Programming Assignment

Assignment 2B: Text Analytics for Health [COMP90090 2024]

Mike Conway & Thinh Truong

1st Feb 2024

1 Introduction

This assessment will require you to complete a Jupyter notebook that will be submitted for grading. The assignment contains a series of questions you need to work through. You will need to provide answers and the code that produces the answers in the Juypter notebook. There are nine questions to examine your learning from the course materials and tutorial sessions.

The file submitted must be a Jupyter notebook file (suffix .ipynb). No other file type will be accepted. Please title your files as follows, replacing {NAME} with your surname, that is:

`NLP4Health_assignment2B_{NAME}.ipynb`

Also, please make sure you begin the Jupyter Notebook with the assignment name ("Assignment 2B"), the date, and most important of all, your name.

2 Assignment 2B

This assignment will primarily explore unsupervised learning, which includes unsupervised clustering, topic modeling, and word embeddings.

1. Load the `dr_review` corpus. How many entries and clinicians being reviewed are there in the corpus? What is the label distribution of the reviews?
2. Preprocess all the reviews in the corpus by performing lowercase tokenisation and removing stopwords and punctuation. Find the most frequent 10 words from all the reviews after preprocessing. Then, calculate the TF-IDF scores and find the top 10 words with highest TD-IDF score. [Hint: use NLTK to perform preprocessing]
3. Select the first 10K reviews and use the TF-IDF scores as features, perform k-means clustering on the reviews to form two clusters. For each cluster, find the cluster centroids (center of the cluster) and get the 10 words with highest TF-IDF scores based on the centroids. Try different numbers of cluster from `[3, 5, 10]` and print out the top 10

words. What are the limitations of using k-means clustering on high-dimensional sparse data like TF-IDF vectors? How might you overcome these limitations? [Hint: use `scikit-learn` for clustering; use `cluster_centers_.argsort()` from kmeans model to obtain the top words].

4. Perform topic modeling with LDA on the 10K reviews to produce 5 topics. Print out the topics in a Jupyter Notebook cell and explore the resulting topics to determine if there are any interesting patterns. What about 10 topics? [Hint: use gensim to train LDA: first create a `Dictionary` for the text, then train `LdaModel`, and finally encode the reviews. You can filter out the most infrequent and infrequent words from Dictionary and use fewer passes in training the LDA model to save time.]

5. Use the two clusters from Q3. Perform dimensionality reduction on the TF-IDF scores through Principal Component Analysis (PCA) to reduce the feature dimension to 2. Then draw a 2D figure to explore these two clusters. [Hint: use `scikit-learn` for PCA and `matplotlib` to draw the scatter plot.]

6. Load the pre-trained word embedding `glove-twitter-25` using `gensim`. Obtain embeddings for the 10 words with highest TF-IDF from Q2. If a word does not have any embedding, try modifying the word (e.g., change the tense) and see if that can resolve the issue. Project them into a two-dimensional space using PCA and draw them in one figure. Explore their relations to each other and see if their positions in the space reflect any semantic meanings. [Hint: use `gensim` downloader API to load the embeddings.].

7. Find the top 10 words with highest TF-IDF in each class (Positive/Negative), then visualize their embedding similar to Q6.

8. Use all reviews from the corpus. Create train/validation/test sets with a 70/20/10 ratio based on doctor id and make sure that each doctor only appears in one set (e.g. the same doctor should not appear in the test set and the validation set). You can simply slice the dataframe and use the original order, say the first 70% of doctors as train set.

9. Preprocess the reviews (or use previous preprocessed data) from Q8 and train a bag-of-word model on the training set using logistic regression. Report accuracy and AUROC on the test set. [Hint: refer to Assignment 2A; calculate AUROC using `scikit-learn`].

3 Learning Objectives

This assignment addresses the following specific learning objectives:

- Evaluate a range of health-related text data sources
- Develop and evaluate NLP pipelines using the Python programming language
- Employ appropriate health-related NLP libraries and knowledge resources