**Chatbot Report**

**System Description:**

This program is an implementation of a very simple and naive chatbot. This chatbot has a domain knowledge around the Los Angeles Lakers and can only speak regarding that subject. At times, the chatbot answers perfectly at other times not so much. This program uses a lot of techniques discussed in class from tokenization, POS tagging, term count, tfidf, and machine learning. The chatbot works like this. It introduces itself and gets the information of the user. The user information is stored and then allows the user the freedom to type anything. The program uses rule based and ML to determine if it is a question, if so, it tries to answer it by doing cosine similarity between the query and the knowledge base. It returns the most common sentence. If the input is not a question, then it determines if it is a command using rule based, if so it tries to do that command by treating it as a question. Otherwise, it performs sentiment analysis using ML to determine if it is an opinion that is negative or positive of the user. It then stores it with the connotation and replies to it. The process then repeats until the user enters a command that makes it exit.

**Logical Steps (Simplified):**

1. Create ML Model for Sentiment Analysis (NLP Tech: ML)
2. Introduce chatbot and store user information
3. Get user input and determine if it is a question (NLP Tech: POS Tagging & ML)
4. If question, then formulate an answer using the knowledge base (NLP Tech: Tokenization, TFIDF, Cosine Similarity)
5. If not a question, determine if it is a command or statement
   1. If command, then treat like a question
   2. If statement, predict sentiment using ML model in step 1 and reply based on the sentiment.
6. Repeat

**NLP Techniques:**

*Machine Learning -* *Sentiment Analysis*

In order to figure out, if a sentence has a specific connotation, I decide to train a model to help me predict that. To train a model, you must have data to work with and be able to learn from. I used the twitter samples from the NLTK library. I retrieved the separated data, did some preprocessing on the tweets, and combined them to form one dataset. Each training example had a label attached to it. I then just simply used the NLTK Naïve Bayes Classifier to train the model. This model was then used to predict the sentiment of the sentence later on in the program. The reason for doing it so early is because it took a long time to train, so this delayed the conservation. However, if it is done earlier before the conversation starts, then the user would be able to conserve seamlessly.

*POS Tagging – Determine if it is a question*

In order to understand the user’s inputs, my program first determines, if it is a question. The way I did that, is by using the library spacy. I did some POS tagging to determine the structure of the sentence. Most questions have a format where a word with the tag W+ is followed by V+. For example, what (WP) is (VBZ) the time. So, my program determines if the input has the following structure (W+V+) by the POS tag using spacy. If so, then it is a question. If not, then the program continues to check if it is a question by using ML.

