Enhancing Book Recommendations in Local Bookstores: A Case Study of Zangak Bookstore

Mane Davtyan

BS in Data Science
American University of Armenia .
mane_davtyan@edu.aua.am

Abstract

This paper suggests a customized approach for bookstore recommendation algorithms using Zangak Bookstore as a case study. With the help of content-based filtering and modern natural language processing algorithms, our models generate personalized book recommendations based on customer interests, past experiences, and book characteristics. The process uses content-based filtering to create item representations based on attributes, including titles, authors, languages, and descriptions. We use the KeyBert model to extract essential keywords from translated descriptions to improve the quality of recommendations. Based on the book's features, the paper presents two primary models mentioned in Chapter 4. Subsequently, the models undergo expansion to consider customers' preferences based on their past purchases. Eventually, the resulting extended model integrates the two previous methodologies. Our methodology presents a fresh approach to tailored book recommendations for local bookstores in Armenia, improving customers' browsing and buying experiences, specifically at Zangak Bookstore, and possibly contributing to the broader e-commerce scene.

Keywords: Recommendation systems, Books, Zangak Bookstore, Content-based filtering, KeyBert, BERT, Translation, Keyword Extraction

1 Introduction

Personalized recommendation systems are essential tools for improving user experience and increasing sales in the e-commerce industry. Unfortunately, the local market lacks recommendation systems based on user preferences. This paper discusses a specific case of bookstores and how to build a recommendation system based on the bookstore data. Zangak Bookstore is one of the biggest bookstore chains in Armenia, selling more than 20,000 books and stationery in their stores. However, it does not have book-recommending tools. The only recommendation from the store is based solely on the employee's knowledge, with the help of a search program based on the book's title and author. The paper addresses this issue by providing an enhanced approach to book recommendations. Our recommendation models use data analysis and natural language processing (NLP) techniques to build customized book recommendations. We introduce two primary versions of recommendation models based on book features and customer purchase history. Additionally, we discuss the data collection, preprocessing, and analyzing process, as well as the usage of Bidirectional Encoder Representations from Transformers (BERT) and KeyBert models in the system [1, 2]. The most common recommendation systems are accepted to be matrix factorization models. However, we do not apply this model due to data feature mismatch. Matrix factorization models are constructed with user preference information, such as individual ratings on each item. Such systems use SVD (singular value decomposition) models as their primary base, mainly used in a wide feature variety of datasets [3]. Since our dataset lacks such information, we decided against using Matrix Factorization. Instead, we consider an alternative content-based filtering method appropriate when user ratings are absent, and other features describing the items are present[4]. Since content-based filtering matches our data when building the recommendation system, we continue with it. Furthermore, we discuss the creation of word embeddings using natural language processing models, such as the BERT pre-trained model [2, 5–7]. We extract keywords from translated descriptions using the KeyBert model [1] to measure book similarity. Cosine similarity is then operated as a comparison metric, defining the similarities between embedding vectors for each keyword [8]. We base our recommendation system on similarity metrics and select the top ten best-resulting similarity scores with the corresponding book information to be recommended to the customer. Similarity scores are taken as valid metrics after we do an experiment checking their performance within purchased and random books. Compared to the random books set, the similarity scores within book embeddings outperform in purchased books. Thus, we are convinced that similarities are favored to show the semantic similarities between the books. As a result, we created a particular Zangak bookstore recommendation system, which builds individualized recommendations for registered and non-registered bookshop customers.

2 Data

We have two sources of data: scrapped books data (Table 1) and user experience data provided by Zangak bookstore (Table 2). We have scraped the book data from the zangakbookstore.am web page using Python. We extract information on more than 21,000 books, which consist of the following features: Title, Author, Price, Publisher, ISBN (13-digit international unique code for each book), Code (unique ID used by the bookstore), Publishing Year, Language, Age License, Weight, Size, Cover Type, Pages Number, Description. The second data source includes a sample from customers' past purchase information provided by Zangak. It consists of the customer's gender, birthday, purchase title, purchase quantity, price, purchase date, and time.

