Received 12 July 2022, accepted 5 August 2022, date of publication 16 August 2022, date of current version 19 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3198706

RESEARCH ARTICLE

A Hybrid Approach to Recommending Universal Decimal Classification Codes for Cataloguing in Slovenian Digital Libraries

MLADEN BOROVIČ[1], MILAN OJSTERŠEK[1], AND DAMJAN STRNAD[2]
[1]Laboratory for Heterogeneous Computer Systems, Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia
[2]Laboratory for Geospatial Modelling, Multimedia and Artificial Intelligence, Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

Corresponding author: Mladen Borovič (mladen.borovic@um.si)

ABSTRACT In this article we present a hybrid approach to recommending the Universal Decimal Classification (UDC) codes for unclassified documents. By recommending UDC codes to librarians, we can provide the decision support as part of a semi-automatic cataloguing process. As current approaches to recommending UDC codes are scarce and limited to certain fields of expertise, our motivation was to create a hybrid approach which covers all fields of expertise within the UDC hierarchy. The cascade hybrid approach combines the BM25 ranking function with a multi-label classifier based on a BERT (Bidirectional Encoder Representations from Transformers) deep neural network architecture. Additionally, lists of the most commonly used UDC codes within a document's origin organization are used as a final content-based filtering method. The BM25 ranking function is used to create an initial list of recommendations. The first cascade step re-ranks the initial list of recommendations using the list of recommendations produced by the multi-label BERT-based classifier. The second cascade step re-ranks the resulting recommendation list from the first cascade step using a list of most commonly used UDC codes within the document's organization. Finally, post-processing steps are applied to obtain the final list of recommended UDC codes. We present in detail the UDC structure, the used text corpus of documents and the functioning of our hybrid recommendation approach. We perform the analysis of the generated recommendation lists for the corpus from the Slovenian Open-Access Infrastructure using the metrics hit-ratio, normalized discounted cumulative gain, mean reciprocal rank and mean average precision. Our hybrid recommendation approach improves reference scores of individual methods for these metrics.

INDEX TERMS Hybrid recommender systems, multi-label classification, digital libraries, universal decimal classification.

I. INTRODUCTION

The interdisciplinary field of digital libraries was formed with the ever growing development of web search engines. This field of science tackles problems such as organization, storage, processing and classification of documents. Even though several different document classification methods exist, few attempts have been made towards automatic classification using library classifications such as the Universal Decimal Classification (UDC) [1], Dewey Decimal Classification (DDC) [2] and Library of Congress Classification (LCC) [3], [4]. There are also other classification systems particularly tailored for use in a certain language (e.g. there are special classifications for Chinese, Japanese and Korean texts). Many documents in libraries across the world are still being classified manually - either due to the lack of trust in automatic classification, or its inadequate performance. The lack of trust in automatic classification is understandable from the librarians' point of view, since a wrong classification during the cataloguing process can bring on a lot of additional work with document record editing. More than that, a misclassified document will be difficult to find with

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

the current search processes. In this article we address the problem of low trust in automatic UDC classification, and introduce a novel method for UDC class recommendation in digital libraries. We propose a semi-automatic classification, with the hybrid recommender system acting as a support system to the librarian, who can use its recommendations during the cataloguing process. To provide recommendations, we use the BM25 ranking function, a fine-tuned BERT-based multi-label classifier and a content-based filtering method that uses the most commonly used UDC classes within the document's origin organization. Furthermore, some recommendation post-processing techniques are applied to create the final list of recommendations.

The paper structure is as follows. In the second section we describe related work and different approaches to recommender systems implementations, as well as their uses in digital libraries. The third section overviews the Universal Decimal Classification. In the fourth section we provide details on the format and processing of our dataset from the Slovenian Open-Access Infrastructure. The fifth section details the functioning of our hybrid approach to make recommendations of the UDC codes. In the sixth section we evaluate the proposed hybrid approach and perform an ablation study based on the HR@k (hit rate at k), MAP (mean average precision), MRR (mean reciprocal rank) and NDCG@k (normalized discounted cumulative gain at k) metrics. The seventh section summarizes the findings and provides some further ideas for improvement.

