EXPLAINABLE RECOMMENDATIONS USING EXTRACTED TOPICS FROM ITEM REVIEWS AND WORD MATCHING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT TUNÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2022

Approval of the thesis:

EXPLAINABLE RECOMMENDATIONS USING EXTRACTED TOPICS FROM ITEM REVIEWS AND WORD MATCHING

submitted by **MERT TUNÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil KALIPÇILAR
Dean, Graduate School of **Natural and Applied Sciences** ____________

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** ____________

Assoc. Prof. Dr. Hande Alemdar
Supervisor, **Computer Engineering, METU** ____________

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU ____________

Assoc. Prof. Dr. Hande Alemdar
Computer Engineering, METU ____________

Assoc. Prof. Dr. Tansel Dökeroğlu
Software Engineering, Çankaya University ____________

Date:

ABSTRACT

EXPLAINABLE RECOMMENDATIONS USING EXTRACTED TOPICS FROM ITEM REVIEWS AND WORD MATCHING

Tunç, Mert
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Hande Alemdar

August 2022, 83 pages

Explanation in the recommendations is a crucial aspect in many applications to share reasoning and context with the users in addition to the recommended item. In this thesis, an innovative method for generating explainable recommendations is designed, implemented, and tested. The proposed design consists of extracting some phrases from the user's written review texts, assigning them to the users as preferences and items as their features, and then generating recommendations using the similarities between these assigned phrases. In such a design, since the recommendations are made using phrases that are understandable by people, the exact same phrases can be used to explain the reasoning behind the recommendations. Not many studies, however, uses keyword extraction techniques and word vectorizers to generate recommendations. Due to the lack of work in the area, it is decided to study such an algorithm that use keyword extraction and word vectorizers to uncover its capabilities. To evaluate the proposed recommender design, alongside of calculating numerical results for the quality of the recommender, a user study with 15 people is conducted. These experiments showed that people like 55% of the recommendations generated by the proposed method, while 58% of the explanations for the recommended items are found meaningful.

Keywords: explainable recommendations, explainable machine learning, topic extraction, KeyBERT, Yake

ÖZ

AÇIKLAMALI ÖNERİLERİN KULLANICI YORUMLARINDAN ÇIKARSANAN KELİMELER VE KELİME EŞLEME İLE ÜRETİMİ

Tunç, Mert
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Hande Alemdar

Ağustos 2022 , 83 sayfa

Ürün önerileri için sunulan açıklamalar, önerilen öğenın yanında kullanıcılarla ürün ile ilgili ek bilgi sağlayan ve önerinin anlamlı bulunmasını sağlayan bir husustur. Bu tezde, açıklanabilir ürün önerileri üretmek için yenilikçi bir yöntem tasarlanmış, uygulanmış ve test edilmiştir. Önerilen tasarım, kullanıcı tarafından yazılan inceleme metinlerinden bazı ifadelerin seçilmesi, bu ifadelerin kullanıcılara tercih, öğelere ise özellikler olarak atanmasından, sonrasında bu atanan ifadelerin yakınlık değerleri kullanılarak öneriler ve açıklamalar üretmeyi amaçlıyor. Bu tezde sunulan tasarımda, ürün önerileri insanlar için anlaşılabilir kelime grupları üzerinden oluşturulduğundan, yapılan önerileri açıklamak için aynı kelime gruplarını kullanmak da mümkün oluyor. Öneri oluşturmak için anahtar kelime çıkarma tekniklerini ve kelime vektörleştiricilerini kullanan çok fazla çalışma yoktu, bu nedenle bu tezde bu teknikleri kullanan bir algoritma ile çalışma yapılmak istendi. Bu tezde üzerinde çalışılan algoritmayı değerlendirmek için, sayısal sonuçların hesaplanmasının yanı sıra 15 kişi ile kullanıcı çalışması yapıldı. Bu deneyler, önerilen tasarım ile üretilen tavsiyelerin %55 kadarının insanlar tarafından beğenildiğini ve önerilen öğeler için yapılan açıklamaların %58 kadarının anlamlı bulunduğunu göstermiştir.

Anahtar Kelimeler: açıklanabilir ürün önerileri, açıklanabilir makine öğrenimi, konu çıkarsama, KeyBERT, Yake

To all contributed to me to be who I am

ACKNOWLEDGMENTS

I have been lucky to have Assoc. Prof. Dr. Hande Alemdar as my supervisor, who has resolutely supported all of my work at every stage. She was, in my opinion, the best possible advisor to work with. It was a pleasure for me to have the chance to work with her.

During the design stage of this thesis, I discussed many aspects of my research and received feedback from many of my friends, especially Alara Karaman, Barış Özkuşlar, and Levent Doğan. I truly feel grateful to have such great friends, and I'm grateful for the time they put into me and the development of this thesis.

This work could not have been possible without a caring and supporting family. It is a chance to have a family that has always supported all my decisions and needs. I thank Ülkü Tunç, Şenol Tunç, Hasan and Hüseyin Tunç, Serpil Oppermann and Basri and Yüksel Tunç and my family-like friend Alara Karaman for their care and love. I wish Yüksel Tunç could have seen the completion of this work, and I am sure that she would be proud, as we are proud to have had her.

I would like to additionally thank my aunt Prof. Dr. Serpil Oppermann, who is an English Literature Professor, and my jury members Assoc. Prof. Dr. Sinan Kalkan, and Assoc. Prof. Dr. Tansel Dökeroglu for providing feedback on this work.

I have asked many friends to attend to the user study that is conducted as a part of the evaluation of this thesis. Their help was invaluable in evaluating the qualities of the recommendations and explanations, which could not have been done without the ones spending their time on this thesis. I will not list their names, but I thank them for their time, help, and input on this work.

I also feel very proud and entitled to be a part of the department of the Computer Engineering of Middle East Technical University. It is not that uncommon to have such successful and professional staff in a department, but I am sure that it is very uncommon for these people to be that friendly and helpful to everybody.

My master's degree and thesis thesis are supported by TÜBİTAK MSc Scholarship 2210/A. I thank TÜBİTAK for supporting my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
MF	Matrix Factorization
CF	Collaborative Filtering
TF-IDF	Term Frequency-Inverse Document Frequency
LDA	Latent Dirichlet allocation
w2v	Word2vec
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The trust built in the recommendations mainly depends on how well the user agrees with the recommended items deserve to be recommended to them. That is, the quality of the recommendations is dependent on how well users can understand the given decision to generate a specific recommendation. In this thesis, we experiment with ways to generate recommendations to the users and, in addition to the recommendations themselves, we try to create a model that serves the reasons for its decisions to the user by using some explanation words. This will hopefully contribute to the user's understanding of why the recommendations are made, which will increase the trust on the recommender. [1] [2]

The primary motivation for this work is to extend and contribute to the literature by studying a novel method for generating explainable recommendations using keyword extraction and word matching techniques. More specifically, the objective is to experiment with a recommender that uses text mining to gather information about the users and the items and generate recommendations solely based on these features.
The proposed recommendation pipeline is designed to fetch the "keywords," which will be the n-grams that can represent the users' review texts best and combine them to create profiles for both users and items. These profiles consisting set of n-grams are then used for calculating similarities between users and items. The causes of these similarities will also be served to the users as the explanations of why the specific item is recommended since the the n-gram phrases that form the user and item pro-

files are meaningful and understandable for humans.

To set an acceptable boundary for this thesis, no additional information for generating recommendations has been used, including the collaborative filtering algorithms. This also allows evaluating the technique adequately since performance of it will be solely based on the user review texts and user ratings.

We expect our recommendation pipeline to perform worse compared to the state-of-the-art recommendation models since we will be using reduced information while generating the recommendations. The loss of information compared to many state-of-the-art models is due to the fact that the information we use is in a form that is understandable by humans, contrary to mostly used latent factor representations. The advantage of using such a representation will be the ability to serve the reasoning behind why specific recommendations are offered to the users.

1.2 Contributions and Novelties

During this study, following contributions are proposed to the literature:

- At the time of the writing, there is no example of using keyword extraction algorithms that we proposed to use for the generation of explainable recommendations even though some similar efforts are shared. These include using attention based models combined with sentiment analysis to similarly capture what users liked or disliked about an item, or some Latent Dirichlet allocation [3] based enhancements over classical recommenders to have some topics to explain the recommendations.
- It is not common to use the exact same information for the generation of the recommendation and the explanations in the literature. Instead, the information used for explaining the recommendations are either a part of the recommender, or used to explain the already generated recommendations. In our recommen-

dation pipeline, we utilize the exact same information for generating recommendations and explanations.

- The algorithm we employed for filtering the generated keywords for the user and item features is also novel, at least in the domain of the recommendations. We simply compare the commonities of the words in our corpus of review text with the word commonities in the English language, measured in a domain agnostic way. We then filter the words that are meaningful in our domain by checking if they are not that common in English.

- We use word vectorizers to match user and item feature words in this thesis and these strategies are not widely used in the domain of recommendations. Still, we can find some related works such as [4] and [5].

1.3 The Outline of the Thesis

In chapter 2, a literature review about generating explainable recommendations utilizing the user written review texts is presented. It is selected to follow a categorization based on the underlying method for generating recommendations. All of the studies selected for the literature review are also able to serve the explanations for their recommendations.

Following the literature review, the methodology of the proposed algorithm for generating explainable recommendations using keyword extraction on item reviews is shared in detail in chapter 3. This chapter contains all parts of the pipeline supported with some pseudo-codes, figures, and discussions around the decisions made.

In the chapter 4, alternative approaches and their effects on the pipeline steps have been shared. The reasoning between the selection of the possible approaches has been discussed and selected in light of the shared numerical studies. In this section, the setup and results of the conducted user studies are also shared.

Considering all the work and the results gathered in chapter 4, discussion around the validity of the proposed algorithm, its performance, and the takeaways have been revealed in the chapter 5. In addition to these final notes, some ideas for future improvements on the success of the pipeline is also shared.

CHAPTER 2

LITERATURE REVIEW

Above all categorization on how the recommender model is built, there are two ways to generate explanations for the recommender systems: model-intrinsic and post-hoc (also called model agnostic) strategies.
In the model-intrinsic explanations, the explanation is partially or solely the answer of why the specific recommendation is made since the recommender is directly based on the information shared by the explanation. In the latter case with the model agnostic strategies, explanations are generated after the recommendation is made [6].

In this section, selected works from the literature that leverage user written review texts for generation of the explanations of the recommended items are presented. We group the literature by the family of the algorithm that the recommender and the explanations is based on.

2.1 Latent factors based explainable recommendation models

Models that are based on predicting latent factors for generating recommendations is the most popular family of algorithms because of their success on achieving high accuracies. The challenge of creating explainable recommendations using these family of algorithms is that they are working on latent features, meaning that the information they use to recommend items does not have any meaning that we can directly infer and thus serve to the users.

To overcome this difficulty, some researchers such as Zhang et al. [7] introduced modifications that is also similar to what we propose; adding features that can be used in explanation of the item to the recommendation features. In their work, these features are extracted from free-format review texts via grammatical and morphological analysis tools and tagged as positive or negative with sentiment analysis algorithms. These features then converted into latent factors and used in the recommender. The similar idea is further improved by a pipeline that is able frequently update the "aspects" that have an influence on the user's opinions on the items, as the authors of similar works Zhang et al. [8] and Bauman et al. [9] stated.

There are some studies about detecting attention over the review text and generate features that matters for the user. In the work of Chen et al. [10], these features are also given to a latent factor based model, while in the proposal paper of Donkers et al. [11], this idea is presented as a way to generate explanations to the users after generating the recommendations.

In the articles above, the explanations were semi model-intrinsic, meaning that the recommender models were fed with a subset of model features to enable the explanations, but still there were other input features to the models that were not explained with the recommended item.

2.2 Deep Learning based explainable recommendation models

In the scope of this thesis, no deep learning based solution is proposed but still, there are some attention based approaches on deep learning based recommenders that are somewhat related with the proposed strategy of keyword extraction.
In the work of Seo et al. [12] and Wu et al. [13], they used an attention based approach to gather the information and parts of the review texts for the items which are the most influential for the ratings given to the item. Having the latent representations of these for both items and users, they match the items with the users to generate recommendations. To serve the explanations for their recommendations, they highlight

the output of the attention model on the review texts given for the item.

2.3 Topic modeling based explainable recommendation models

In the work of Wu et al. [14], similar to what Zhang et al. [7], Zhang et al. [8] and Bauman et al. [9] did, they tried to capture the user's opinions about some aspects on the domain. After having the aspects of the items and the opinions of the users about these aspects, they used collaborative filtering to generate recommendations for the users and using the item properties as some form of the explanation about the recommended item.

In another work, done on the paper of McAuley et al. [15], their Matrix Factorization [16] based model is supported with generated topics using LDA [3]. LDA is a technique that generates and assigns some topics to the given document set. In the paper, authors promptly state that using the assigned topics of the review texts during the recommendation generation also improves the performance of the recommender especially for the users with few reviews. They claim that the output of the LDA's generated topics are clean and easily interpretable so the recommendations can also be explained with these topics if the topics are named manually beforehand.

2.4 Graph based explainable recommendation models

The problem that recommendation algorithms are trying to solve are widely thought as graphs to have a better understanding over the modeling and for the ability to access huge literature beyond the graph related problems [17].
In this work, even though we do not employ any graph based approach, some related methodologies using similar techniques that are used in our proposed recommendation pipeline are found.
One of such works are presented by He et al. [18]. In their work, in addition to modelling the recommendation problem as a knowledge graph, they also proposed using

extracted aspects to be able to generate explanations and better recommendations using collaborative filtering based methods. They extract the aspects for the users and items from a sentiment based approach, and mention that they have also applied filtering to the aspects that they gathered from the aspect extractor by the TF-IDF values for the aspects. These aspects are then integrated to the knowledge graph and used to serve explanations while generating recommendations.

In the work presented by Suzuki et al. [19], they similarly integrated item aspects extracted from the review texts of the users to their knowledge graph for the ability of explaining the generated recommendations. In their work, they define a predefined set of categories and employ an algorithm called LARA [20] to fetch the opinionated review texts about these aspects.

2.5 Model Agnostic models to explain recommendations

One exemplary work for generating post-hoc explanations, after recommendations are made, through a MF based algorithm is presented on Abdollahi et al. [21]. In this work, the authors mention a MF based recommender combined with a CF based explanation generator as a way of achieving high accuracies on the recommendations while being able to make some explanations. Different than the others, their explanations are not based on the review text, and only give explanations on how other similar users rated the recommended item. A very similar idea of using CF techniques to explain the recommendations are presented by Heckel et al. [22] for their knowledge-graph based recommender.

Donkers et al. [11] presents a noteworthy idea to generate explanations to the users after generating the recommendations using an attention based text analysis model. In their paper, researchers mention that a separate explanation model can be placed to work with a recommender to explain the recommendations.

CHAPTER 3

GENERATING EXPLAINABLE RECOMMENDATIONS USING KEYWORD EXTRACTION FROM REVIEW TEXTS

In this thesis, we have designed and experimented with a recommender that can generate model-intrinsic explanations using keyword extraction and word vectorization.

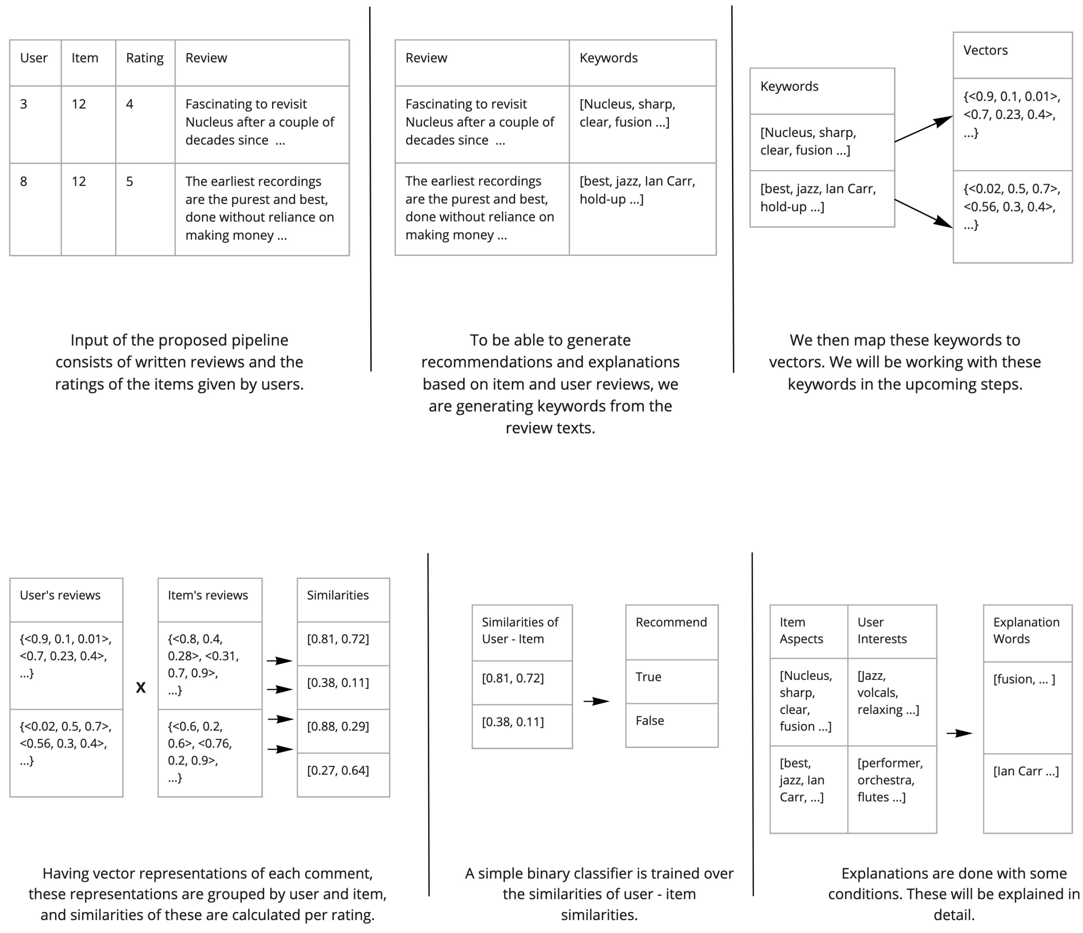
For generating the recommendations, our proposed algorithm solely depends on the features that are in the form of n-grams. Since these features are already in a form that is understandable by humans, they can also be served with recommended items as their explanations. As a direct consequence, the explanations generated for a recommended item directly reflects why that item is recommended to the user.