Table 1: Sample of Scraped Dataset of Books

Title	Author	Price (AMD)	Publisher	Language	Description
Dinner Time	Sophie Giles	2300	Award	English	The Snuggle Bun...
The brain	Jack Challoner	6200	Welbeck ...	English	The next incredible ...
Armenian Genocide	Nikolay Hovhannisyan	1000	Zangak	Armenian	NaN
Marilyn Monroe	Maurice Zolotow	3800	Parragon	Russian	Dieses Buch beric...
Изумруд раджи	Agatha Christie	1600	Эксмо	Russian	В этот сборник ...

Table 2: Sample from Customer Data

ID	GENDER	BIRTH_DATE	PURCHASE	QUANTITY	PRICE	AGE
1	female	1977-08-17	Egypt Guidebook	1	7660	46
1	female	1977-08-17	Double Net	1	2700	46
1	female	1977-08-17	Chair	1	3763	46
1	male	1985-02-01	Gift for Number	1	2364	39
1	male	1985-02-01	White Jewel	1	2361	39

2.1 Data Preprocessing

The missing values for the bookstore scraped data for description, title, author, publisher, and language are filled with an empty string, while the numerical features fill the nulls with 0s. The data types are corrected, taking ISBN, Code, and the other numerical categories as int32 or float64 type. The customer data is changed using regular expressions, and the customer's gender, birthday, and purchases are extracted from the provided initial concatenated strings. Columns such as Purchase, Purchase Date, Purchase Time, Quantity, Price, gender, Birth Date, and ID are created. The missing values are filled for the customers as well.

In the book data set, around 33% of the descriptions are scraped in English, while the rest are given in Russian, Armenian, Spanish, Italian, German, and various other languages. The non-English descriptions are a particular issue to address since the keywords of each description of the books have to be extracted for later usage. To handle descriptions in different languages, we use a deep-translator library in Python to translate the non-English descriptions into English. The translation aims for higher accuracy in later keyword extraction [6]. Books that do not have descriptions are replaced with an empty script, out of which no keywords can be later extracted. Next, the keywords are extracted on top of the translated descriptions. The Age column is created based on the Birth Date feature for customer data. All null values are filled with either the average values or empty strings to avoid errors in the future usage of the models.

Since there are books from different publishers with the same author, titles, and language, there is a need to handle the duplicates of such books. The repetition of books may result in inefficient recommendation results by repeating the book variations instead of recommending other choices to the customers.

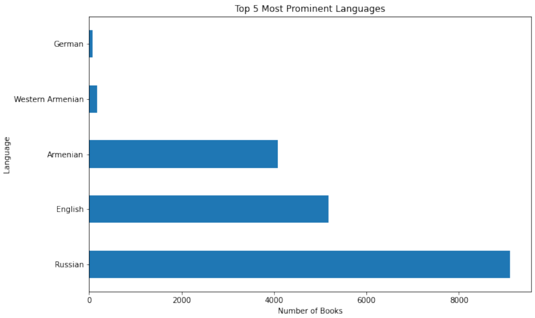


Figure 1: Languages of the Books

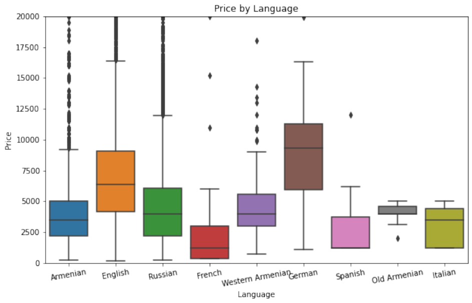
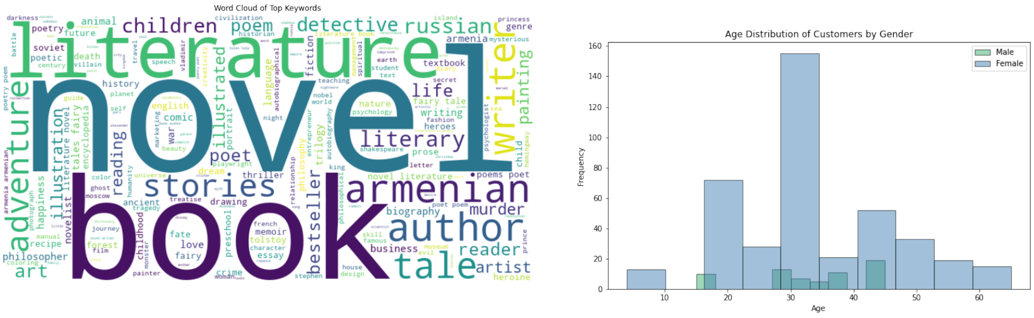


