

Group 13 - CP468

Final Project Report

Amber Dsilva - 173199200
Kieara Miranda - 170802500
Alexander Ojo - 170750510
Hitanshi Shroff - 170492430

Introduction:

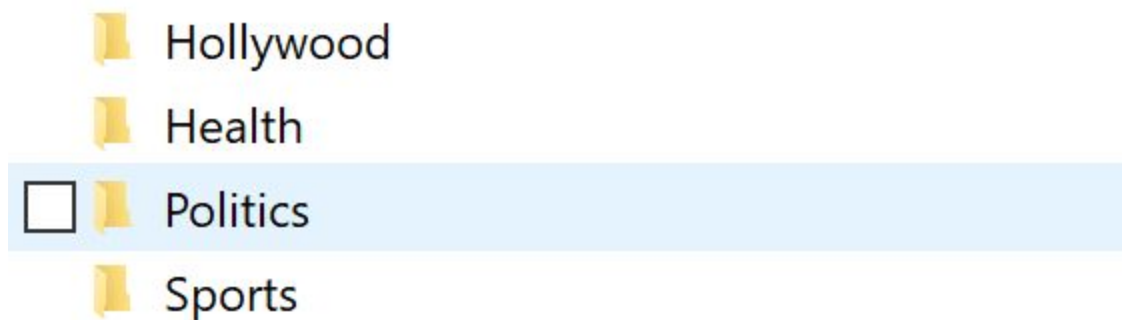
The purpose of this project is to investigate the tagging and classification of web pages through automatic categorization into specific topic categories. We named our project PHiSH. PHiSH is a bot that uses artificial intelligence to help users categorize popular news into topics that you care about such as politics, health, intense Sports, and Hollywood. These topics spell out the acronym PHiSH. 'PHiSH' is also a pun off the word fishing as one can end up scrolling for minutes without seeing an article that interests them. Similar to how you can spend hours fishing without catching any fish. PHiSH has a 92% accuracy rate in correctly categorizing new articles. PHiSH's output can be used to determine which of the specified topics are most popular at the time by counting the number of articles written about the topics.

PHiSH was created with python. We developed a web crawler which scraped the latest articles from a popular news website, CNN. From The CNN's site, our bot extracts relevant article information (article title, text, and category name) from the articles available and puts it into a dataset. The four topics we chose to focus on for our project are politics, health, sports, and Hollywood. These topics were chosen as they were of interest to our group members and are currently popular topics that are being discussed on the internet. We then classified 60% of the dataset to create a test set of data. Utilizing the K-Nearest Neighbour algorithm, PHiSH learns about the patterns within the articles information and proceeds to classify the remaining 40% of articles on it's own.






Project Description

From web to database:

Our web crawler gets a list of all the news articles and then looks at all the elements from each article. It then extracts the text from each url and adds it to a csv file which acts as a database. Our goal was to get the information to be split by title, text, category name for easier access by other functions. We used beautiful soup to search through html contents of a URL and parse through the elements by classifying them by their attributes and specific tags. We first tried to scrap the news article titles from the cover page based on specific CNN sections and then we retrieved the text out of them. Each topic was broken into the following folders with the respective data:



Each containing various text files which was the organized into a csv file based on the article data, title, url and category.

 001	Text Document
 002	Text Document
 003	Text Document
 004	Text Document
 005	Text Document

Features:

For classification we are using word frequency vectors (word count of each term in a document) as well as TF-IDF vectors (represent the importance of a term in a document). Term Frequency (TF) is the total count of the unique words available within a document. Inverse Document Frequency (IDF) is used to identify the weights of the rare words or important words. In order to make this process easier, before creating features we have to clean the raw text such as it being case-sensitive, punctuations, special characters, and filler words such as 'the', and other pronouns . Since classification models require numeric features to provide an accurate prediction we have also created the following mapping:

<u>Category</u>	<u>Value code</u>
Politics	0
Health	1
Sports	2
Hollywood	3

Training:

We have chosen a random split with around 60% of observations being put in the training set and 40% being added to the test set. We are following the model, K-nearest neighbour which is a supervised classification algorithm that computes the distance between each data point. It then classifies the article based on the features. Using the TF-IDF approach we got the following words that are most correlated to each category.

We attempted to do two word vectors as well, however, they did not provide additional information and were unrelated to the topics. Thus, we stuck with one word vectors.

```
from sklearn.feature_selection import chi2
import numpy as np

for Product, category_id in sorted(category_codes.items()):
    features_chi2 = chi2(features_train, labels_train == category_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf.get_feature_names())[indices]
    common_words = [v for v in feature_names if len(v.split(' ')) == 1]
    print("{}' Topic:".format(Product))
    print(common_words)
```

```
Sports Topic:
[Teams, Win, MLB, NBA, matches, games]

Health Topic:
[covid-19, coronavirus, fitness, CDC, workout, home]

Politics Topic:
[trump, vaccine, government, economy, president, election]

Hollywood Topic:
[celebrity, music, designer, movie, kardashian]
```

Experimental Analysis:

Outputs:

Once the similarities have been measured and determined from the training data that has been classified, we use the k nearest neighbour algorithm to find the

neighbours that are closest to k based on how similar their scores are. When this has been found, the weight of the ranked scores acts as the weight of each of the candidate categories and these categories are then used to predict the categories of the test data.