In the proposed algorithm for generating explainable recommendations using keyword extraction from review texts, features of the recommender are generated using well known keyword extraction algorithms. Then, using word vectorizers, the strengths of relations between users and items can be calculated, and the recommendations are made based on the strengths of these matches.
In a way, our method can be viewed as a content-based recommendation algorithm, but unlike many content-based recommendation algorithms,

- the content that the recommendations are generated based on is also mined from the review texts in an unsupervised manner.

- we calculate and use a similarity distance between users and items, instead of expecting an exact content match.

3.1 An overview of the proposed algorithm



User	Item	Rating	Review
3	12	4	Fascinating to revisit Nucleus after a couple of decades since ...
8	12	5	The earliest recordings are the purest and best, done without reliance on making money ...

Input of the proposed pipeline consists of written reviews and the ratings of the items given by users.

Review	Keywords
Fascinating to revisit Nucleus after a couple of decades since ...	[Nucleus, sharp, clear, fusion ...]
The earliest recordings are the purest and best, done without reliance on making money ...	[best, jazz, Ian Carr, hold-up ...]

To be able to generate recommendations and explanations based on item and user reviews, we are generating keywords from the review texts.

Keywords
[Nucleus, sharp, clear, fusion ...]
[best, jazz, Ian Carr, hold-up ...]

Vectors
{<0.9, 0.1, 0.01>, <0.7, 0.23, 0.4>, ...}
{<0.02, 0.5, 0.7>, <0.56, 0.3, 0.4>, ...}

We then map these keywords to vectors. We will be working with these keywords in the upcoming steps.

User's reviews
{<0.9, 0.1, 0.01>, <0.7, 0.23, 0.4>, ...}
{<0.02, 0.5, 0.7>, <0.56, 0.3, 0.4>, ...}

X

Item's reviews
{<0.8, 0.4, 0.28>, <0.31, 0.7, 0.9>, ...}
{<0.6, 0.2, 0.6>, <0.76, 0.2, 0.9>, ...}

Similarities
[0.81, 0.72]
[0.38, 0.11]
[0.88, 0.29]
[0.27, 0.64]

Having vector representations of each comment, these representations are grouped by user and item, and similarities of these are calculated per rating.

Similarities of User - Item
[0.81, 0.72]
[0.38, 0.11]

Recommend
True
False

A simple binary classifier is trained over the similarities of user - item similarities.

Item Aspects	User Interests
[Nucleus, sharp, clear, fusion ...]	[Jazz, volcals, relaxing ...]
[best, jazz, Ian Carr, ...]	[performer, orchestra, flutes ...]

Explanation Words
[fusion, ...]
[Ian Carr ...]

Explanations are done with some conditions. These will be explained in detail.

Figure 3.1: Proposed recommendation pipeline's outline is illustrated. In each step, the description of the step is given with the state of the flowing data.

3.2 Formal definition of the proposed algorithm

The input of the recommendation pipeline is a vector of $(U_i, I_j, r_{ij}, R_{ij})$ where

- U_i is the distinctive user id
- I_i is the distinctive item id
- r_{ij} is the rating user U_i gave to item I_j on a scale of $[1, 5]$

- R_{ij} is the free format text, English review that user U_i has written to the item I_j

There are two outputs of the recommendation pipeline, with more focus being put on the latter one in the scope of this thesis:

- Predicting if a given U_i liked an item I_j based on the information r_{ik}'s and R_{ik}'s given by the user U_i to any item I_k (other than I_j), and R_{mj}'s written to I_j by any user U_m (other than U_i).
- If the recommender decides to recommend item I_j to user U_i, explaining the recommendation using R_{ix}'s of user U_i and R_{yj}'s of item I_i.

Please note that the steps for generating these two different bits of information are the same until the very end of the proposed pipeline.

In this section, the proposed pipeline will be discussed with the best options. Detailed reports on the hyper-parameter optimizations and algorithms selections can be found under chapter 4.

3.3 Keyword extraction from the user reviews

As the first step of the pipeline, the review texts, R_{ij}'s, are converted to summarizing phrases by keyword extraction algorithms. At the end of this step, we will have a mapping from each review text R_{ij} to its generated keywords, K_{ij}. Extracted keywords from the free format English texts will be stored as n-grams.
The generated set of K's are then affiliated with the items as the items' aspects and affiliated with the users along with the rating given together with the review text as the users' interests. By affiliating users with the extracted keywords together with their rating for the review, we aim to be able to distinguish between the aspects users would like or not.

In this thesis, we have experimented with three different methods for extracting summarized information from free format English texts: KeyBERT [23], Yake [24] [25]

[26] and TF-IDF. These algorithms are fed with the texts of item reviews given by the users and generate keywords of these texts.

3.3.1 KeyBERT

KeyBERT is a model built on top of Google's well-known BERT model to generate text summarization. It uses a small and pretrained version of the BERT encoder. Internally, KeyBERT tries to find the n-grams that can be used to summarize the given full text by creating a vector encoding of each n-gram in the text via BERT. It assumes that the document can be represented as an average of these vectors. This average is then compared with the individual BERT encodings of each of the n-grams of the text. The most similar ones are expected to be the summary of the given text. [27]

3.3.2 Yake!

Yake is a model that relies on the statistical features extracted from texts. It is significantly faster compared to KeyBERT and serves better or equal extracted keywords and overall performance according to the experiments we have conducted.
Yake is adopting a more classical approach for generating the keywords of the sentences by employing a rating that is based on the structure of the sentences, term frequencies, and co-occurrences.[24]

3.3.3 TF-IDF

TF-IDF is a classical algorithm for extracting keywords from a given corpus. TF-IDF is used as a baseline algorithm, and it is observed that using KeyBERT or Yake! outperforms the TF-IDF on keyword extraction.

In the Table 3.1, the outputs of KeyBERT and Yake, when given the abstract of this thesis, is shared for demonstration purposes

At the end of the stage of keyword extraction, we have the extracted phrases from

Table 3.1: Key phrases extracted from the abstract of this thesis using KeyBERT and Yake with n-gram lengths limited to 1 and 2

Keyword Extractor	n of n-grams	Extracted Keywords
KeyBERT	1	recommender, recommendations, vectorizers, phrases, explanations, keyword, review, liked
KeyBERT	2	recommender generate, generate recommendations, generating recommendations, explainable recommendations, recommendations using, explanation recommendations, recommender user, recommender performs
Yake	1	recommendations, phrases, reasoning, design, users, user, study, recommender
Yake	2	recommendations, important aspect, marketplace applications, phrases, share reasoning, generating recommendations, explainable recommendations, generate recommendations

each review text in the training dataset.

3.4 Preparation for Similarity Calculation

3.4.1 Rearranging the user interests and item features

As a preparation for the match distance calculation using the extracted keywords, the format of the data is rearranged for users and items. A list of the aspects of each item is stored, while a map is stored for each user. The maps of users store the keywords per the rating of the review that keyword is extracted. This way, we aim to distinguish between the preferable and non-preferable features for the users. In this step, the features of the items with more than 40 elements are trimmed after these features are sorted according to the returned confidence scores from the keyword extraction algorithms. Main purpose of trimming features to have at most 40 elements is to reduce the training and test times of the recommendation pipeline. More discussion on the required time for these operations are shared in the chapter 4.

These rearranged structures will be referred as the *profiles* of the users and items. The profiles will be consisting of the extracted keywords in the forms of n-grams, structured as demonstrated in the algorithm 1.

Algorithm 1 Rearranging the user interests and item features

```
procedure REARRANGE_KEYWORDS($List < U, I, r, List < K >> ds$)
    $user\_interests \leftarrow Map < U, Map < r, List < K >>>$
    $item\_aspects \leftarrow Map < Item, List < K >>$

    for $row \leftarrow ds.rows$ do
        $U_i \leftarrow row.user$
        $I_i \leftarrow row.item$
        $r_i \leftarrow row.rating$
        $K_i \leftarrow row.keywords$

        $user\_interests[U_i][r_i].append(K_i)$
        $item\_aspects[I_i].append(K_i)$

        ▷Trim user and item features to have at most 40 items
    end for
    return $user\_interests, item\_features$
end procedure
```

3.4.2 Selection of eligible words for explanation

Another critical step before calculating the match distances between items and users is to eliminate the words from the calculation that carries little to no meaning in our domain. Since the recommender shouldn't decide in accordance to words that carries no information in our domain, we filter these words that are gathered from the keyword extraction algorithms.

We observed that the meaningful words for a domain consist of the words that are *common* on that domain but *not common* in the overall language.
This observation relies on the following sub-parts:

- Very common words hold no information. They are either too general, such as "good" or stop words, such as "to"
- Very uncommon words hold information that we cannot utilize, such as "cactus" (In an unrelated domain such as album recommendations), or some words with typos such as "progvessive"
- Some somewhat-common English words still create some issues since they are not common in our corpus and not meaningful in our domain. It would be absurd to explain the recommendations in the music domain with the words such as "keep" or "water."

These cases are demonstrated in the Figure 3.2.

To decide the thresholds for common and uncommon, two percentiles are tuned manually for both the word frequency distribution on the training set corpus and the English language word frequency database [28].
Manually tuned percentiles are

- 97%+ words on the English language word frequency database is considered as "common."
- 98%+ words on our training user reviews corpus are considered as "common."

Possible future work for improvements on the proposed approach are discussed in the chapter 5

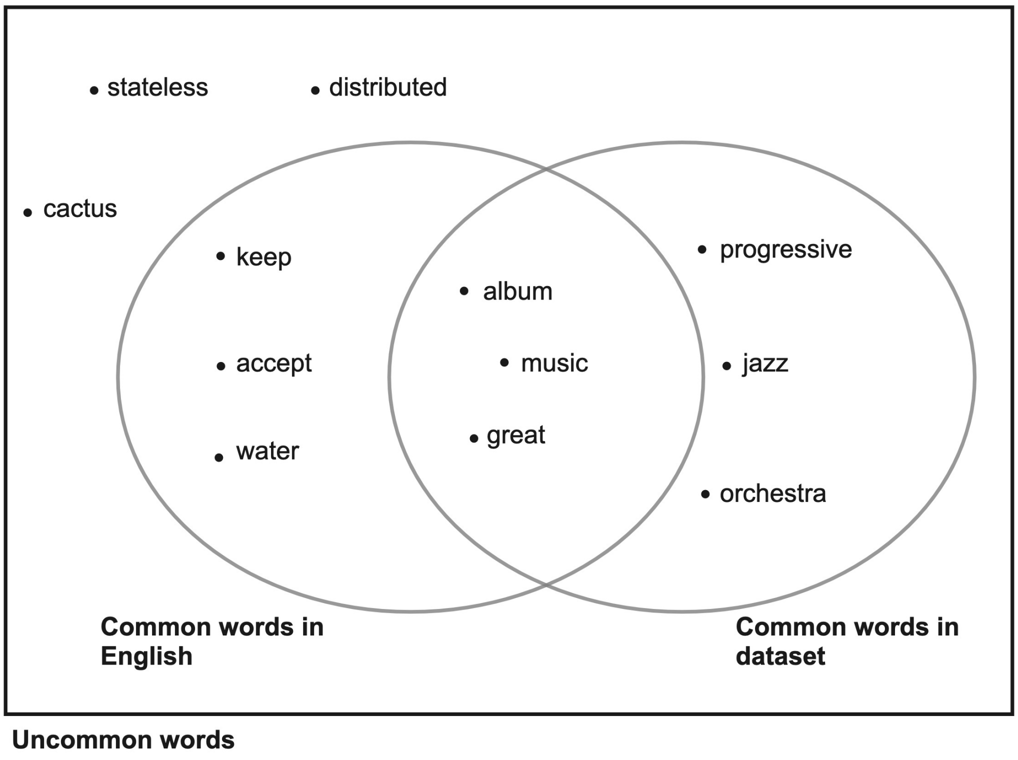


Figure 3.2: An example of Venn diagram demonstrates some sample words by their frequencies in English and our corpus on music album reviews. In this diagram, the area representing the common words in the dataset that is not contained by the common words in English denotes the words that are eligible for using in recommendations and explanations.

3.5 Word Vectorizer

To derive the recommendations and the explanations, the pipeline requires a component to numerically evaluate the strengths of the matches between the users and the items. For this purpose, word vectorization techniques over the generated keywords and cosine similarity over the encodings of the extracted phrases are used in the proposed pipeline.

It is important to note that since the recommendations will be made directly over the keyword similarities between users and items, the explanation step becomes trivial when we score the similarities between item aspects and user interests. While reducing the qualities of the recommendations (since we restrict ourselves to a finite subset of human readable n-grams for each text), we gain the ability to perfectly explain the reasons for the recommendations made by the proposed model.

In the following subsections, alternatives for the word vectorizers that are evaluated during this work are briefly shared.

3.5.1 word2vec

Word2vec [29] is a model that uses neural networks to learn the associative relations between the words in a corpus to map the words to vectors. These vectors, then, can be used to encode the words.
Word2vec uses local statistics and co-occurrences of the words to adjust the encodings of the words [29].

3.5.2 GloVe

GloVe is, similarly, an algorithm to generate vector encodings for words so that the generated encoding vectors will have some linear properties.
Different from the word2vec, GloVe additionally uses global co-occurrence statistics as well as still employing local co-occurrence statistics to generate the representations of the n-grams [30].

In this work, instead of using transfer learning techniques on pre-trained word vectorizer models, another model has been trained on the dataset to be used as a "fallback" model in case the pre-trained model does not include the specific words in its corpus. The pre-trained word vectorizer model is trained over the whole corpus instead of the extracted keywords because of the word vectorizers' context aware nature.
Transfer learning over the pre-trained versions of both word2vec and GloVe is possible, but you would need to have the internal model parameters, and these were not provided with most of the pre-trained models found online. Instead of investing time

on applying transfer learning over the pre-trained models, we have decided to implement a strategy to fall back to the model that is trained from scratch considering the very low rate of fallbacks and the small difference on the accuracies between the model that is trained over our corpus and the pre-trained models.

For calculating the effect of using a fallback model instead of applying transfer learning techniques, the rates of the fallback are calculated. The rate of requiring the fallback model was 0.64% in the dataset that is being used. This is the ratio of a word encountered during the test phase of the algorithm that couldn't be found on the pre-trained models but found in the model I trained over the training set. Also, the percentage of the cases in which the words on the test set couldn't be found on either the pre-trained model or our fallback model was 0.64%.
In the case when there is a representation of a word on the model I trained while there is none in the pre-trained model, the loss in the accuracy is equal to the difference between the accuracy of my model and the pre-trained model. The effect of this difference is negligible especially considering the very small percentage of the case occurrence and the slight accuracy difference between the accuracy of the model I trained and the pre-trained model.

For the selection of the word vectorizer algorithm, the effect of the selection to the precision score of the recommender is calculated during the experiments. In these experiments, it is observed that there is no significant difference between using Glove or Word2Vec. In the presented state of this work, GloVe vectors are used as the word vectorizer as an arbitrary choice.

It is worth noting that not using transfer learning techniques on the pre-trained word vectorizers also hurt the performance of these models in a general sense since the dataset we use has different dynamics of the words then the datasets that these models are trained on. This discrepancy can be solved either with using a context aware word vectorizer such as BERT [31], or fine-tuning the pre-trained models we use over our dataset. These improvements are left as a future work and discussed in chapter 5.

3.6 Match Distance Calculation

Algorithm 2 Match Distance Calculation - Mean of All

```
1: procedure MATCH DISTANCE($U_i, I_j$)
2:     $X \leftarrow []$
3:     $positive\_interests \leftarrow []$        ▷ Positive means keywords of $R$ with $r >= 4$
4:     $negative\_interests \leftarrow []$
5:     $item\_aspects \leftarrow []$
6:
7:     $temp\_distances \leftarrow []$
8:     for $positive\_interest \leftarrow positive\_interests$ do
9:         for $item\_aspect \leftarrow item\_aspects$ do
10:            $pair\_dist \leftarrow word2vec.dist(positive\_interest, item\_aspect)$
11:            $temp\_distances.append(pair\_dist)$
12:        end for
13:    end for
14:    $X[0] \leftarrow mean(temp\_distances)$
15:
16:    $temp\_distances \leftarrow []$
17:    for $negative\_interest \leftarrow negative\_interests$ do
18:        for $item\_aspect \leftarrow item\_aspects$ do
19:            $pair\_dist \leftarrow word2vec.dist(negative\_interest, item\_aspect)$
20:            $temp\_distances.append(pair\_dist)$
21:        end for
22:    end for
23:    $X[1] \leftarrow mean(temp\_distances)$
24:
25:    return $X$
26: end procedure
```

Match distance calculation can be seen as the core of the proposed pipeline. In a sense, all of the steps before the distance calculation has been done as a preparation for having the ability to numerically evaluate the strength of a match between given (U_i, I_j) pair.
As an output, the match distance calculation procedure produces a vector of two numbers for the relatedness of the item's aspects to the positive and negative interests of the user.
The returned vectors will form the features of the ML algorithm and will be used to predict the user's opinion on the given item.

Algorithm 2, named *Mean of All*, introduces the best performing distance function according to the numerical comparisons made in the chapter 4. The proposed algorithm compares each of the user's interests with each of the item's features via the cosine similarity function provided by the word vectorizers. Recall that the profiles of users and items are consist of n-grams generated through keyword extraction algorithms.

As drafted in the algorithm 2, taking the mean distances of all the preferences of the user with all aspects of item provided the best results with a high significance compared to the other algorithms we test. Experiments with the algorithms considering a subset of these matches shown significant performance losses on precision of the predictions of ML model.
More details on the selection of the distance function are shared in section 4.

3.7 Predicting user's opinion on items based on the interest similarity vector

To map the similarity vectors to the recommendation predictions, we train a simple ML model to act as a recommender.
The recommender is an ML model that needs to map two numeric values in the range $[0, 1]$ to $\{0, 1\}$. Outputs of the model is used as a signal for the pipeline to decide if the item should be recommended or not, so the output 1 means that the item should be recommended to the user.