Figure 2: Price through the Languages

The rows of duplicated Titles are extracted from the book's data set, and then the keywords and descriptions are selected correspondingly. Using the BERT model, the keyword embeddings are defined, and the embeddings for the descriptions are created[2]. Later, based on the embedding values, the cosine similarity scores are calculated between each row's keywords and the description. Based on the highest cosine similarity score, the row with the most outstanding results is kept in the data set. The rest of the rows with the lower cosine similarity score are dropped and do not participate in the recommendation process.

2.2 Data Analysis

The most widespread language in the database is Russian, followed by English and Armenian books (Figure 1). On average, the prices vary from 2000 AMD to 10,000 AMD, but depending on the language in which the books are published, the price range may slightly change (Figure 2). For example, German books have a higher average price value than Spanish, Armenian, or French books, which are the cheapest. We also visualize the most used words in the books and their descriptions. The words concerning literature are the most common, such as a novel, book, author, writer, bestseller, and poetry. Afterward, words such as detective, illustration, stories, poet, and others are detected (Figure 3a). Another visualization done for the customer's sample data provided presents the distribution of cardholders based on their gender and age range (Figure 3b). Most customers in our data sample are females, with an average age of 30 to 40.

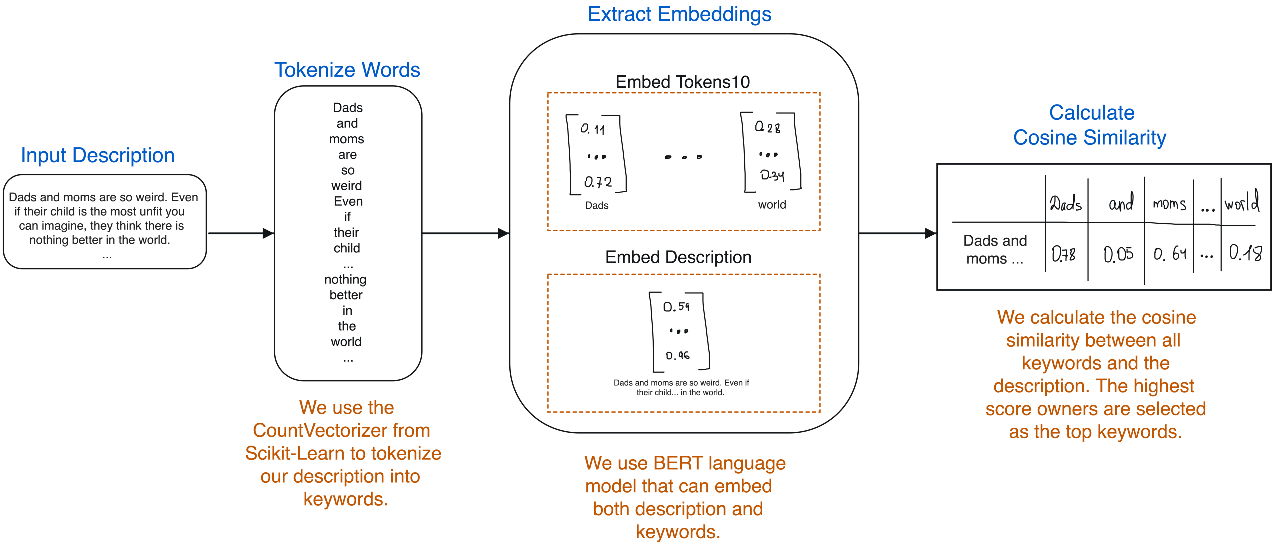


(a) **(b)**

Figure 3: (a) Wordcloud of Keywords in the Books, (b) Customer Age Range

3 Methodology

We use content-based filtering to build feature-based profiles for books and compare them to the user's past experiences[4]. We consider the feature similarities between the book's author, title, language, and description keywords to indicate possible recommendations. For customers, we extract past purchased books and consider the similar features of such books. We recommend based on the user's past purchase and/or the book title provided. First, the descriptions are translated into English to achieve the working models, as mentioned in the dapreprocessinging section. Later, the descriptions are processed through the KeyBert model to extract keywords. KeyBert is a minimal method for keyword extraction using the BERT language model[1]. BERT is first used to extract document embeddings to obtain a document-level representation[6]. Next, word embeddings for N-gram words and phrases are extracted. Lastly, KeyBert applies cosine similarity to identify the terms or phrases most closely resembling the document. You can see the structure in more detail in Figure 4.



