# Categorization of Web Pages using Text Classification Methods.

Ashish Mohapatra
Department of Computer Science
The University of Texas at Dallas
axm160031@utdallas.edu

***Abstract*—This project aims to come up with a system to categorize Web pages based on the content available. I have used Wikipedia API to categorize the Wikipedia web pages on 5 different topics likes Business, Politics, Sports, Technology or Travel. This project would allow a user to get the correct categorization of a page without having to manually find it.**

***Index Terms*—Wikipedia, NLTK, Hypernyms, Meronyms, Chunking, Stopwords, Tokenizer**

## I. Introduction

The internet has a lot of web pages and these web pages are made up of a lot of different content that it is not easy to categorize them under different topics. To do this, a person has to manually index and categorize the web page based on the topic that the page is dealing with. The problem is with the current size of the web, it has become cumbersome to try and manually index and categorize its content. For example, a page on Football Club Barcelona can be broadly classified under the topic Sports. But, to label a page, the user has to read the contents and decide the label later on.

## II. Proposed Approach

I have created a manually tagged dataset that broadly classifies a feature word into one of the five categories namely - Business, Politics, Sports, Technology or Travel.
Initially, I went with a simple classification system using Wikipedia API and later followed it up with a more improved strategy that makes use of Natural Language Processing features in lexical, syntactic and semantic terms and Machine Learnings Nave Bayes Classification.

Text classification has traditionally been one of the most popular problems in information retrieval, natural language processing, and machine learning. In the simplest case, the task of text classification [1] is as follows: A set of training documents T = X1, X2, ...Xm, each labeled with a class value from a set of k distinct labels, from the set 1, 2, ..k, is used to learn a classification model, that captures the relationship between features in the documents and their labels.

## III. Programming Tools

- Python.
- Pycharm IDE
- Natural Language Toolkit (NLTK)
- Wikipedia API for Python.

## IV. Implementation : Baseline

In the first program, I have used Wikipedia API to obtain the corpus of the feature word and then classify the category based on the frequency of the Category term occurring in the corpus. If the term with the highest frequency matched with the Tagged Category we have in the dataset, the feature word in classified correctly else it will be updated with the obtained category from the corpus parsing.
Example: If in the Database, Barcelona is tagged as Category Sports, but when we parse the corpus of Barcelona obtained from Wikipedia we see that the term Travel occurs more time than Sports then the term Barcelona will be categorized as Travel.

## V. Improved Strategy

Once the baseline system has been computed, I improved the classification system by performing

additional operations by adding lexical, syntactic, semantic features to the corpuses associated with each feature word.

### A. Lexical Feature.

I used the following two lexical processes by making use of the functions in the Natural Language Toolkit

1) Tokenization : Tokenization is the method of breaking the text into words, phrases and other meaningful elements. Tokenization of the text allows us to use the text for further processing and parsing.
   I tokenized the corpus obtained from Wikipedia for the specific feature word. Tokenization was performed using the NLTK tokenize package (nltk.tokenize package).

```
tokens=word_tokenize(wiki_content)
```

2) Removing stopwords : Following tokenization of the words, all the stop words were removed from the word list. To acquire this I used stopwords() function from the nltk.corpus library

```
stop=set(stopwords.words('english'))
```

3) Lemmatization : Lemmatization is the process of grouping together the different forms of the same word so they can be analyzed as a single element and identified by dictionaries. The goal of lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:
   *am, are, is =¿ be*
   *car, cars, car's, cars' =¿ car*
   The result of this mapping of text will be something like:
   *the boy's cars are different colors =¿ the boy car be differ color*
   For lemmatization, I used the WordNetLemmatizer from the Python library NLTK.stem for textual processing.

```
lemmatizer = WordNetLemmatizer()
```

### B. Syntactic features.

1) POS Tagging : Following Lemmatization of the words, I went for the pos_tag function of the NLTK library. Using the function, I was able to get POS tagging for each of the lemmatized words.

```
list = pos_tag(lemmatized_tokens)
```

2) Chunking of the POS Tagged List: Following tagging of the words. I went to chunking. Chunking helps in Returning the best chunk structure for the given tokens and return a tree.