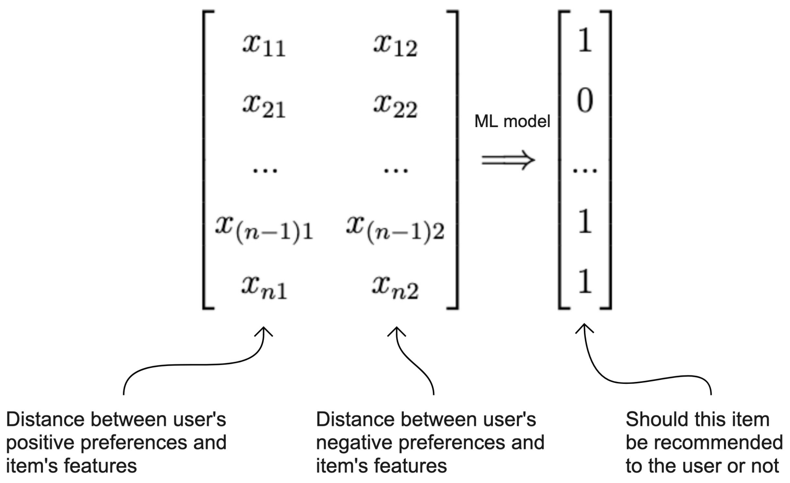

ML model
Distance between user's positive preferences and item's features
Distance between user's negative preferences and item's features
Should this item be recommended to the user or not

Figure 3.3: Mapping that the recommender should learn from. A good distance function is expected to generate distances that can be linearly relatable with the labels of the recommender.

As it is demonstrated in Figure 3.3, there is a linear relationship between the outputs of the distance function and the probability of a user liking an item. In the recommender, in addition to the match distances of the user's positive preferences with the items' aspects, we also use the match distances of the user's negative preferences with the item's aspects. So the model is expected to learn the relation between the features and the labels where the features are linearly correlated with the labels.

The best performing model for the described task was a Support Vector Machine optimized for high precision on the positive label for the "recommend" signal. The requirement for high precision on the positive values comes from the nature of generating recommendations. There is high toleration for predicting false negatives when recommending items but much less tolerance for predicting false positives. Aiming for high precision reduce the recall of the model and cause recommender to recommend less items as a side effect. This is not an important concern when generating recommendations.
Optimization of the prediction thresholds for achieving high precision on the positive label is shared in detail in the chapter 4.

3.8 Generation of Explanations

For the generation of explanations, a slightly modified version of the match distance calculation algorithm is used, which outputs similar user interests with corresponding item aspects instead of returning the mean similarities between user's positive and negative preferences with the item's features.

Algorithm 3 Generation of Explanations

```
1: procedure GENERATE EXPLANATIONS FOR A MATCH($U_i, I_j, k$)
2:     $dists \leftarrow []$
3:     for $interest, rating \leftarrow U_i.interests$ do
4:         if $rating \geq 4$ then
5:             for $aspect \leftarrow I_j.aspects$ if $eligible\_world(aspect)$ do
6:                 $dists.append(\ tuple(cos(interest, aspect), interest, aspect)\ )$
7:             end for
8:         end if
9:     end for
10:
11:     $explanations \leftarrow []$
12:     $dists.sort()$
13:     while $len(explanations) < k$ and $len(dists) > 0$ do
14:         $relation \leftarrow dists.pop()$
15:         if $relation.aspect$ not in $explanations$ then
16:             $explanations.append(\ relation.aspect\ )$
17:         end if
18:     end while
19:     return $explanations$
20: end procedure
```

For demonstrating the strategy we employ for generating explanations, algorithm 3 is shared. In this proposed algorithm, each of the distances between the user's interests and item's features are calculated and saved for all pairs in a list together with user's interest and item's feature. After this list is built, algorithm selects the smallest k

distinct elements, and these are served to the user with the recommended item.
In the presented state of the work, k on the algorithm 3 is set to four.

3.9 Discussion on proposed pipeline

3.9.1 An example recommendation and explanation

In Figure 3.4, user has written reviews to Albums 1 and 2. The keyword extractor has selected a set of 1-grams from these reviews, which includes the words *relaxing* and *interesting*. Due to the similarity between the full set of n-grams that are extracted from the user's reviews and Album 3's extracted n-grams, using the *Mean of All* distance calculation strategy, the recommender decides to recommend Album 3 to the Test User.
After a recommend signal is generated via the recommender explained in the section 3.7, an explanation for this recommendation is generated using the most similar words of the user's set of n-grams and the item's set of n-grams.
The most similar words found in item's n-grams are *relaxing* and *enjoyable*. These words are used to explain why the item is recommended to the user.

3.9.2 Discussion on the performance of the proposed solution for generating explanations

The proposed solution has some strong characteristics as well as some weak ones.

The generation of the explanations is solely based on unsupervised machine learning methods. This fact brings great flexibility to mine a set of labels for the items and users without any effort in a domain agnostic way. These sets of labels generated from the reviews of users will be much more diverse and rich in comparison to methodological classifications done by humans.

This fact will help serving unique and authentic explanations for the recommended items from a wide spectrum of phrases, but also result in using unrelated or absurd words or phrases to serve explanations.

The success of the explanation words in the proposed pipeline heavily depends on the quality of the written reviews by the community in the platform we train our pipeline, and heavily relies on the success of the keyword extraction algorithms. During the self-evaluation of the thesis, it is observed that many instances of false positives on the recommendations and unsuccessful explanations are due to these two reasons.

Also, the unsupervised fashion of generating the labels for items and users ends up with many labels that are generic for explaining items. In the above example, the second explanation word *enjoyable* is an example for this case. Even a filtering mechanism described in the section 3.4.2 is implemented and dramatically overcomes this issue, there are still many such words that are eligible to get used in explanations. As it is also observed in the user study in section 4.3.3, even though in average the explanations are perceived as accurate and successful, a significant percentage of the served explanations are found irrelevant to the recommended item. This shortcoming is also discussed in the conclusion section 5

One other trade-off point was around the selection of n of n-grams of the keyword extraction process. Using 1-grams, namely single words, for the text summarization and profile creation for users and items causes the explanations to stay on the safer side. All words have some meaning if they are used separately from the context. Still, they can hold much less meaning compared to using several words together to explain a recommendation. However, it is observed that when the n of n-grams is increased to 2 or 3, there is an increased chance of serving non-sense phrases as the explanations of the items. In summary, it is observed that with an increase of n of n-grams, the quality of the explanations of the best cases increases, but the worst cases decreases.

In the example Figure 3.5 from one of the user study participants' recommendations, the trade-off discussed above is observed. In this example case, one of the positive interests of the user is selected as *monotonous* and the recommendation of the album is explained to the user with the phrases *Relaxing background* and *Tedious household*. In the first phrase, we can observe that the *Relaxing background* serves more meaning and a more precise explanation to the recommendation compared to its 1-gram conjugate *Relaxing*; while the *Tedious household* is not an appropriate explanation for explaining an album, even the recommender selected that phrase to be close to the word *monotonous*.

In the same scenario presented in Figure 3.5, another critical shortcoming of the proposed algorithm in this thesis can be observed: The word *monotonous* is selected by the keyword extraction algorithm as a representative word for the user who wrote it, but the user used that word as ".. *not repetitive and monotonous*". Since this word is used on a highly rated review text, the extracted word *monotonous* is perceived as a positive interest of the user while, in reality, it is a feature that the user does not like. There may be some possible solutions for this issue that we discuss in the section 5.

In the chapter 4, results for comparing the recommenders that are serving 1-gram explanations as well as serving explanations formed as 2-grams is presented and compared. It is seen that a recommender using 1-gram explanations is liked more, but the difference was very low and found to be statistically insignificant.

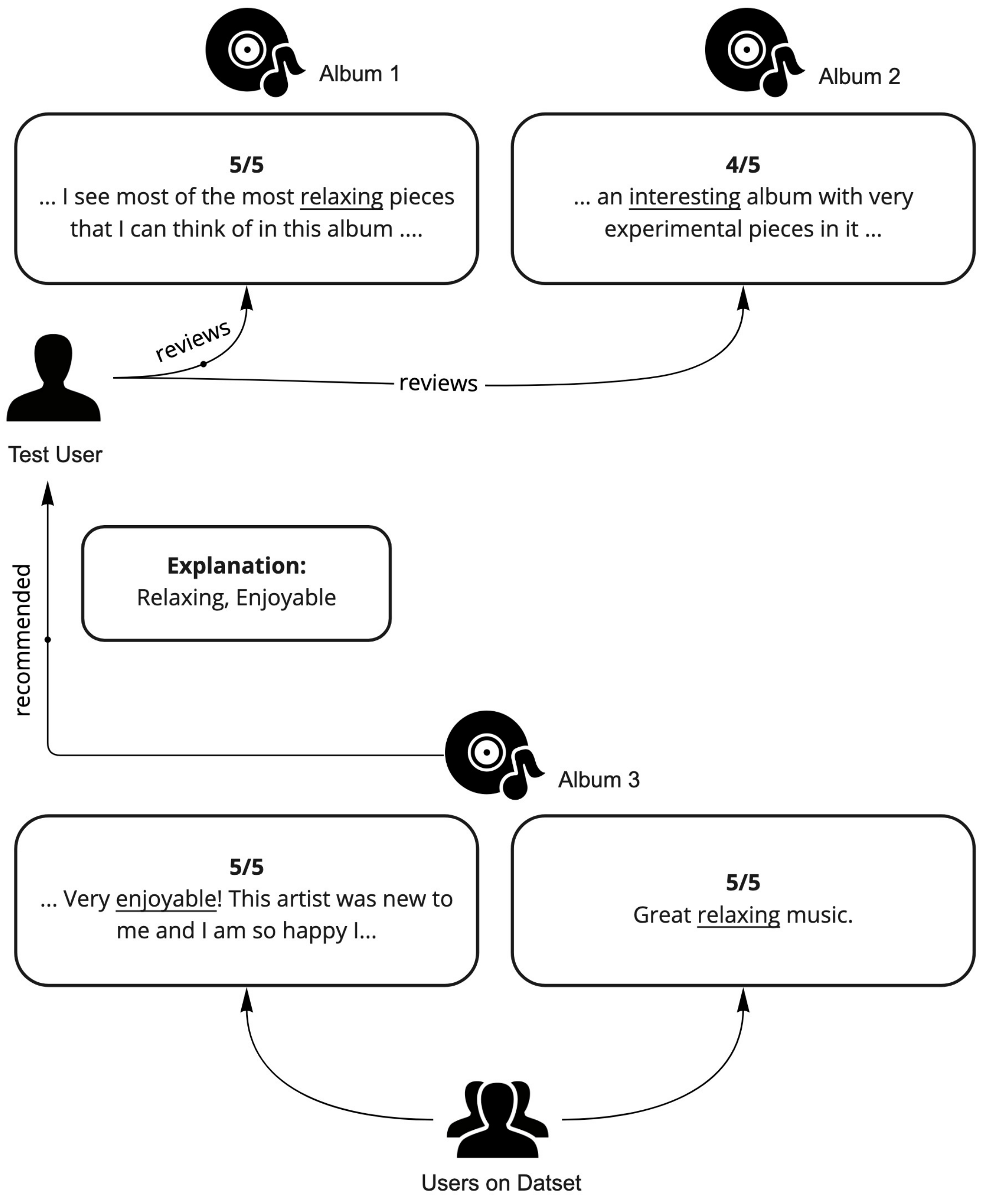

Album 1
Album 2
5/5
... I see most of the most relaxing pieces that I can think of in this album ....
4/5
... an interesting album with very experimental pieces in it ...
reviews
reviews
Test User
recommended
Explanation:
Relaxing, Enjoyable
Album 3
5/5
... Very enjoyable! This artist was new to me and I am so happy I...
5/5
Great relaxing music.
Users on Datset

Figure 3.4: An example scenario, taken from the user study, for a recommendation to a user on our case study group with a recommender based on extracted 1-grams. User is getting a recommendation based on the extracted keywords "relaxing" and "interesting" in the shared case. The Album 3 is recommended due to its profile that includes the keywords "relaxing" and "enjoyable".

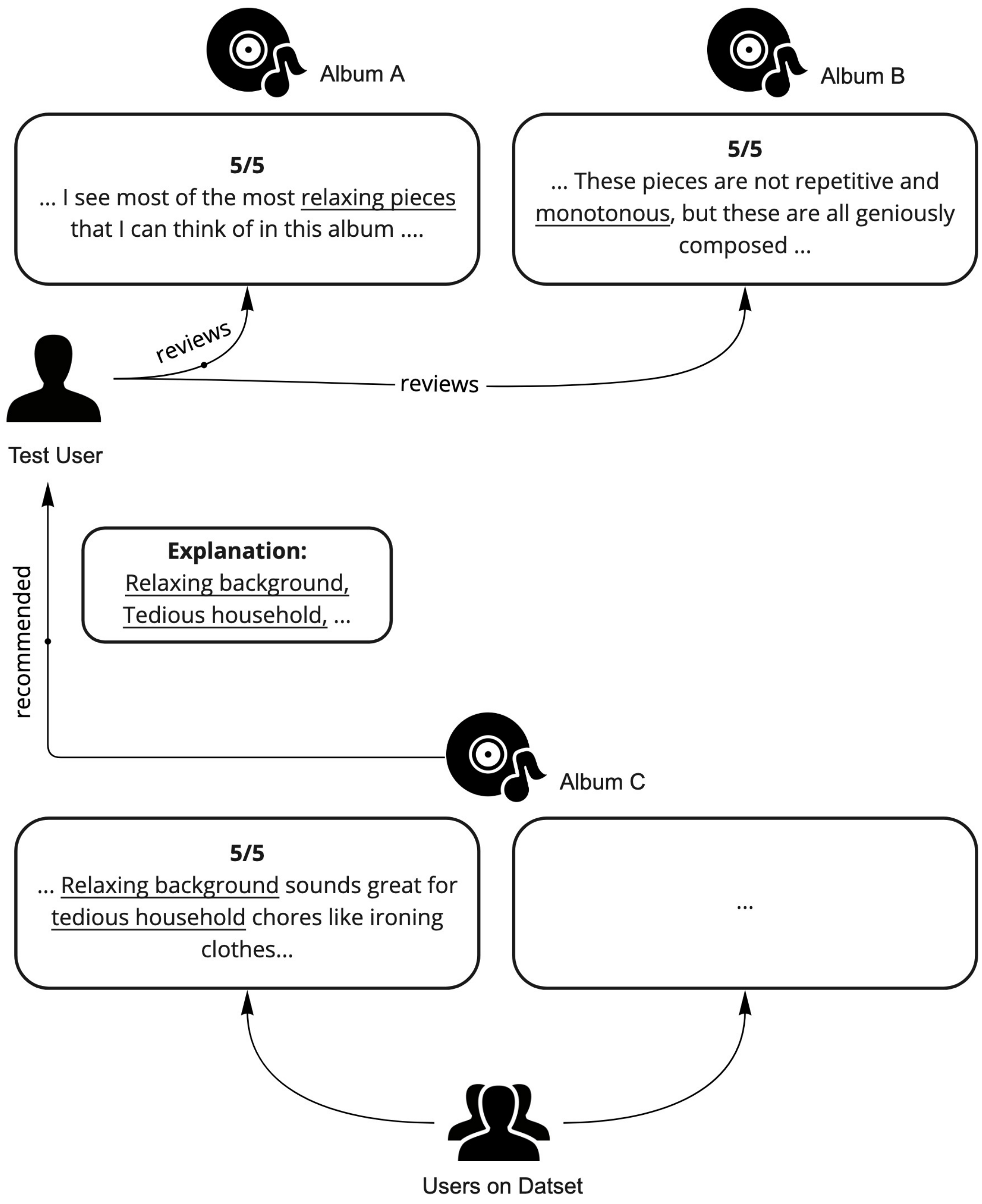

Album A
5/5
... I see most of the most relaxing pieces that I can think of in this album ....
Album B
5/5
... These pieces are not repetitive and monotonous, but these are all geniously composed ...
reviews
reviews
Test User
recommended
Explanation:
Relaxing background,
Tedious household, ...
Album C
5/5
... Relaxing background sounds great for tedious household chores like ironing clothes...
...
Users on Datset

Figure 3.5: Another example scenario, taken from the user study, for a recommendation to a user on our case study group generated with a recommender based on extracted 2-grams. User is getting a recommendation based on the extracted keywords "relaxing pieces" and "monotonous" in the shared case. The Album C is recommended due to its profile that includes the keywords "relaxing background" and "tedious household".

CHAPTER 4

EXPERIMENTAL EVALUATION

The presented work has two outputs that can be evaluated; the quality of the recommendations and the quality of the explanations. These two outputs are generated by the same pipeline; the words that are used in the explanation given an item and a user are the words that the item has as *aspects* and also similar to the user's *interests*. These same set of n-grams are also used for generating match scores and recommendations.
While selecting between alternative approaches for the parts of the proposed pipeline, we have tried to pick the best options to improve the recommendation precision. Since the rating prediction and the explanation generation use the exact same pipeline, optimizing the rating predictor also improves the quality of the explanations.

In this chapter, two kinds of experiments that are carried out in the scope of this thesis are presented; numerical optimizations and a user study.
For optimizing the recommendation precision and selecting most of the components of the proposed pipeline, numerical evaluation and optimizations are made. These experiments are done on the dataset with a validation test split.
After having some optimized alternative recommendation pipelines with a very similar performance on the test set performances, a user study is also conducted with 15 people to evaluate the quality of the recommendations and the explanations.

In the following sections, these experiments are explained in more detail.

4.1 Dataset

This thesis's evaluations have been made with the Amazon Reviews Dataset for the category "CDs and Vinyl" [32]. This dataset is selected over many other alternatives since it has enough data points and allows conducting a user study, and since it is much easier to rate music album recommendations and explanations compared to restaurants, books, movies, etc.

CDs and Vinyl category of the Amazon Reviews Dataset consist of $1,443,475$ reviews of $112,391$ users to $73,713$ items. About 86% of the reviews in the dataset are positive (4+ stars).

Since the evaluation of the models on unbalanced datasets are harder, all of the graphs and values on the test and validation sets are presented for two different sampling strategies:

- **Balanced sampling**: 500 rows from each of the ratings has been randomly selected in this sampling strategy. The final subset of data will consist of $2,500$ data points that equally include reviews from all ratings.
- **Random sampling**: In this sampling strategy, $15,000$ rows from the original dataset have been selected randomly.

