CS403 Project Report

Classification Of News Articles Based On Headlines

150050004, 150050054, 150050057, 150070004

Motivation

In many real-world scenarios, the ability to automatically classify documents into a fixed set of categories is highly desirable. This categorization of news articles poses a significant challenge to user-driven news aggregator applications interested in determining the interests of their users and thereby provide the most relevant content. We aim to use machine learning to automate this process of binning news articles into appropriate topics.

Problem Formulation

Given a news headline string, classify it into one of the following -
1) business
2) science and technology
3) entertainment
4) health
Dataset: News Aggregator Dataset - UCI machine learning repository
Key Characteristics of the data set

- It contains headlines, URLs, and categories for 422,937 news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014.
- There are 152746 news of business category, 108465 news of science and technology category, 115920 news of entertainment category, 45615 news of health category.
- 2076 clusters of similar news for entertainment category, 1789 clusters of similar news for science and technology category, 2019 clusters of similar news for business category, 1347 clusters of similar news for health category.

Approach

The various approaches studied have been mentioned below.
The steps that need to be followed:

1. Data Processing

- Punctuation and number Removal - replace punctuation and number with spaces
- Stop Words Removal - based on a list of words or word frequency
- Words Stemming - Reduce derived words to their word stem/root

2. Feature Extraction
    - CountVectorizer
    - TFIDFVectorizer
3. Feature Selection (algorithms to avoid overfitting)
    - LSA (Latent Semantic Analysis)
    - Chi Square Stats
    - Variance threshold
    - Mutual Information
4. Word2Vec (Feature Selection and Extraction)
5. Classification Techniques
    - Naive bayes
    - SVM Classifier using different kernels
    - Neural network

Data Processing

Firstly, the data is converted into lowercase characters, and the punctuations removed. Stop-words removal and stemming help in improving the accuracy of the classifier by removing unimportant words and details of a word. Stop-words are removed in the next step of processing. Stop-words are words that are very common (e.g. the) and can be ignored for our problem of classification

Stemming reduces the words to their word bases (word roots). This allows the classifier model to treat two different words with same word bases as the same and thus having the same effect on classification which is what we'd expect.

Feature Extraction

This task requires us to extract features in the form of matrices (usable data format and that can be later used by various feature selection and machine learning algorithms to correctly classify data into different classes).

- TfidfVectorizer() : **tf–idf**, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. In our case we did not decide upon a fixed **corpus** (*a collection of words*) and allowed the algorithm to generate it's own corpus which in our case is all the unique words that have occurred so far. (To check the functionality of tf-idf we counted the unique words and the dimension of the feature. For the case of 10,000 samples it turned out to be 9582 unique words and hence 9582 features.)
- CountVectorizer(): Convert a collection of text documents to a matrix of token counts. The number of features in this case is also equal to the total number of unique words occurring in the whole dataset. But unlike TF-IDF it does not gives weights to each letter occurring in that sentence rather it just assigns a 1 if the number is there in that line and 0 otherwise.

Which one to use - CountVectorizer or TF-IDF?

We decided to use CountVectorizer instead of TfidfVectorizer because of the following results that we got:

The accuracy obtained on a dataset of 10,000 samples was as follows-

1. TfidfVectorizer - 0.931
2. CountVectorizer - 0.9525

The accuracy obtained on a dataset of 1,00,000 samples was as follows-

1. TfidfVectorizer - 0.9378
2. CountVectorizer - 0.94155

We expected TFIDF to do better than CountVectorizer, given that instead of simple one hot encoding, it assigns normalised weights based on frequency. One reason could be that it is just a coincidence, given that dataset is very large, the performance can vary. Other reason could be that the some words that do occur more frequently might be the drivers of the classification contrary to what TFIDF assumes, again because the dataset is quite large.

Feature Selection Algorithms

In any form of NLP, the number of words that we need to work with or the number of words encountered is simply very big, which is why it becomes tough and much complex for classifiers to work smoothly and can even in fact result in overfitting. For this, we carry out effective feature selection process and reduce dimensionality of the data set The basic idea is that the transformed data should preserve most of the relevant information while at the same time having low dimensionality which in turn allows better machine learning treatment as against using the basic one-hot encoding of data.

- Principal Component Analysis
- Variance threshold
- Chi Square with the help of SelectKbest
- Mutual info classification

Principal Component Analysis

It is most basic implementation of feature selection. In this, since the data set was sparse so we ended up using LSA (Latent Semantic Analysis). It is almost similar to PCA in the fact that both of them rely on Singular Value Decomposition but the former processes the data first before applying SVD to calculate most significant features and thus can be used with sparse matrices for which SVD is not defined (also called Truncated SVD). Since features extracted from LSA contained negative values, we could not use Naive Bayes with it. So we used randomm forest as ML classifier.

Variance Threshold

Variance Threshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples. As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by so we can select using the threshold .8 * (1 - .8).
In the project we have used the default value of threshold as present in sklearn

Chi-Square Test

Compute chi-squared stats between each non-negative feature and class.
This score can be used to select the n_features features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification), relative to the classes.
The chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

Mutual Information Classification

Estimate mutual information for a discrete target variable. Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances. Thus it is later used with K-means clusterization.

Word2vec

We have also implemented word2vec as an option for the feature extraction + feature constrained technique. Word2vec uses two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of predefined text and produces a vector space, with each unique word in the corpus being assigned a corresponding vector in the space. In our implementation we were able to get an accuracy of 0.72 on a dataset of 10,000 and using a corpus of less words (glove.6B.50d). But we were not able to do this on a bigger sized corpus because it was requiring too much computation power which our machines were not able to handle.

