Downloading and saving data

Download data from the link and save it in a folder StackOverflow Bot in Desktop

https://github.com/MLWhiz/data_science_blogs/tree/master/chatbot

(download .ipynb and .py files from raw and save it by manually typing the extension)

Create a GitHub repository for our project directory

1. Download git software from https://git-scm.com/download/win

2. Create a new repository in GitHub named StackOverflow-Bot.

3. Open Git Bash.

4. Change the current working directory to your local project.
   (cd Desktop/ StackOverflow Bot)

5. Initialize the local directory as a Git repository. ($ git init -b main)

6. Add the files in your new local repository. This stages them for the first commit.
   ($ git add .)

7. Commit the files that you've staged in your local repository.
   (git commit -m "First commit")

8. Set the new remote by adding the URL for the remote repository where your local repository will be pushed.
   ($ git remote add origin https://github.com/arun5461/StackOverflow-Bot.git)

9. Verify the new remote URL. (git remote –v)

10. Push the changes in your local repository to GitHub. ($ git push origin main)

https://docs.github.com/en/github/importing-your-projects-to-github/adding-an-existing-project-to-github-using-the-command-line#

Telegram setup

1. Download and Install the Telegram App on your Laptop from https://desktop.telegram.org/

2. Talk with BotFather by opening this https://telegram.me/BotFather in Chrome and subsequently in your Telegram App that will take you to a Chatbot called Botfather which can help you create a new bot.

3. Give "/start " and set up a new bot using command and "/newbot"

4. Create a name for Your bot.

5. Create a username for your bot 'ArunStackBot'.

6. You will get an access token for the bot. Copy the Token at a safe place.

7. Click on the "t.me/ ArunStackBot" link to open Chat with your chatbot in a new window.

8. Run main.py in atom (command line) to make our Chatbot communicate with Telegram using the Access token.

Use Google Colaboratory to clone a GitHub Repository to your Google Drive

1. Open notebook as GitHub url by entering the link

   https://github.com/arun5461/StackOverflow-Bot

2. Connect to a GPU runtime. Go to the 'Runtime' menu, select the 'Change Runtime Type' option. Select the 'GPU' option under the 'hardware accelerator' field in the dialog box that appears and click 'save'.

3. Mount your google drive in to this runtime. Type in the first cell of your colab and run the cell.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

This would prompt a URL with an authentication code. After you insert that authentication code in the provided space, your google drive will be mounted.

4. Check the contents of the current folder. (! ls)

If the drive is mounted correcly, you would see that the current folder has a directory called 'gdrive'. This is where you can find your google drive contents.

5. Access the drive to create the project folder. (%cd gdrive/My Drive)

6. Clone our GitHub repository in this folder.

   (! git clone https://github.com/arun5461/StackOverflow-Bot)

7. Change directory to

(%cd StackOverflow-Bot)

https://medium.com/@ashwindesilva/how-to-use-google-colaboratory-to-clone-a-github-repository-e07cf8d3d22b

Download GoogleNews-vectors from here

https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

(Download and paste it in the same folder as the code main.py and upload it in Google drive in StackOverflow Bot folder)

Download the resources folder from Google drive and paste it in the same folder as the code main.py

Making main.py work

Install python if not already available - https://www.python.org/downloads/

python --version – to check python version

To install ChatterBot python library, we need a 3.4 < = python version < = 3.8.

To setup-a-virtual-environment-with-a-different-python-version

https://medium.com/swlh/how-to-setup-a-virtual-environment-with-a-different-python-version-windows-4876738e7aa0

(If it doesn't work, uninstall the previous version and install the required one)

Install pip, if not already available - https://bootstrap.pypa.io/get-pip.py

Save it in a folder and run the code

(To check pip version)

pip --version

python -m pip --version – (If installed for a different python version, to check pip version for that python version)

(Install the required libraries)

pip install chatterbot

pip install gensim

pip install pickle-mixin

pip install nltk

pip install requests

pip install -U scikit-learn

pip install chatterbot-corpus

(Optional – in case needed)

pip install pytz

pip3 uninstall PyCrypto

pip3 install -U PyCryptodome

(Include this in code)

import nltk

nltk.download('stopwords')

python sqlite3 OperationalError: attempt to write a readonly database

- Change ownership of the folder where the database is being written

https://www.laptopmag.com/articles/take-ownership-folder-windows-10-using-file-explorer

ChatterBot error- OSError: [E941] Can't find model 'en'

https://stackoverflow.com/questions/66087475/chatterbot-error-oserror-e941-cant-find-model-en

Unable to allocate array with shape and data type

https://stackoverflow.com/questions/57507832/unable-to-allocate-array-with-shape-and-data-type#:~:text=A%20better%20solution%20is%20to%20adding%20more%20memory%20to%20the%20computer.&text=I%20came%20across%20this%20problem,RAM%20(2%5E32).