4.2 Hyper-parameter optimization and numerical evaluations

In this section, the emphasis will be on improving the quality of the recommendations.
In the proposed pipeline, the quality of the recommendations are dominated with the following parameters, so the experiments will done over their selections:

- **Keyword extraction algorithm:** We have experimented with Yake and KeyBERT. Both of these algorithms support generating n-grams while summarizing given texts. In this thesis, experiments with the following combinations are presented:
    - KeyBERT with 1-grams
    - KeyBERT with 1 or 2-grams
    - Yake with 1-grams
    - Yake with 1 or 2-grams
- **Distance function to calculate the match strengths between users and items:** Experiments with many alternative distance functions are completed, but formally the following three are presented:
    - Mean of All: returns the mean of distances between all interests and all features of an item. Positive and negative interests are treated differently and returned separately.
    - Mean of Half: returns the mean of the half of the smallest distances between all interests of a user and all features of an item. Positive and negative interests are treated differently and returned separately.
    - Three smallest: returns the mean of the three smallest distances between all interests and all features of an item. Positive and negative interests are treated differently and returned separately.

  We provide the pseudo-codes of these algorithms in Appendix A
- **Machine Learning algorithm that is used to predict if a user would like an item:** Experiments with a few algorithms are conducted, but in the corresponding section, only the Support Vector Machine based classifier will be presented. The mapping ML model should need to learn is a linear mapping, and the selection of the ML algorithm was not very effective due to the uncomplicated nature of the mapping. Most of the emphasis was on having high precision on positive predictions because of the nature of our problem. More details will be shared in the following subsection.

4.2.1 Evaluation of distance functions

The distance function that calculates the distance between given user and an item is the core of the pipeline. It plays a role in both recommendation and explanation generation.

Having a step to compare the distance functions without needing to compare ML models improves the required time for conducting experiments and eliminates potential adverse effects from the ML model's performance. Due to this preliminary evaluation step, ML model optimization pipeline can be run only for the best-performing distance function and keyword extraction algorithm pairs.
In this work, experiments with three alternative match distance calculation algorithms are made. Pseudo codes of these algorithms are shared under Appendix A.

To evaluate possible distance functions, Pearson and Spearman correlation coefficients between the output of the distance function and the ratio of the likes to all reviews are calculated. These two coefficients indicate how strong the correlation is between two variables.
Both of the correlation coefficients have the value range of $[-1, 1]$. Negative values indicate negative correlation, and it is generally considered that values between 0.3 to 0.5 indicate a medium-strength correlation while values between 0.5 to 1 indicate a strong correlation between the variables [33].

Our distance function should produce higher values if we know that a user does not like an item and vice versa. To test and compare the performances of the distance functions, we measure the strength of the negative correlation between the change of percentage of the users liked items with the change of distances between users and items.

Table 4.1 and Table 4.2 presents the calculated values of the Pearson Coefficient & Spearman Coefficient values of the experimented distance function - keyword extrac-

tion pairs. In Table 4.1, calculated coefficients are given for a balanced sample taken from the dataset whereas in Table 4.2, same calculations are shared for a random sample taken from the dataset.
Each table presents Pearson & Spearman Coefficients in descending order by the absolute value of the found correlation.

Table 4.1: Pearson and Spearman correlation coefficients for different distance functions and keyword extractors on balanced sample

Distance function	Keyword extractor	Pearson Coefficient	Spearman Coefficient
Mean of All	yake1	-0.7922	-0.7763
	bert1	-0.5213	-0.4307
	bert2	-0.5259	-0.2920
	yake2	-0.3898	-0.2328
Mean of Half	yake1	-0.6220	-0.6540
	bert2	-0.3694	-0.2794
	bert1	-0.3108	0.0790
Three Smallest	yake1	-0.2198	-0.2102
	yake2	0.0398	0.3236

According to the numerical results presented on Table 4.1 and Table 4.2 for the correlations between the rate of positive reviews by the distance function's value, it can be seen that Yake1, combined with the distance function *Mean of All* outperforms all other combinations.
There are also other cases that show weaker correlation but these correlations are not consistent in both balanced and random samples. For example, although Bert with 2-grams performs good on balanced dataset, it fails to produce results on randomly sampled dataset that preserves the required correlation.

From the values on the Table 4.1 and 4.2, following generalizations can be concluded:

Table 4.2: Pearson and Spearman correlation coefficients for different distance functions and keyword extractors on random sample

Distance function	Keyword extractor	Pearson Coefficient	Spearman Coefficient
Mean of All	yake2	-0.4984	-0.6926
	yake1	-0.4720	-0.5869
	bert1	-0.2939	-0.2107
	bert2	-0.0221	-0.1191
Mean of Half	yake1	-0.1871	-0.3531
	bert2	-0.0888	-0.1601
	bert1	-0.0476	-0.0487
Three Smallest	yake1	0.0836	0.1135
	yake2	0.4657	0.3824

- Using 1-grams is better than using 2-grams: With an increase in the length of the phrases we compare, the weights of the closer words will become less. Still, using phrases will allow the pipeline to capture more context with the provided explanation and might positively impact explanation performance.

- KeyBERT performs worse than Yake. Combining the results we took here with the user studies, Yake is slightly better than KeyBERT on both recommendation and explanation quality, especially when n of n-grams is more than 1.

Figure 4.1 shows the significant negative correlation of the Mean of All function using Yake with 1-gram for keyword extraction. In subfigures (a) and (b), the negative correlation is clearly visible inside the area of interest which is the ranges with high values on the corresponding graphs (c) and (d).
Despite the fact that Yake with 1-grams shows significant correlation with the rate of the positive reviews, there are other combinations showing meaningful correlation. In the following experiments, other pipelines that shows good performance in this experiment will be presented together with Yake using 1-grams. Correlation graphs of the all tests presented on the Table 4.1 and Table 4.2 is presented in Appendix B.

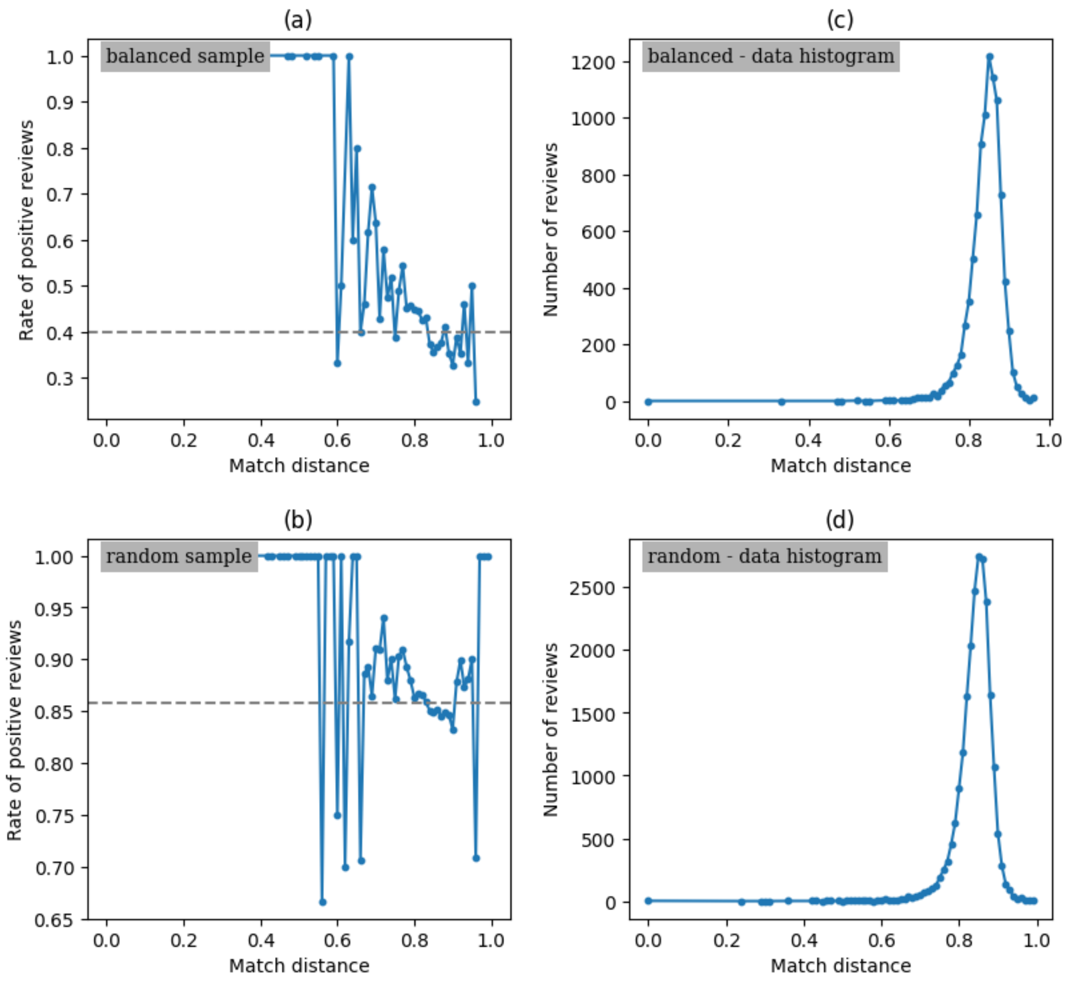


Figure 4.1: The graphs (a) and (b) in the left column visualizes the relation between the percentage of the positive ratings for values distance function can take. In these graphs, intensity of inclination to the right indicates that the distance function performs well. In the right column, number of ratings per the calculated match distances are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

The setup with Yake keyword extractor using 1-grams and distance function of *Mean of All* is the best selection due to the high negative correlation between the two axes.

4.2.2 Optimization of recommender

In the proposed pipeline, recommendations are made with a ML model that is trained over the output of the distance function.

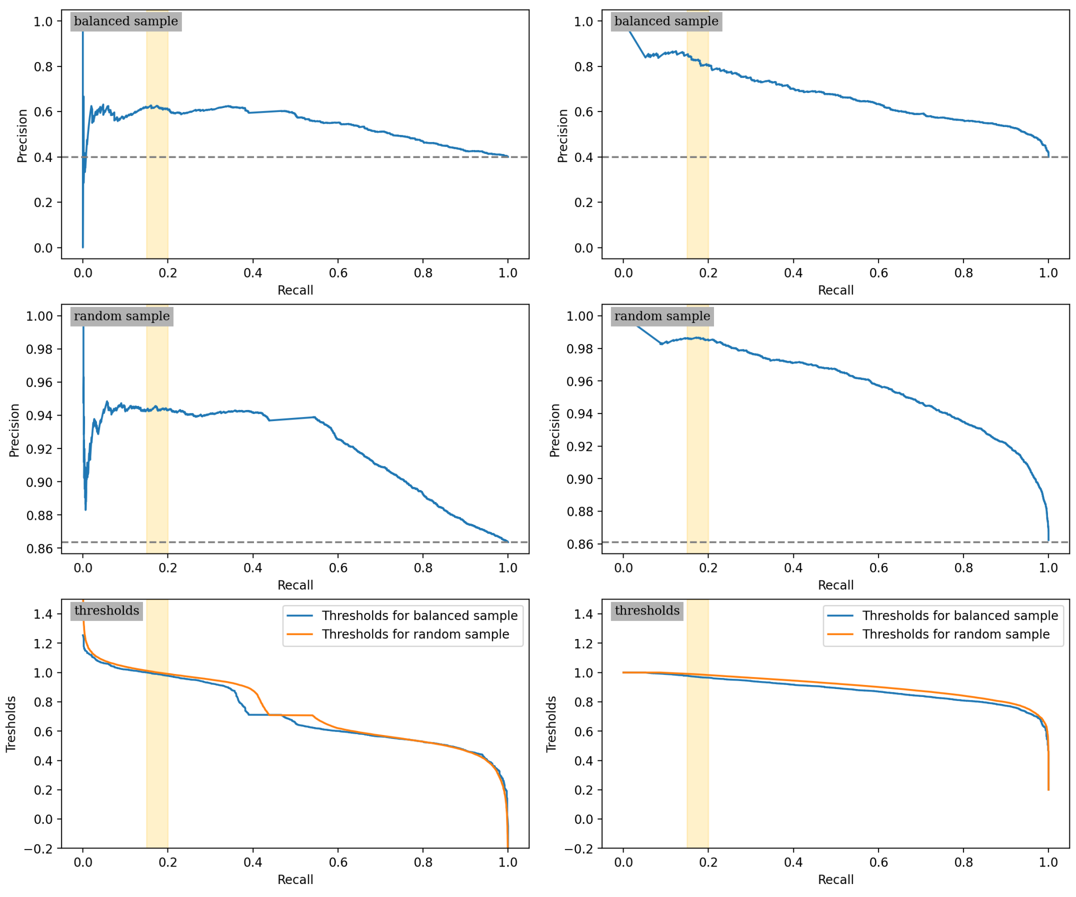
In our application, precision of the true predictions was the most important metric to optimize. This is due to the fact that the generation of recommendations does not require high recall for the true predictions and thus high overall accuracy but requires high true class prediction precision.

To optimize the model with the highest possible precision for the true labels while having a recall value that would not prevent our model to generate recommendations, precision-recall graphs of the recommender for different keyword extraction algorithms are generated. The optimal thresholds for deciding the predicted outputs label are selected from the following graphs generated from the validation set, and the performances of these thresholds in the test sets are presented.

This section presents one of the best performing models in terms of positive class prediction precision. Other experiments and a complete picture for comparisons on the validation and test sets for selecting sensitivity are shared in the Appendix C.

Parallel to what is found in the correlation experiments, the best model to receive high precision on positive predictions is based on Yake with 1-grams, and the distance function to calculate the match distances between users and the items is *Mean of All*. Still, Yake with 2-grams and Bert with 1-grams, when using *Mean of All* strategy performs very close to Yake with 1-grams. Their precision-recall graphs are presented in Appendix C and a further comparison is presented under user study in section 4.3.3.

Figure 4.2 presents the best performing model, which is the pipeline using Yake with 1-grams as keyword extraction algorithm and *Mean of All* as its distance function. In Table 4.3, the parameters used for the recommender is shared in detail.



(a) Our model (b) SVD

Figure 4.2: In figure (a), precision-recall graphs of the recommender developed in the thesis are presented, while in figure (b), precision-recall graphs of the well-established matrix factorization based SVD model are presented [34], [35]. The top graphs are generated on the balanced test set. Following them, the second graphs are generated on a randomly sampled test set, and the last graph shows the required thresholds for the corresponding points on the precision-recall graphs. In the precision-recall graphs, the dashed gray lines represent the dummy predictor's precision-recall values that always predict 1. In the graphs, yellow bars represent the range where the true class prediction thresholds are selected. Position of the yellow bar is selected from the validation split, such that the precision is around its maximum value for a precision that is greater than 0. This will result in the model being tuned to have high precision values for positive predictions.

Table 4.3: Hyper-parameter selections for the best recommendation pipeline using Yake with 1-grams

Keyword Extractor	
Algorithm	Yake
Max number of features to extract	8
n of n-grams	1
SVM based Classifier	
Confidence score for true class	0.99
Common word filter	
Lower % on training set	98%
Upper % on English word frequencies	97%
Glove	
Size of Latent Vectors	200
Window	5
Epochs on training data	100

Table 4.4 and Figure 4.3 summarizes the performance of the tuned recommender. The presented evaluations are taken after the true class prediction threshold is tuned.

As it can be seen from Figure 4.2, trained ML model based on the extracted keywords from user review texts is performing worse than a well-known matrix factorization based model while performing better than dummy models on true class prediction precision.
We expected this result since the methodology we use drastically reduces the amount of information on the training set by extracting only a small number of words from the review texts and making predictions based on these incomplete features. Still, seeing that the best performing model performs significantly better than a dummy predictor indicates that the model learns and generalizes well on the training set.

Table 4.4: Prediction performance of the best recommender on test sets

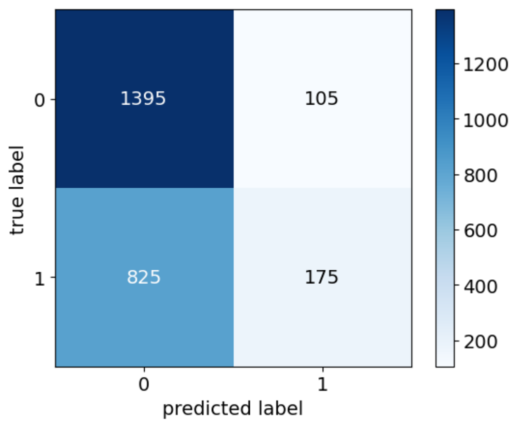
(a) Test set with balanced sampling

	precision	**recall**	**support**
False	0.63	0.93	1500
True	0.62	0.17	1000
Accuracy		0.63	2500

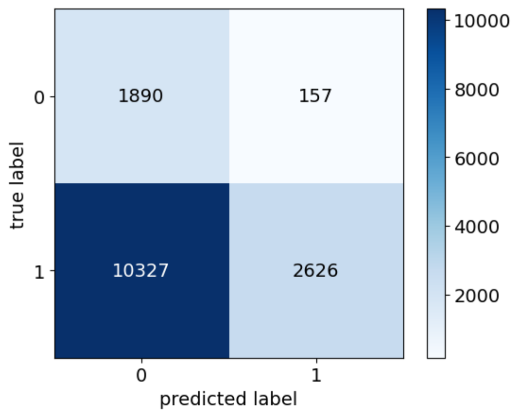
(b) Test set with random sampling

	precision	**recall**	**support**
False	0.15	0.92	2047
True	0.94	0.20	12953
Accuracy		0.30	15000

Training and generation of predictions for the test set is done on a 2019 Macbook Pro with 2.6 GHz CPU on a single core. The training process requires around 20 GBs of Memory when using the Amazon Cds and Vinyl review dataset consisting of 1,443,475 rows. [32] The whole training and predictions on the test set took 2.5 hours on a single CPU core, excluding keyword extraction from review texts.



(a) balanced sample



(b) random sample

Figure 4.3: Confusion matrices of the best recommender using Yake with 1-grams on test sets with two sampling strategies.

4.3 User Study

The work presented in this thesis aims to serve personalized explanations together with the recommendations it generates. To evaluate the success of the explanations and compare the success of recommendations to what we numerically calculated, we have conducted a user study with 15 people.

