LING48 Computational Linguistics
Dartmouth College. Spring 2023

Homework 3
Features and Unsupervised Learning

Instructions

This week we have three exercises related to sentiment analysis, K-Means clustering and embeddings. Remember to include basic documentation (your name, what the program does, its input and its output).

You can use code that you find online. If you do this, please include a link to the original code in your documentation.

You will need to submit **four** files: The code for each of the three exercises, and a PDF document with screenshots and explanations of your results.

Exercise 1: Darth Vader's Feelings

You will find four files in your Canvas module:

(1) `NRC-Emotion-Lexicon-Wordlevel-v0.92.txt`: This contains the NRC emotion features for approximately 14K words.
(2) `sw4.txt`, `sw5.txt` and `sw6.txt`: The scripts of the original Star Wars movies. The file has the name of the character and the lines.

You need to find the words in the Darth Vader lines that have emotion values in the NRC file. You will then add up the emotions for those words so that you can get the sum of emotions for each movie.

Make sure your file reports these findings. You can do this as simply as printing the results of each emotion in each movie to the Console/Terminal/CommandPrompt.

Finally, write a short description of your results in the written submission file. Please make it at least 50 words, and 200 words at most. For example, is Darth Vader angrier in any of the movies? Is he sadder in any of them? Also, please include screenshots of the output of your code.

Exercise 2: Clustering Shakespeare Plays

You will find two files in your Canvas module:

(1) The file `shakespeare.txt` has all the plays of William Shakespeare. They are divided by the token `<<NEWTEXTSTARTSHERE>>`.
(2) The file `shakespearePlayTitles.txt` has the titles for the plays.

You will need to do the following:

(1) Open the starter code. It will have commands for how to open the Shakespeare files.

(2) Calculate the tf-idf matrix for the plays and their tokens. I recommend you use the `SciKit-Learn` function `TfidfVectorizer`, but you can use any function you want. (There's a URL on Canvas with examples of how to do this).

(3) Use a k-means algorithm to cluster the plays according to lexical similarity. I suggest you use ten clusters, but you can use a different number if you believe it gives you clearer results. I suggest you use the function `KMeans` in `SciKit-Learn`, but again, you can use any function you want.

(4) Print the name of each play and the cluster it was assigned to. You can get this using the method `labels_` of the object that `KMeans` returns.

(5) Try to predict the cluster that two new "documents" would fall into. You will need the function `transform` from the `TfidfVectorizer` object, and the method `predict` from the `KMeans` object. The new documents are the following:

Doc1: `battle and king`
Doc2: `wit and love`

(6) Make a hierarchical clustering dendrogram of the plays. If you'd like, you can use the code in the Indo-European clustering example. You'll have to make two changes: (a) You should apply the method `todense()` to the matrix that goes into linkage (e.g. `X.todense()`). (b) Instead of the `single` method, please use the `ward` method in the linkage.

Write a short description of your results in the written submission file. (Please make it at least 50 words, and 200 words at most). Do your clusters make sense? Are similar plays grouped together? Also, please include screenshots of the output of your code, as well as the dendrogram.

If you want to work in an Indigenous or a non-European language, **please let me know** so we can find a suitable collection of documents to cluster.

Exercise 3: Word Embeddings

The website https://fasttext.cc/docs/en/crawl-vectors.html has word embedding models for 157 languages. In this exercise, you will download the models for one of these languages and see if these have the same relationships and biases that the English models do.

(1) Download the starter code from Canvas.

(2) Download your chosen `.bin` file. (There is code for this in the notebook, but you will need to give it the new URL). Remember that you need to do this with a language that is not English. You will use the function `fasttext` from the `fasttext` package. (This is a variant of word embeddings; similar to Word2Vec. It saves words and subword units).

(3) Get the 25 most similar words to *man* and to *woman* in the language you are working with.

(4) Try to perform the arithmetic operation "king-man+woman" (in the language you've chosen) and report the 25 top results. Is the word for *queen* included in the results?

(5) Print the t-SNE chart for the words ['man', 'woman','king','queen','child','boy','girl'] in the language that you've chosen. I recommend that you use the functions `TSNE` from `sklearn` and `pyplot` from `matplotlib`.

Write a short description of your results in the written submission file. (Please make it at least 50 words, and 300 words at most). What language did you use? What are the differences between the embeddings for man and woman? Does the operation woman+king-man work as expected? Please include screenshots of your output and also the t-SNE chart.

(You can use languages with CJK characters. In order to do so you will need to uncomment a few lines in the code).

Evaluation

You will need to submit **four** files: The code for each of the three exercises (either `.py` or `.ipynb`), and a PDF document with screenshots and explanations of your results.

25% First task (Darth Vader's feelings)

- 10% Emotions are correctly calculated for all three movies.
- 10% Output is well organized (i.e. it's present and it's easy to read).
- 5% Explanation of the results. The student effectively summarized the results. The explanation is within the word limits.

35% Second task (Shakespeare clustering)

- 10% Names of plays and the cluster for each of them is calculated and presented.
- 10% Correct calculation of the cluster of the "new documents".
- 10% Plot for the hierarchical clusters.
- 5% Explanation of the results. The student effectively summarized the results. The explanation is within the word limits.

30% Third Python script (embeddings in a language other than English)

- 5% Calculation of most similar words to man/woman
- 5% Explanation of man/woman results.
- 5% Calculation of king-man+woman arithmetic
- 10% Explanation of arithmetic results.
- 5% Plot for t-SNE

10% Other aspects of the code

- 5% Following instructions. E.g. Did the write-up include screenshots of their work?
- 5% Code quality. E.g.: Does it include their name, email, and basic comments throughout the code so you can follow the flow of the code?