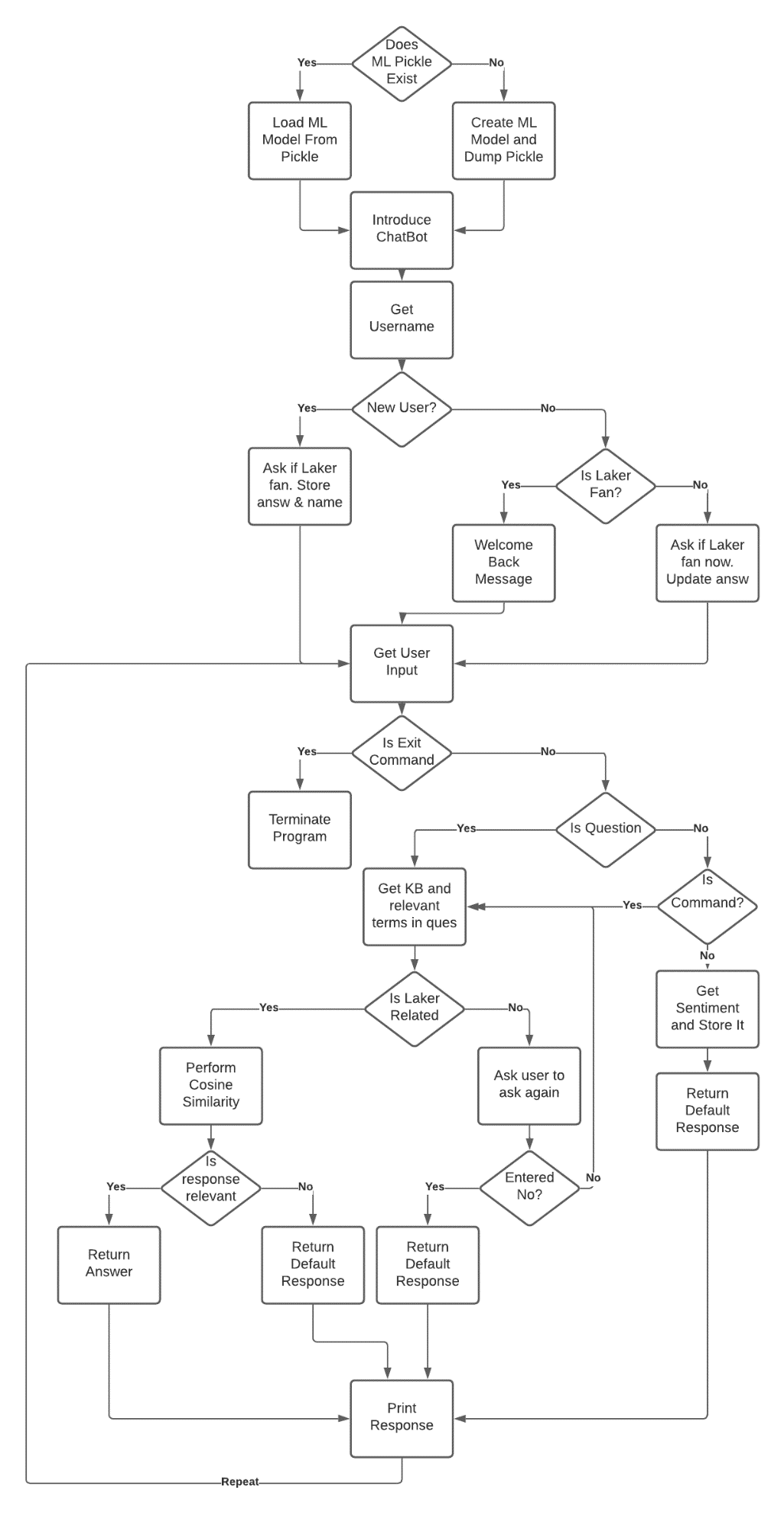
*Machine Learning with Count Vectorizer – Determine if it is a question*

If the POS tagging cannot determine if it is a question, then a machine learning model is used to determine if the sentence is a question. There are two files I have created to train my model. One file is filled with different types of questions and the other is filled with just statements. So, I first read the questions and statements into a data frame using pandas. I understand this is going to be a binary classification either question or not, so I added a label to each question and statement. One for question (positive) and zero for the statements (negative). I think combine the data frames to form one data frame to use for the training model. I then extract the features (sentences) and label from the data frame to use separately. I then use a count vectorizer that is fitted to the features. This will allow us to then transform our features into vectors and use it a machine learning algorithm. So that is what I do. I transform the features and the user input. With the training count vectors, I then train a logistic regression model using scikit-learn. Finally, I predict the classification of the user input and output the appropriate response.