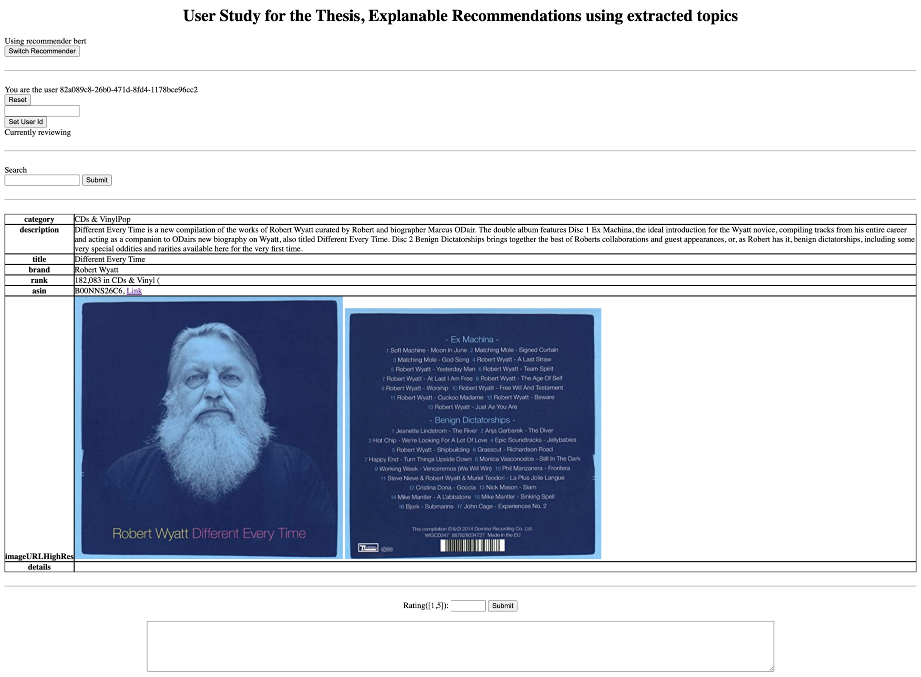
The main reason to conduct a user study is to evaluate the success of the explanations, which cannot be done numerically. In the study, candidates were asked to write reviews to some items of their choices and then provided with recommendations and explanations and were asked to rate the recommended items and the served explanations.

In this study, three of the selected recommenders by their numerical evaluation are used to serve recommendations to the users. This way, we both had a chance to verify the numerical results we found for the precisions of the models on the quality of the recommendations and had a chance to measure and compare the explanations they serve.

4.3.1 Setup

A web interface with three types of pages has been implemented to have a real-life like scenario. These pages are:

1. **Review page** to provide a way to participants to write reviews about the items
2. **Search page** to provide a way to participants to search items in the dataset
3. **Feedback collection page** to collect participants' opinions about the recommendations and explanations.



User Study for the Thesis, Explanable Recommendations using extracted topics

Using recommender bert

Switch Recommender

You are the user 82a089c8-26b0-471d-8fd4-1178bce96cc2

Reset

Set User Id

Currently reviewing

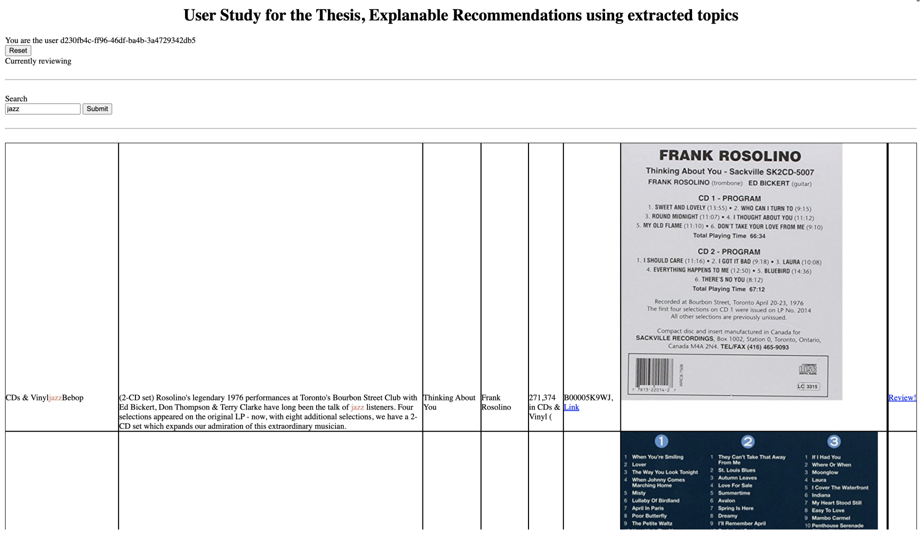
Search

Submit

category	CDs & VinylPop
description	Different Every Time is a new compilation of the works of Robert Wyatt curated by Robert and biographer Marcus ODair. The double album features Disc 1 Ex Machina, the ideal introduction for the Wyatt novice, compiling tracks from his entire career and acting as a companion to ODairs new biography on Wyatt, also titled Different Every Time. Disc 2 Benign Dictatorships brings together the best of Roberts collaborations and guest appearances, or, as Robert has it, benign dictatorships, including some very special oddities and rarities available here for the very first time.
title	Different Every Time
brand	Robert Wyatt
rank	182,083 in CDs & Vinyl (
asin	B00NNS26C6, Link
imageURLHighRes	
details	

Rating([1,5]): Submit

Figure 4.4: A sample review page that allows users to write a textual review and rate the item with a rating in the range $[1, 5]$. When the user clicks the submit button after the review is completed, the review is saved for generating the recommendations.



User Study for the Thesis, Explanable Recommendations using extracted topics

You are the user d230fb4c-ff96-46df-ba4b-3a4729342db5

Reset

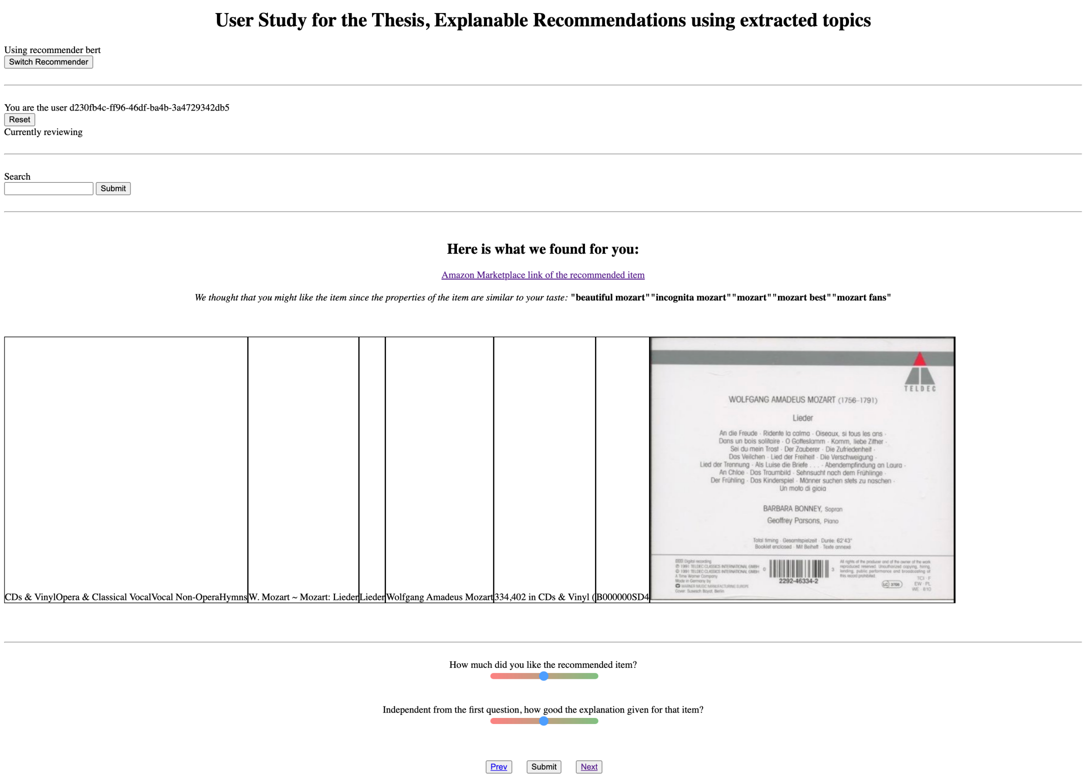
Currently reviewing

Search

jazz Submit

CDs & VinyljazzBebop	(2-CD set) Rosolino's legendary 1976 performances at Toronto's Bourbon Street Club with Ed Bickert, Don Thompson & Terry Clarke have long been the talk of jazz listeners. Four selections appeared on the original LP - now, with eight additional selections, we have a 2-CD set which expands our admiration of this extraordinary musician.	Thinking About You	Frank Rosolino	271,374 in CDs & Vinyl (	B00005K9WJ, Link		Review!

Figure 4.5: A sample search page that allows users to search for an item by a key phrase. The given phrase is searched among the dataset, and the matches are returned in an arbitrary order. Users can view the returned items in a listed fashion and pick any items to review. The provided review button navigates to the review page of that item.



User Study for the Thesis, Explanable Recommendations using extracted topics

Using recommender bert

You are the user d230fb4c-ff96-46df-ba4b-3a4729342db5
Currently reviewing

Search

Here is what we found for you:

Amazon Marketplace link of the recommended item

We thought that you might like the item since the properties of the item are similar to your taste: **"beautiful mozart""incognita mozart""mozart""mozart best""mozart fans"**

CDs & VinylOpera & Classical VocalVocal Non-OperaHymns	W. Mozart ~ Mozart: Lieder	Lieder	Wolfgang Amadeus Mozart	334,402 in CDs & Vinyl (	B000000SD4	

How much did you like the recommended item?

Independent from the first question, how good the explanation given for that item?

Figure 4.6: A sample rating page that allows users to rate their recommendations and the explanations provided for these recommendations. On that page, users adjust the sliders to indicate their opinions about the two questions on the item. When the user is ready with their opinion, they can click the submit button to submit the rating they gave. These ratings are saved with the recommender's information that generated the recommendation and the user's id.

4.3.2 Methodology

4.3.2.1 Design of the user study session

For the user study, attendants are asked to write reviews to 5 to 7 music albums consisting of 2-5 sentences on why or what they like or not like about these albums to imitate a real world scenario. Reviews do not need to be written for items in the dataset that is used since the thesis pipeline only depends to the numerical ratings and the review texts about the review. No other item metadata is used.

To formally ask for help, the following text is shared with the possible attendants:

> For evaluating my thesis, "Explainable Recommendations using extracted topics and word matching," through conducting a user study, I am kindly asking you to write review texts for 5 to 7 music albums. The albums themselves do not need to be in the training set. Also, you don't also need to write extensive reviews. Generally, 2-5 sentences on why you did or did not like the album are sufficient. You should rate the album you reviewed from 1 to 5. (It is better if you mainly review the albums you would give higher scores.)
>
> After you have written the reviews, I'll need to create a user for you and generate the recommendations for that user from the reviews you give. You can share your reviews with me via mert.tunc@metu.edu.tr in any format that I can understand. (e.g, Album names as titles, reviews as paragraphs under these)
>
> At that point, I'll ask you to schedule a time with me to conduct the study via Zoom. The thesis program will run on my end and I'll screen share so that you can see and rate the music albums recommended to you.
>
> The rating of the recommendations and their explanations is expected to take 20-40 mins.

If the attendants are sending the reviews through e-mail, a user is prepared with the shared reviews and made the thesis pipeline generate recommendations for them. Otherwise, if the reviews are written through the web UI, attendants are provided with some recommendations on the fly, during the study.

Candidate's opinions on each recommended item with the following two questions are asked for collecting data about the perceived performance of the recommender and the provided:

- How much did you like the recommended item?
- Independent from the first question, how good is the explanation given for that item?

4.3.2.2 Design of the experiment

The user study's main aim is to test if any of the best performing three recommenders are performing better than the others. The emphasis is on evaluating the explanations since the recommendation precision can be testable numerically, and the test results are shared at the beginning of this chapter.

Each of the three recommenders that we compare generates at most eight items. Each of the items has one to eight explanation words. No more than one item can be recommended with the same explanation words to ensure users can rate different explanation words.
In all three alternative pipelines, the match distance calculation function *Mean of All* is used since it was the best performing algorithm. The keyword extractor algorithm Yake is used with 1-gram and 2-grams generation configurations. In contrast, the KeyBERT is used only with 1-grams due to KeyBERT's lower performance on the correlation and precision performances on the numerical evaluation when used with 2-grams.

As the evaluation metric of the conducted experiment, Friedman test [36] [37] is used to determine if any of the recommenders perform better than others. Friedman test allows checking if any of the entities in question is consistently rated higher compared to the others in the evaluation group. In our test configuration, three alternative pipelines for generating recommendations and explanations were being compared, and the ratings are given to the recommendations and the explanations by 15 people. The configuration and the number of attendants is just enough to use the Friedman test with a high confidence since the Friedman test requires at least 15 participants to provide a precise p-value.
For the Friedman test, the followings are our hypotheses:

- H_0 - Null hypothesis: The mean rating given to each recommender is equal.
- H_{alt} - Alternative hypothesis: At least one of the recommenders has consistently rated lower or higher.

These hypothesis are tested for recommendation ratings and explanation ratings independently.

4.3.3 Results

In the user study, sessions with 15 people were conducted. The attendants are given at most 8 music albums to rate from each recommender totalling at 24, as described in the section 4.3.2. It is important to note that none of the attendants were a native speakers of English Language.

In this section, collected data and the conclusions of the user study is shared.

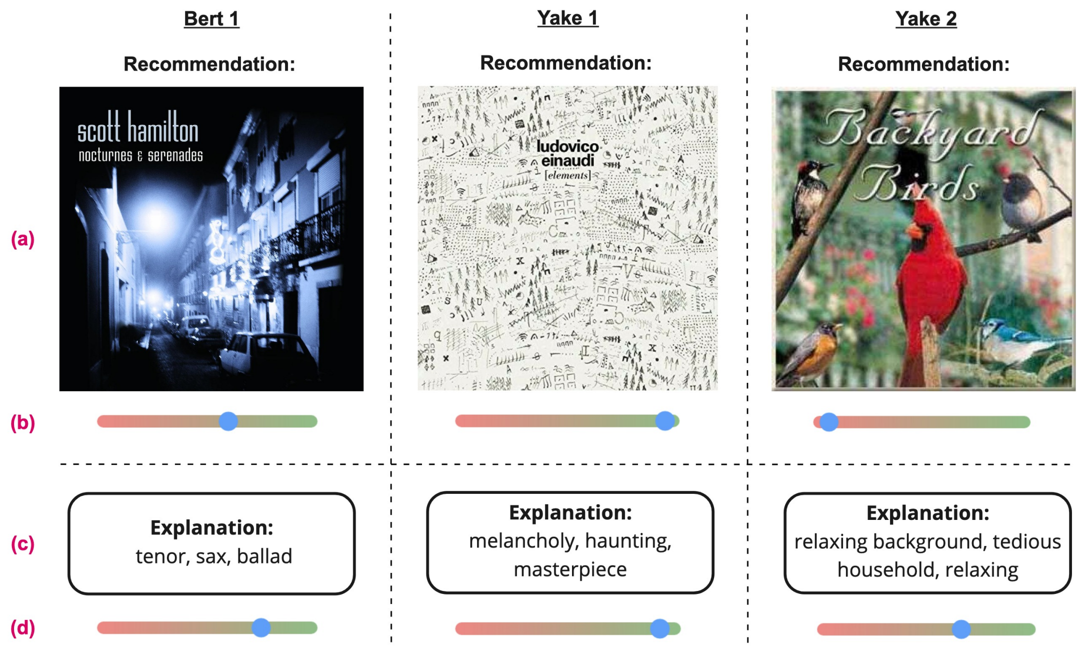
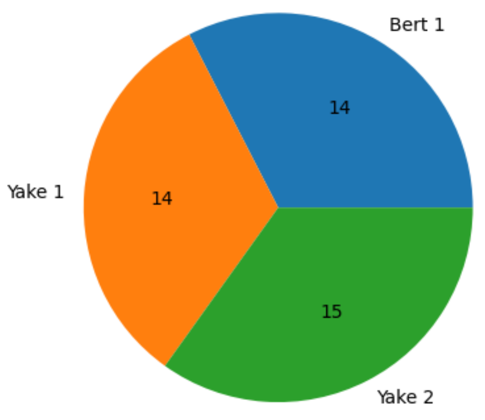
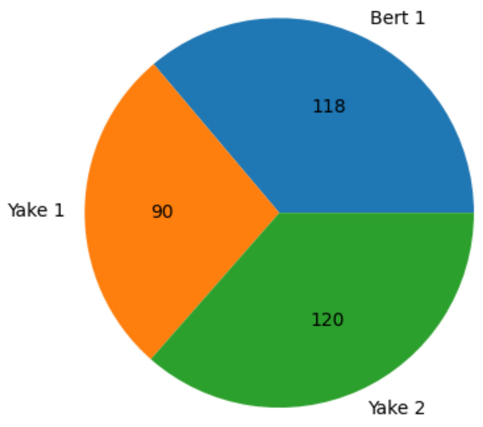


Figure 4.7: Figure presents examples of the given ratings for each of the recommenders by three attendants. In the row (a), album covers of the recommended items are presented. Row (b) contains the snapshots of the ratings given to the items by the attendants. In row (c), the explanations generated for the recommendations are presented, and the snapshots on row (d) presents the perceived rating of these explanations given by the attendants. The interface provided to attendants does not involve numerical scores. Values selected in these sliders are converted to numbers in the scale [0-100] for numerical comparisons and analysis done in this chapter.

Figure 4.7 presents three samples taken from the responses in the user study.



(a)



(b)

Figure 4.8: In subfigure (a), number of participants that is served recommendations from the noted recommenders are shared. In subfigure (b), number of total recommendations generated by the corresponding recommenders are shared. Since the recommenders are tuned to be restrictive to achieve high precision on "recommend" signal, recommenders sometimes generate less then the maximum limit of 8 recommendations. This also leads to inability to serve any recommendations, as it can be seen in subfigure (a).

In Figure 4.9, average of the ratings given for the recommenders are shared. In this bar graph, it can be seen that the performances of the recommenders on two aspects, the quality of the recommendations and the quality of the explanations, are scored very close. Still, configuration with Bert 1 is slightly scored better compared to other two recommenders.