3) Shallow Parsing of the chunked Tree: I employed shallow parsing since the only parts of speech in the corpus I was interested in are the nouns. For the Shallow Parser, I used specific grammar rules that allow the program to choose only parts of the text that conforms to the given grammar rules. I use the RegExParser on NLTK to produce shallow parsed output.

### C. Semantic features.

I choose using hypernyms and meronyms for getting the semantic features from the processed text. I made use of the NLTK Wordnet API for semantic features. Wordnet is the lexical database for the English language. I used the hypernyms and Meronyms from the NLTK Wordnet features.

1) Hypernyms A hypernym is a word which includes the meanings of other words may contain more specific words that fall under it. For example, *the word Technology is a hypernym for any technology related term including Bluetooth and Wi-Fi.*

2) Meronyms: Meronymy is used to describe a part-whole relationship between lexical items. Thus *cover and page are meronyms of a book.*

### D. Naive Bayes Classification.

The output from the above processes, specifically the meronym features gives us a list of words which are closely related to the topics we have to classify. This is given as input to the Nave Bayes Classifier and is used to create a training model. The model is further used to test the reasonably large test dataset and classify the results there. I stored the model as a pickle file so that it can be used later for faster testing of datasets.

## VI. Outputs

```
Tokenizing Words :
['uber', 'technologies', 'inc.', 'is', 'a', 'global',
'transportation', 'technology', 'company',
'headquartered', 'in', 'san', 'francisco', ',',
'california', ',', 'united', 'states', ',', 'operating',
'in', '633', 'cities', 'worldwide', '.', 'it', 'develops',
',', 'markets', 'and', 'operates', 'the', 'uber', 'car',
'transportation', 'and', 'food', 'delivery', 'mobile',
'apps', '.', 'uber', 'drivers', 'use', 'their', 'own',
'cars', 'although', 'drivers', 'can', 'rent', 'a', 'car',
'to', 'drive', 'with', 'uber', '.', 'the', 'name', '``',
'uber', "'", 'is', 'a', 'reference', 'to', 'the',
'common', '(', 'and', 'somewhat', 'slangy', ')', 'word',
'``', 'uber', "'", ',', 'meaning', '``', 'topmost', "'",
'or', '``', 'super', "'", ',', 'and', 'having', 'its',
'origins', 'in', 'the', 'german', 'word', 'über', ',',
'meaning', '``', 'above', "'", '.', 'uber', 'has',
'been', 'a', 'pioneer', 'in', 'the', 'sharing', 'economy',
'and', 'the', 'changes', 'in', 'industries', 'as', 'a',
'result', 'of', 'the', 'sharing', 'economy', 'have',
'been', 'referred', 'to', 'as', '``', 'uberification',
"'", 'or', '``', 'uberisation', "'", '.', 'uber', 'has',
'also', 'been', 'the', 'subject', 'of', 'protests', 'and',
'legal', 'actions', '.']
```

Fig. 1: Tokenization the corpus

```
After Removing stopwords :
['uber', 'technologies', 'inc.', 'global',
'transportation', 'technology', 'company',
'headquartered', 'san', 'francisco', 'california',
'united', 'states', 'operating', '633', 'cities',
'worldwide', 'develops', 'markets', 'operates', 'uber',
'car', 'transportation', 'food', 'delivery', 'mobile',
'apps', 'uber', 'drivers', 'use', 'cars', 'although',
'drivers', 'rent', 'car', 'drive', 'uber', 'name', 'uber',
'reference', 'common', 'somewhat', 'slangy', 'word',
'uber', 'meaning', 'topmost', 'super', 'origins',
'german', 'word', 'über', 'meaning', 'uber', 'pioneer',
'sharing', 'economy', 'changes', 'industries', 'result',
'sharing', 'economy', 'referred', 'uberification',
'uberisation', 'uber', 'also', 'subject', 'protests',
'legal', 'actions']
```

