NLP coursework

Maxim Khovansky

November 2024

1 Q1

I modified the pre_process function to add a number of preprocessing techniques, namely EOL token removal, punctuation separation, punctuation removal, tokenisation, lowercase normalisation, stopword removal, and lemmatisation. How many of these techniques are used is determined by the "tech" parameter, which is a list of the techniques to use, that I added for convenience of use. I used regular expressions for punctuation separation and list comprehension for everything else. Of these techniques, EOL token removal and stopword resulted in the biggest performance increase. This makes sense: TFIDF does not know the order of the tokens, so the EOL tokens provide no information and merely constitute noise. Similarly, stopwords are generally not very informative and constitute noise as well.

2 Q2

I modified both the to_feature_dictionary function and the pre_process function, as well as the create_document_matrix_from_corpus functions, to add a number of additional features to my vector; additionally, I modified the create_character_document_from_dataframe function as well as the subsequent cell to accomodate gender classification. The additional features I added were bigrams, POS tags (I appended a copy of the tokens with the corresponding POS tags attached to them to the vector), POS_counts (count of each POS tag), a pre-trained gender classifier's prediction, average line length, total sentiment (based on a lexicon dataset), and k-best selection. Additionally, I implemented TFIDF instead of simple counts using sklearn's TfidfVectorizer. I additionally tried using TfidfVectorizer together with the default Counter method, but this produced worse results.

I added bigrams, POS tags, and POS tags in the pre_process function for convinience; k-best selection had to be implemented in the create_document_matrix_from_corpus function, while the rest were implemented in the to_feature_dictionary function. For gender classification, I pre-trained a Linear SVC model on a separate dataset constructed from the data, which I did in the create_character_document_from_dataframe function.

Out of the tested techniques, gender classification and POS tags were the only ones to improve performance. This makes sense as my gender classifier achieved 94 pct accuracy on the training data, meaning it practically eliminated half of the characters from the similarity rankings, while POS tags helped disambiguate the meaning of words.

As before, which techniques to use could be specified in each of these function's arguments, variously called "extra_features" or "ef"

3 Q3

I modified the create_character_document_from_dataframe function to add context from the same scene to each of the character documents. I did this by first creating a database of the lines from all the scenes, and then sequentially adding lines from other characters to each character document if they were in the same scenes. I stopped adding context when the total number of lines added per character exceeded the specified maximum line count.

The question was worded ambiguously with regard to how exactly the maximum line count should be implemented, but this was the most plausible interpretation for me.

As before, whether to use context or not is specified in the "context" argument of the function.

Adding context significantly decreased performance, as my grid search later found; therefore, I didn't use it in my final configuration. This makes sense: adding context greatly reduces the number of lines

```
For CHRISTIAN
Most similar character: JANE, Least similar character: RONNIE
For CLARE
Most similar character: TANYA, Least similar character: RONNIE
For HEATHER
Most similar character: MINTY, Least similar character: CHRISTIAN
For IAN
Most similar character: MAX, Least similar character: RONNIE
For JACK
Most similar character: MAX, Least similar character: CHRISTIAN
For JANE
Most similar character: IAN, Least similar character: RONNIE
For MAX
Most similar character: TANYA, Least similar character: RONNIE
For MINTY
Most similar character: HEATHER, Least similar character: RONNIE
For OTHER
Most similar character: TANYA, Least similar character: CHRISTIAN
For PHIL
Most similar character: JACK, Least similar character: RONNIE
For RONNIE
Most similar character: JACK, Least similar character: CHRISTIAN
For ROXY
Most similar character: JACK, Least similar character: CHRISTIAN
For SEAN
Most similar character: MAX, Least similar character: RONNIE
For SHIRLEY
Most similar character: MINTY, Least similar character: RONNIE
For STACEY
Most similar character: TANYA, Least similar character: RONNIE
For TANYA
Most similar character: MINTY, Least similar character: RONNIE
```

Figure 1: Most and least similar characters per character

spoken by the relevant character, while the context mostly adds noise, as it is spoken by other characters and provides little relevant information.

4 Q4

For grid search, I created one general-purpose function, grid_search, which takes the search space as its main argument and then iteratively tries each combination given the parameters specified in the search space. I then implemented the function 3 times: once for all the techniques implemented in Q1, Q2, and Q3, respectively, with the initial parameters being updated to the optimal ones found by the previous grid searches each time. I ran the grid search for Q3 first as Q3 modifies the part of the code that comes the earliest (create_character_document_from_dataframe).

My grid search found the following configuration of techniques to be optimal: EOL token removal, lowercase normalisation, stopword removal, POS tagging, punctuation separation, and gender classification.

5 Q5

I made the "cm" and "labels" variables within the plot_heat_map_similarity funcion global so that I could use them to generate similarity rankings. I then created a dictionary, called "extremes", which gave the training set characters as keys and the most and least similar validation characters (excluding the character themselves for the most similar) as values. The contents of the dictionary were printed and are displayed in Figure 1.

It can be seen that Jack, Max, and Tanya often feature as the most similar characters while Christian and Ronnie dominate the least similar characters. This is because Jack, Max, and Tanya's speech is quite generic and uses a lot of common words, such as "alright" and phrases, such as "enough is enough". This means they are less distinguished from other characters than average.

On the other hand, Christian's speech features a lot of sarcasm, which makes it quite distinct, while Ronnie doesn't use a lot of emotional words such as "amazing", distinguishing her from other characters.

6 Q6

I simply used my best configuration of techniques identified by the grid search, which was already specified above. I achieved an accuracy of 100 pct and a mean rank of 1 on the test data, indicating perfect performance.