cmpe 493 Assignment #3

Kayacan Vesek 2016400114

January 20, 2021

Problem description: "Building a recommendation system is a common task in many modern applications. The goal of a recommendation system is to identify relevant data for their users. These systems are generally based on two methods: collaborative filtering and content based filtering. While collaborative filtering exploits similar users' rates, content based filtering considers the contents of the items liked by the same user."

In this assignment, you will build a simple book recommendation system from scratch. Firstly, you will extract the contents and recommendations of books from Goodreads. To represent the contents of the books, you will implement the vector-space model. For the book descriptions, you are required to use tf-idf weighting, while you are free to choose any model for the genres. After vectorizing all books' contents, you will make recommendations for a given book by getting the most similar K books based on cosine similarity. Finally, you will calculate the evaluation metrics considering the Goodreads recommendations as the relevant (ground truth) books.

1 Model

1.1 Crawling and Preprocessing

I crawled the data and tokenize it with multithreading, so all the preproccessing for 1800 books takes 2 minutes in total.

- Page is downloaded.
- html is parsed with regex.(definetly it is an absurd thing to do, we should have used some libraries)
- removed punctuations
- removed stop-words
- tokenized words.
- calculated the occurance of each word
- calculated tf-idf scores for each books
- save books and word counts in the pickle.

1.2 Calculating Similarity

I used cosine similarity between descriptions of each books. After calculating the score, I added some constant point for each genres matches.
Descriptions are short so I did implement with different way
Because I did not keep a very long vector for each book(I did not use vector embedding for). I only compared the set of words exists in both of two books.
So it is really fast to calculate. I think 10x times faster than others.

```
def cossim_desc(bookA, bookB): # calculates the score for a given book
    for i in set(bookA.description + bookB.description):
        x = bookA.tf_idf[i]
        y = bookB.tf_idf[i]

        xx += x*x
        yy += y*y
        xy += x*y
    score = xy/math.sqrt(xx*yy)

    score += sum(myutils.GENRE_POINT for i in bookB.genres if i in bookA.genres)

    return score
```

1.3 Recommending Books

Given book data set, and the book url. It compares then sorts all the scores for the given book. Than recommends 18 similar books for a book and then calculate the average precision for these recommendations.
For a given url it calculates the recommendations in 3 seconds.

1.4 Scores

I calculated scores for each books in book.txt then I get the mean value of the scores.
Mean results are calculated as followings:

- Precision = 0.22388888888888897
- AP@18 = 0.41920883567296385

1.5 Genre Coefficient

After calculating similarity of the descriptions, to add genres' similarity I have coefficient. You can find it under myutils class named GENRE_POINT

1.6 Computation Speed

The main difference of my code is the speed of it. The full process is holding 6 minutes in total.(2 minutes parsing and 4 minutes calculating all the scores across all the books). So adding new feature and testing it is easier than ever.