Fig. 2: After Removing Stopwords

```
Lemmatization :
['uber', 'technology', 'inc.', 'global', 'transportation',
'technology', 'company', 'headquartered', 'san',
'francisco', 'california', 'united', 'state', 'operating',
'633', 'city', 'worldwide', 'develops', 'market',
'operates', 'uber', 'car', 'transportation', 'food',
'delivery', 'mobile', 'apps', 'uber', 'driver', 'use',
'car', 'although', 'driver', 'rent', 'car', 'drive',
'uber', 'name', 'uber', 'reference', 'common', 'somewhat',
'slangy', 'word', 'uber', 'meaning', 'topmost', 'super',
'origin', 'german', 'word', 'über', 'meaning', 'uber',
'pioneer', 'sharing', 'economy', 'change', 'industry',
'result', 'sharing', 'economy', 'referred',
'uberification', 'uberisation', 'uber', 'also', 'subject',
'protest', 'legal', 'action']
```

Fig. 3: Lemmatization of Corpus Words

```
POS Tagging :
[('uber', 'NNP'), ('technology', 'NN'), ('inc.', 'NN'),
('global', 'JJ'), ('transportation', 'NN'), ('technology',
'NN'), ('company', 'NN'), ('headquartered', 'VBD'),
('san', 'JJ'), ('francisco', 'JJ'), ('california', 'NN'),
('united', 'JJ'), ('state', 'NN'), ('operating', 'VBG'),
('633', 'CD'), ('city', 'NN'), ('worldwide', 'NN'),
('develops', 'VBZ'), ('market', 'NN'), ('operates',
'VBZ'), ('uber', 'JJ'), ('car', 'NN'), ('transportation',
'NN'), ('food', 'NN'), ('delivery', 'NN'), ('mobile',
'NN'), ('apps', 'IN'), ('uber', 'JJ'), ('driver', 'NN'),
('use', 'NN'), ('car', 'NN'), ('although', 'IN'),
('driver', 'NN'), ('rent', 'NN'), ('car', 'NN'), ('drive',
'NN'), ('uber', 'NN'), ('name', 'NN'), ('uber', 'JJ'),
('reference', 'NN'), ('common', 'JJ'), ('somewhat', 'RB'),
('slangy', 'JJ'), ('word', 'NN'), ('uber', 'IN'),
('meaning', 'VBG'), ('topmost', 'NN'), ('super', 'NN'),
('origin', 'JJ'), ('german', 'JJ'), ('word', 'NN'),
('über', 'NNP'), ('meaning', 'NN'), ('uber', 'NNP'),
('pioneer', 'NN'), ('sharing', 'VBG'), ('economy', 'NN'),
('change', 'NN'), ('industry', 'NN'), ('result', 'NN'),
('sharing', 'VBG'), ('economy', 'NN'), ('referred',
'VBD'), ('uberification', 'JJ'), ('uberisation', 'NN'),
('uber', 'NNP'), ('also', 'RB'), ('subject', 'JJ'),
('protest', 'JJ'), ('legal', 'JJ'), ('action', 'NN')]
```

Fig. 4: POS Tagging of Lemmatized words

```
Chunked Tree:
(S
  (NP uber/NNP)
  (NP technology/NN)
  (NP inc./NN)
  (NP global/JJ transportation/NN)
  (NP technology/NN)
  (NP company/NN)
  headquartered/VBD
  (NP san/JJ francisco/JJ california/NN)
```

Fig. 5: Shallow Parsing of Chunked Tree

```
Hypernym :
['uber', 'technology', 'inc.', 'transportation',
'technology', 'company', 'california', 'state', 'city',
'worldwide', 'market', 'car', 'transportation', 'food',
'delivery', 'mobile', 'driver', 'use', 'car', 'driver',
'rent', 'car', 'drive', 'uber', 'name', 'reference',
'word', 'topmost', 'super', 'word', 'über', 'meaning',
'uber', 'pioneer', 'economy', 'change', 'industry',
'result', 'economy', 'uberisation', 'uber', 'action']
```