II. RECOMMENDER SYSTEMS IN DIGITAL LIBRARIES AND RELATED WORK

In recent years, recommender systems have been implemented successfully in various fields for a variety of uses. They are used commonly in web search engines, social networks and multimedia services such as YouTube, Netflix, Spotify and Last.fm. The two main approaches to computer-aided recommendation are content-based filtering and collaborative filtering [5].

Content-based filtering can work on structured descriptions of items to be recommended (e.g., objects with predefined properties), as well as on unstructured data such as text. The key component of this approach is a good choice of features that describe the recommended items, so they can be compared using similarity metrics. The usual metrics used with structured data are cosine similarity, Jaccard index and Pearson correlation [6]. Unstructured data is usually represented as text, so similarity metrics are those traditionally used in natural language processing. A popular choice in this case is the *tf-idf* metric with the BM25 ranking function.

Contrary to content-based filtering, collaborative filtering uses the data about user interactions with recommendation objects. In this case, the user views, viewing times, downloads and other user actions are used to determine similar objects to recommend. For example, in an online store, a purchase is a major user interaction that can be used. Likewise, in digital libraries, a download of a document can be considered a major user interaction.

Both approaches have their disadvantages. The most prominent disadvantage of collaborative filtering methods is the cold start problem. This is a situation that occurs in the starting phase of a recommender service distribution, when the active user count is low. Consequently, user interactions are also sparse, making it difficult to provide users with quality recommendations. The main weakness of content-based filtering methods is overspecialization, where a recommender system provides users with a limited type of recommendation objects. Usually, hybrid recommender systems are employed to resolve specific disadvantages of content-based and collaborative filtering. Hybrid recommender systems make use of two or more different filtering methods. Several content-based filtering methods or collaborative filtering methods can also be used in a hybrid recommender system. Different types of hybridization approaches exist [7], [8], [9]. Using weight hybridization, a single ranking is calculated from many different used methods [10]. Switching hybrids [11] use a switching mechanism between different methods, either on demand, or depending on the goal of the recommender system. Mixed hybridization [12] is commonly used to present the results of several techniques in a single recommendation list. Feature combination hybridization combines features from several sources and inputs them into a single recommendation method [13]. In feature augmentation hybridization [14], a single recommendation method is used for obtaining features which are then used as inputs to a different recommendation method. Cascade hybridization [15] introduces a fixed sequence of recommendation techniques which lead to a final recommendation list that is a result of re-ranking. Finally, meta-level hybridization uses a single recommendation technique to build a model, which is used as an input to the next recommendation technique [16].

Recommender systems in digital libraries are used commonly to recommend documents and other digital library content [17], [18]. The recommender systems in [19] and [20] were developed specifically for use in digital libraries, with the aim of helping researchers find relevant publications. Similar recommender systems can be found in academic social networks such as Mendeley [21]. In Slovenia, a cascade hybrid recommender system using content-based and collaborative filtering was developed within the national infrastructure of open-access [22], with the aim of providing recommendations across the digital libraries and repositories of all Slovenian universities.

Although some research efforts towards automatic UDC classification have been made [23], most were limited to traditional machine learning methods [24]. Some related work has been done on the automatic classification of DDC classes. This is relevant, because DDC and UDC are similar in structure as they are both decimal classification systems. In [25] the authors used traditional machine learning methods to perform an automatic DDC classification on Swedish documents. Modern deep neural network architectures within the

TABLE 1. Main classes of the universal decimal classification.