Use your own access token of the bot created in place of token.

Understanding the code

Model_Creation.ipynb

tagged_posts.tsv — StackOverflow posts, tagged with one programming language (positive samples).

dialogues.tsv — dialogue phrases from movie subtitles (negative samples).

pandas - used for data analysis, allows importing data from various file formats such as .csv etc.

re (REGular EXpression) - to check if a particular string matches a given regular expression (pattern).

NLTK (Natural Language ToolKit) - a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language

nltk.corpus - creates a set of corpus reader instances that can be used to access the corpora in the NLTK data package

corpus - large and structured set of machine-readable texts

nltk.corpus.stopwords() – a collection to access stopwords in many languages

stopwords - the English words which do add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. (e.g. ‘a’, ‘an’, ‘his’)

pickle - used in serializing and deserializing a Python object structure.

Serialization/Pickling - converting a Python object into byte streams (0s and 1s) to store it in a file/database, maintain program state across sessions, or transport data over the network.

Deserialization/Unpickling - converting the byte stream (generated through pickling) back into python objects.

pd.read_csv - to read csv files into a dataframe (2-dimensional labeled data structure with columns of potentially different types) and do operations on it (sep – delimiter used).

dataframe.head() – returns the first 5 rows.

len(dataframe) – to get the number of rows.

list(dialogues[:200000].text.values) – list of first 200000 values of text column in dialogues dataframe.

list(posts[:200000].title.values) - list of first 200000 values of title column in posts dataframe.

texts = [Okay -- you're gonna need to learn how to lie, ……, Calculate age in C#,…….]

['dialogue']*200000 – a list of 200000 elements containing ‘dialogue’.

['stackoverflow']*200000 - – a list of 200000 elements containing ‘stackoverflow’

labels = [‘dialogue’, ‘dialogue’, ‘dialogue’, ….., ‘stackoverflow’, ‘stackoverflow’, ‘stackoverflow’, …….]

pd.dataframe() – creates a dataframe (rows and columns) with the given lists.

text_prepare() – function that performs tokenization and pre-processing.

re.compile(pattern) - Compile a regular expression pattern into a regular expression object, which can be used for pattern matching using its search(), findall() and other methods.

```
s = 'AK says AK is the don’
pattern=re.compile('AK')
pattern.findall(s)  // [‘AK’, ‘AK’]
t= 'I am AK’
pattern.findall(t)  //[‘AK’]
```

instead of

```
s = 'AK says AK is the don’
re.findall(‘AK’, s) // [‘AK’, ‘AK’]
t= 'I am AK’
re.findall(‘TP’, t) //[‘AK’]
```

stopwords.words('english') – list of stopwords in English.
['i', 'me', 'my', 'myself',.....,]

set(stopwords.words('english')) – set of stopwords in English (unordered and non-repeating).

text = ‘if Arun is {$7}’

text.lower() - returns the lowercased string from the given string.

text = ‘if arun is {$7}’

re.sub() - to replace occurrences of a particular sub-string with another sub-string.

replace_by_space_re.sub(' ', text)- replace those symbols with ‘ ‘ (whitespace) in text.

text = ‘if Arun is $7 ’

bad_symbols_re.sub('', text) - remove other than those symbols in text.

text = ‘if Arun is 7 ’

join() - takes all items in an iterable and joins them into one string.

‘-‘.join[‘1’, ‘2’, ‘3’] = 1-2-3

text.split() – splits a string into a list (of words)

‘Arun 7 ’

text.strip() – removes the spaces at the beginning and end

returns ‘Arun 7’

lambda function - a single-line function declared with no name, which can have any number of arguments, but it can only have one expression.

apply() function calls the lambda function and applies it to every row or column of the dataframe and returns a modified copy of the dataframe

data['text'] = data['text'].apply(lambda x : text_prepare(x))
applies text_prepare function on every text element in dataframe

train_test_split(X, y, test_size, random_state) - a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data (with given test_size) and returns four sequences (NumPy arrays) in this order :
X_train, X_test, y_train, y_test.

