A Project Report on

“Why choose Vegan or Paleo?”

By

Aishna Agrawal
Avani Arora
Sanjana Woonna

Guided by

Suresh K Lodha

Department of Computer Science
University of California, Santa Cruz

I. Data Wrangling:

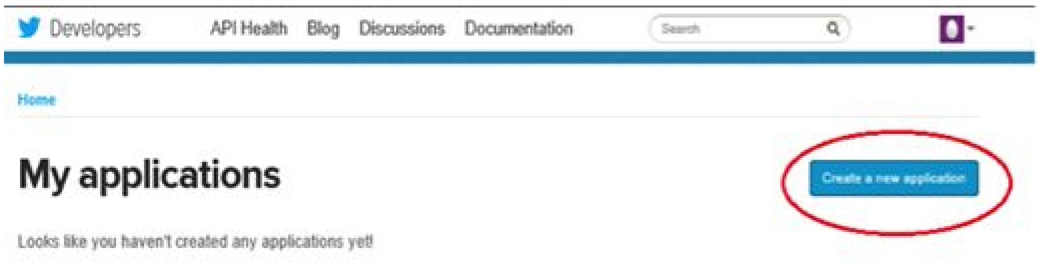
A. Data Extraction from Twitter's REST API:

Prerequisites:

1. In order to extract tweets, you will need a Twitter application and hence a Twitter account.
2. If you don't have a Twitter account, please sign up.
3. Use your Twitter login ID and password to sign in at Twitter Developers.

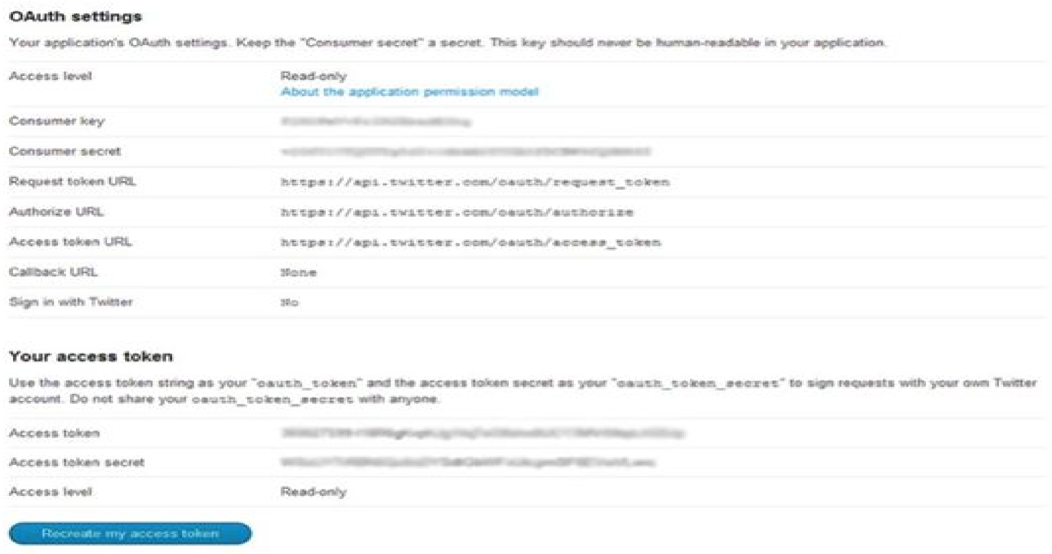
Step 1: Create a twitter application

- Navigate to My Applications in the upper right hand corner.
- Create a new application.



My applications

Looks like you haven't created any applications yet!



OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read-only About the application permission model
Consumer key	[redacted]
Consumer secret	[redacted]
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token
Callback URL	None
Sign in with Twitter	No

Your access token

Use the access token string as your "oauth_token" and the access token secret as your "oauth_token_secret" to sign requests with your own Twitter account. Do not share your oauth_token_secret with anyone.

Access token	[redacted]
Access token secret	[redacted]
Access level	Read-only

- Fill out the new app form. Names should be unique, i.e., no one else should have used this name for their Twitter app.
- Give a brief description of the app.
- Enter your website or blog address.
- Callback URL can be left blank.
- Once you've done this, make sure you've read the "Developer Rules Of The Road" blurb, check the "Yes, I agree" box, fill in the CAPTCHA and click the "Create Your Twitter Application" button.
- Click on "Create my access token" button.
- Note the values of consumer key and consumer secret key and keep them handy for future use. You should keep these secret.

