Information Retrieval Project

Elise Kuylen　　*Robin Verachtert*

1 Introduction

For this project, we implemented a *content-based* recommendation system for books. Based on which books a user has like before, we tried to recommend new books, that were similar in content and would (hopefully) also interest the user.

2 Algorithms implemented

To create our recommendation system, we first had to index a set of documents - the summaries and reviews of the books in our database. Next, we constructed tf-idf vectors for these books, which we could then compare to determine the similarity of different books. We also implemented dimensionality reduction. To perform a dimensionality reduction we had to use Singular Value Decomposition. We consulted the chapter in the book describing the dimensionality reduction.

3 Design choices

3.1 Dataset collection

To get a sufficiently large dataset of books and user reviews of those books, we scraped the Goodreads database. Goodreads is a social platform that allows users to rate and review the books they have read.
We saved the information we got from Goodreads in JSON format, since this is easy to parse.

3.2 Lucene for indexing

To index the summaries and reviews of the books that we had scraped from Goodreads, we used the Lucene library. It has a lot of already implemented methods, making so that we could focus on the recommender part and most importantly the evaluation. For indexing we used a English Analyzer to pre-process the data and perform stemming.

3.3 Creating tf-idf vectors

To store the TF-IDF vectors of our books, we create the class TFIDFBookVector. Since many of the terms in our document collection will not be present in the term vector of one document, we used a sparse vector to save the TF-IDF weights of these terms. Apart from that, we also saved some information about the book, such as title, ISBN and author.

TF-IDF weights were calculated from the information present in the index: the term vectors of the documents and the overall frequency of terms in the document collection.
We saved

3.4 Recommendation

To make recommendations we used two different strategies:

- **Based on one book:** To make recommendations based on one book in the user's profile, we treated the text of the summary and the reviews for this book as a query. Next, we constructed a TF-IDF vector for this query, and compared it to the TF-IDF vectors in our matrix, using the cosine similarity measure. Once this was done, we sorted the books according to their similarity with the TF-IDF vector for the user's book and returned the first 10 results.

- **Based on all books in the user's profile** While making recommendations on the basis of one book the user has liked is interesting, it would be more interesting if we could recommend books based on the entire set of books that a user has enjoyed. However, we could not discover any algorithms that used a Vector Space Model to do this. Thus, we experimented with three different techniques to try to find a solution to this problem:

3.5 Dimensionality reduction

4 Evaluation results

4.1 Indexing

To evaluate the quality of our indexing process, we timed how long it took for different sizes of datasets. We checked the time consumption for regular indexing as well as for indexing followed by dimensionality reduction.

4.1.1 Regular

4.1.2 With dimensionality reduction

4.2 Recommendation

To evaluate our recommendation process, we looked at the precision of our results.[1] We used two users to test our system: a real user account, that of Robin Verachtert, and 'IR Test', a user account that we created for this project, and which mainly contains Fantasy novels.

4.2.1 Regular

Recommendation based on single book To evaluate the results of recommendation based on one book, we let the system make a top-10 of recommendations

[1] Recall is less relevant here, and it is also nigh impossible to check: we would have to decide manually, for each book in our dataset, whether a book is relevant for a particular user or not.

for each book for which our test user had given a score of 4 stars or higher. We then compared the books in this top-10 with the other books the user had liked previously, and calculated the precision @ 10. The average precision @ 10 for user 'IR Test' with a small of books to recommend from was 0.567. The average precision @ 10 for user 'IR Test' with a larger set of books to recommend from was 0.247.

The average precision @ 10 for user 'Robin Verachtert' with a small of books to recommend from was 0.448. The average precision @ 10 for user 'Robin Verachtert' with a larger set of books to recommend from was 0.173.

We also checked the books that were recommend manually, to find an explanation for the rather low results for precision. We found that things like names of people and countries were very influential in deciding which books were recommended - since they are often rare in the complete set of documents that was indexed. However, names and countries often do not decide the genre or subject of a book, an thus are not very relevant for recommendation.

Recommendation based on multiple books To evaluate recommendation based on a set of books, we used 80% of the user's 'likes' (books that were given 4 stars or more) as a training set to base our recommendations on. We used the 3 techniques described above to generate a top-10 of recommendations for our test user. Then we compared the content of this top-10 with the remaining 20% of the user's 'likes', which we used as our test set. From this, we again calculated precision @ 10. This gave us the following results:

- **Comparing top-10s of separate books:**
- **Adding vectors of books:**
- **Adding cosine similarity results:**

4.2.2 With dimensionality reduction

Because SVD is a very costly operation, we only used a very small data set to test this on. Otherwise we got a Heap Memory Overflow. We ran the same tests as mentioned above on this datasat and got the following results.

Recommendation based on single book The average precision @ 10 for user 'IR Test' with a small of books to recommend from was 0.411. The average precision @ 10 for user 'Robin Verachtert' with a small of books to recommend from was 0.370.

Recommendation based on multiple books

- **Comparing top-10s of separate books:**
- **Adding vectors of books:**
- **Adding cosine similarity results:**

5 Code

The code created in this project can be found at `https://github.com/verachtertr/IR_project`