2018 Machine Learning Project

for CS 4375.001

Team Half & Half

Web Crawling and Use of Supervised and Unsupervised Algorithms

Abstract

Tasked with demonstrating machine learning algorithms using supervised and unsupervised learning methods, we will display the results of KMeans, Decision Tree, and Random Forest algorithms

.Project Overview

This project will demonstrate text analysis using web crawlers and different classification algorithms. We elaborated on the sample code using URL library request to obtain text from the New York Times. We collected data, gathered features, and used three algorithms to generate results. The process used was the word2vec model to extract features to be prepared for the algorithms. All the algorithms used were taught in class. After the results were obtained, evaluation and validation were used to properly check the results. Lastly, the data was visualised using TSNE.

KMeans Algorithm

The first algorithm used was the KMeans algorithm. This is an unsupervised learning algorithm that takes k initial clusters and groups similar data together into clusters. We used the sklearn.clusters.KMeans from sckikit learn to achieve this. We converted text from the New York Times using the Count Vectorizer, which is also from sklearn. From there we performed the KMeans algorithm to form clusters by frequencies of certain words. To visualize it, we used a tfidf matrix to plot the clusters on a plt figure. The clusters are labeled based on what keywords were used to cluster them and a corresponding color.

Code Example:

```
# Instantiate parameters for vectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.95, max_features=10000,
                                   min_df=0.005, stop_words='english',
                                   use_idf=True, tokenizer=tokenize_and_stem, ngram_range=(1,3))

tfidf_matrix = tfidf_vectorizer.fit_transform(df['lead_paragraph'].values.tolist()) #fit the vectorizer to synopses
terms = tfidf_vectorizer.get_feature_names()
dist = 1 - cosine_similarity(tfidf_matrix)

# KMeans
num_clusters = 5
km = KMeans(n_clusters=num_clusters)
km.fit(tfidf_matrix)
clusters = km.labels_.tolist()

# Dump into pickle file
joblib.dump(km, 'kmeans-cluster.pkl')
km = joblib.load('kmeans-cluster.pkl')

clusters = km.labels_.tolist()
section_list = df['section'].values.tolist()
section_cat = { 'section': section_list, 'main_paragraph': df['lead_paragraph'], 'cluster': clusters}
frame = pd.DataFrame(section_cat, index = [clusters] , columns = ['section', 'main_paragraph', 'cluster'])
```

Decision Tree Algorithm

The second algorithm we used was the decision tree algorithm. This algorithm can be used for supervised learning methods and can be used for solving regression and classification problems. We used countvectorizer to count the number of occurrences for each term.

```
cvec = CountVectorizer(stop_words='english', min_df=.0025, max_df=.1, ngram_range=(1,2))
cvec.fit(data_features)
```

We then used tfidf to calculate weights for each term and then averaged the weights before tokenizing the data. It was the same processed for Naive Bayes only we used the decision tree classifier instead.

Code Example:

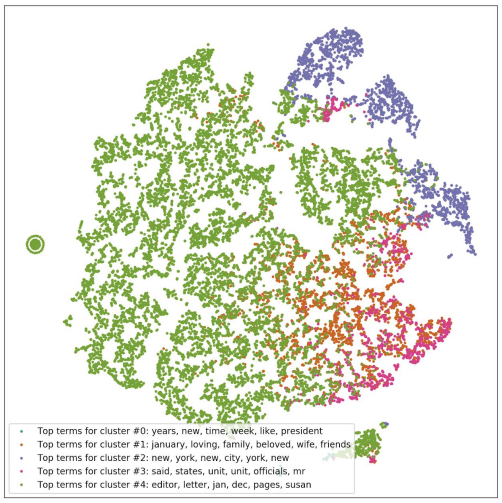
```
# tokenize train and test text data
vect = CountVectorizer()
X_train_tokens = vect.fit_transform(X_train)
X_test_tokens = vect.transform(X_test)

dt = DecisionTreeClassifier()
dt.fit(X_train_tokens, y_train)
y_pred_dt = dt.predict(X_test_tokens)
# Precision, Accuracy, and Recall model_eval_calculations for NB and SVM.
def model_eval_calculations(y_test, y_pred_dt):
    print('Precision:', precision_score(y_test, y_pred_dt, average='weighted'))
    print('Accuracy:', accuracy_score(y_test, y_pred_dt))
    print('Recall:', recall_score(y_test, y_pred_dt, average='weighted'))

model_eval_calculations(y_test, y_pred_dt)
```

Random Forest Classification

Using the bag of words as features and labels, we fit the forest to the training set. This fits a number of decision tree classifiers on various sub-samples of the dataset, using the average to improve the predictive accuracy and to control potential over-fitting. The value for number of trees in the forest we used is 1000.

```
# Clustering the data with Random Forest Classifier
rf = RandomForestClassifier(n_estimators=1000, random_state=0)
rf.fit(X_train, y_train)
```

K-Means Result



Decision Tree Result

	precision	recall	f1-score	support
0	0.75	0.79	0.77	379
1	0.54	0.50	0.52	284
2	0.52	0.57	0.54	299
3	0.53	0.53	0.53	327
4	0.26	0.23	0.25	30
5	0.56	0.58	0.57	205
6	0.59	0.55	0.57	78
7	0.52	0.49	0.50	254
8	0.42	0.27	0.33	48
9	0.40	0.41	0.40	239
10	0.33	0.16	0.22	37
11	0.16	0.40	0.23	40
12	0.51	0.39	0.44	69
13	0.32	0.28	0.30	43
14	0.31	0.26	0.28	35
15	0.53	0.50	0.51	54
16	0.00	0.00	0.00	5
micro avg	0.53	0.53	0.53	2426
macro avg	0.43	0.41	0.41	2426
weighted avg	0.53	0.53	0.53	2426

Decision Tree accuracy = 52.926628194558944%

Random Forest Result

	precision	recall	f1-score	support
0	0.43	0.36	0.40	44
1	0.60	0.76	0.67	359
2	0.61	0.63	0.62	274
3	0.32	0.21	0.25	33
4	0.55	0.16	0.24	38
5	0.84	0.85	0.84	393
6	0.50	0.50	0.50	233
7	0.57	0.49	0.53	275
8	0.47	0.62	0.54	248
9	0.50	0.11	0.19	35
10	0.69	0.65	0.67	205
11	0.52	0.42	0.46	74
12	0.45	0.23	0.30	40
13	0.79	0.58	0.67	57
14	0.71	0.66	0.68	67
15	0.71	0.42	0.53	40
16	0.00	0.00	0.00	3
micro avg	0.61	0.61	0.61	2418
macro avg	0.54	0.45	0.48	2418
weighted avg	0.62	0.61	0.61	2418

Random Forest accuracy = 61.49710504549214%

Summary

In summary, this project demonstrated implementations of three different machine learning algorithms; KMeans, Decision Tree, and Random Forest to classify a dataset of New York Times news articles. Initially a web crawler was used to read text from the articles into a file. Features were extracted using the word2vec model. Then the three different learning algorithms were implemented on the datasets, evaluated for precision and accuracy, and visualized using a TSNE scatter plot.