ADVANCED COMPUTING SESSION

MASTERBOT

Dušan Todosijević

Singidunum University,
Belgrade, Serbia

Abstract:
In this paper the web application will be shown that can classify URL using Naïve Bayes and chatbot which is combination of rule-based chatbot and artificial intelligence (AI). Chatbot can answer to user's queries based on URL corpus. Web application is developed for users regardless of their technical knowledge. The paper presents my solution and opens questions for further development in this scientific field..

Keywords:
Chatbot, Naïve Bayes, cosine similarity, classification, TfidfVectorizer.

1. INTRODUCTION

Technological innovations in the last decade have brought with them changes in almost all aspects of society. The availability of information has been significantly improved, which caused the mentioned changes.

A large part of the company's business has moved to the Internet in terms of sales, advertising and partial or complete provision of services. Today, it can be said that economic entities that are not present on the Internet are almost invisible on the market. Companies that want to be competitive must follow innovations and trends in order to preserve and then improve their position in such a system.

New trends of social networks have emerged, with new platforms, forms and contents every day. The development of social networks has led to new sociological needs of humanity. The strong influence is reflected in the attention caused by posts, comments and every other form of publishing of both celebrity and anonymous personalities, which became sensation almost overnight.

The education system is trying to adapt to the new wave of IT. Technological innovations have created new job positions and scientific fields.

Correspondence:
Dušan Todosijević

e-mail:
dusan88todosijevic@gmail.com

DOI: 10.15308/Sinteza-2020-295-300

Today in Serbia there are already computer science subjects in primary school. Due to the need for education of future employee, various IT programs have emerged in high schools and colleges. In addition to regular education, there is an increasing presence of courses and specialist training.

In order for an ordinary person to be able to keep up with the changes, he/she is forced to spend a good part of his time on the Internet. It can be complicated and confusing for the older generations, especially those who are less able to cope with new technologies, while the younger generations find it difficult to grip.

Inspired by the wishes and difficulties of different generations, in this paper I will try to make a practical solution that will be likable, innovative and able to grip users, while on the other hand I will try to make the usage of this application easy regardless of users technical knowledge.

In search of information, users often type a question or keyword into the search engine, after they open one of the offered websites and finally search the content of the page for information that interest them. This way of searching requires time, unnecessary energy consumption and it's boring.

In this paper, I will present my web application with the symbolic name MasterBot. It was named MasterBot because it is a chatbot and was the subject of my master's thesis. MasterBot is designed precisely as a solution to the before mentioned needs and problems of users.

2. WEB APPLICATION

I created a web application, symbolically named MasterBot, using the 3.8.2 version of the python programming language. The final layout of the web application can be seen in Figure 1.

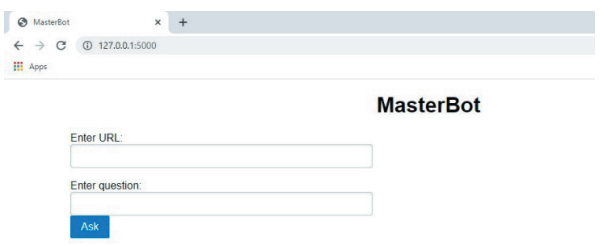


Figure 1. MasterBot

- Frontend was developed in html and css
- Backend was developed in python using flask

MasterBot consists of two basic segments:

1. URL classification
2. Chatbot

First steps:

We expect from user to enter the desired URL and we place it in a variable.

After entering the desired URL, user can make a query to the chatbot, that query is placed in a variable.

It is necessary to use some tool for manipulating text in order for the aplication to be usable. I use the well-known NLTK (Natural Language Toolkit) package. I use packages punkt and wordnet.

When a user enters a URL, we want to use all the text of that URL in the application. To do this, I used the Article library that is applied to the variable in which the URL was placed. The article is downloaded, parsed and nlp content is applied to it.

We're building a corps. These types of objects typically contain raw strings annotated with additional metadata and details [1].

We do tokenization and we make a list of sentences.

3. URL CLASSIFICATION

The URL classification determines which category the content of the website belongs to. The fetch_20newsgroups library was used for classification, which is a frequently used library for various text analyzes, document classifications and similar purposes. The library is composed of a sample of 18,446 articles. It consists of a training and test subset and is divided into 20 categories.

I open a training set that includes a training subset from the library. The training set consists of 11,314 articles.

I open a test set that includes a test subset from the library. The test set consists of 7,532 articles.

