Introduction to Recommendation Systems

Armin Pousti
Dmitrii Vlasov
Mcgill Department of Computer Science
COMP 480
Professor: Joseph Vybihal
May 21, 2023

In this report, we will overview recommendation systems, their principles, classifications, and the different methods for data collection they use.

Recommendation systems can be found anywhere; any website recommending articles, music, movies, people, or webpages uses recommender systems that predict ratings or preferences (Kumar, P. Pavan, 2021).

A recommendation system is an algorithmic approach that uses different information filtering techniques to suggest a relevant product to a consumer. These recommendations are made on various criteria, including past purchases, search history, demographic information, and other factors to suggest the item closest to a customer's preferences, interests, or behavior patterns.

Recommendation systems are becoming essential in our digital world, where consumers are overwhelmed by choice and need help in decision-making. There are several techniques for filtering data, but some main ones include collaborative filtering, content-based filtering, hybrid approaches, knowledge-based filtering, and context filtering which are also referred to as the architecture of recommendation systems (Robillard, Martin P, 2014).

Each recommendation system follows a set of principles regardless of the filtering approach that the developer implements. The first principle is personalization; each recommendation system aims to give personalized recommendations to individual users based on their interests, enhancing the user experience and finding relevant items for each individual which separates a recommender system from a search engine. The second principle is data collection; each recommendation system also relies on big data on users and items to make proper recommendations, even the best recommender systems cannot give relevant suggestions unless they have an abundant amount of good data to work with. The third principle is filtering and ranking; the recommender systems need to filter and rank the available items to provide the most relevant suggestion to users using one of the information filtering techniques available. Lastly, it is essential to continually evaluate each recommendation system's performance and gather user feedback. Some techniques used for feedback are click-through rates, conversion rates, and user satisfaction.

Recommendation systems start with an extensive collection of data, and using recommendation engines, they use algorithms and the data to recommend the most relevant item to a particular user. This user-item interaction that shows the working principles of the recommendation system can happen in two ways: User-Item interactions and exploration of diverse content. For User-Item interaction, information about the user is obtained by analyzing the user's profile and preferences. For the exploration of diverse content, the information about the product is retrieved by taking into account the product's characteristics, such as rating, the number of purchases made, and likes and dislikes.

Despite the extensive research and agreed-upon standards, concerns have been raised about the lack of reproducibility of recommender systems algorithms in research (Cremonesi, Jannach 2021). Scholars have found it difficult to reproduce the research results, with a significant percentage of papers being almost impossible to verify or falsify. Moreover, even minor variations in evaluation setups can significantly impact the outcome of the experiment which makes the analysis even more challenging. Surprisingly, recent studies have shown that certain long-known methods outperform the latest deep learning methods in several cases suggesting major issues with the field. The paper shows the signs of stagnation in the field of algorithm research with new models being proposed without consistent improvements. Researchers sometimes compare their methods to weak baselines, and flexible experimental designs can introduce researcher bias. Despite all these problems, there is a lack of crisis awareness, and the current progress in AI and machine learning research leads to more and more publications of algorithms that may not adequately demonstrate progress.

The paper suggests several improvements to the existing process of algorithm research to practically eliminate these issues. To ensure reproducibility, the authors suggest that researchers make their data publicly available and properly documented. This also includes sharing the preprocessed data along with data splits and negative samples used in experiments. In addition, authors should share all relevant code, including the new method, baselines, data preprocessing, clear instructions, software/hardware requirements, installation scripts and configuration files. Standardized benchmarks (although researchers should be cautious not to overly focus on specific problem settings that may potentially limit exploration of other important research questions) can help alleviate some of the reproducibility issues. However, it is crucial to provide a specific train-test split and detailed information about the evaluation procedure and metrics. Even with all the mentioned crises that have risen in recommendation research, it can not be denied that the field has made substantial improvements over the years that contributed to improvements in marketing and opened doors for more research.

In this report, as an introduction to recommender systems and how they function, we will focus on content-based filtering, one of the architectures of recommendation systems.
This architecture recommends items based on their attributes or contents. The algorithm will suggest items to the user that are similar to the contents and attributes of items that the user prefers. This recommender system assumes that a user preferred a specific item in the past and is likely to prefer a similar item in the future. Content-based filtering relies on two types of background data: 1)a set of users, and 2) a set of categories or keywords extracted from item descriptions. The recommender system then calculates a set of items most relevant and similar to the item already known to the user. Similarities between the items are the keywords that were extracted from the item’s content descriptions.

In the next part of the report, we look at the TF-IDF model analysis, which is an application of content-based filtering, how it works, and the code implementation of this approach on a dataset of movies.

TF-IDF model analysis

Term Frequency - Inverse Document Frequency is a technique used in natural language processing and information retrieval to measure the importance of a term within a document that is part of a corpus of documents. It aims to balance the relative significance of a given term compared to its occurrence across the entire document collection.

The input and output of a given TF-IDF model depend on the specific application of the model. In the code provided with this report, the input and output are the following:

Input: A corpus of documents

Output: A similarity score matrix indicating the similarity between documents based on their TF-IDF representation. This is obtained by computing the cosine similarity on the TF-IDF vectors of the documents.

Procedure

Step 1: Tokenization

The first step is to break down each document in the corpus into individual words or tokens. In the vectorizer.py module provided with this report, the documents went through an initial preprocessing of removing certain words, which was done to better demonstrate how the model works on simple examples. This is not necessary, however, as the model can deal with commonly used words that carry little to no meaning (with the IDF term).

Step 2: Term Frequency (TF)

TF measures the frequency of a term within a single document. It is calculated using the following formula:

$TF(t,\ d)\ = \frac{number\ of\ time\ the\ term\ t\ occurs\ within\ the\ document\ d}{total\ words\ in\ document\ d}$ where t is a given term in document d

The idea of TF is that the more the term appears within the document the more important it is to that document.