Step 2: Install Python Packages

Packages to be installed: json, twitter

Code: *pip install <name_of_the_library>*

Step 3: Data Extraction

- After installing the required packages, run the following code using the credentials provided by Twitter.(Refer **DataExtraction.py).**
- The hashtags used to extract data for Vegan are **#vegan, #veganlifestyle**, **#vegandiet**, **#vegano,#veganforfood** and for Paleo are **#paleo, #paleodiet, #paleolifestyle, #cavemandiet, #whole30, #primal, #grainfree, #paleolithiclifestyle.**

Step 4: Save and Load the data

- Save the data extracted in form of json file using the following command in the command prompt. First navigate to the folder where the data extraction code resides and then type the following command.

  python DataExtraction.py > SampleDataFile.json

  For data, refer folders named **Vegan** and **Paleo.**
- The packages required are **json, os** and **pandas.** Initially we had multiple data files which were read and the data was combined into a list object.

Step 5: Pre-processing the data:

Packages: nltk

Steps:

- Remove repeated tweets.
- Convert the text to a uniform case(like upper or lower). In our project we have converted the tweets to lower case.
- Remove the unicode characters since they do not help in the process of mining.
- Remove the urls , newline characters, '#', 'rt @'(term to indicate retweet), extra spaces and punctuation.
- Remove the stop-words like pronouns, prepositions etc and a

Ref : **Paleo: MiningPaleoTweets.py, Vegan: MiningVeganTweets.py**

II. Data Analysis:

- We used an incremental model to develop our project.
- The models are mentioned below:
    a. **Stanford Dependency Parser:** We used this parser to find dependencies of head words like Vegan and Paleo on other words to find reasons. But the parser did not identify the dependencies on Vegan and Paleo that could help us to find the reasons.
    b. **Hierarchical Clustering(Unigrams and Bigrams):** We created a corpus of unigrams and bigrams and measured the Jaccard Distance between all the words to find similar words . But the similarity was not based on semantics , hence we did not go forward with this model.
    c. **Word2Vec**(Deep learning): This model uses neural networks to build dictionary using the words from the tweets and clubs similar meaning words. However, the drawback is that it requires enormous amount of data which in our case was not enough.
    d. **PyDictionary with Hash Tag Splitter:** This model uses an inbuilt dictionary called PyDictionary which is based on WordNet. We used the model to find the synonyms for each word in the corpus of tweets and later, replaced similar meaning words with a single word. We also used 'Hash tag splitter' to split the hash tags into meaningful words. This process was not time efficient(eg: for 1000 tweets it took around 1 hour).

e. **Latent Dirichlet Allocation (LDA):** This model performs topic modelling i.e. it find the topics that best represent the documents. In our case we found redundancy and irrelevancy of topics.

f. **Word2Vec(PreTrained):**
   - This model includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features.
   - We used the pre-trained model to find the synonyms for each word using the **model.similarity()** with a threshold of 0.85 to replace similar meaning words with a single word.
   - However if the word is not present in the dictionary we ignore it.
   - We refined it further manually by using the word counter for similar words .

You can download Google's pre-trained model here. The packages required are **gensim, gensim.models, collections(Paleo: MiningPaleoTweets.py, Vegan: MiningVeganTweets.py)**

III. Data Visualisation:

Packages: wordcloud, PIL
Software: Numbers

- We created a masked wordcloud to represent the relative frequencies between the words in the corpus using the word counter.
- We masked the Vegan corpus as a **leaf** and the Paleo corpus as a **caveman** to represent the wordcloud.
- We created barcharts for both Vegan and Paleo as well as a stacked barchart to differentiate between them using the software **Numbers**

References:

1. http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/
2. https://github.com/amueller/word_cloud
3. https://nlp.stanford.edu/software/lex-parser.shtml
4. https://radimrehurek.com/gensim/models/word2vec.html

5. https://pypi.python.org/pypi/hcluster/0.2.0
6. https://pypi.python.org/pypi/PyDictionary/1.3.4
7. https://github.com/matchado/HashTagSplitter
8. http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/