Information Organization and Retrieval
Final Project Report

Benjamin Fell

Project Summary:

My project aims to evaluate the genre classification of the 69 most popular eBooks in the Gutenberg project collection (https://www.gutenberg.org/browse/scores/top). Each book in this collection has multiple genres included in its metadata, and my analysis will dive into evaluating the consistency of these genres on a high level.

Given the high specificity of genres, we will only study whether the highest-level categories available, which are Fiction (At least one Subject contains Fiction as a direct category or hierarchical category), vs Non-Fiction (None of all Subject contains Fiction as a direct category or hierarchical category). We will analyze if books classified as Fiction are similar to all other fiction books in this collection from a bag-of-words perspective. In other words, we will see if books classified as Fiction, share a similar vocabulary to all other books under that same category.

I was able to scrape and categorize the following number of books:

#Fiction Books: 43
#Non-Fiction Books: 26

We created a Fiction and Non-Fiction vector representation by aggregating all these 69 documents into vectors that represent each of these two categories (Fiction vs Non-Fiction). After that, we compared new documents (embedded with the same model) to these vectors by using cosine similarity to check if they were classified as Fiction or Non-Fiction in a consistent way. If a document was classified as some kind of Fiction and is more similar to the Fiction vector than to the Non-Fiction vector, then that would mean that that specific book was categorized consistently from a vocabulary perspective. In other words, this book has similar vocabulary as the books under the same high-level category.

The results of this evaluation were promising: Most categories in our test set were consistent with their group (>65%) from a vocabulary perspective. Considering this was a small experiment, we could improve the genre classification accuracy and genre representation vectors if we train this model with more data.

If this model is to be taken to a next step, the following applications could be performed:

- Auto-classification of e-books with this model, which could assign genres when doc embedding similarity with the genre vector is over a threshold.

- High-specificity category evaluations could be performed if the model is trained with enough documents for each label

Both these applications could reduce manual work related to genre classification, and also improve consistency among genre classification of books, which would improve searching experience of Gutenberg users.

Technical Specification

The analysis process consists of the following steps:

1. Scrape 69 of the 100 most popular books in Gutenberg project.
2. Manually assign the most relevant genres associated for each book. These are stated as "Subject" in the e-book page for each book (Eg: See https://www.gutenberg.org/ebooks/84 > Bibliographic Record > Subjects). Because we are only analyzing a low number of books, we only assigned low-specificity genres and not location-related were assigned in our analysis.
3. We re-classified each book into Fiction or Non-Fiction categories, based on if any genre of that book contained the word "fiction" or not.
4. We create a Gensim doc2vec model to build genres and document embeddings. Gensim doc2vec library creates a bag-of-words (dense vector) representation of each document, that accounts for the order and semantics of words in the document.
5. We train a model using three types of labels: Fiction, Non-Fiction, or other. The latter are those documents which will be used to evaluate our model (hence, these book's classifications into one of the two categories). With this model, we are able to obtain a vector representation for Fiction books, Non-fiction books, and individual books (which will be used for the evaluation).
6. Finally, we calculate cosine similarity between each TEST book against Fiction and Non-Fiction vectors, to see where it should be classified from a bag-of-word perspective, vs how it was actually classified.

More details are included in the python code section.

¿How does this project relates to the class?

With this project, I was able to improve my understanding of this course by developing a category evaluation analysis that considered several key concepts reviewed in class:

- **Metadata**: I was able to navigate, understand, extract and transform metadata from a books collection.
- **Collections**: I searched and selected an interesting collection of books that was appealing to evaluate.

- **Faceted Categories**: Genres in the Gutenberg collection were assigned in a faceted way. Several categories were assigned as labels, which significantly improved filtering and searching capabilities of this collection website.
- **Web Search and Crawling**: I was able to create a real scrapper that could read and download a whole collection of books.
- **Automated Semantic Similarity with Word Embeddings:** I was able to see first-hand how to create document embeddings with a bag-of-word representation, and evaluate similarity with certain categories they were assigned to.

References:

Scrapping Gutenberg project books

- https://katherinepully.com/project-gutenberg-scraper/
- https://codereview.stackexchange.com/questions/141898/sample-scraping-project-gutenberg-using-beautiful-soup-and-requests
- https://stackoverflow.com (several)

Document embeddings and Gensim doc2vec models

- https://radimrehurek.com/gensim/models/doc2vec.html#module-gensim.models.doc2vec
- https://arxiv.org/pdf/1405.4053v2.pdf
- https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4
- https://www.kaggle.com/currie32/comparing-books-with-word2vec-and-doc2vec/notebook#Doc2Vec
- https://stackoverflow.com (several)

Cosine similarity:

- https://stackoverflow.com/questions/61596101/calculating-cosine-similarity-from-a-gensim-model
- https://stackoverflow.com/questions/55035560/sklearn-cosine-similarity-convert-1d-array-to-2d-array-in-python