Main classes	Discipline/branch of knowledge
0	Science and knowledge. Organization. Computer Science. Information Science. Documentation. Librarianship. Institutions. Publications.
1	Philosophy. Psychology.
2	Religion. Theology.
3	Social Sciences
5	Mathematics. Natural Sciences.
6	Applied Sciences. Medicine. Technology.
7	The Arts. Recreation. Entertainment. Sport.
8	Language. Linguistics. Literature.
9	Geography. Biography. History.

field of natural language processing such as BERT [26] have also been used to classify documents into the DDC. In [27], the authors performed a BERT-based automatic DDC classification on German documents, however the study was limited to hierarchy levels 3 and 4 within the DDC. Research has also been done on automatic classification of UDC classes. In [28] the authors present a recommender system for UDC class recommendations on mathematical articles, while in [29], a system for automatic UDC classification of older documents in the Slovenian language was presented. Based on the promising results of those efforts, we have developed a novel approach to semi-automatic UDC classification using a hybrid recommender system for documents. Another thing to note is that all of the mentioned related work defines the automatic classification as a multi-class problem, e.g. they return a single predicted UDC class from a list of candidates as a result. Our approach redefines the classification problem as a multi-label problem and therefore always returns a list of predicted UDC classes. This is relevant because documents can be classified into multiple UDC classes. Additionally, it is helpful for the end-users to receive a list of recommended UDC classes, which can aid them during the cataloguing process.

III. UNIVERSAL DECIMAL CLASSIFICATION

Universal Decimal Classification (UDC) is a bibliographic and library classification system used for content indexing and information retrieval. Its licensed paid version spans more than 70,000 classes, however, a free license version of approximately 2,600 classes also exists. Both are managed by the UDC Consortium. Each document can be assigned a class which classifies it in a certain discipline and branch of knowledge. UDC consists of main classes and auxiliary tables, where the main classes represent disciplines and branches of knowledge, and the auxiliary tables represent additional information (e.g., time, place, language and format).

A UDC code can be simple or complex. In the latter case, connecting signs are used to define the type of connection between different classes in a UDC code. Thus, interdisciplinary documents can be classified appropriately. Tables 1-3 show examples of UDC classes and codes.

TABLE 2. Hierarchical structure of UDC classes for Computer science (004), branch Computer communication, computer networks (004.7). A document with the UDC code "004.73" is implicitly classified into classes "0", "00", "004", "004.7" and "004.73", with the latter being the most specific class, "0" being its top-level class, and all the rest being intermediate classes within the UDC hierarchy.

Class	Detailed description
004	Computer science and technology. Computing. Data processing.
004.7	Computer communication. Computer networks.
004.73	Networks according to area covered.
004.738	Network interconnection. Internetworking.

TABLE 3. Example of a simple and complex UDC code. The simple UDC code contains a common auxiliary of form (043.2). The complex UDC code contains a relation connecting sign (denoted with a colon ":"), a consecutive extension (denoted with a slash "/"), a common auxiliary of form (043.2) and a common auxiliary of place (497.4).

	UDC code
Simple	519.85(043.2)
Complex	336.778(043.2):336.713/.717(497.4)

A cataloguing process is needed to obtain a UDC code for a document. This is done with a request to a librarian. The request consists of a title, abstract and keywords at a minimum, but a table of contents, authors, supervisors and other information may also be provided, depending on the type of document. Librarians then look up the provided information in glossaries and UDC mappings to define the suitable UDC code. Usually, this process can take a few days. The quality of the UDC code depends on the used glossary and mapping, the number of available UDC classes librarians can choose from, and also the cataloguing experience of the librarian.

IV. TEXT CORPUS AND PREPROCESSING

In this article, we used a text corpus from the Slovenian open-access infrastructure [22], which was established in 2013 and consists of theses and scientific publications from all Slovenian universities. The text corpus [30], [31] consists of 114,485 documents in the Slovenian language, and is segmented into keywords, titles, abstracts, full-texts and additional bibliographic metadata, including UDC codes.

Because the majority of documents had complex UDC codes, we implemented a UDC code parser that extracts the main UDC codes from a complex UDC code. The logic for parsing relation, coordination and addition connections, consecutive extensions and subgrouping was implemented. Other syntax features of the UDC were not supported, because a negligible amount of documents in the corpus featured UDC codes with them.