In Figure 4.10, histogram of the rating distributions for the generated recommendations per recommender is shared. Similar to what is observed from the Figure 4.9, the distributions of Bert 1 and Yake 1 are slightly left-skewed while Yake 2 is closer to a symmetric distribution. This means that the probability of a recommendation is liked more for Bert 1 and Yake 1 is slightly higher compared to a recommendation from Yake 2.

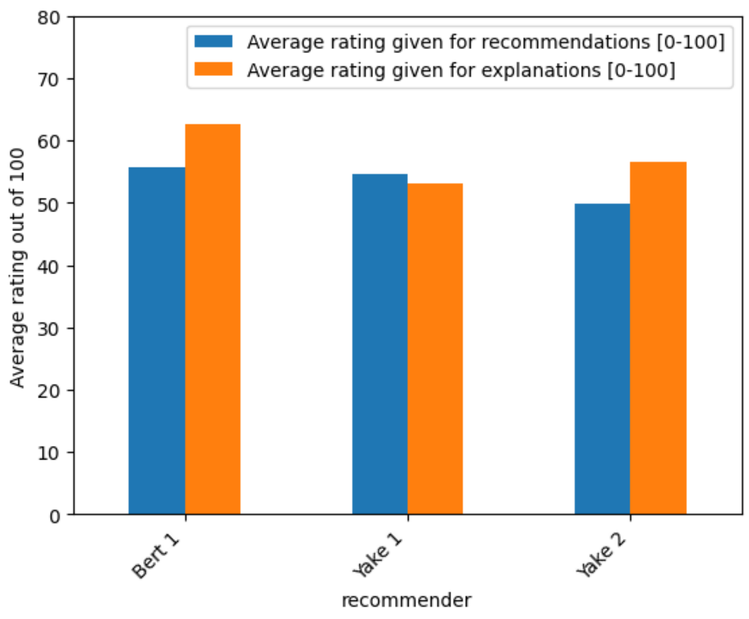
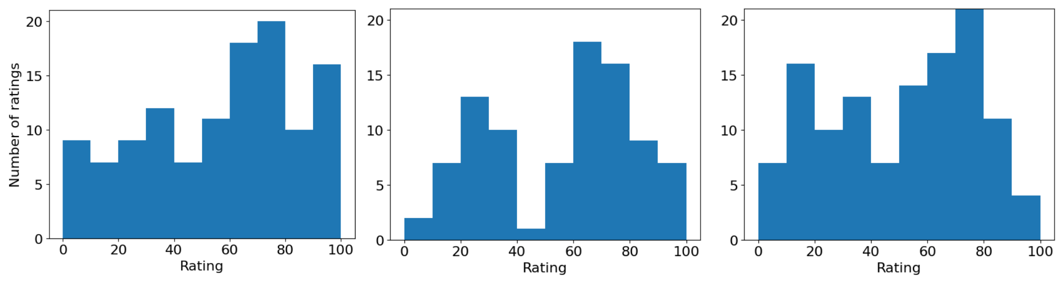


Figure 4.9: Average ratings given to the recommenders for the two questions asked during the user study. The blue bars represents the average ratings given for the quality of the generated recommendations whereas orange bars represents the average ratings given to the qualities of the explanations.

The similarity between the distributions of the recommendation qualities for the three configuration and the average scores given to them are also found consistent with the numerical experimentation shared in the section 4.2.2.

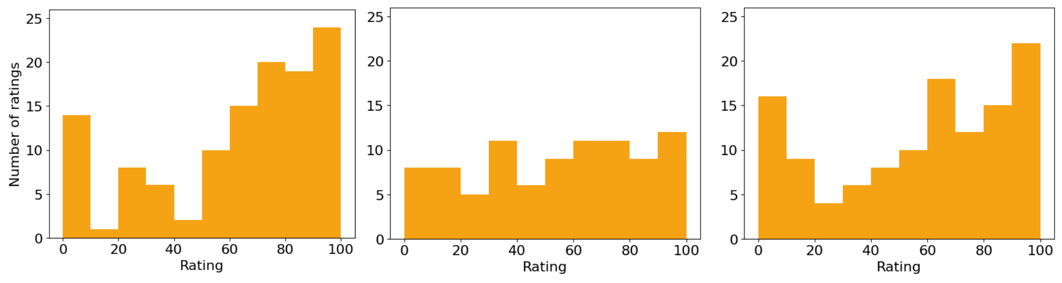
In Figure 4.11, we share histograms for the scores of the qualities of the explanations. Compared to the distributions of the qualities of the recommendations shared in Figure 4.10, Bert 1 on the perceived explanation quality is much better than the Yake 1, and slightly better than Yake 2. While the distribution of Yake 1 is symmetric, distributions of Bert 1 and Yake 2 is left-skewed, meaning that the perceived scores of their explanations were higher on average.

Table 4.5 presents a simplified version of the information on the histograms of the rating distributions given for the recommenders.
In the table, percentiles of the scores given by the participants of the user study are shared for two cases; score ≥ 60 and score ≥ 75. This table can be read as "30% of the attendants scores for Bert 1 on the quality of the recommendations were higher



(a) Bert 1 (b) Yake 1 (c) Yake 2

Figure 4.10: The histograms of the rating distributions for the question *"How much did you like the recommended item?"* for the three recommender.



(a) Bert 1 (b) Yake 1 (c) Yake 2

Figure 4.11: The histograms of the rating distributions for the question *"How good the explanation given for that item?"* for the three recommender.

than 75 out of 100, while 42% of the scores for the Bert 1 on the quality of the explanations are higher than 75 out of 100.". As it is discussed for the Figure 4.10 and 4.11, we observe the same fact that Bert 1 is performing slightly better than the other two recommenders.

Until that point, figures of the distribution graphs, Figure 4.10 and Figure 4.11 suggests that Bert 1 is performing slightly better than other two combinations while the significance is not obvious. To calculate the significance, we use Friedman test as described in detail in section 4.3.2.2.
Table 4.6 shows the calculated p-values from the collected data from the user study using the Friedman test. Since the p-values are higher than 0.05, we can conclude that there is no statistically significant evidence to support reject H_0.

Table 4.5: In these tables, successes of the individual ratings are presented as percentiles. The percentiles are calculated using the thresholds presented in the left-most column. In sub-table (a), percentiles for the quality of the recommendation is presented. In the sub-table (b), perceived quality of explanations are presented.

	(a)		(b)	
	Recommender	**Percentile**	**Recommender**	**Percentile**
Score ≥ 60				
	Bert 1	55%	Bert 1	58%
	Yake 1	54%	Yake 1	42%
	Yake 2	49%	Yake 2	52%
Score ≥ 75				
	Bert 1	30%	Bert 1	42%
	Yake 1	23%	Yake 1	22%
	Yake 2	25%	Yake 2	33%

The calculated p-values concludes to the fact that the observed differences in the performances of the recommenders are not statistically significant even though we observe some differences of the rating distributions on the perceived quality of the explanations in favor of the recommender with Bert 1.

Table 4.6: Calculated p-values from the Friedman tests. Values lower than 0.05 means that the Null Hypothesis can be rejected. In the test we conducted, p-values are significantly greater than 0.05, meaning that we cannot reject the H_0

Test	***p*-value given by Friedman test**
Recommendation ratings	0.247
Explanation ratings	0.420

4.3.4 Discussion of the lack of comparisons on explanation performance

In this thesis, we are not able to present any comparison of the generated explanation performance with the other works.

During our literature review, two ways to evaluate the explanations are found; using numerical metrics and conducting online experiments such as user studies or A/B tests to evaluate the perceived success of the explanations.

There is no widely accepted numerical metric for evaluating recommendation explanations [38]. Our literature review found that commonly used metrics over generated explanations are for evaluating the structural integrity of the generated explanation sentences. BLEU [39], METEOR [40] and ROUGE [41] are such metrics. In addition to these metrics, a metric called model fidelity is used for evaluating the percentage of cases where the recommender can generate explanations. Despite the wide usage of these metrics in the explanation evaluation for the recommendations, these metrics do not directly measure the explanation's correctness or usefulness.

In this thesis, since we do not generate explanations in the form of sentences, it is not possible to report values generated via the first group of evaluation metrics. Also, our proposed solution offers explanations on all recommendations, so the model fidelity on the explanations of this work is 100%, which also shows that the metric is not ideal for evaluating generated recommendation explanations.

The results of the user study conducted in this thesis are also not comparable with the other studies in the literature. The main reason for this is the form difference of the explanations between our proposed recommender produces and the others in the literature.
In the literature, we reach some exemplary work that conduct user study to evaluate the explanations generated in the form of sentences generated from templates using fetched opinions and predefined aspects [42], using predefined topic sentiments [43]

or using collaborative filtering techniques, mostly on knowledge graph relations [44]. In our literature search, we are not able to find any work that conducts a user study for a recommender with keyword based explanations. Also, no study that reports user study results on the dataset we selected is found even if we do not assert that the form of the generated explanations is similar to ours.

Considering the limited number of similar studies that serve keyword based recommendations and the limited number of literature on explainable recommendations that have conducted a user study, our confidence in the fact that there is no comparable work for the user study that is done in this thesis is high. It is also worth noting that, due to the lack of alignment on the evaluation metrics, the style of the user study, and the form of the explanations, the studies in the literature generally do not include comparisons for their user study results and explanation evaluations.

CHAPTER 5

CONCLUSIONS

In this research, a recommender that can serve model-intrinsic explanations that is powered by the extracted keywords from the user written review texts has been studied. The aim of the thesis was to experiment with the novel strategy of using keyword extraction algorithms to extract and summarize user written review texts and use these extracted keywords to generate recommendations and explanations. It was expected to have a recommender that performs worse than the state-of-the-art recommenders but still can serve sensible recommendations while serving explanations with the recommended items.

The proposed method has some core parts that are novel in the domain of generating recommendations. This work contributes to the literature by studying keyword extraction algorithms to generate recommendations. The matching of users and items from their profiles built by the keyword extraction algorithms is made using word vectorizers, and this strategy is also not widely used in this domain.

Reported experiments and evaluation of the proposed recommendation generation pipeline included the performance comparisons of recommenders based on KeyBERT [23] and Yake [24] [25] [26] with the extracted phrase lengths of 1 to 2. In addition to the different approaches to keyword extraction, results and effects of three different match distance calculation algorithms to the recommendation pipeline are shared.
A user study is conducted for the evaluation of the quality of explanations and the comparisons of the explanations between the numerically selected configuration alternatives for the recommendation pipeline. All of the experimentation is done with

the Amazon Review Data [32], using the subset of product group "CDs and Vinyl."

It is observed that the recommender that is proposed and studied in this thesis can significantly outperform a dummy recommender and can serve sensible recommendations as well as serving sensible explanations. It is shown that the configuration based on the KeyBERT extraction algorithm with 1-grams performing slightly better at generating liked explanations and recommendations. Despite the fact that the scores given for the recommendations and the explanations of the recommender utilizing KeyBERT extraction algorithm with 1-grams were slightly better than other tested combinations, it is shown that there is no statistically significant evidence to favor one of these combinations over another.

During the development and evaluation of the proposed pipeline, several points have been discovered to have an opportunity to improve the performance of the recommender and the served explanations further.

The first one of these was the possible improvements to the method we employed for filtering the words that carry no meaning in the domain of the dataset. The current strategy falls short on its task by removing many of the words that the recommender can make use of and use in the explanations as well as failing to filter many words that shouldn't be used by the recommender and the explanations. It would be a good future work to test with a predefined set of words that are allowed to be used by the recommender and the explanations that are tailored for the domain of the recommendations. Also, it would be good to generate a more extensive set of words using the proposed algorithm and filter the words that carry little to no meaning in the recommendation domain.

One of the ways that can significantly boost the performance of the recommender would be integrating many more features to the recommender or integrating the proposed model to a state of the art recommender. The current recommender does not take any data other than the extracted keywords from the reviews and the ratings of the review into account. The performance gain from having more features to generate recommendations, such as item metadata as predefined categories, number of ratings, average ratings, etc, is very likely. Also, some other well-established methods for generating recommendations, such as collaborative filtering based algorithms can be used together with the proposed pipeline to boost the recommendation precision further. It would be interesting to integrate the proposed model into a state-of-the-art recommendation engine as a feature and explanation generator.

Also, in the current proposed and implemented structure, usage of context aware word vectorizers, such as BERT [31] encoder, is not possible. After keyword extraction step to get n-gram summarization of the review text, the fetched keywords are separated from their context. It would be a good addition to the literature to evaluate the pipeline's performance that utilizes the context aware word vectorizers on the match score predictions. The context can be referenced from the extracted keywords and the vector encodings of the extracted keywords can be fetched from the original context they are taken.

Bibliography

[1] Zachary C. Lipton. *The Mythos of Model Interpretability*. 2017. arXiv: 1606.03490 [cs.LG].

[2] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[4] Makbule Gulcin Ozsoy. *From Word Embeddings to Item Recommendation*. 2016. arXiv: 1601.01356 [cs.LG].

[5] Cataldo Musto et al. “Word Embedding Techniques for Content-based Recommender Systems: An Empirical Evaluation”. In: *RecSys Posters*. 2015.

[6] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101. ISSN: 1554-0677. DOI: 10.1561/1500000066. URL: http://dx.doi.org/10.1561/1500000066.

[7] Yongfeng Zhang et al. “Explicit Factor Models for Explainable Recommendation Based on Phrase-Level Sentiment Analysis”. In: *Proceedings of the 37th International ACM SIGIR Conference on Research amp; Development in Information Retrieval*. SIGIR ’14. Gold Coast, Queensland, Australia: Association for Computing Machinery, 2014, pp. 83–92. ISBN: 9781450322577. DOI: 10.1145/2600428.2609579. URL: https://doi.org/10.1145/2600428.2609579.

[8] Yongfeng Zhang et al. “Daily-Aware Personalized Recommendation Based on Feature-Level Time Series Analysis”. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 1373–1383. ISBN: 9781450334693. DOI: 10.1145/2736277.2741087. URL: https://doi.org/10.1145/2736277.2741087.

[9] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. “Aspect Based Recommendations: Recommending Items with the Most Valuable Aspects Based on User Reviews”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’17. Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 717–725. ISBN: 9781450348874. DOI: 10.1145/3097983.3098170. URL: https://doi.org/10.1145/3097983.3098170.

[10] Jingwu Chen et al. “Attention-Driven Factor Model for Explainable Personalized Recommendation”. In: *The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval*. SIGIR ’18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 909–912. ISBN: 9781450356572. DOI: 10.1145/3209978.3210083. URL: https://doi.org/10.1145/3209978.3210083.

[11] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. *Explaining Recommendations by Means of User Reviews*. Tech. rep. 2018.

[12] Sungyong Seo et al. “Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys ’17. Como, Italy: Association for Computing Machinery, 2017, pp. 297–305. ISBN: 9781450346528. DOI: 10.1145/3109859.3109890. URL: https://doi.org/10.1145/3109859.3109890.

[13] Libing Wu et al. *A Context-Aware User-Item Representation Learning for Item Recommendation*. 2017. DOI: 10.48550/ARXIV.1712.02342. URL: https://arxiv.org/abs/1712.02342.

[14] Yao Wu and Martin Ester. “FLAME: A Probabilistic Model Combining Aspect Based Opinion Mining and Collaborative Filtering”. In: *Proceedings of*

the Eighth ACM International Conference on Web Search and Data Mining. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 199–208. ISBN: 9781450333177. DOI: 10.1145/2684822.2685291. URL: https://doi.org/10.1145/2684822.2685291.

[15] Julian McAuley and Jure Leskovec. "Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text". In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: Association for Computing Machinery, 2013, pp. 165–172. ISBN: 9781450324090. DOI: 10.1145/2507157.2507163. URL: https://doi.org/10.1145/2507157.2507163.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.

[17] Janneth Chicaiza and Priscila Valdiviezo-Diaz. "A Comprehensive Survey of Knowledge Graph-Based Recommender Systems: Technologies, Development, and Contributions". In: *Information* 12.6 (2021). ISSN: 2078-2489. DOI: 10.3390/info12060232. URL: https://www.mdpi.com/2078-2489/12/6/232.

[18] Xiangnan He et al. "TriRank: Review-Aware Explainable Recommendation by Modeling Aspects". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: Association for Computing Machinery, 2015, pp. 1661–1670. ISBN: 9781450337946. DOI: 10.1145/2806416.2806504. URL: https://doi.org/10.1145/2806416.2806504.

[19] Takafumi Suzuki, Satoshi Oyama, and Masahito Kurihara. "Explainable Recommendation Using Review Text and a Knowledge Graph". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 4638–4643. DOI: 10.1109/BigData47090.2019.9005590.

[20] Hongning Wang, Yue Lu, and Chengxiang Zhai. "Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach". In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. Washington, DC, USA: Association for Comput-

ing Machinery, 2010, pp. 783–792. ISBN: 9781450300551. DOI: 10.1145/1835804.1835903. URL: https://doi.org/10.1145/1835804.1835903.

[21] Behnoush Abdollahi and Olfa Nasraoui. “Using Explainability for Constrained Matrix Factorization”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys ’17. Como, Italy: Association for Computing Machinery, 2017, pp. 79–83. ISBN: 9781450346528. DOI: 10.1145/3109859.3109913. URL: https://doi.org/10.1145/3109859.3109913.

[22] Reinhard Heckel et al. *Scalable and interpretable product recommendations via overlapping co-clustering*. 2016. DOI: 10.48550/ARXIV.1604.02071. URL: https://arxiv.org/abs/1604.02071.

[23] Maarten Grootendorst. *KeyBERT: Minimal keyword extraction with BERT*. Version v0.5.0. 2020. DOI: 10.5281/zenodo.4461265. URL: https://doi.org/10.5281/zenodo.4461265.

[24] Ricardo Campos et al. “YAKE! Keyword extraction from single documents using multiple local features”. In: *Information Sciences* 509 (2020), pp. 257–289. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.09.013. URL: https://www.sciencedirect.com/science/article/pii/S0020025519308588.

[25] Ricardo Campos et al. “A Text Feature Based Automatic Keyword Extraction Method for Single Documents”. In: *Advances in Information Retrieval*. Ed. by Gabriella Pasi et al. Cham: Springer International Publishing, 2018, pp. 684–691. ISBN: 978-3-319-76941-7.