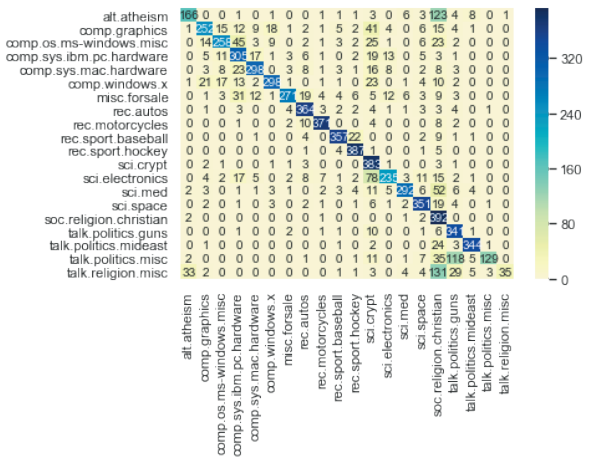
I use pipeline to organize the work so that everything TfidfVectorizer does fits into the basic model of Multinomial Naive Bayes.

TfidfVectorizer simply measures the frequency of a word in a document with the intention of showing how important that word is to the corpus. Conjunctions, auxiliary verbs and in general all the words that appear often will not have the same weight as words that are

not used as often as some professional terms or specific terms from different fields.

In a multinomial naive Bayes the features are assumed to be generated from a simple multinomial distribution. The multinomial distribution describes the probability of observing counts among a number of categories, and thus multinomial naive Bayes is most appropriate for features that represent counts or count rates [2].

I trained the model and made a prediction.



Graph 1. Graphical representation of category predictions

I have graphically shown the success of the prediction in Graph 1. On the abscissa (y-axis) there are the predictions, and on the ordinate (x-axis) there are the true values. Both numerically and visually, the colors evoke the accuracy of the model. The highest values are expected on the diagonal of the square and the color should be dark blue. There are minor deviations where the model inaccurately classifies:

- 'soc.religion.christian' and 'talk.religion.misc',
- 'talk.politics.misc' and 'talk.politics.misc.guns',
- 'soc.religion.christian' and 'alt.atheism'.

Since these categories are very similar we can say that the model is quite successful.

I have defined a function in which can be any string sent, that string is pushed into model pipeline and text classification will be made, so as a result it returns a category.

When user enters the URL, the textual content of that page is downloaded, processed and a corpus is created, which is forwarded to a defined function that performs our classification, that is determine the category to which the URL belongs. The obtained category is displayed with the appropriate message to user.

4. CHATBOT

The chatbot in this paper is a combination of artificial intelligence (AI) and rule-based chatbot. The predicted conversation is in English.

I defined in the variable some of the expected initial messages of users in the form of greetings: "hi", "hello", "greetings", "hey", "howdy", "hey there", "what's up".

Chatbot should respond to this greeting with: "hi", "hey" or "hello".

To make the chatbot a little more intelligent, I made a greeting function. The first thing I wanted was to avoid the trap of users entering in the form of an initial uppercase or lowercase letter, different users have different spelling habits. User input is converted to lowercase. With a simple loop and a logical condition, it is determined whether the entry corresponds to one of the expected greetings. If the greeting is in the list chatbot will respond with a randomly selected greeting from the list.

The next thing that was important to me was to extract all the punctuation marks and make all the text in lower case, which I defined as a function, e.g. if user asks the question: "What is Singidunum University?", it first turns into “what is singidunum university?” than transforms in "what is singidunum university". This will later be used as a tokenizer.

This brings us to the concept of word tokenization, which is simply the process of separating a single string object, usually a body of text of varying length, into individual tokens that represent words or characters that we would like to evaluate further [3].

When a user enters a query which is not predicted it is necessary for the chatbot to give some answer. To make this possible, I defined a response function. The query from user is added to the end of the list of sentences made from the URL. We use TfidfVectorizer to which our defined tokenizer is added and to reduce noise we add "stop_words = 'english'".

What does "stop_words = 'english' actually do? It excludes certain words. NLTK package can exclude words that have little influence on text classification. The words that will be excluded are:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

The idea is for TfidfVectorizer to determine word weights as correctly as possible, and to make the model more accurate. Excluded words are often repeated and have little weight for the entire corpus.

When we convert user's query into vectors, then we measure the similarity of the entered query with the sentences from the URL through cosine similarity. After processing TfidfVectorizer the values can be from 0 to 1, it means that the angle between the vectors can be from 0° to 90°. If the angle is 0°, the value is 1 (cos 0° = 1), which means that the two vectors are identical. If the angle is 90°, the value is 0 (cos 90° = 0), which means that the two vectors have no similarities.

The smaller the angle between the vectors, the greater the similarity of the documents is.

I create a loop that, if the result of cosine similarity is 0, returns a message to user: "I apologize, I don't understand.". In any other case, it returns the most similar sentence, that is sentence with index -2. A sentence with index -1 is actually a user query and it has a similarity of 1, i.e. it is completely identical. I sort the similarity of the sentences so it ranges from the least similar to the most similar, as we said that in the end of list is the query itself, the next most similar sentence is to the index -2 and therefore it is returned to user.

When all the necessary functions have been defined, a loop can be created which actually represents user's communication with application.

If user writes anything except "bye", then a new loop opens, if he writes "bye", then the message will appear "Chat with you later!".

In this new loop, if user enter "thanks" or "thank you", then will appear the message "You're welcome!". In any other case, a different new loop opens.