We linked the free version of the UDC to our corpus using the available linked open data [32]. It consists of approximately 2,600 UDC classes with English and Slovenian translations. Because of the restricted number of freely available classes, the parsing was limited in depth for some branches of the Universal Decimal Classification, as demonstrated in Table 4.

After processing the UDC codes with the parser, we analyzed the distribution of processed UDC classes in the chosen

TABLE 4. Example of inputs and outputs of our UDC code parser. The underlined UDC code 621.952.8 was parsed as 621.9 due to limitations of the freely available classes in the linked open data collection. The example also shows that our UDC code parser ignores the common auxiliaries - in this case, the auxiliary of form (043.2) and place (497.4Koper).

Input	Output
[004.94:<u>621.952.8</u>]+658.8(043.2)	004.94
	<u>621.9</u>
	658.8
711.4:711.1:158.937:003.63(497.4Koper)(043.2)	711.4
	711.1
	158.937
	003.63

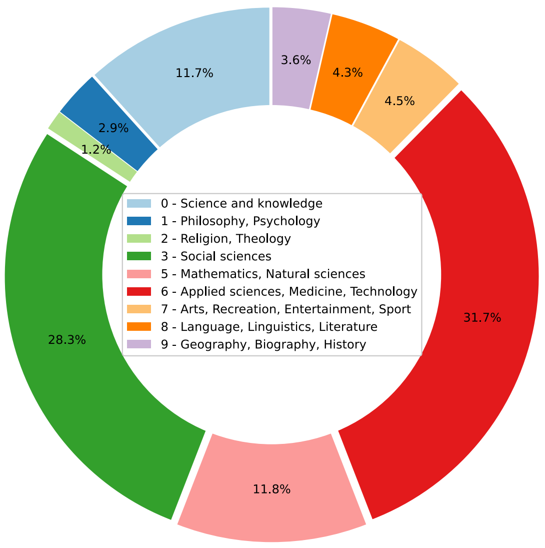

FIGURE 1. Distribution of documents in the corpus based on the top-level UDC classes.

text corpus. Specifically, we studied the hierarchy depth of processed UDC codes, since it directly affects the categorization specificity. For example, these UDC codes appear as "004" for depth of 3, "7.03" for depth 4, "004.7" for depth 5, "004.73" for depth 6, "004.738" for depth 7 and "004.738.5" for depth 9.

Figure 1 shows the distribution of documents in the text corpus with respect to the main UDC classes. We can observe that the top represented UDC branches are 6 - Applied sciences, Medicine, Technology (31.7%), 3 - Social sciences (28.3%), 5 - Mathematics, Natural sciences (11.8%) and 0 - Science and technology (11.7%). Figure 2 shows the distribution of UDC classes in the text corpus after processing with respect to the hierarchical depth of the UDC classes. The most frequent UDC code hierarchical depths were 5, 6, 4, 3 and 7 respectively. This means that the majority of catalogued documents with a UDC code in the corpus contained a good degree of specificity. Furthermore, documents in the corpus, catalogued with a UDC code of hierarchical depth 1 are very rare (0.3%).

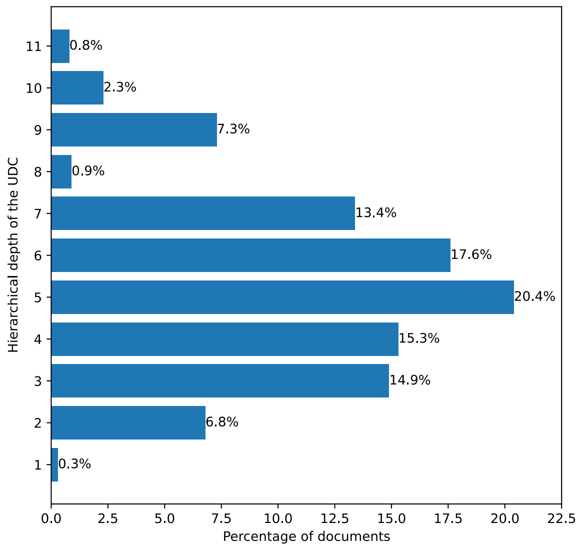

FIGURE 2. Distribution of documents in the corpus based on the hierarchical depth of the UDC class after processing.