The training subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

random_state - used for reproducing your problem the same every time it is run (when set to 0). If you do not use a random_state in train_test_split, every time you make the split you might get a different set of train and test data points and will not help you in debugging in case you get an issue.

mkdir – to create new directories

sklearn.feature_extraction.text.TfidfVectorizer - Convert a collection of raw documents to a matrix of TF-IDF features.

dtype - Type of the matrix returned by fit_transform() or transform().

min_df – to remove terms that appear so frequently

max_df – to remove terms that appear so frequently

(float value – proportion of documents, int value – number of documents)

max_features - If not None, build a vocabulary that only consider the top max_features ordered by term frequency across the corpus. This parameter is ignored if vocabulary is not None.

strip_accents = 'unicode' - replace all accented unicode char by their corresponding ASCII char (performing normalization – base form).

analyzer='word' - Whether the feature should be made of word or character n-grams, here word.

token_pattern=r'\w{1,}' - Regular expression denoting what constitutes a "token", only used if analyzer == 'word'.

\w (word character) - matches any single letter, number or underscore (same as [a-z A-Z 0-9_])

{1,} - sequences of length one or more

ngram_range = (1,3) - The lower and upper boundary of the range of n-values for different n-grams to be extracted. Here, 1-gram, 2-gram and 3-gram can be extracted

use_idf=1 - Enables inverse-document-frequency reweighting. While computing TF, all terms are considered equally important. IDF weighs down the frequent terms while scaling up the rare ones.

smooth_idf=1 - Weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.

sublinear_tf=1 - Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).
(It seems unlikely that twenty occurrences of a term in a document truly carry twenty times the significance of a single occurrence, so to normalise the bias of "where a term that is X times more frequent shouldn't be X times as important").

stop_words = 'english' – to remove stop words from the resulting tokens.

To center the data (make it have zero mean and unit standard error), you subtract the mean and then divide the result by the standard deviation:

$$x' = \frac{x - \mu}{\sigma}$$

fit(): to calculate the mean and variance of each of the features present in our data.

transform(): to transform all the features using the respective mean and variance.

tfv.fit_transform(): used on the training data so that we can scale the training data and also learn the scaling parameters (mean and vaiance) of that data.

tfv.transform() - to use the same mean and variance as it is calculated from our training data to transform our test data.

pickle.dump(object, file) – the object needed to be pickled and the file to which the object has to be saved

open(file,'wb') – writing to the file in binary mode

X_train_tfidf/X_test_tfidf - (document_id, token_id) tfidf value

with open(filename,'rb') as f – opens the file in read binary mode with f as the file handle/object.

pickle.load() - to load pickled data from a bytes string

Linear Regression – used for regression problems (to predict continuous valued output).

Logistic Regression – used for classification problems (to predict disrete valued output)..

LogisticRegression(C, random_state, solver) - implements logistic regression using liblinear, newton-cg, sag of lbfgs optimizer.

> C - Inverse of regularization strength; must be a positive float. Regularization is applying a penalty to increasing the magnitude of parameter values in order to reduce overfitting.
>
> random_state - represents the seed of the pseudo random number generated which is used while shuffling the data (seed is to generate pseudo random number).
>
> solver - Algorithm to use in the optimization problem for fitting the model.

intent_recognizer – Logistic Regression model

intent_recognizer.fit(X,Y) – the model is fit (trained).

> Model fitting - the process of determining the coefficients that correspond to the best value of the cost function.

intent_recognizer.predict(X_test)- predict the values (classes – 0/1)
for the test set by using the trained model.

accuracy_score(y_true, y_pred) - return a float value that describes (number of points classified correctly) / (total number of points in your test set)

posts['title'].values – list of title in Posts

posts['tag'].values – list of tag in Posts

X= ['Calculate age in C#', 'Filling a DataSet or DataTable from a LINQ query result set', ..., 'Can I safely convert struct of floats into float array in C++?']

y= ['c#' 'c#' 'c#' ... 'javascript' 'javascript' 'c_cpp']

OneVsRestClassification – for multi-class classification where there are more than two classes by splitting into many different binary classification problems (one vs rest)
(for n classes, n models are required)

OneVsRestClassifier(LogisticRegression) – Using Logistic Regression, a binary classification algorithm with the One-vs-Rest Classifier to perform multi-class classification.

tag_classifier - Logistic Regression model for One-vs-Rest Classifier to perform multi-class classification.

gensim.models.KeyedVectors.load_word2vec_format – to load a pre-trained word2vec model

GoogleNews-vectors-negative300.bin - It's a pre-trained word2vec model by google for sentiment analysis.

groupby - A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

by - mapping, function, label, or list of labels. Used to determine the groups for the groupby.

		count
Python	Qn	2
Python	Qn	
.		
.		
Java	Qn	3
Java	Qn	
Java	Qn	

enumerate() - The enumerate() method adds counter to an iterable and returns it (the enumerate object).