We use the CountVectorizer from Scikit-Learn to tokenize our description into keywords.

We use BERT language model that can embed both description and keywords.

We calculate the cosine similarity between all keywords and the description. The highest score owners are selected as the top keywords.

Figure 4: The Structure of KeyBert Model[1]

We create the word embeddings using the BERT model. BERT is a powerful pre-trained language model developed by Google that can understand the context and semantics of natural language text [2]. The tokenization done at first involves separating the input text into smaller tokens. BERT uses a technique called WordPiece tokenization to break words up into smaller subword units. It can process non-vocabulary words by breaking them down into recognizable subword components. We break our descriptions into sentences and later words and extract their embedding vectors using tokenization, which maps high-dimensional vectors in an embedding space[9]. BERT uses previously learned word embeddings to represent these tokens. Significant texts transfer these embeddings to the BERT model during the pre-training phase. Those embeddings capture different grammatical features of words, including their semantic and lexical meanings, based on the context in which they appear[5]. The vector embedding of each token, typically with dimensions of 768 or 1024, captures a rich representation of the meaning of each phrase. You can see more details in Figure 5.

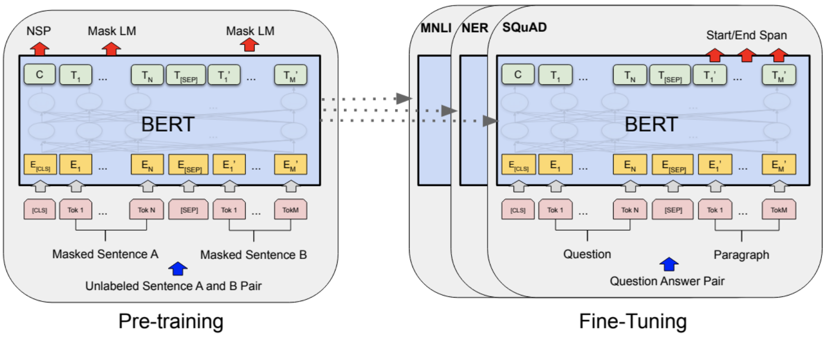


Figure 5: Pre-training and fine-tuning procedures for BERT[2]

After tokenizing descriptions and extracting the word embeddings, similarities between those embeddings are counted using cosine similarity. A cosine similarity metric compares two vectors in a multidimensional space[10]. Cosine similarity is frequently used in natural language processing (NLP) to evaluate how similar word embeddings, phrase embeddings, or document embeddings are to one another[11]. The cosine of the angle formed by two vectors is measured by cosine similarity. Euclidean norms are computed by dividing the product of the magnitudes of the two vectors by their dot product. The cosine similarity (similarity$(\mathbf{A}, \mathbf{B})$) between two vectors, $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$, can be found mathematically as follows:

$$\text{similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

In this case, the notation $\mathbf{A} \cdot \mathbf{B}$ indicates the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$. The magnitudes (lengths) of vectors $\mathbf{A}$ and $\mathbf{B}$ are denoted by the symbols $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$, respectively.