In this new loop, if user enters any of the defined greetings, he/she will get the response from defined list. If user enters query which is not predicted, application will return the most similar sentence from website or message "I apologize, I don't understand." if cosine similarity is 0.

5. USAGE

MasterBot is developed as an application that will make websites interactive. Instead of reading static content of website with the application, content can become interactive. It is not rare for users to type keywords or questions into a search engine when they need information. Once they find a website with the mention content, it often happens that:

1. The content of the website is large, so users who are less able to search over content can waste a lot of time to extract information that interests them.

2. The page is not related to the area in which they are interested, similar terms are used in different areas.

MasterBot is intended to overcome such difficulties for users.

1. The user does not have to read the content of the page, he/she can ask a specific question. He/she can ask more questions and extract all the information that is potentially of interest to them from a website without too much effort and energy needed for applying to some other method of extracting information. No prior technological knowledge is required to use MasterBot, so users who are less proficient on the Internet can easily use this application.

Example: The current topic, as in previous months, is about COVID-19. In this example, I will present a simulation of a possible user conversation with MasterBot using the URL [access date 09.06.2020.]:

https://www.aljazeera.com/news/2020/01/coronavirus-symptoms-vaccines-risks-200122194509687.html

Figure 2. Example 1

Figure 3. Example 2

Figure 4. Example 3

Figure 5. Example 4

In the figures above we can see the answers of MasterBot from which we can conclude that MasterBot has answered with appropriate sentence from the website to each question. Using chatbot the user quickly has come to the desired information about COVID-19.

2. By classifying the URL, the user can see which category a website belongs to. Classification will save user's time and effort which would be needed to infer from the context of the website to determine category. If the page does not belong to the category that interests them, he/she can quickly continue the search on the Internet.

Example: I entered "danger operation" in the search engine, as keywords that potentially interest the user. On the first page I got websites with different contents.

First URL [access date 09.06.2020.]:

https://www.scientificamerican.com/article/hidden-dangers-of-going-under/

Figure 6. Example 5

The content of the website refers to the dangers that anesthesia can cause. The classification of this URL is medicine, which corresponds to its content.

Second URL [access date 09.06.2020.]:

https://www.asahq.org/whensecondscount/anesthesia-101/anesthesia-risks/

Figure 7. Example 6

The content of the website also refers to the dangers that anesthesia can cause. The classification of this URL is medicine, which corresponds to its content.

Third URL [access date 09.06.2020.]:

https://en.wikipedia.org/wiki/Able_Danger

Figure 8. Example 7

The content of the page refers to the military operation "Able Danger" aimed at the fight against terrorism by the institutions in the USA. The classification of this URL is a policits-guns that corresponds to its content.

The user could enter "danger operation" in the search engine, with the intention of obtaining content related to a military-political dangerous operation. By classifying the URL, the first two pages related to medicine would be recognized by MasterBot and a message would be printed to the user that the category is medicine. The user could immediately see that it is not the content of interest to him and he/she could continue the search further and come to the third website that potentially is interesting to them.

6. CONCLUSION

In this paper, I presented my solution for the needs and problems of users, which I explained in the introduction. More importantly I pointed out the potential of modifying the way information is obtained. Although I have presented a successful application in the paper, the application has some limitations. Some of the limitations are:

- The intended language of the application is English. The application can be used in other languages, but with much less success.
- There is a limit to the classification of URL to twenty categories, because this is the maximum of the library used in the paper.
- Classification can miss label, i.e. return wrong category, especially for websites that have the theme of religion and politics.
- If the selected website does not have information that user is interested in, only the appropriate message will be displayed: "I apologize, I don't understand.".

These limitations can be overcome, but it requires a lot of knowledge and effort, also for each individual it would be an endeavor.

The project can go a step further, my vision for further improvement of the project is reflected in several aspects:

- The application should not be limited to one page, but should have access to all content on the Internet.
- When user enters a query, he/she should not receive an answer based on cosine similarity of the documents, but much more sophisticated techniques and methods of machine learning should be applied.
- URL classification was just idea of showing the category as a user guide about the content of the page. This idea arose due to the use of similar terms in different areas. In a more advanced application, after user makes a query, a chatbot subquestion could be asked in order to profile user's query, i.e. to determine a category.

Undoubtedly, text mining will become increasingly important. The paper presents my solution and opens questions for further development in this scientific field. This paper is a solid starting point for further improvement of chatbots, all with the aim of obtaining a better and more interesting application, but also easier to use.

REFERENCES

[1] Julia Silge, David Robinson, Text Mining with R, O'REILLY, 2017., pp. 2

[2] Jake VanderPlas, Python Data Science Handbook - Essential Tools for Working with Data, O'Reilly, 2017., pp. 386

[3] Taweh Beysolow II, Applied Natural Language Processing with Python - Implementing Machine Learning and Deep Learning Algorithms for Natural Language Processing, Apress, 2018., pp. 44-45