[26] Ricardo Campos et al. “YAKE! Collection-Independent Automatic Keyword Extractor”. In: *Advances in Information Retrieval*. Ed. by Gabriella Pasi et al. Cham: Springer International Publishing, 2018, pp. 806–810. ISBN: 978-3-319-76941-7.

[27] Maarten Grootendorst. Oct. 2020. URL: https://towardsdatascience.com/keyword-extraction-with-bert-724efca412ea.

[28] Rachael Tatman. *English Word Frequency*. Sept. 2017. URL: https://www.kaggle.com/datasets/rtatman/english-word-frequency.

[29] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[30] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[31] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[32] Jianmo Ni, Jiacheng Li, and Julian McAuley. "Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 188–197. DOI: 10.18653/v1/D19-1018. URL: https://aclanthology.org/D19-1018.

[33] Stephen Kokoska and Daniel Zwillinger. "CRC Standard Probability and Statistics Tables and Formulae, Student Edition". In: 1999.

[34] Nicolas Hug. "Surprise: A Python library for recommender systems". In: *Journal of Open Source Software* 5.52 (2020), p. 2174. DOI: 10.21105/joss.02174. URL: https://doi.org/10.21105/joss.02174.

[35] Simon Funk. *Netflix Update: Try This at Home*. Dec. 2006. URL: https://sifter.org/simon/journal/20061211.html.

[36] Wikipedia. *Friedman test — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Friedman%20test&oldid=1079111940. [Online; accessed 02-July-2022]. 2022.

[37] Milton Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.

1937.10503522. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522.

[38] Bingbing Wen et al. “ExpScore: Learning Metrics for Recommendation Explanation”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 3740–3744. ISBN: 9781450390965. DOI: 10.1145/3485447.3512269. URL: https://doi.org/10.1145/3485447.3512269.

[39] Kishore Papineni et al. “BLEU: A Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL ’02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[40] Satanjeev Banerjee and Alon Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”. In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 65–72. URL: https://aclanthology.org/W05-0909.

[41] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[42] Nan Wang et al. “Explainable Recommendation via Multi-Task Learning in Opinionated Text Data”. In: *The 41st International ACM SIGIR Conference on Research &amp Development in Information Retrieval*. ACM, June 2018. DOI: 10.1145/3209978.3210010. URL: https://doi.org/10.1145%2F3209978.3210010.

[43] Jesse Vig, Shilad Sen, and John Riedl. “Tagsplanations: Explaining Recommendations Using Tags”. In: *Proceedings of the 14th International Conference on Intelligent User Interfaces*. IUI ’09. Sanibel Island, Florida, USA: Associa-

tion for Computing Machinery, 2009, pp. 47–56. ISBN: 9781605581682. DOI: 10.1145/1502650.1502661. URL: https://doi.org/10.1145/1502650.1502661.

[44] Rose Catherine Kanjirathinkal. “Explainable Recommendations”. PhD thesis. Stanford University, 2018.

APPENDIX A

MATCH SCORE FUNCTIONS

In the presented thesis, three scoring functions are mentioned: *Mean of All*, which is already presented in methodology chapter 2, *Mean of Half* and *Three smallest*. In the following section, pseudo codes for *Mean of Half* and *Three smallest* are shared.

Algorithm 4 Match Score Calculation - Mean of Half

```
procedure MATCH SCORE($U_i, I_j$)
    $X \leftarrow []$
    $positive\_interests \leftarrow []$    ▷ Positive means keywords of $Rw$ with $Rt >= 4$
    $negative\_interests \leftarrow []$
    $item\_aspects \leftarrow []$

    $temp\_distances \leftarrow []$
    for $positive\_interest \leftarrow positive\_interests$ do
        for $item\_aspect \leftarrow item\_aspects$ do
            $pair\_dist \leftarrow word2vec.dist(positive\_interest, item\_aspect)$
            $temp\_distances.append(pair\_dist)$
        end for
    end for
    $temp\_distances \leftarrow$ Smallest half of $temp\_distances$
    $X[0] \leftarrow mean(temp\_distances)$

    $temp\_distances \leftarrow []$
    for $negative\_interest \leftarrow negative\_interests$ do
        for $item\_aspect \leftarrow item\_aspects$ do
            $pair\_dist \leftarrow word2vec.dist(negative\_interest, item\_aspect)$
            $temp\_distances.append(pair\_dist)$
        end for
    end for
    $temp\_distances \leftarrow$ Smallest half of $temp\_distances$
    $X[1] \leftarrow mean(temp\_distances)$

    return $X$
end procedure
```

Algorithm 5 Match Score Calculation - Three Smallest

```
procedure MATCH SCORE($U_i, I_j$)
    $X \leftarrow []$
    $positive\_interests \leftarrow []$    ▷ Positive means keywords of $Rw$ with $Rt >= 4$
    $negative\_interests \leftarrow []$
    $item\_aspects \leftarrow []$

    $temp\_distances \leftarrow []$
    for $positive\_interest \leftarrow positive\_interests$ do
        for $item\_aspect \leftarrow item\_aspects$ do
            $pair\_dist \leftarrow word2vec.dist(positive\_interest, item\_aspect)$
            $temp\_distances.append(pair\_dist)$
        end for
    end for
    $temp\_distances \leftarrow$ Smallest 3 value from $temp\_distances$
    $X[0] \leftarrow mean(temp\_distances)$

    $temp\_distances \leftarrow []$
    for $negative\_interest \leftarrow negative\_interests$ do
        for $item\_aspect \leftarrow item\_aspects$ do
            $pair\_dist \leftarrow word2vec.dist(negative\_interest, item\_aspect)$
            $temp\_distances.append(pair\_dist)$
        end for
    end for
    $temp\_distances \leftarrow$ Smallest 3 value from $temp\_distances$
    $X[1] \leftarrow mean(temp\_distances)$

    return $X$
end procedure
```

APPENDIX B

MATCH SCORE FUNCTION CORRELATION GRAPHS

In this section, for most of the keyword extractors using 1 and 2-grams and scoring function implementations, a correlation graph is shared. For the success of the operation, there should be a negative correlation between the match score and the ratio of positive reviews to all reviews
This is best satisfied with Yake1 with a scoring function of *Mean of All*. Correlation graph of this combination is also presented in chapter 4 as Figure 4.1.

Also, numerical values of the correlation coefficients betweem ratio of positive reviews and the output of the scoring function is shared in the table 4.1 and 4.2 presented in chapter 4.

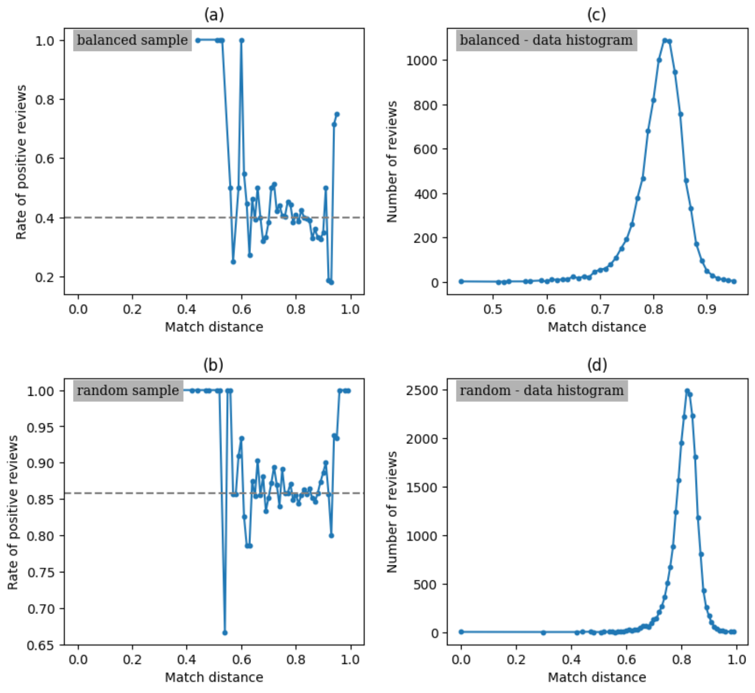


Figure B.1: Scoring function response - Rate of positive reviews to all reviews graph for KeyBERT with 1-grams, using scoring function *Mean of All*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

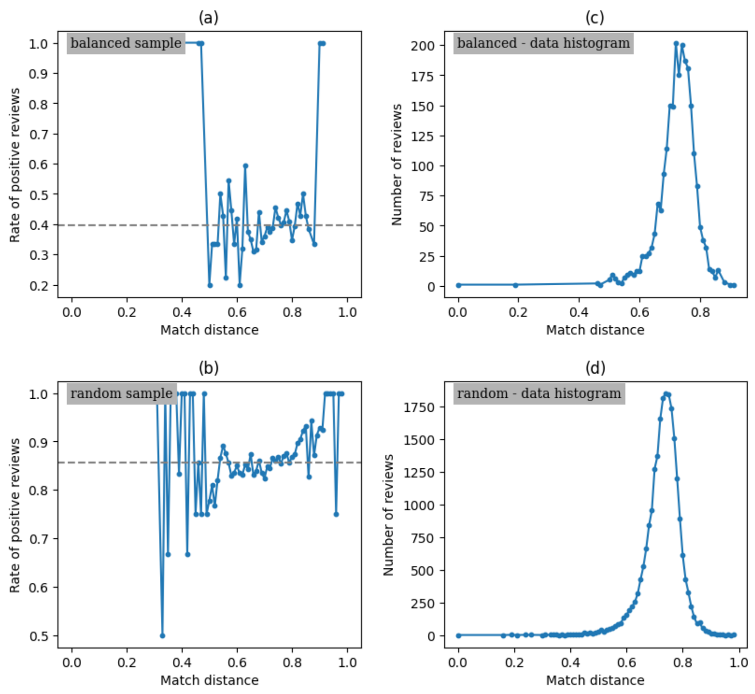


Figure B.2: Scoring function response - Rate of positive reviews to all reviews graph for KeyBERT with 1-grams, using scoring function *Mean of Half*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

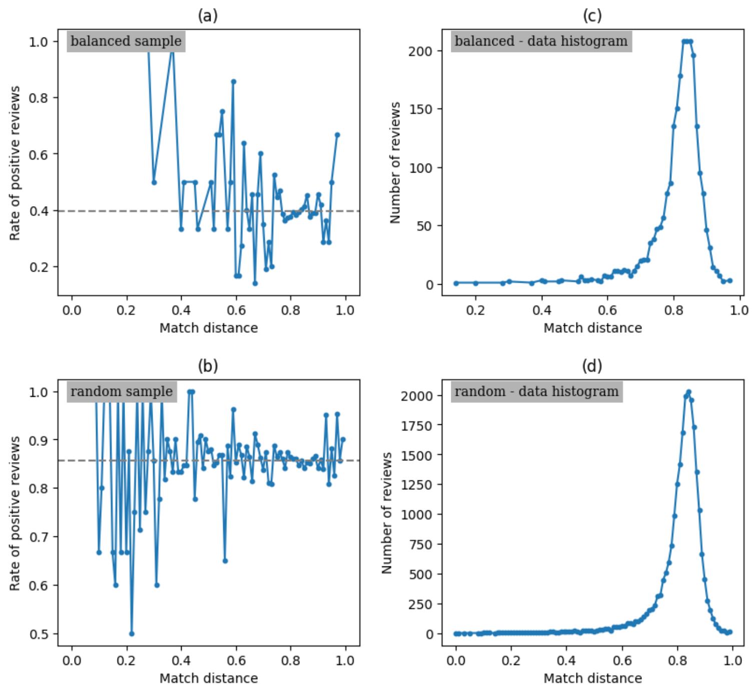


Figure B.3: Scoring function response - Rate of positive reviews to all reviews graph for KeyBERT with 2-grams, using scoring function *Mean of All*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

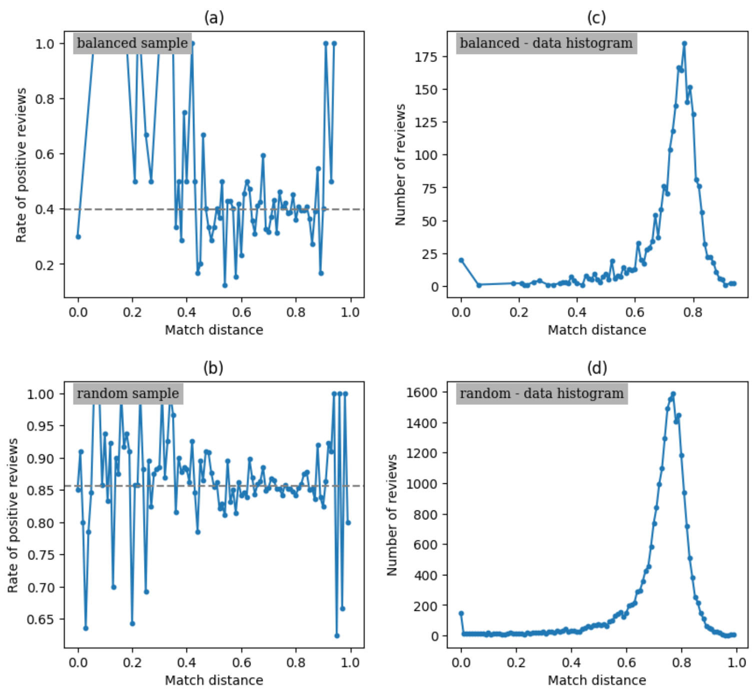


Figure B.4: Scoring function response - Rate of positive reviews to all reviews graph for KeyBERT with 2-grams, using scoring function *Mean of Half*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

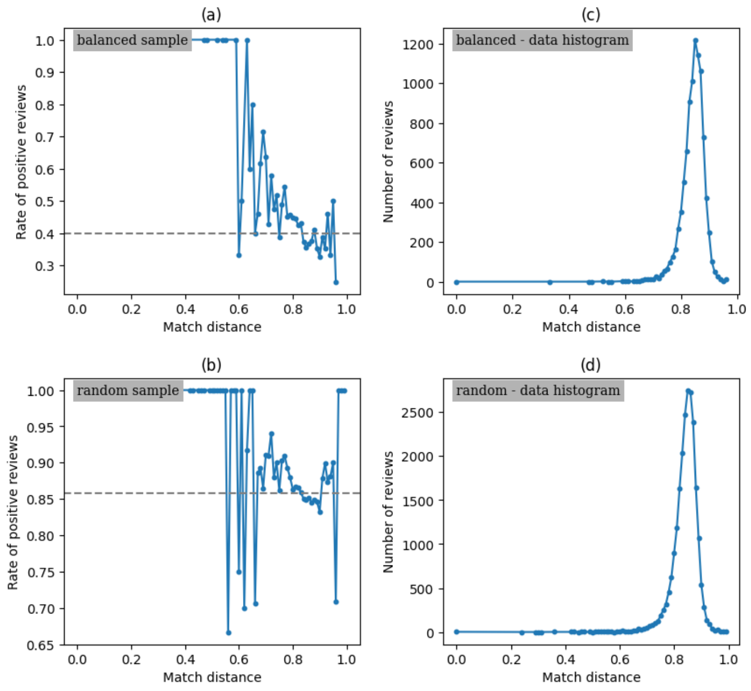


Figure B.5: Scoring function response - Rate of positive reviews to all reviews graph for Yake with 1-grams, using scoring function *Mean of All*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

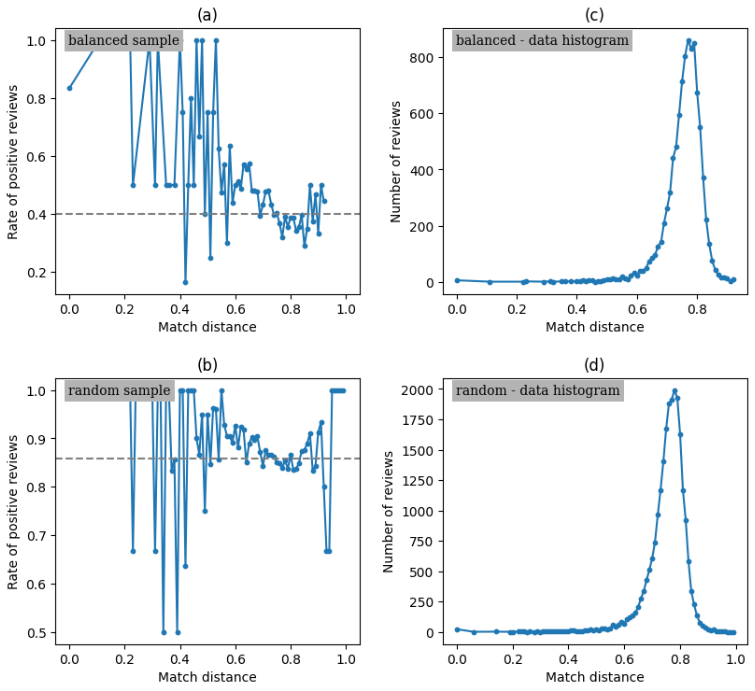


Figure B.6: Scoring function response - Rate of positive reviews to all reviews graph for Yake with 1-grams, using scoring function *Mean of Half*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

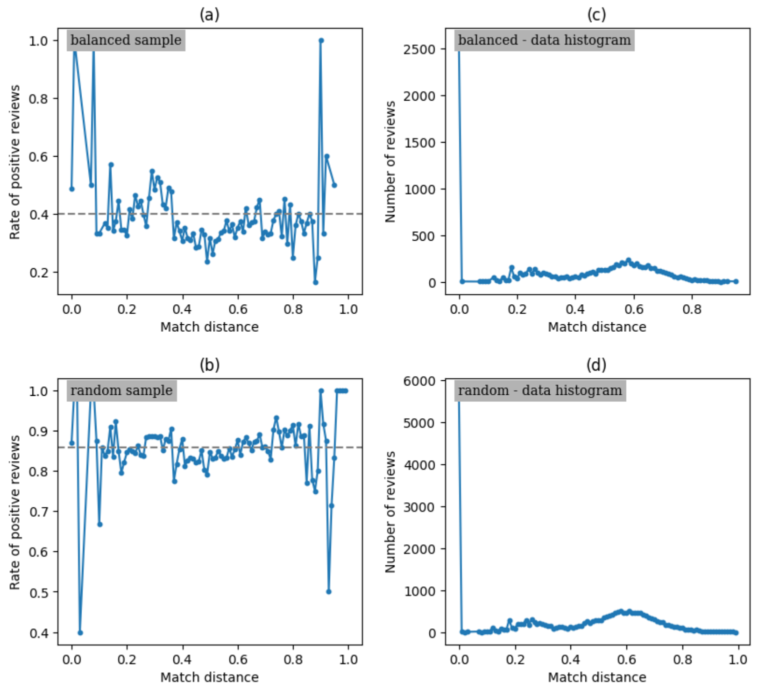


Figure B.7: Scoring function response - Rate of positive reviews to all reviews graph for Yake with 1-grams, using scoring function *Three Smallest*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