V. HYBRID APPROACH TO RECOMMENDATIONS

In the proposed hybrid recommendation approach (Figure 3), we use the cascade hybridization with the BM25 ranking function and a multi-label BERT-based classifier fine-tuned on the hierarchical structure of the UDC. The input into the recommendation process is a document feature structure, consisting of the title, keywords, abstract and the document's origin organization code. The output of the recommender system is a list of recommended UDC codes. Within the cascade hybridization, the first step is to obtain the initial list of UDC codes with the BM25 ranking function. Similarly, a list of UDC codes is obtained using the multi-label BERT-based classifier using the branch hierarchy of the UDC. This list then re-ranks the initial list of UDC codes. The UDC codes that appear in both lists are ranked higher. Next, the second step of the cascade hybridization re-ranks the resulting list of the first step using the most commonly used UDC codes depending on the document's origin organization. The lists of the most commonly used UDC codes for each organization have been pre-generated. Finally, the post-processing steps of top-level boosting, hierarchical specificity boosting and cut-off are applied. Each of the post-processing steps defines parameters which can be optimized. Top-level boosting increases the ranking scores of all UDC codes in the recommendation list that match the top 3 ranked UDC codes in the recommendation list by their top-level UDC branch. Hierarchical specificity boosting proportionally increases the ranking scores of the UDC codes within the recommendation list, with respect to their depth within the UDC hierarchy, such that deeper UDC codes are boosted more. The cut-off procedure removes the UDC codes from the recommendation list, that have their ranking scores lower than a dynamically set cut-off value, which is calculated as a proportion of the highest ranking score (e.g. 50%).

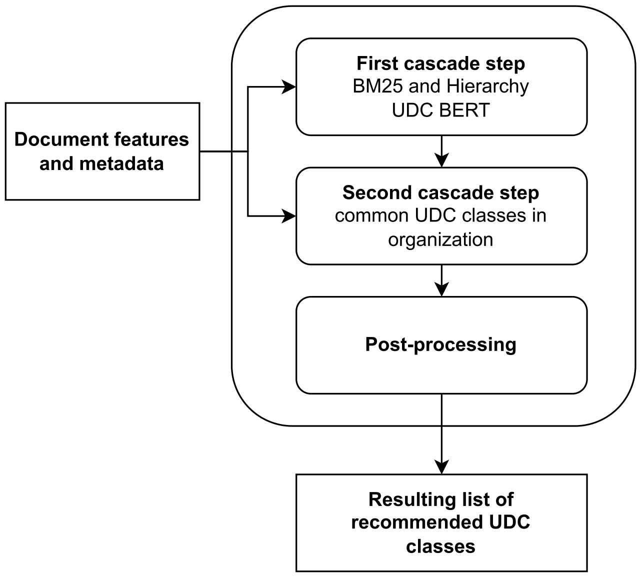
First cascade step
BM25 and Hierarchy UDC BERT
Second cascade step
common UDC classes in organization
Post-processing
Document features and metadata
Resulting list of recommended UDC classes

FIGURE 3. **Proposed hybrid recommendation approach, employing two cascade hybridization steps and post-processing. The inputs are the document features constructed from titles, keywords and abstracts. The first cascade step generates an intermediate list of recommendations which is the input to the second cascade step. The second cascade step produces a list of recommended UDC classes which is post-processed and returned as a result of the hybrid recommendation.**

A. FILTERING METHODS

The idea of our approach is to use the BM25 ranking function to find the k most suitable UDC codes and construct an initial list of recommendations. Following this initial ranking, we rely on a fine-tuned multi-label BERT classifier to produce an independent recommendation list using the levels of hierarchy within the UDC.

