Laboratory Exercise 2
FRA 311 Artificial Intelligence for Robotics and Automation
Text Classification and Clustering

Warasinee Chaisangmongkon, PhD

April 3, 2018

1 Objective

In this lab exercise, we will learn how to build a machine learning model from unstructured data such as text. We are constructing two models: sentiment analyzer and text clustering.

Sentiment analysis is the process in which documents are automatically classified based on the positive or negative sentiment of the document. For example, you retrieved 5000 pieces of movie reviews (documents) from IMDB and you want to classify each document into positive versus negative review. This is exactly what we want to accomplish in this lab exercise.
We are also going to learn to cluster text data and automatically categorize emails (newsletters) according to their contents.

2 Sentiment Analysis

2.1 Overview

In order to perform sentiment analysis, we are going to write 3 code sections.

1. Read and organize raw data, turning raw review text into lists that can be fed to feature extraction function.
2. Feature extraction, generate vectors of meaningful features from review texts for classification.
3. Classification. We will use various classifiers to categorize movie reviews.

2.2 Movie Review Data

Let us first start by looking at the data provided with the exercise. We have positive and negative movie reviews labeled by human readers, all positive and negative reviews are in the 'pos' and 'neg' folders respectively. If you look inside a sample file, you will see that these review messages have been 'tokenized', where all words are separated from punctuations.

There are approximately 1000 files in each category with files names starting with cv000, cv001, cv002 and so on. You will split the dataset into training set and testing set.

1. Write some code to load the data from text files.

2.3 TF-IDF

From a raw text review, you want to create a vector, whose elements indicate the number of each word in each document. The frequency of all words within the documents are the 'features' of this machine learning problem.

A popular method for transforming a text to a vector is called tf-idf, short for term frequencyinverse document frequency.

1. Conduct a research about tf-idf and explain how it works.

2. Scikit-learn provides a module for calculating this, this is called TfidfVectorizer.

   You can study how this function is used here:

   `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

   Write code to transform your text to tf-idf vector.

2.4 Classification

Use 4 different models to classify each movie into positive or negative category.

1. K-Nearest neighbor model, using module sklearn.neighbors.KNeighborsClassifier
2. RandomForest, using module sklearn.ensemble.RandomForestClassifier
3. SVM, using module sklearn.svm.SVC

4. Neural network, using sklearn.neural_network.MLPClassifier

You may pick other models you would like to try. Just present results for at least 4 models.

Please provide your code for model fitting and cross validation. Calculate your classification accuracy, precision, and recall.

2.5 Model Tuning

Can you try to beat the simple model you created above? Here are some things you may try:

- When creating TfidfVectorizer object, you may tweak sublinear_tf parameter which use the tf with logarithmic scale instead of the usual tf.
- You may also exclude words that are too frequent or too rare, by adjusting max_df and min_df.
- Adjusting parameters available in the model, like neural network structure or number of trees in the forest.

Design at least 3 experiments using these techniques. Show your experimental results.

3 Text Clustering

We have heard about Google News clustering. In this exercise, we are going to implement it with Python.

3.1 Data Preprocessing

Let's switch up and use another dataset called 20newsgroup data, which is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data is collected from a university's mailing list, where students exchange opinions in everything from motorcycles to middle east politics.

1. Import data using sklearn.datasets.fetch_20newsgroups
2. Transform data to vector with TfidfVectorizer

3.2 Clustering

We are going to use the simplest clustering model, k-means clustering, to do this task. Our hope is that this simple algorithm will result in meaningful news categories, without using labels.

1. Fit K-Means clustering model to the text vector. What is the value of K you should pick? Why?

2. Use Silhouette score to evaluate your clusters. Try to evaluate the model for different values of k to see which k fits best for the dataset.

3.3 Topic Terms

We want to explore each cluster to understand what news articles are in the cluster, what terms are associated with the cluster. This will require a bit of hacking.

1. Use TfidfVectorizer.get_feature_names to extract words associated with each dimension of the text vector.

2. Extract cluster's centroids using kmeans.cluster_centers_.

3. For each centroid, print the top 15 words that have the highest frequency.