Algorithms for classification:

To classify the data we have used the following standard off the shelf libraries from Sklearn.

1) Naive bayes
2) SVM Classifier using different kernels - linear, polynomial, rbf.
3) Neural network

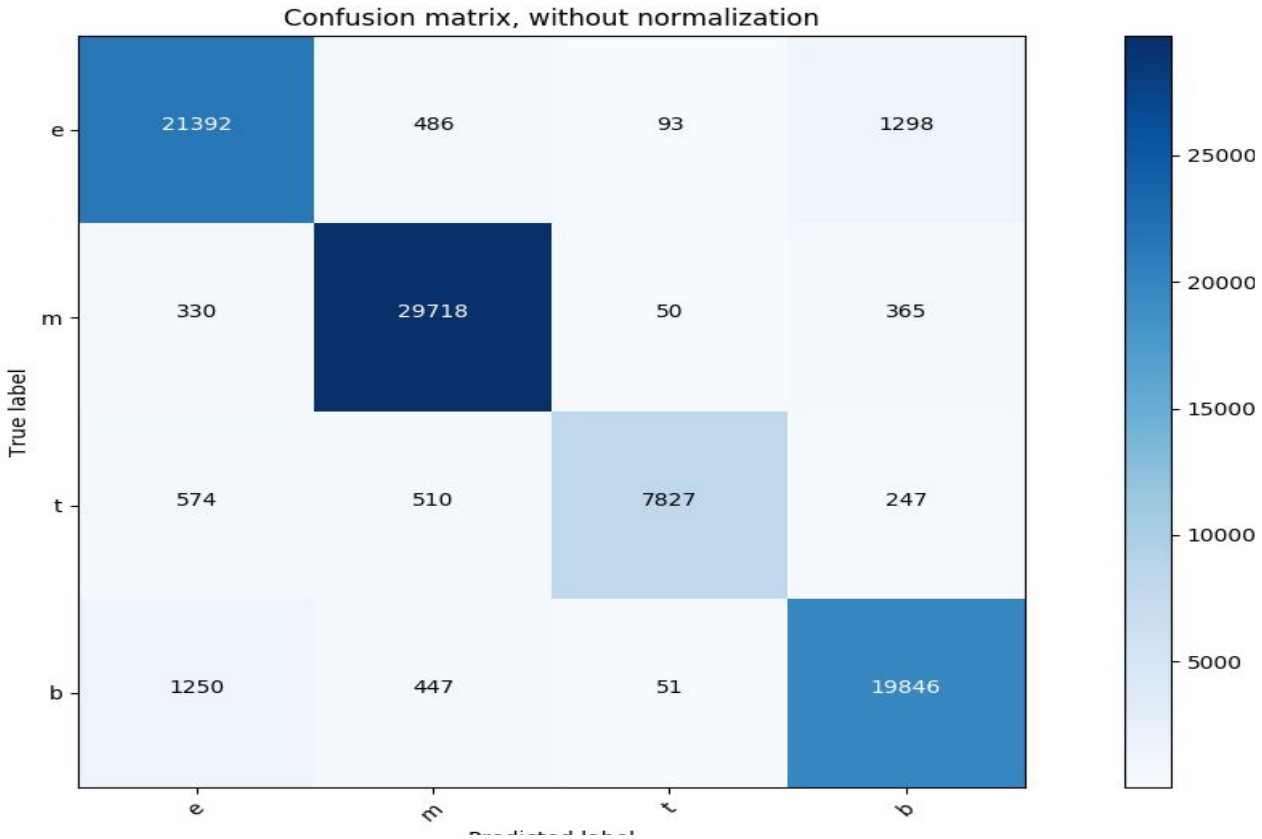
Among these the best results were given by SVM classifier using linear kernel.

Results

Scores obtained with different feature extraction techniques on training dataset of 1 lakh samples:

1. Chi square :
    A. Without Chi square : 0.93815
    B. With Chi square : 0.94175
2. Mutual Information Gain: 0.9404
3. Variation Threshold : 0.9373
4. RBF Kernel : 0.74365
5. Linear SVM : 0.9525

Confusion Matrix of Results:



Conclusions

So on the basis of various functionalities that we tried and tested we were able to conclude the following best algorithms to do their respective jobs for classifying news headlines to different classes.

- Feature Extraction - CountVectorizer
- Feature Selection - Mutual Info implemented using K-means (although it works best when data set is really big otherwise variance threshold works fine)
- ML Classifier - Linear SVM

Shortcomings

Our projects has some shortcomings

- The dataset has just 4 categories and so news headlines of other categories (like sports) cannot be classified.
- The dataset is primarily of US news headlines and so classifications of other countries news headlines will not optimal due to mention of US agencies, states etc in the dataset.
- We have not classified the news headlines based on semantics and have only used words and their patterns of occurrence to decide

References

- Scikit Learn Documentation
- NLTK Documentation
- http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7097339
- http://stackoverflow.com/questions/5512765/removing-punctuation-numbers-from-text-problem
- http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html#sphx-glr-auto-examples-text-document-classification-20newsgroups-py
- http://scikit-learn.org/stable/modules/feature_extraction.html
- http://textminingonline.com/getting-started-with-word2vec-and-glove-in-python
- http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7097339&tag=1
- https://link.springer.com/chapter/10.1007/978-3-642-35527-1_27
- http://www.iosrjournals.org/iosr-jce/papers/Vol18-issue1/Version-3/D018132226.pdf
- https://pdfs.semanticscholar.org/aa96/9114cf6e4d77c5bb3dd62a20bee3446f33ab.pdf

Individual contribution

Manas Bhargava 100%	Charles Rajan 100%
Shaan Vaidya 100%	Devansh Shah 100%