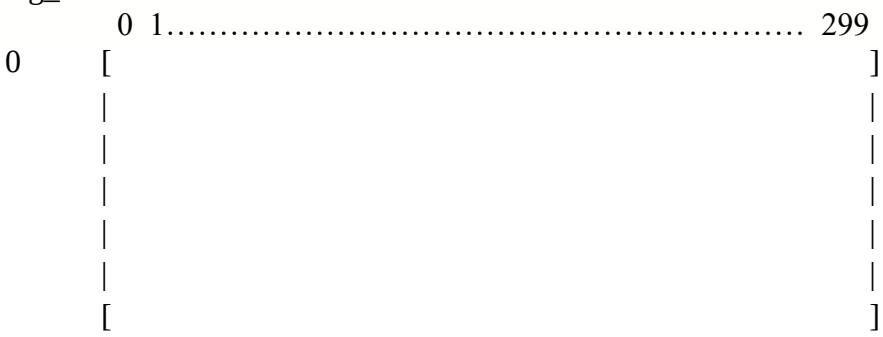
Syntax : enumerate(iterable, start=0)

e.g.

grocery = ['bread', 'milk', 'butter']

list(enumerate(grocery)) => [(0, 'bread'), (1, 'milk'), (2, 'butter')]

tag_vectors

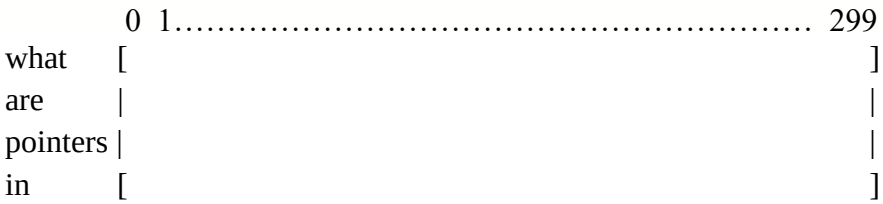


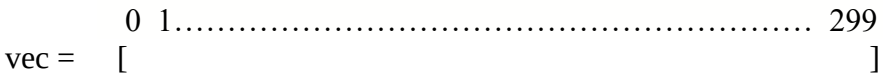
question_mat (For qn ‘what are pointers in C++’)

np.all - Test whether all array elements along a given axis evaluate to True.

~ - other than these

(rows => axis = 1, columns => axis = 0)





pairwise_distances_argmin - Compute minimum distances between one point and a set of points.This function computes for each row in X, the index of the row of Y which is closest (according to the specified distance).

main.py

chatterbot - a Python library that makes it easy to generate automated responses to a user's input

chatbot – to create a chatbot instance

ChatterBotCorpusTrainer - Allows the chat bot to be trained using data from the ChatterBot dialog corpus (chatterbot.corpus.english)

offset - refers to the update_id (or message_id?)

requests – a python library to make HTTP requests simpler and more human-friendly

BotHandler class implements all back-end of the bot using Telegram's API. It has three main functions:

- get_updates - checks for new messages sent by the user.
- get_answer - computes the most relevant answer to a user's question using a SimpleDialogueManager class.
- send_message – posts the answer computed as a new message to the user.

SimpleDialogueManager class is where we will write our bot logic that fits the pieces together to build one wholesome logic.

- All the models and TFIDF objects are instantiated (.pkl files)
- A chatbot is instantiated using chatterbot and trained on the provided English corpus data for chit-chat type questions.
- get_similar_question function - given the question and the question's programming language (tag), loads the particular tag's post_ids and post_embeddings from the .pkl file, converts the question to a vector, and computes the minimum distance between this vector and post_embeddings (set of vectors) to find post id of the most similar question in the dataset.
- generate_answer function - transforms the question using the loaded tfidf_vectorizer and determines the intent of the question. For a dialogue question, it generates a response using chatterbot, and for a programming question, it finds the tag

(programming language) and using get_similar_question function generates the thread (StackOverflow link) of the question that is the most similar as a response.

Doubts

Model_Creation.ipynb

Num Posts:len(posts)

Solver to be used in Logistic Regression

https://github.com/arun5461/StackOverflow-Bot

https://drive.google.com/drive/u/0/folders/15wQdunmsy7j9ZcoUl8092GZwVCGnwIsp

https://docs.github.com/en/github/managing-large-files/configuring-git-large-file-storage