Jeffrey Levesque

Ist-652 | Final project

Can Wikipedia Articles Be Classified

IST-652

Jeff Levesque

Professor Landowski

IST 652

**Introduction**

Often studies associated with natural language processing are closely related with the availability of public data. Social media platforms such as facebook, and twitter are common examples. Because more studies are usually conducted when data is available, efforts to collect from other sources, are sometimes limited. This is apparent in competition such as the Kaggle platform. Specifically, more popular competitions are generally associated when data is easily provided to competitors. Very few competitions are found, when data is both scarce, and similar competitions are limited.

In this study, rather than using available data, to repurpose an already accomplished study, data was manually harvested. This allowed a more unique situation, where a new question could be asked. *Can Wikipedia content predict the article category*? Though questions like these may not generally be novel, it has not directly been applied to Wikipedia articles. Therefore, manual labelling of the collected articles was required.

The process of collecting, then manually labeling data, to provide context for classification, is time consuming. Thus, a secondary question will address whether the collected data was enough. Specifically, *was the amount of data enough, and was the category of labels appropriate*? If article categories can be predicted with tolerable accuracy, then findings can be applied to a more involved study.

**Analysis**

Data Preparation:

Before proceeding, various package dependencies are needed:

apt-get update

apt-get install -y python3-pip

pip3 install lxml \

twitterscraper \

beautifulsoup4 \

python-dateutil \

wikipedia \

nltk \

scikit-learn \

numpy \

scipy \

matplotlib

python3 -c 'import nltk; nltk.download("punkt")'

However, if both vagrant, and virtualbox are installed locally, then only two commands are needed, to automate the necessary environment on any host machine:

git clone https://github.com/jeff1evesque/ist-652.git

cd ist-652 && vagrant up

After the necessary dependencies installed, the entire project can be executed through a single python3 run.py command in the terminal console. Using the wikipedia library, the top 1000 articles per month was collected, for the duration between August 1, 2016 through September 1, 2018:

r = requests.get(

'{}/metrics/pageviews/top/{}/all-access/{}'.format(

'https://wikimedia.org/api/rest\_v1'

'en.wikipedia.org',

date

)

)

top1000 = r.json()

articles = json.loads(r.text)['items'][0]['articles']

# report top 1000 article

with open(outfile, 'w') as jsonfile:

json.dump(top1000, jsonfile, indent=4)

This aggregated data provided context to generate a sample dataset. Specifically, the first month was used to generate a sample dataset. However, since the aggregated data did not provide the article category, this was done manually. Due to time restriction, only the first 500 articles from the first months top 1000 articles was categorized. Therefore, the refactored 2016-08-01--sample-train.json, required the last 500 articles to be truncated, and an extra category field to be added:



Next, the sample data was loaded into python:

with open('data/2016-08-01--sample-train.json', 'r') as f:

articles = json.load(f)['items'][0]['articles']

articles = [a for a in articles if a['category'] != 'other']

Then, a vectorized word count was computed, and normalized using the tfidf. At various times, the execution of the necessary scripts was throttled by either the corresponding endpoints, or limitations of the local machine hardware. Therefore, the corresponding scripts were git clone on an ec2 AWS medium instance. Since the IP address was not assigned using an elastic IP (or similar), a new public IP address was assigned each time the machine rebooted. This reduced the unexpected problems experienced when the corresponding scripts were run on local hardware.

**Results**

Before generating svm models, the tfidf matrix was split into train and test:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X,

y,

test\_size=test\_size,

random\_state=42

)

