Lab 05: Exploratory Data Analysis (EDA) with Dimensionality Reduction
CS3300 Data Science

Learning Outcomes

1. Extract features for non-tabular data
2. Apply dimensionality reduction methods to visualize high dimensional data
3. Compare and determine appropriateness of different data structures for a given situation

Overview

In the first half of the class, we've been focusing on exploratory data analysis (EDA). You've primarily been using data sets with only a handful of variables. You could analyze each variable with visualizations and statistics to find relationships. High dimensional data sets, however, have too many variables for you to analyze each variable individually. We need to turn to more sophisticated techniques such dimensionality reduction and clustering.

In this lab, you are going to analyze 63,542 emails. You will convert the raw text into a feature matrix using a "bag of words" model. Each column of the feature matrix corresponds to one word, each row corresponds to one email, and the entry stores the number of times that word was found in that email. You will perform dimensionality reduction using the Truncated SVD method, cluster the emails, and compare the "inherent" structure to the given class labels.

Instructions

Part I: Load the Data

a. Extract the provided email_json.zip file. It should create a directory called "email_json". Each email is stored as a separate JSON document with a name in the format "message_XXXXX.json".

b. Write some code to find and load all of the JSON documents. You should have a list of dicts when done. (Hint: Review the built-in Python glob and json libraries.)

c. Convert the list of dicts into a Pandas DataFrame. The DataFrame should have 5 columns and 63,452 rows. What are the column names and their types? (Hint: use the DataFrame.from_records() or DataFrame.from_dict() functions.)

Part II: Extract Features

By themselves, strings of the message bodies are not amenable to analysis. We need to convert them to a feature matrix.

a. Use Scikit Learn's CountVectorizer class with the binary=True flag to create a feature matrix from the message bodies. The "bodies" column of the DataFrame can be used as a list of strings and passed directly into the fit_transform() method of the CountVectorizer.

b. How many rows and columns does the feature matrix have? How many nonzero entries are in the matrix?

c. Inspect the features. How many entries are there in the vocabulary_ dict? What are the column indices of the words "work", "love", and "different"? Print out the columns for each of the three words.

Part III: Dimensionality Reduction

In part III, we will perform dimensionality reduction.

a. Use the fit_transform() function of Scikit Learn's [TruncatedSVD]() class to transform the original feature matrix into a new feature matrix with 10 columns (variables or components). Scikit Learn's TruncatedSVD method is similar to its PCA method, but it works with sparse matrices.

b. Plot the explained variance ratios of the components. Which two components have the highest explained variance ratios? (Hint: use the explained_variance_ratio_ property of the TruncatedSVD class.)

Part IV: Visualization

In part IV, we will plot the points along the two components you identified.

a. Create a scatter plot using the two components with the highest explained variance ratios. (Hint: plt.scatter(proj_matrix[:, i], proj_matrix[:, j]).

b. Create a second scatter plot using the same two components. This time, color the points based on the category column of the DataFrame. (All spam messages should be one color; all ham messages should be a second color.)

Reflection Questions

1. Assume that 32-bit (4-byte) floating point values are used to store the counts. Calculate the memory usage of a dense matrix with those dimensions.

2. The vectorizer returns a sparse matrix in compressed row format (CSR). Assume that the sparse matrix uses one 32-bit (4-byte) floating point number and one 32-bit (4-byte) integer for each nonzero entry and one 32-bit (4-byte) integer for each row. Calculate the memory usage for the sparse matrix.

3. Calculate the sparsity ratio (100 * number of nonzero entries divided by maximum possible entries).

4. Based on your analysis, do you think that the sparse matrix is better suited for this situation? If so, why?

5. In the scatter plot with label classes, how you describe the relationship between the pattern you observe and the ham and spam messages?

Submission Instructions

Save the Jupyter notebook as a PDF and upload that file through Canvas.

Rubric

Followed submission instructions	5%
Formatting: Report is polished and clean. No unnecessary code. Section headers are used. Plots are described and interpreted using text. Plots have axis labels. The report contains an introduction and conclusion.	5%
Part I: Load the Data	
Found and loaded all JSON documents	10%
Converted JSON documents to a DataFrame	10%
Part II: Extract Features	
Extracted word counts to create feature matrix	10%
Inspected feature matrix	5%
Inspected the features	5%
Part III: Dimensionality Reduction	
Extracted 10 components using SVD	10%
Plotted the explained variance ratios and choose the 2 correct components	10%
Part IV: Visualization	
Scatter plot (without labels)	10%
Scatter plot (with labels)	10%
Response Questions	10%