Maximum similarity is indicated when the vectors are aligned in the same direction, as evidenced by a cosine similarity value of 1. Maximum dissimilarity, conversely, is shown when the vectors are aligned in opposing directions when cosine similarity obtains the value of -1 [8].

Lastly, we consider the similarity scores as an indicator of book resemblance and recommend future purchases based on the highest similarity scores achieved between the embeddings.

4 Experiments

We have two main versions of the working model for the recommendation system, with one of their extensions. In the versions, we consider book features such as title, author, language, keywords, and customer ID, through which the books purchased in the past are extracted.

4.1 Version 1

The first version of the model takes the book's title, author, language, and the corresponding keywords extracted from translated descriptions and concatenates them. Next, BERT is used to construct embeddings for these concatenated strings. Within our model, BERT converts the concatenated strings into dense, high-dimensional vectors that effectively represent the text's semantic meaning. The associations between the books' numerous characteristics—such as their names, authors, languages, and keywords taken from their descriptions—are encoded in these embeddings. By embedding the concatenated strings, our model represents the multi-dimensional characteristics of each book in a continuous vector space. This enables the model to capture minor similarities and relationships between books that may not be apparent from individual features alone. The detailed architecture of the model is presented in Figure 6.

Based on their embeddings, we calculate the similarity scores between the input book and Zangak's book database. The similarity scores allow the model to identify books with embeddings that align closely with the grammatical features of the user's input. The model selects the top 10 highest similarity scores in descending order and later identifies the corresponding books from our dataset. The selected ten books are then recommended to the user as personalized recommendations. As an instance, if the book with the title "The Metamorphosis and Other Stories" is requested, we embed the concatenated title, author, language, and keywords ("The Metamorphosis and Other Stories Franz Kafka English Gregor insect salesman Samsa monstrous") into a word embedding, calculate the similarity scores between our dataset's 18,753 such word embeddings. We get an array of similarity scores of 18,753, where the top 10 similarity scores with their corresponding book information are recommended to the customer.