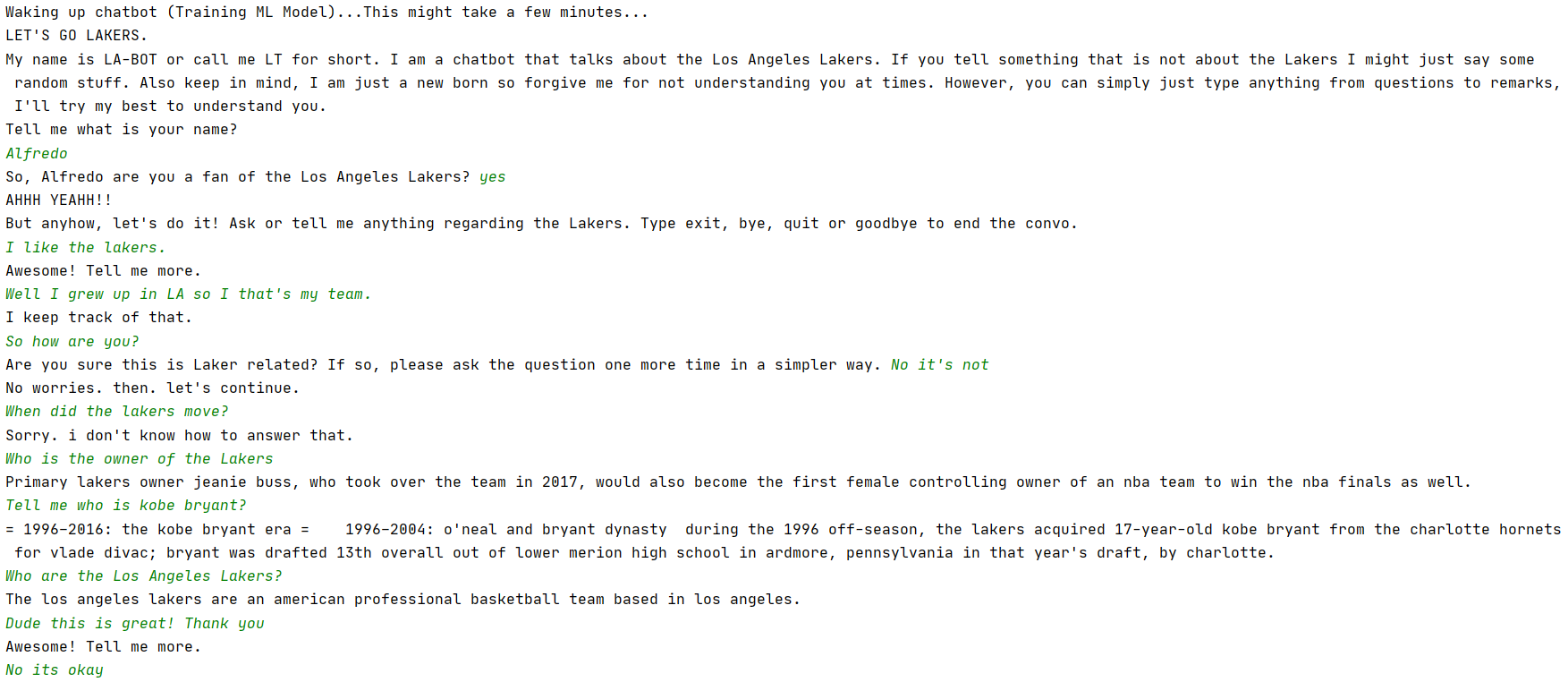
*Cosine Similarity with Tokenization & TF-IDF – Getting an answer to the question*