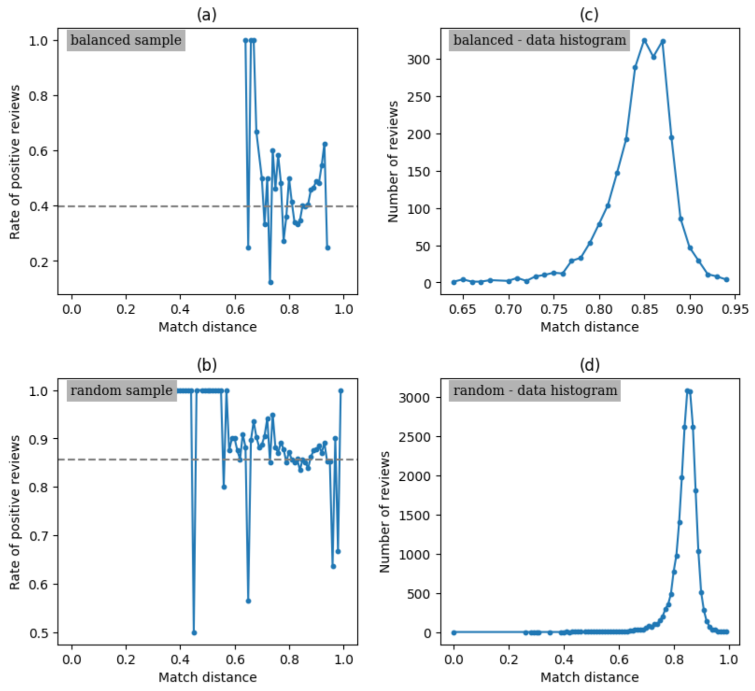


Figure B.8: Scoring function response - Rate of positive reviews to all reviews graph for Yake with 2-grams, using scoring function *Mean of All*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

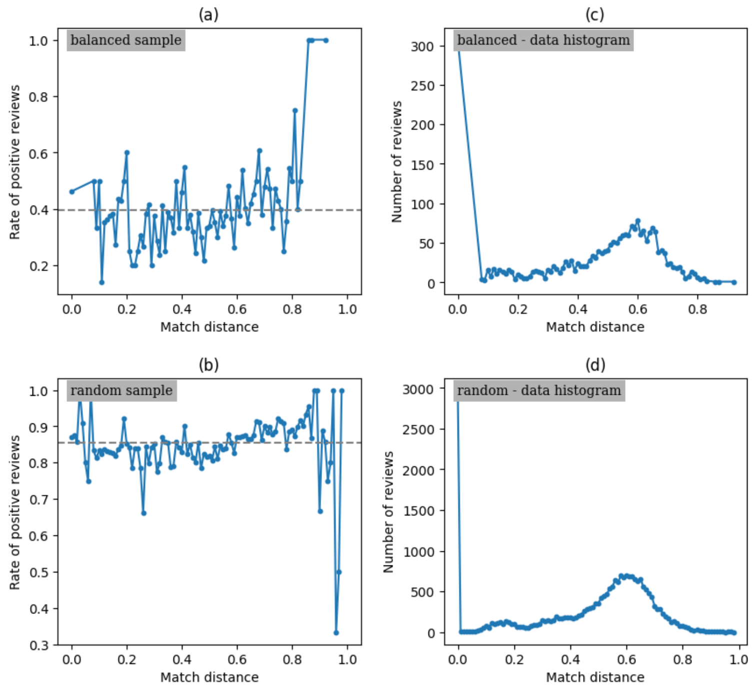


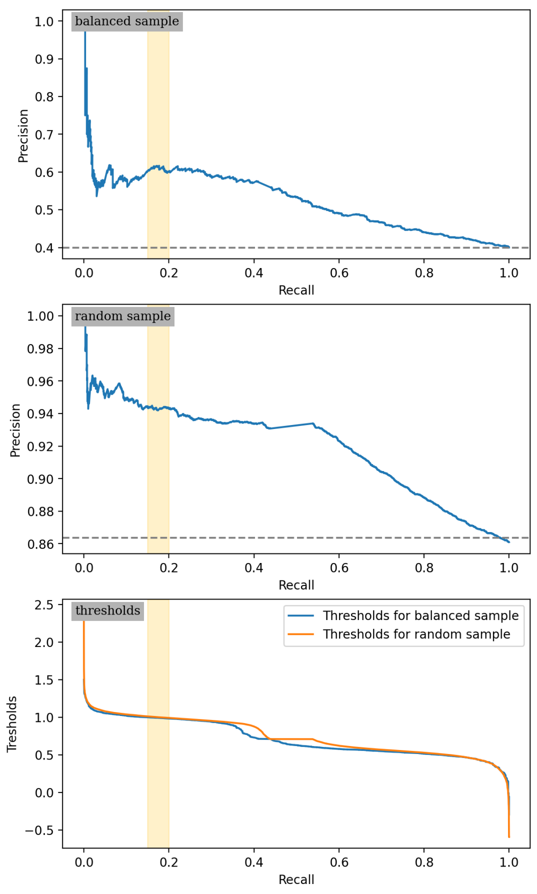
Figure B.9: Scoring function response - Rate of positive reviews to all reviews graph for Yake with 2-grams, using scoring function *Three Smallest*. The graphs in the left column visualizes the relation between the percentage of the positive ratings for values scoring function can take. In these graphs, intensity of inclination to the right indicates that the scoring function performs well. In the right column, number of ratings per the calculated match scores are presented. The figures (c) and (d) suggests that the range we are looking for a correlation should be searched around the scores $>> 0$ due to the high number of samples around these points.

APPENDIX C

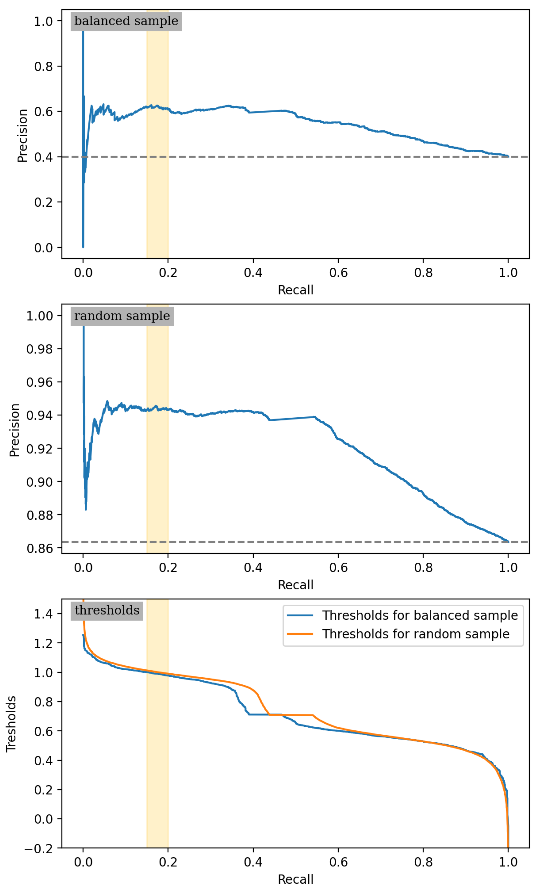
PRECISION-RECALL GRAPHS

In this section of the appendix, precision-recall graphs of the recommenders with Yake and Bert algorithms for keyword extraction is presented for both 1-grams and 2-grams of extracted phrases.

In the figures, graphs on the left columns contain the precision-recall plots of the recommenders on the validation set. The yellow bar indicates the selected threshold to achieve a good precision while having valid but a low value of recall. In the right column, same graphs for the test set is presented. The values for precision is received from these graphs by finding y axis value of the plot inside the yellow range, meaning that the resulting performance on the test set according to our decision made on validations set. There are two graphs per data set: balanced and random. As it is described in Chapter 4, balanced set consists of 2.500 data points in which the rating distribution is uniform, while the random set contains 15.000 data points sampled randomly from the dataset. The gray dashed lines shows the percentage of positive votes to all votes inside the dataset's sample.

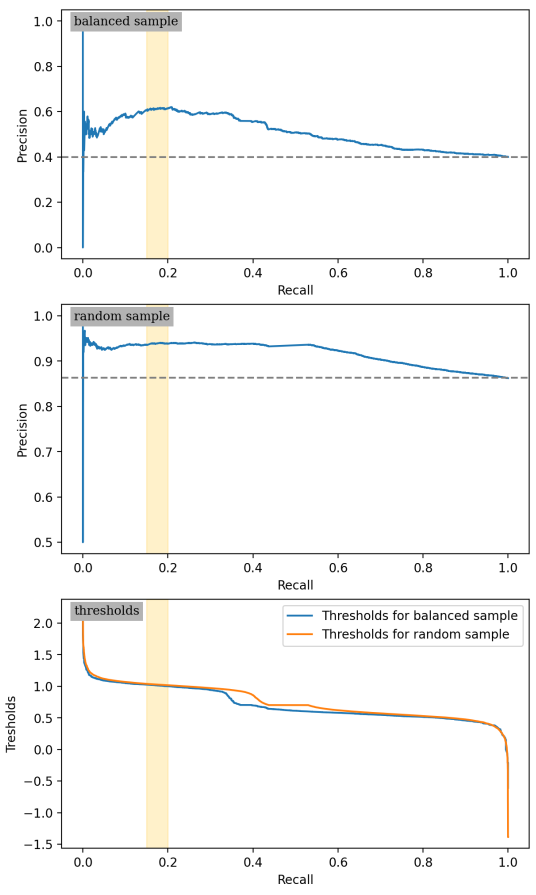


(a) Yake1 on Validation Set

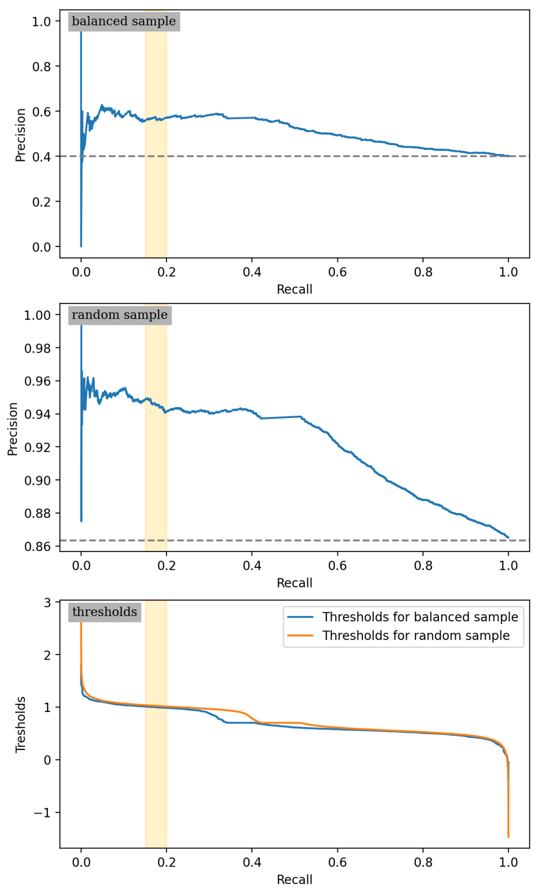


(b) Yake1 on Test Set

Figure C.1: Precision Recall Graphs of Yake with 1-grams

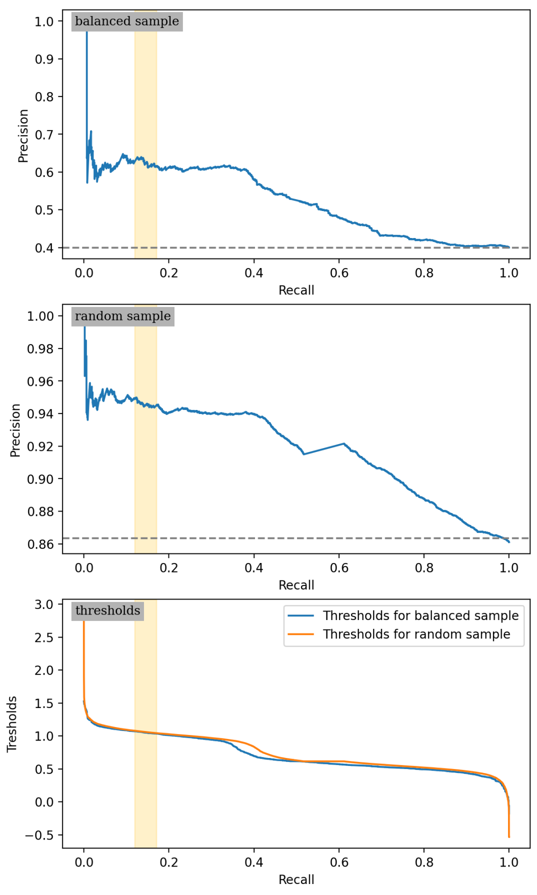


(a) Yake2 on Validation Set

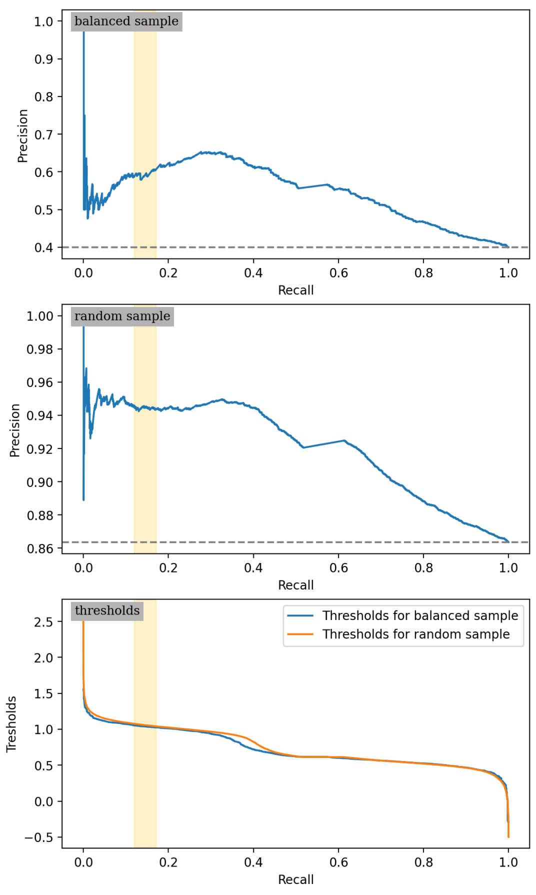


(b) Yake2 on Test Set

Figure C.2: Precision Recall Graphs of Yake with 2-grams

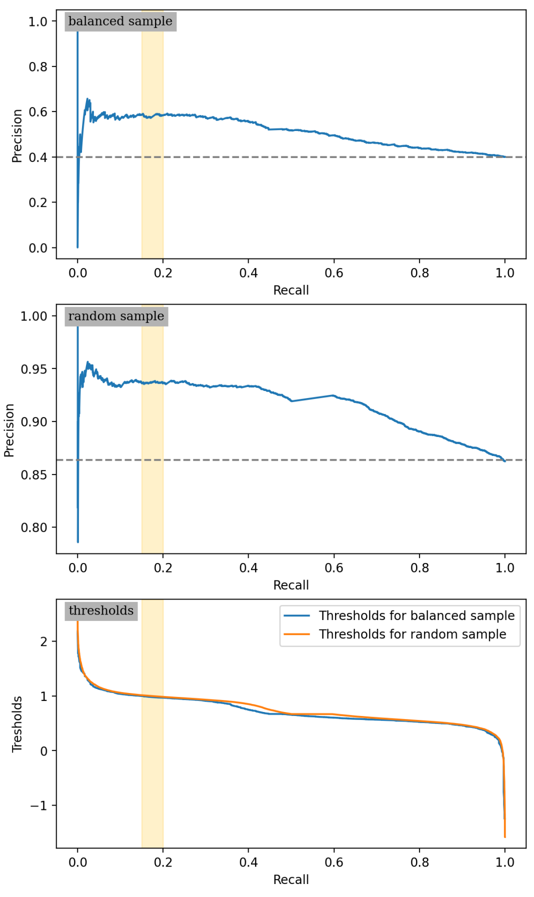


(a) Bert1 on Validation Set

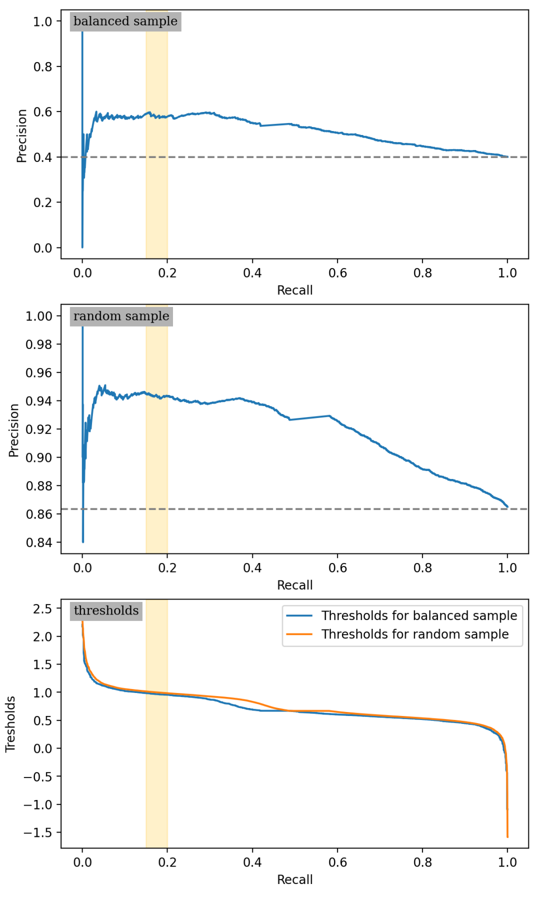


(b) Bert1 on Test Set

Figure C.3: Precision Recall Graphs of Bert with 1-grams



(a) Bert2 on Validation Set



(b) Bert2 on Test Set

Figure C.4: Precision Recall Graphs of Bert with 2-grams