Fig. 6: Hypernyms of the Shallow Parsed List

```
Meronym matching :
['uber', 'technology', 'inc.', 'transportation',
'technology', 'company', 'california', 'state', 'city',
'worldwide', 'market', 'car', 'transportation', 'food',
'delivery', 'mobile', 'driver', 'use', 'car', 'driver',
'rent', 'car', 'drive', 'uber', 'name', 'reference',
'word', 'topmost', 'super', 'word', 'über', 'meaning',
'uber', 'pioneer', 'economy', 'change', 'industry',
'result', 'economy', 'uberisation', 'uber', 'action']
```

Fig. 7: Meronyms of the Hypernyms parsed words

## VII. Results

```
Topic : Tianjin
Assigned Category :  Travel
Obtained Category : Technology
Topic : Tim Cook
Assigned Category :  Business
Obtained Category : Business
Topic : Tokyo
Assigned Category :  Travel
Obtained Category : Sports
Topic : Toronto
Assigned Category :  Travel
Obtained Category : Sports
Topic : Uber (Company)
Assigned Category :  Business
Obtained Category : Business
Topic : Vienna
Assigned Category :  Travel
Obtained Category : Business
Topic : Vladmir Putin
Assigned Category :  Politics
Obtained Category : Business
Topic : Washington DC
Assigned Category :  Travel
Obtained Category : Sports
Topic : Windows PC
Assigned Category :  Technology
Obtained Category : Business
Topic : Zurich
Assigned Category :  Travel
Obtained Category : Sports
Accuracy : 40.909090909090914%
```

Fig. 8: Baseline Output

## VIII. Conclusion:

The project experimented with different techniques to identify the category of web pages based on content. All of the techniques attack a different area of Natural Language Processing or information extraction from text. From the result we can easily conclude that processing the corpus through different Natural Language Processing algorithm we tend to get better classification result than initially obtained from the baseline program.

```
Topic : Tianjin
Assigned Category : Travel
Obtained Category :  Travel
Topic : Tim Cook
Assigned Category : Business
Obtained Category :  Travel
Topic : Tokyo
Assigned Category : Travel
Obtained Category :  Travel
Topic : Toronto
Assigned Category : Travel
Obtained Category :  Travel
Topic : Uber (Company)
Assigned Category : Business
Obtained Category :  Travel
Topic : Vienna
Assigned Category : Travel
Obtained Category :  Travel
Topic : Vladmir Putin
Assigned Category : Politics
Obtained Category :  Politics
Topic : Washington DC
Assigned Category : Travel
Obtained Category :  Travel
Topic : Windows PC
Assigned Category : Technology
Obtained Category :  Technology
Topic : Zurich
Assigned Category : Travel
Obtained Category :  Travel
Accuracy : 80.0
```

Fig. 9: Improved Program Output

## IX. Future Work

For this project I have only used webpages from the wikipedia as its better categorished. In future this project can be extended to include any webpages.
I have taken into consideration only 5 category for the categorization of the webpages. In future more category can be involved.This result is based on 100+ datasets. For better and efficient result we can increase our dataset to more literals. Using better classification methods including Support Vector Machines instead of Nave Bayes Classifier can provide more accurate results.

## X. Acknowledgments

The author would like to thank Professor Dan Moldovan and TA for the class Linrui, for their support and advice on this project.

## References

[1] Wikipedia Python API [Online]. Available: https://pypi-.python.org/pypi/wikipedia
[2] Chunking. Available: http://www.bogotobogo.com/
[3] Dao, Thanh Ngoc, and Troy Simpson. "Measuring similarity between sentences." WordNet. Net, Tech. Rep. (2005).
[4] Wu, Zhibiao, and Martha Palmer. "Verbs semantics and lexical selection." Proceedings of the 32nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1994.
[5] Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML. Vol. 14. 2014.
[6] Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.