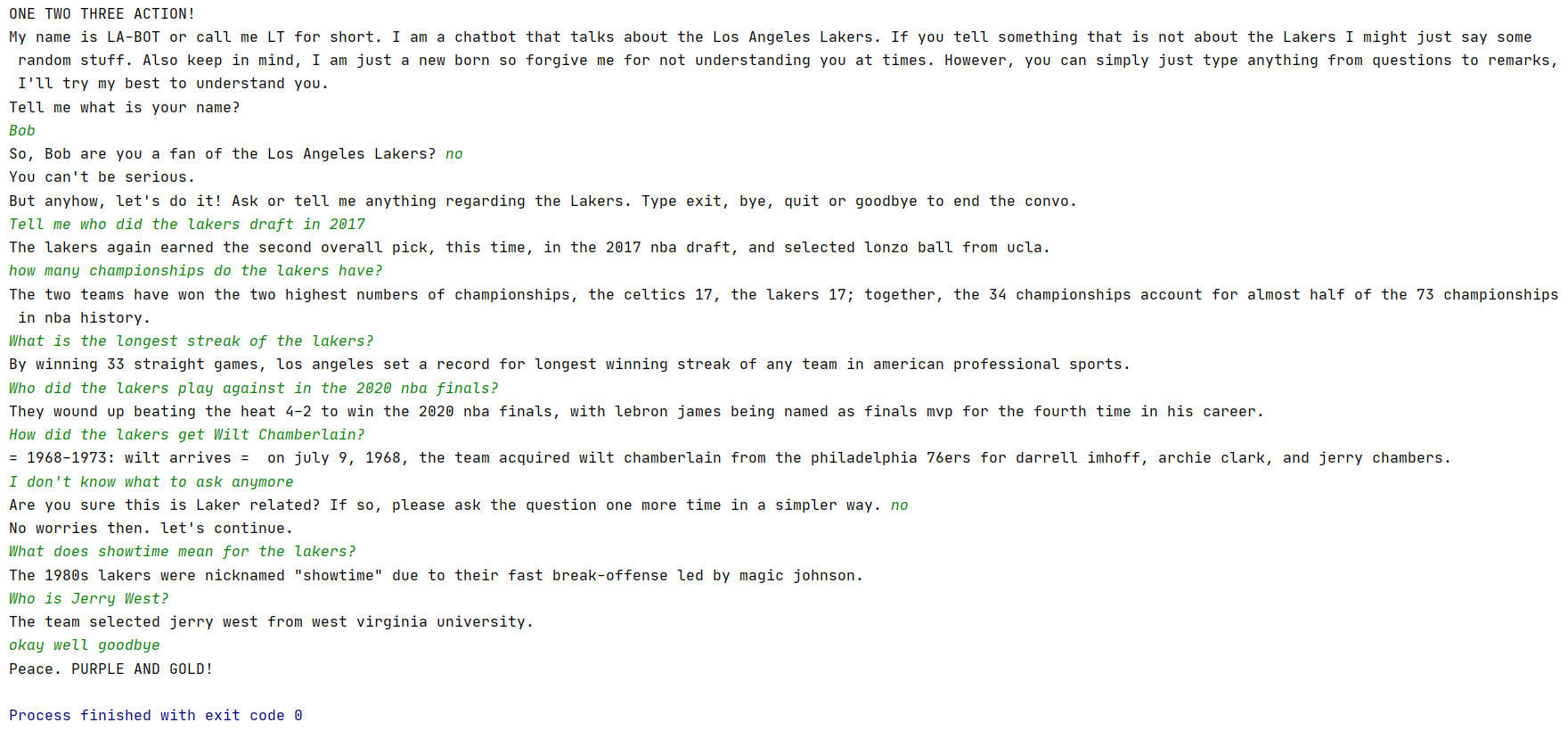
The way my program answers question is to find the closest sentence from the knowledge base to the question. In order to do this, I must perform cosine similarity. The first thing I do is tokenize my knowledge base and the user input into sentence tokens. Each sentence is going to act like its own document. The next thing I do, is I perform TF-IDF to all the sentences or “documents” to only get the relevant terms of the knowledge base. I then fit and transform the knowledge base to this TF-IDF vectorizer from scikit-learn to transform the sentences and user input into vectors. Finally, I using scikit’s learn cosine similarity, I am able to get the vector that is closest to the question.