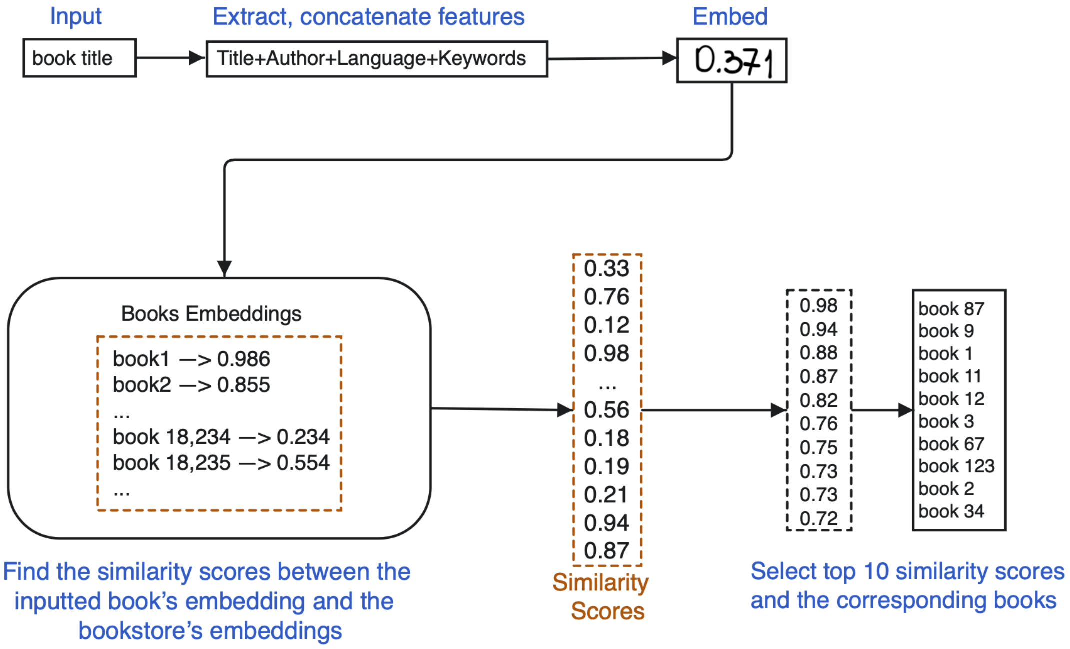


Figure 6: The Structure of Version 1

4.2 Version 2

Customer ID is inputted in Version 2 of the model instead of the book title compared to Version 1. The model first searches the customers' dataset for previous purchases and then extracts the corresponding embeddings using the BERT model. For each purchase, similarity scores are calculated between the purchased books and the data set scrapped in the data collection process. So, if there are **n** books in our data set, and the customer made **m** purchases, **mxn** similarity scores are being calculated. See more details about the architecture in Figure 7. The similarity scores obtained from comparison with **m** purchases are averaged; thus, we end up with an array of averaged similarity scores of length **n**. The top 10 highest similarity scores are selected again with their corresponding book information and recommended to the customer. For instance, the first customer has made five purchases before. The indices of purchased books are found according to their Title, Author, Language, Keyword concatenation, and word embeddings. Based on these five embeddings, we calculate similarity scores within 18,753 non-duplicated books from our data set. Thus, a matrix of size [5:18,753] is generated. We calculate the average similarity score for each book from our dataset based on past purchases, axis = 0, and get the final array of similarities with a length of 18,753. Out of the 18,753 similarity scores, the top 10 are selected accordingly, with their corresponding book information.

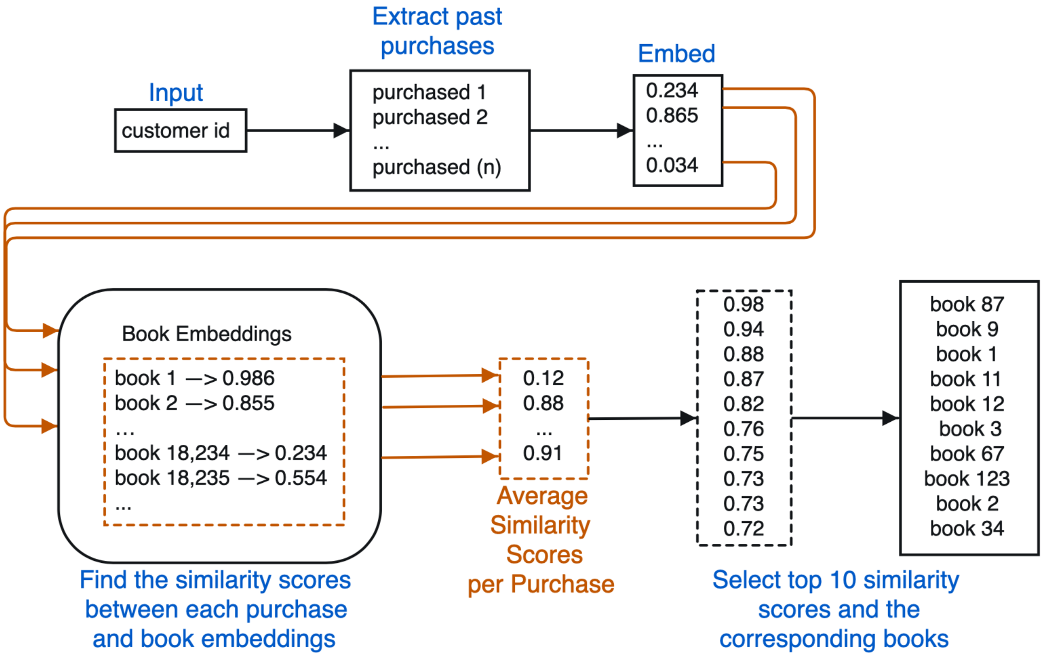


Figure 7: The Structure of Version 2

4.2.1 Version 2: Extension

The extended Version 2 combines the parameters of the past two versions. The book title and customer ID are inputted into the model. The customer's purchases and corresponding embeddings are extracted. The embeddings for the inputted book title are also extracted. The embeddings for the customer's past purchases are averaged and treated as a single embedding. If only one book title is provided, embeddings of it are used for analysis. These two embeddings are then combined using a weighted average approach, where the input title embedding is given a weight of 0.7, and the average purchased embedding a weight of 0.3. This combined embedding calculates similarities with other books in our database. Please see the architecture in Figure 8. For all the versions of the models, the embeddings for the purchased books are excluded from the book dataset during the recommendation process.