When presented with training data that is supposed to be classified, the algorithm searches for the K nearest neighbours in the pre-classified training data. This is ultimately based on the similarities and ranks that the k neighbours have based on their similarity scores. Further, the categories consisting of the k nearest neighbours are then used to predict the category of the test data. This is done by using the ranked scores of each as the weight of candidate categories.

Training analysis: Looks at the accuracy of the training datasets:

The training accuracy is: 0.9436284962

Test accuracy: Looks at the accuracy of the actual testing files:

The test accuracy is: 0.9152716293

Classification report:

```
from sklearn import metrics

y_pred = ["Politics", "Health", "Sports", "Hollywood"]

y_act = ["Politics", "Health", "Sports", "Hollywood"]

print(metrics.confusion_matrix(y_act, y_pred, labels=["0",
"1", "2", "3"]))

print(metrics.classification_report(y_act, y_pred, labels=["0",
"1", "2", "3"]))
```

Our outcome was as followed:

	precision	recall	f1-score	support
0	0.95	0.90	0.92	19
1	0.88	0.84	0.86	17
2	0.90	0.92	0.91	11
3	0.93	0.87	0.90	14
avg / total	0.92	0.88	0.90	61

*In the above outcome, 0 represents Politics, 1 represents Health, 2 represents Sports and
3 represent Hollywood.*

Table of actual outputs:

The y-axis represents the predicted and the x-axis represents actual.

					TOTAL
Predicted Politics	18	1	0	0	19
Predicted Health	2	15	0	0	17
Predicted Sports	0	0	10	1	11
Predicted Hollywood	1	0	0	13	14
	Actual Politics	Actual Health	Actual Sports	Actual Hollywood	61

Analysis by topics:

Politics:

- In total there were 19 articles that are supposed to be classified under the politics category. 18 of the 19 articles were correctly classified, however, 1 was incorrectly labelled into the Health category. This may be because it mentioned a lot of words that were similar to politics such as president, government, and trump while speaking about the health measures taken for covid-19.

Health:

- In total there were 17 articles that were supposed to be classified under the health category. 15 out of the 17 articles were correctly, classified, however, 2

were not. They were classified as politics instead. This could be because a lot of news surrounding covid-19 and the effect it will have on the US elections is found in the Politics category, hence, that could have impacted the result.

Sports:

- In total there were 11 articles that were supposed to be classified under the sports category. 10 out of the 11 articles were correctly classified, however, 1 was not. It was incorrectly classified as Hollywood instead. This could be because some of the articles revolving around sports mention the relationship the players have with people in Hollywood. Therefore, this could be a possible reason the article was misclassified.

Hollywood:

- In total there were 14 articles that were supposed to be classified under the Hollywood category. 13 out of the 14 articles were correctly classified, however, 1 was not. It was incorrectly classified as Politics instead. A reason for this could be the relationship between politics and hollywood. For example, the latest news for Kanye deciding to run for president could be classified as either politics or hollywood.

Conclusion

Throughout this assignment, we have been given the opportunity to learn more about supervised machine learning via designing and implementing this web document classifier. PHiSH enables the tagging and classification of web pages through automatic categorization into specific topic categories via the application of a K-Nearest Neighbour algorithm approach. With our bot, users can spend less time scrolling through the recent page and more time reading the articles that you want. With a 92% accurate categorization results, PHiSH's outputs can be utilized to determine what topics out of politics, health, intense Sports, and Hollywood are hot based on the number of articles being written about the topic.

References

1. Sadangi, Siddhant. "Introduction to Text Classification in Python." Medium, Analytics Vidhya, 29 June 2020, <https://medium.com/analytics-vidhya/introduction-to-text-classification-in-python-659eccf6b2e>.
2. Kulshrestha, Utkarsh. "NLP - Feature Selection Using TF-IDF." Medium, Analytics Vidhya, 2 June 2020, <https://medium.com/analytics-vidhya/nlp-feature-selection-using-tf-idf-db2f9eb484fb#:~:text=TF-IDF%20acronym%20for%20Term%20Frequency%20%26%20Inverse%20Document,Classification%2C%20Information%20Retrieval%20Systems%2C%20Text%20Data%20Mining%20etc.>
3. "K-Nearest Neighbors (KNN) For Iris Classification Using Python." Indowhiz, 25 June 2020, www.indowhiz.com/articles/en/implementation-of-k-nearest-neighbors-knn-for-iris-classification-using-python-3/.
4. "Sklearn.metrics.confusion_matrix¶." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html