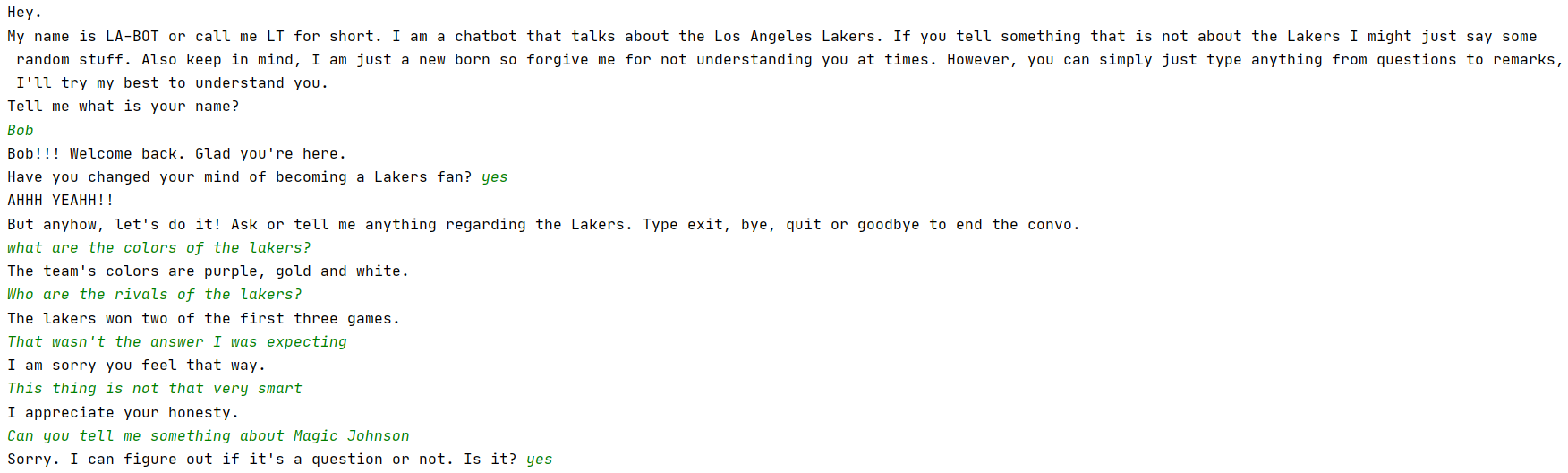
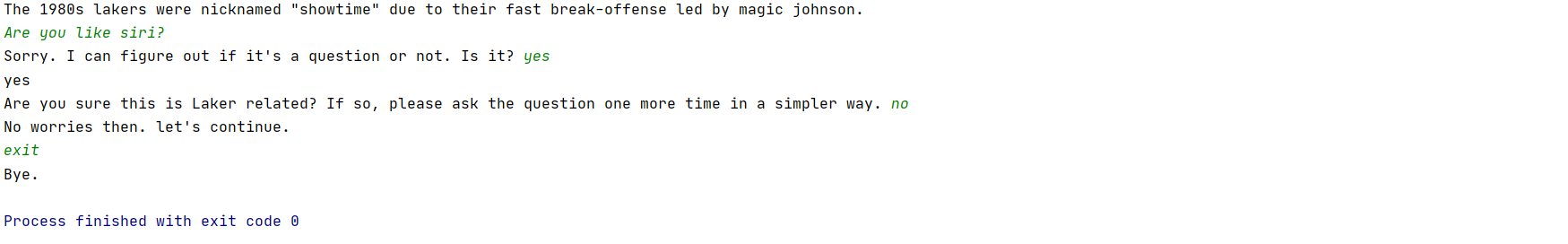
**Flow Chart:**

