics"?
year
What is the name of the song again?
cry me a river
```

In any case, a feedback feature after the answer was added to solicit feedback from the user. If the answer is not correct, the user can input no for this request. The program will then ask about the intent and the song name (entity) again. The program will confirm with the user its input and the correct answer will be outputted as an answer.

```
1.:
cry me a river


2.:
dont let me get me


3.:
how do you like me now


4.:
ride wit me


5.:
follow me
Is it any of the above? (Type 1 thru 5 to select, or No)
1
```

```
The song "cry me a river" by "justin timberlake" came out in "2003"
```

Afterwards, this feedback feature will take the received information, add it in the training data and re-train the model. This process of re-training the model takes about half a minute. Next time when you ask a similar question, the program will be able to identify the intent and entity correctly, and return the correct answer.

```
----------------------------------------------------------------------------------------------------"


Type your question here:
justin timberlake sang cry me a river in what year
```

```
Answer:
The song "cry me a river" by "justin timberlake" came out in "2003"
```

This feature of requesting for feedback and updating the training data serves as a way to continuously collecting new data to train and to improve the model.

Other improvements:

(iii) When the program cannot classify the intent from the processed question, or with confidence less than 0.45, or when it cannot identify the entity, the program also asks for confirmation from the user similar to (ii). This step also triggers the program to obtain information from the user's feedback, update the training data and re-train the model.

7. CONCLUSION

In this report we have discussed in details the pipeline in the development of this QA Bot. We walked through step-by-step (i) setting up of the training (question) data and training the model using RASA, (ii) using an interactive environment to receive user's input, predict intents and entities from the questions, extracting the necessary information from the database and returning answers to the user as outputs, (iii) a feedback request and updating the training data, re-train model and improve results.

We have explored the capabilities of the QA Bot and also the limitations. Future investigations and improvements to the QA Bot include:

• Deploying this QA Bot on the internet for other users to use.

• The feedback feature step triggers an update of the training data and re-trains the entire model, which takes close to a minute. An improvement is to change the model partially instead of the entire model to make the process more efficient.

• Design QA Bot to be able to perform functions that require multiple intents or multiple entities. For example, "what song did Justin Timberlake release in year 2005?" given he has multiple songs in different years.

• Expand functions of what the QA Bot can do. For example, be able to ask "how many billboard top 100 songs does Beyonce have?"

• Expand the domain of the QA Bot.

• Explore how other NLP techniques such as including parts of speech, Stemming and Lemmatization, word2vec, may help improve this bot.

THE END