Step 3: Inverse Document Frequency (IDF)

IDF measures the uniqueness of a term in the corpus. It helps to identify terms that are important in a specific document but occur rarely across the corpus. It is calculated with the following formula:

$$IDF(t,\ D)\ = log(\frac{number\ of\ documents\ in\ corpus\ D}{number\ of\ documents\ in\ D\ that\ contain\ term\ t})$$

The IDF component of the TF-IDF model allows the model to downweigh or eliminate the words that are frequently used in language, such as stop words (ex. “the”, “is”, “and” etc). These words typically do not provide much meaningful information; therefore, by eliminating them, the model can focus more on the words specific to individual documents.

Step 4: TF-IDF

The TF-IDF score of a given term is obtained by multiplying its TF value with its IDF value. The TF-IDF score is directly proportional to the frequency (high TF) and to the rarity of a term across the corpus (high IDF).

Step 5: Document-Term Matrix

Create a matrix, where each row corresponds to a document and each column corresponds to a unique term in the entire corpus. The matrix values represent the TF-IDF scores of the terms in the respective documents.

Step 6: Cosine Similarity

For each pair of documents calculate their cosine similarity. The cosine similarity between documents A and B is calculated by taking the dot product of A and B divided by the product of their magnitudes. In the context of the TF-IDF matrix, each row is an n-dimensional vector of a single document (that still contains all n unique tokens of the corpus with values 0 if the terms are not present in a given document)

Step 7: Provide Recommendation

The cosine similarity matrix is a square matrix.

- Diagonal elements of the matrix represent the document's similarity to itself with a score of 1.
- Off-diagonal elements represent the similarity between pairs of documents. The similarity scores range from -1(high dissimilarity) to +1 (high similarity).

Providing k recommendations for a given document involves taking k highest similarity scores for a given document.

Vectorizer.py documentation

```python
def preprocess(text):
    """
    Preprocesses the given text by converting it to lowercase, splitting it
into words,
    and removing common stopwords.
    :param text: (str) text to preprocess
    :return: (list) preprocessed list of words
    """
def compute_tf(words):
    """
    Computes the term frequency (TF) for each word in the document
    :param words: (list) the list of words in the document
    :return: (dict) the term frequency dictionary
    """
def compute_idf(tokenized_docs):
    """
    Computes the inverse document frequency for each term
    :param tokenized_docs: (list) the list of tokenized documents
    :return: (dict) the inverse document frequency dictionary
    """
def compute_tfidf(tf, idf):
    """
    Computes the TF-IDF values for each term in the document
    :param tf: (dict) term frequency dictionary
    :param idf: (dict) inverse document frequency dictionary
    :return: (dict) TF-IDF dictionary
    """
def compute_cosine_similarity(docs):
    """
    Computes cosine similarity matrix for a list of documents
    :param docs: (list) the list of documents
    :return: (numpy array) cosine similarity matrix
    """
def get_recommendation(cos_sim_matrix, k, index):
    """
    Gets the best k recommendations based on the cosine similarity matrix for a
given index
    :param cos_sim_matrix: (numpy array) cosine similarity matrix
    :param k: (int) the number of the best recommendations
    :param index: (int) index of the document for recommendations
    :return: (numpy array) indices of the best k recommendations
    """
```

TF-IDF model application to film recommendations

Input: imdb_top_1000.csv dataset from Kaggle, name of the film, number of recommendations for the given film

Output: a list of k recommendations for a given film

The csv file contains certain information on top 1000 films from IMDb. Given the purpose of the experiment, only the following columns were used: Series_title, Genre, Overview, Director, Star1, Star2, Star3, and Star4 as they provide the best information for content-based filtering. These columns were concatenated into a single column called text_data. This new column was then used as a corpus in the TF-IDF model, where a single entry in this column represents a single document/film. This approach is not perfect and would benefit from a more exhaustive text_data column (a movie plot instead of a short overview could be used instead to provide the best recommendations) or from a bigger list of films. However, this technique still provides adequate recommendations, especially for multi-chapter films or when k is low. A possible improvement to this model would be creating a list of movies the user likes and then getting the recommendation based on multiple films. This can be achieved using the same model by creating a new film list entry containing information about all liked items. As an example, if a user prefers two films, “The Godfather” and “The Dark Knight”, then a new entry to the dataset can be created by concatenating the titles, the overviews, and the names of the directors and actors in both movies. This new entry could then be used as an input to the get_recommendation function.

Work Cited:

- Kumar, P. Pavan, et al. *Recommender Systems: Algorithms and Applications*. CRC Press, 2021.
- Robillard, Martin P., et al. An Introduction to Recommendation Systems in Software Engineering. Springer Science and Business, 2014.
- Cremonesi, Paolo, and Dietmar Jannach. "Progress in Recommender Systems Research: Crisis? What Crisis?" *AI Magazine*, ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/18145. Accessed 20 May 2023.
- "What Is a Recommendation System?" *NVIDIA Data Science Glossary*, www.nvidia.com/en-us/glossary/data-science/recommendation-system. Accessed 20 May 2023.
- "An In-Depth Guide to How Recommender Systems Work." *Built In*, builtin.com/data-science/recommender-systems. Accessed 20 May 2023.
- Sharma, Nikita. "Recommender Systems with Python-Part I: Content-Based Filtering." *Medium*, 21 Sep. 2021, heartbeat.comet.ml/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831.
- "The TF*IDF Algorithm Explained." *Onely*, 30 Jan. 2023, www.onely.com/blog/what-is-tf-idf/.