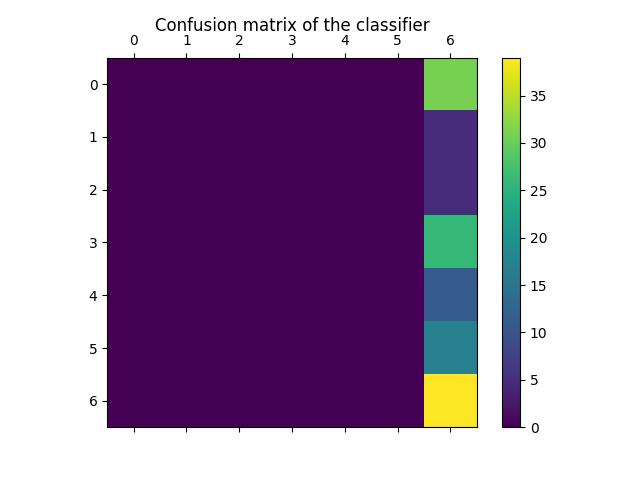
However, out of the 500 original articles, 2/3 was dedicated into the train set. Additionally, 101 articles were labelled other, and removed. This was done, since the other category comprised of wikipedia api endpoints, as well as various extraneous articles. Therefore, (500-101) \* 2/3 = 263 articles were used for the train dataset, while (500-101) \* 1/3 = 133 articles were used for testing.

Next, the chi-squared was used for feature selection. This meant various redundant words/features were removed, and only the top n features were used for the modeling and prediction. Specifically, five different cases were used to select the top features – 5, 25, 50, 300, all features. During the prediction for each of these cases:

clf.fit(X\_train, y\_train)

pred = clf.predict(X\_test)

it was found for all chi-squared cases, the pred returned an array with all elements being tv\_movie. The corresponding confusion matrix indicated this with the corresponding error rate being identical at 0.708:



Ideally, the confusion matrix should yield only diagonal values. This was not the case. When reviewing other related nlp studies, corresponding datasets often did not fall under several thousand records. For example, the 20 newsgroups dataset[[1]](#footnote-1) comprised of 20,000 documents. However, in this study the train dataset was not more than 263. Additionally, the distribution of articles in this study was not balanced. Specifically, the tv\_movie category had roughly 50 articles. However, two categories had less than ten articles. The poor distribution, along with limited data, likely prevented enough variances along distance measures, for the corresponding support vectors. On a similar note, an attempt to use chi-squared during the feature selection, likely did not have enough data to make an accurate judgement of which bag of words were closely related. Therefore, the computed approximations were likely error prone.

When returning to the original question posed: Can Wikipedia content predict the article category, the short answer is no. With the current distribution of articles, and corresponding vectorized tfidf bag of words, the accuracy rate 30%, is roughly double what would be expected by chance. Nevertheless, the accuracy is significantly poor. Some factors that could have contributed to this poor result include an unbalanced set of classifiers. As stated earlier, some categories contained under ten articles, while others had near 50 articles.

Due to time constraints, the grouping of categories may not be as standardized as needed. For example, many articles were categorized as other, if they did not fall into one of the predefined categories. Similarly, some articles may have been poorly categorized. Movie actors, and actresses, were labelled as celebrity, instead of falling under the tv\_movie category. However, musical artists were grouped under music, rather than the celebrity category. Parallel problems occurred for various situations, such as immediate political figures being labelled as politics, while corresponding political family members being labelled as either celebrity (Melania Trump), or politics (Michelle Obama).

The discrepancies for labelling, was largely due to the fact a standard scoring scheme was not determined. To further answer the second question, more labelled data is required, if a future study is conducted. Otherwise, the unbalanced categories can have a large negative impact on the overall model.

**Conclusions**

Being able to categorize text into predefined categories is one base for natural language processing. Social media platforms such as facebook, and twitter often perform tasks such as these. However, websites like Wikipedia may not perform these tasks as frequently. Specifically, when an article is created, authors can choose to categorize their work with any arbitrary category and associate it to any number of other categories. This unstructured hierarchy can pose some difficulties when trying to aggregate articles, and corresponding data.

Overall, the results found in this study does not provide new, or exciting insight in a given field. However, the process of trying to harvest custom data, with the challenges of labelling, allowed a greater holistic picture for data mining. Often as students, or beginner data scientists, dataset(s) obtained from the web, or Kaggle, are nicely structured with labels. However, in the context of research, and industry, many times these niceties are not be provided, but expected.

1. <http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html#loading-the-20-newsgroups-dataset> [↑](#footnote-ref-1)