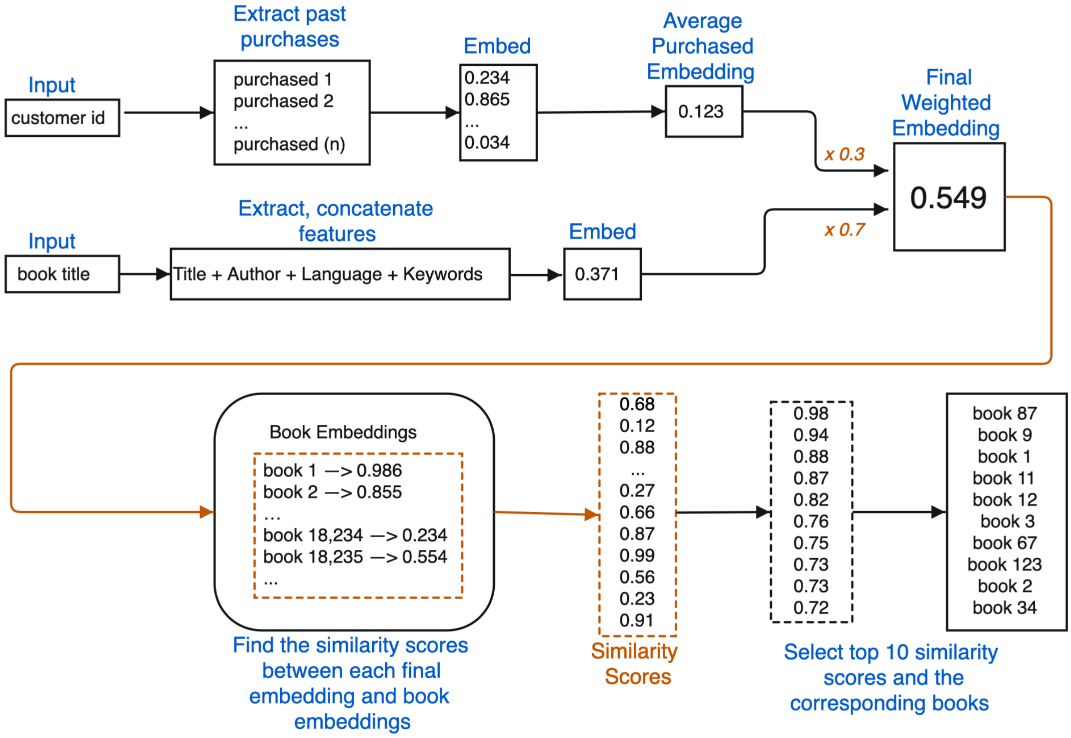


Figure 8: The Structure of Version 2 Extension

5 Evaluation

The recommendations are evaluated based on the similarity scores. This metric is indicative of the quality of the recommendations. It is chosen after a thorough experiment with similarity evaluation. We first compute the average similarity of past purchases for every customer. Next, other books with similar vector sizes (equal to the number of books the customer purchased minus one) are randomly selected for each customer's randomly selected purchase, and their corresponding similarities are calculated. We create a hundred such pairs of random book tables and calculate the average cosine similarity scores between their keyword embeddings. We compare all the similarity scores with those of the original books purchased. For the first customer, as an instance, the purchased books similarity average score is 0.746. The average similarity scores for the randomized 100 book tables are 0.515, 0.497, 0.586, and so on. The results for this customer represent how semantically connected their purchased books are based on similarity, compared to a randomly taken group of books. For all customers in our sample data, the average similarity scores of purchases consistently outperformed the scores of books chosen randomly during the experiment. This contrast highlights the better performance of the similarity score within purchased books compared to randomized experiments as a reliable indicator of how close books with similar features are to one another (Figure 9).

