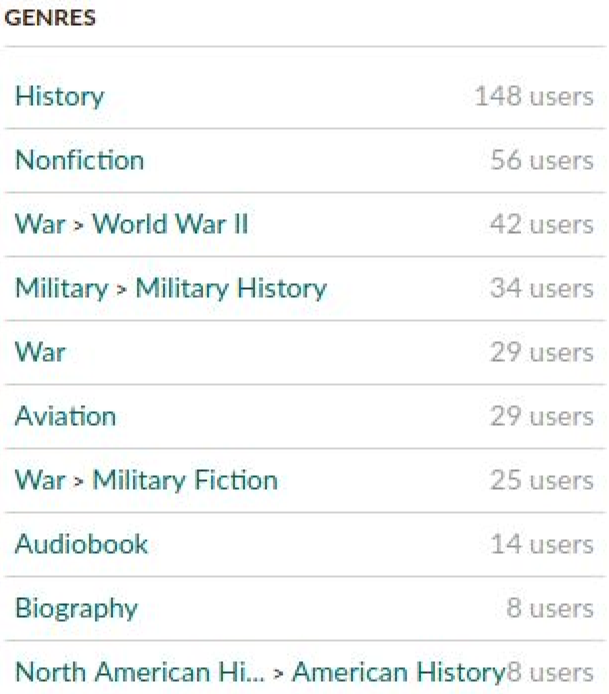
CmpE 493 - Assignment 3
Emre GİRGİN - 2016400099

Genres Encoding:

Some of the books do not have a **genre** section. (Example) For those books, they are counted as they are, which is an empty list of genres.

1. For each book, all the keywords in the genres section are extracted. Like History, Nonfiction, so on...



GENRES	
History	148 users
Nonfiction	56 users
War > World War II	42 users
Military > Military History	34 users
War	29 users
Aviation	29 users
War > Military Fiction	25 users
Audiobook	14 users
Biography	8 users
North American Hi... > American History	8 users

2. For each book, sub-genres are counted as a separate genre. For the example above, “War” and “World War II” counted as **two separate genres** and added to the genres list separately. This means for the lines “War > World War II” and “War > Military Fiction”, the “War” genre added twice. This creates a **duplication** for this step but they will be removed later on.
3. For all the books used while creating the recommendation system (let’s call it **trainset**), their genre list is summed up into a single list.

4. The duplications in the genre list generated from the trainset are removed. As the result, we end up with a list of genres that covers all of our trainset. Let’s call this list the **corpus genre list**.

History	Fiction	Horror	Romance	Business	Politics	Self Help	...

Corpus Genre List

5. A vector of zeros, with the size of the corpus genre list, is generated for each book. Let’s call it **genre vector**.
6. The genre list of each book is encoded into its genre vector. For each genre that the book’s genre list contains, we look at the index of it from the corpus genre list, then put a 1 into the genre vector of that book in that index. In other words, the genre list of each book is **one-hot encoded** into its genre vector, based on the corpus genre list. Thus, the duplications are discarded automatically.

War	History	Romance	...

Genre List

1	0	0	1	0	0	0	...

One-Hot Encoded Genre Vector (See Corpus Genre List indices above)

7. The genre vector of each book is normalized based on **Euclidian Distance (L2Norm)**.
8. The similarity score between two genre vectors is calculated using **Cosine Similarity**.

Model Parameters:

- If a crawled XML does not have a **title** or **recommendations** section, it is **discarded**. They can not be integrated into the recommendation system. The malfunction can be faced due to server-side problems.
- All the logarithms are calculated on **base 10**.
- Some punctuations which could not be decoded, are decoded manually:
    - &quot -> “
    - &#39 -> ‘
    - amp& -> &
- Some of the HTML tags like <br>, <p>, <div> are places inside the description in the website. Those are replaced with whitespace.
- Before applying TF-IDF, some **normalization** steps are adopted:
    - Punctuation removal (using *string* module)
    - Case folding (lowercase)
    - Tokenization
    - Stopword removal (Stopwords are also included to project files. They are adopted from the NLTK library.)
- If a vector is a vector of zeros (ie. the vector of the empty description or empty genres) its normalized value is again vector of zeros.
- The default **alpha** is 0.75, meaning that 75% of the similarity score comes from the cosine similarity between descriptions and the rest from genres.
- Any maximum or minimum threshold is adopted.
- The number of terms in the corpus depends on the books in the trainset.