****

**Sample Dialog Interactions**

*Case 1:*

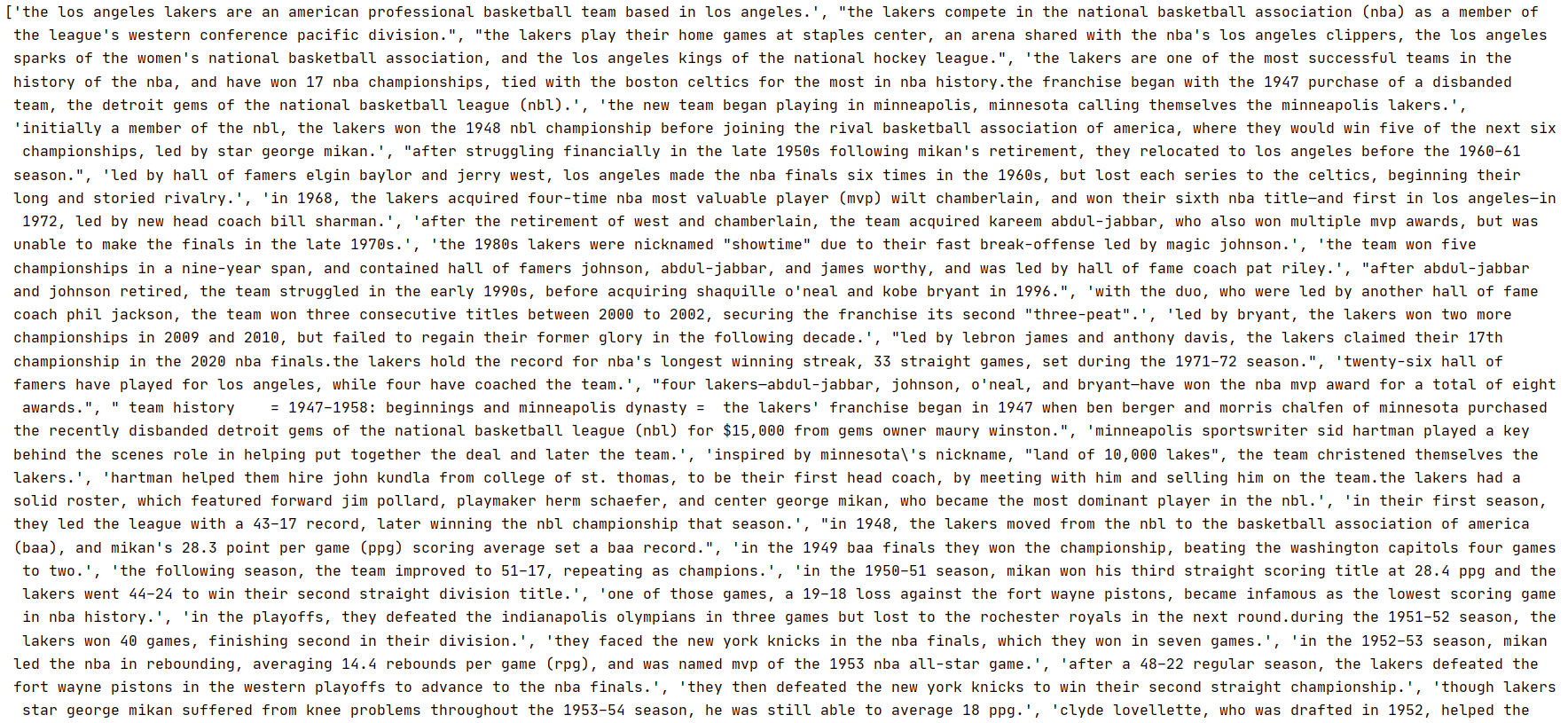
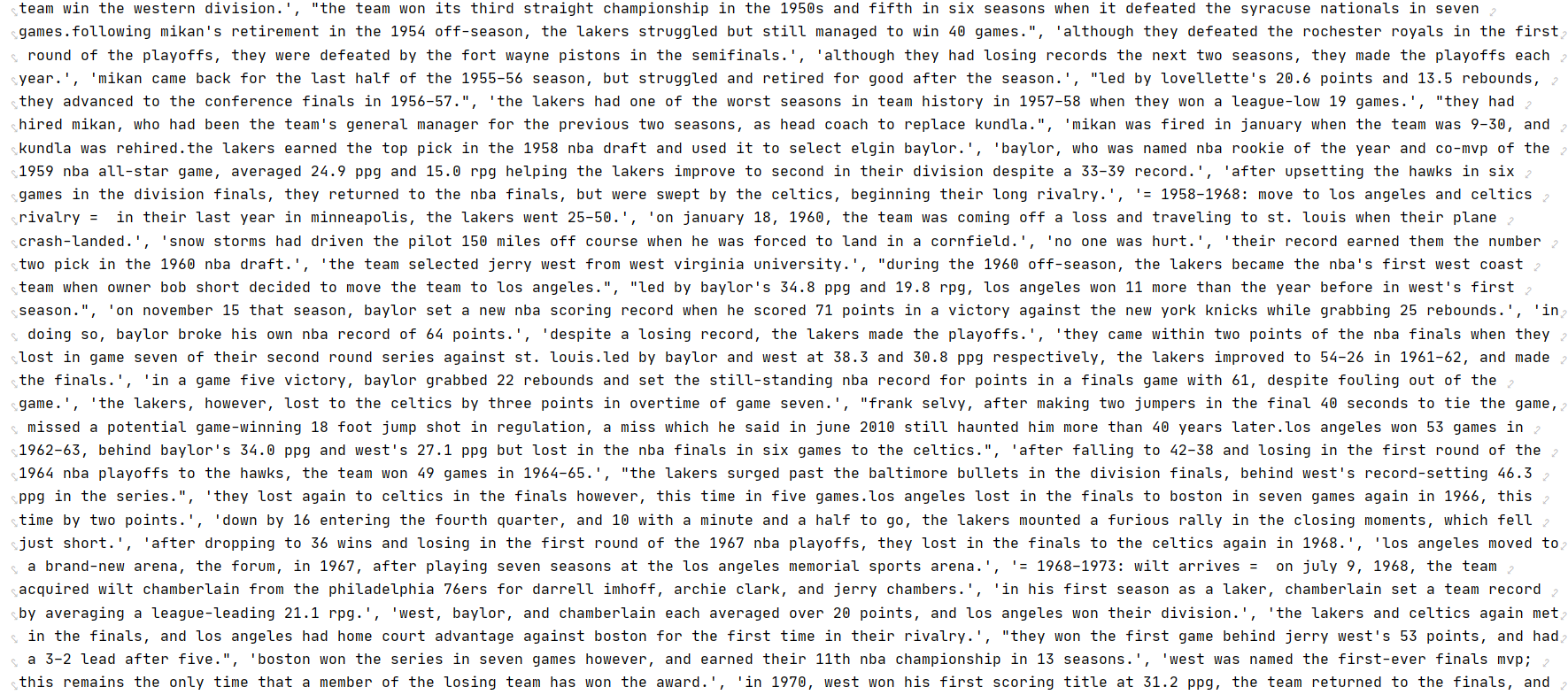
*Case 2:*