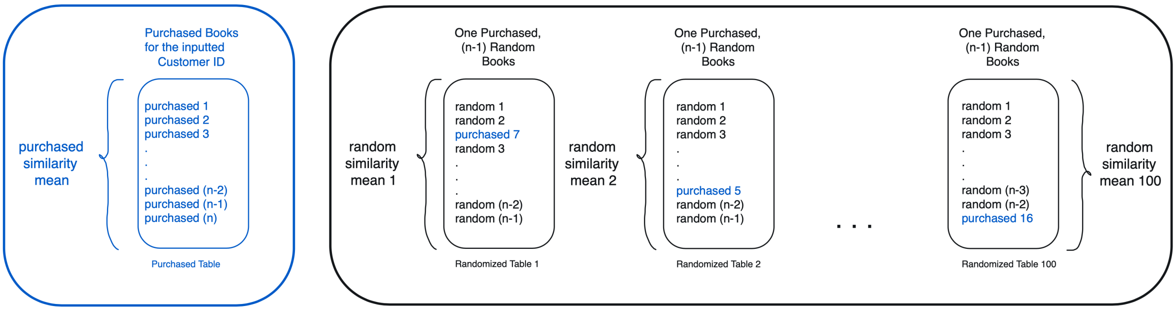


Figure 9: The Structure of Evaluation Experiment

6 Conclusion

We develop a solid structure for personalized recommendation systems for Zangak bookshop, with valuable applications for improving user experience and promoting customer satisfaction. The recommendation engine analyzes several features and uses content-based filtering to build feature-based profiles for each book and user. The system recommends ten books based on the provided book title, customer ID, or both. It considers the user experience with past purchases from the bookstore and the book features, such as the title, author, language, and keywords describing the book. Our research spreads the foundation for a refined recommendation system customized to Zangak Bookstore's and its customers' needs. Further research assumes that other factors like user reviews, ratings, and demographic information should be considered to improve recommendation efficacy and accuracy. Expanding the scope of data features, the recommendation system can offer even more personalized and relevant suggestions to Zangak Bookstore customers. More specifically, customers' demographic features, such as gender, age, and the price ranges of usually purchased books, can help future research cluster the customers and, in the recommendation, consider the similarities between the books and similar customers simultaneously. The mentioned changes can be applied after thoughtful data processing and more profound research on advanced applications of content filtering models.

References

[1] Grootendorst, M.: KeyBERT: Minimal keyword extraction with BERT (2020)

[2] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[3] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)

[4] Son, J., Kim, S.B.: Content-based filtering for recommendation systems using multiattribute networks. Expert Systems with Applications **89**, 404–412 (2017)

[5] Lebret, R.P.: Word embeddings for natural language processing. Technical report, EPFL (2016)

[6] Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., Cao, G., Jiang, D., Zhou, M., et al.: K-adapter: Infusing knowledge into pre-trained models with adapters. arXiv preprint arXiv:2002.01808 (2020)

[7] Golchin, S., Surdeanu, M., Tavabi, N., Kiapour, A.: A compact pretraining approach for neural language models. arXiv preprint arXiv:2208.12367 (2022)

[8] Soyusiawaty, D., Zakaria, Y.: Book data content similarity detector with cosine similarity (case study on digilib. uad. ac. id). In: 2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA), pp. 1–6 (2018). IEEE

[9] Chen, Y., Perozzi, B., Al-Rfou, R., Skiena, S.: The expressive power of word embeddings. arXiv preprint arXiv:1301.3226 (2013)

[10] Xia, P., Zhang, L., Li, F.: Learning similarity with cosine similarity ensemble. Information sciences **307**, 39–52 (2015)

[11] Li, B., Han, L.: Distance weighted cosine similarity measure for text classification. In: Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings 14, pp. 611–618 (2013). Springer