*Case 3:*

**Appendix for Knowledge Base**

The knowledge base used is from the Wikipedia API. What the program does, is that it retrieves the Lakers Wikipedia page and sentence tokenize the content. Then cosine similarity is performed on each sentence with the user input to get the response. The knowledge base is shown below.

**Note**: The knowledge base shown is below is NOT the complete knowledge base used in the program. The whole KB is not placed in here because it would take too much space.

**Appendix for User Models**

The appendix for the user models is divided into two database files. The first database file only has one table with two columns, one for the username which is the primary key, and the other one to that simply says “yes” or “no” to the question if the user is a Laker fan. This information is used in the introduction of the chatbot. The other database file could potentially have many tables. Each table is named after the username that is currently in session with the chatbot. Each table has three columns. The first column is an autoincrement unique id, the second column is the statement the user has made, and the third column is simply a sentence saying “positive” or “negative” describing the sentiment of the sentence. It is difficult to open the files through windows, so I have extracted the database files in a python program and converted them to a list to print them. The following two databases are shown below.









**Evaluation**

*Strengthens:*

* Can answer simple questions that involve the lakers, especially when the key words are align with the type of answer you want.
* Can handle any input and it will generate a response.
* It can sometimes tell when the question is not related to the lakers.
* It can determine if the answer is relevant or not, if not it doesn’t show it.
* Use machine learning to perform sentiment analysis and predict if it is a question or not.
* Understands that some queries are not in a question format, such as “tell me about the lakers”.

*Weakness:*

* Does not actually understand the conversation occurring.
* Can give unexpected answers that have nothing to related to the question but because some key words were in there, it prints that out.
* Not a lot of human interaction, for example, it lacks the small talk.
* Can sometimes confuse statements as questions
* Can sometimes predict the sentiment incorrectly

*Survey Results:*

1. In a scale from 1 to 5, rate the answers given by the LA-BOT.

Responses: 3,2,3

1. In a scale from 1 to 5, rate the responses given by the LA-BOT.

Reponses: 3, 2, 2

1. In a scale from 1 to 5, how satisfactory are you with LA-BOT.

Responses: 4,3,3

1. In a scale from 1 to 5, rate the understanding of what you were trying to ask to LA-BOT.

Responses: 2,1,2

1. In a scale from 1 to 5, how confident are you this can be used commercially.

Responses: 1,1,1