The BM25 ranking function is widely used in search engine implementations, and as a full-text search functionality in commercial solutions such as Microsoft SQL Server and MySQL. Fine-tuned BERT models are most commonly associated with the categorization and classification tasks within the field of natural language processing. In our hybrid approach, BERT served to improve the rankings of UDC codes that were obtained by the BM25 ranking function. Similarly, the aggregation of the most commonly used UDC codes by the document's origin organization serves to capture suitable UDC codes that could have been missed in the previous passes.

1) BM25

BM25 is a ranking function used to determine document similarity [33]. It was first used in the 1970's as part of a probabilistic information retrieval framework. BM25 is actually a family of methods that differ in weight schemes and parameter values. The tf and idf weights are usually used across all variants of the ranking function. There are many variants of BM25 which bring improvements in specific cases [34], [35], [36]. We used two variants of the BM25 ranking function, to test the performance of different query weight schemes. Let us denote by D the corpus of documents, and by Q the input search query to the recommender system. In the first variant, the BM25 score $s(d, Q)$ is calculated as:

$$s(d, Q) = \sum_{i=1}^{|Q|} \mathrm{idf}(q_i) \cdot \frac{\mathrm{tf}(q_i, d) \cdot (k_1 + 1)}{\mathrm{tf}(q_i, d) + k_1 \cdot B} \tag{1}$$

where $q_i \in Q$, $d \in D$ and

- $\mathrm{tf}(q_i, d)$ is the frequency weight, i.e. the number of occurrences of a query term q_i in document d
- k_1 is a parameter with a typical value between 1 and 3. For instance, in [37] the authors recommend the value $k_1 = 1.2$, [33] used $k_1 = 2.0$, and [38] used $k_1 = 2.5$.
- $\mathrm{idf}(q_i)$ is the inverse document frequency weight for the query term q_i. It indicates the frequency of the term q_i within the complete corpus, and is calculated as:

$$\mathrm{idf}(q_i) = \log \frac{|D| - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{2}$$

  Here, $|D|$ is the number of documents in the corpus D, and $n(q_i)$ is the number of documents that contain the term q_i.
- B is a normalization factor, computed as:

$$B = 1 - b + b \cdot \frac{l_d}{l_{avg}} \tag{3}$$

  where l_d is the length of document d, and l_{avg} is the average length of documents in the corpus. The length of a document is the number of terms in the document. The values for parameter b usually range from 0 to 1. In [37] the authors recommend the value $b = 0.75$, which is also used by other studies [33]. In [38], the value $b = 0.8$ is used, while earlier variants of this ranking function used $b = 0$ (BM11) or $b = 1$ (BM15).

The second variant of BM25 we used calculates the BM25 score $s'(d, Q)$ as:

$$s'(d, Q) = s(d, Q) \cdot \frac{(k_3 + 1) \cdot \mathrm{qtf}(q_i)}{k_3 + \mathrm{qtf}(q_i)} \tag{4}$$

where $q_i \in Q$ and $d \in D$. The meaning of new quantities introduced on the right side of Equation (4) is:

- $\mathrm{qtf}(q_i)$ is the query term frequency weight for the term q_i. The value is the number of occurrences of the term q_i within the query Q.
- k_3 is a parameter with a commonly used value $k_3 = 0$, although $k_3 = 1,000$ [39] and $k_3 = 7$ [38] have also been used. It is worth noting that Equation (1) is a special case of Equation (4), when $k_3 = 0$, i.e., the query frequency term is ignored.

Using the BM25 ranking function scores $s(d, Q)$ we can find the documents in the corpus that are most similar to the input document, and use the UDC parser on their UDC codes to extract the UDC classes they were catalogued in. In this case, higher BM25 scores translate to higher similarity between the documents. A list of UDC classes, sorted by the decreasing frequency of the occurring class, is then generated, and the first k elements are returned as recommended classes.

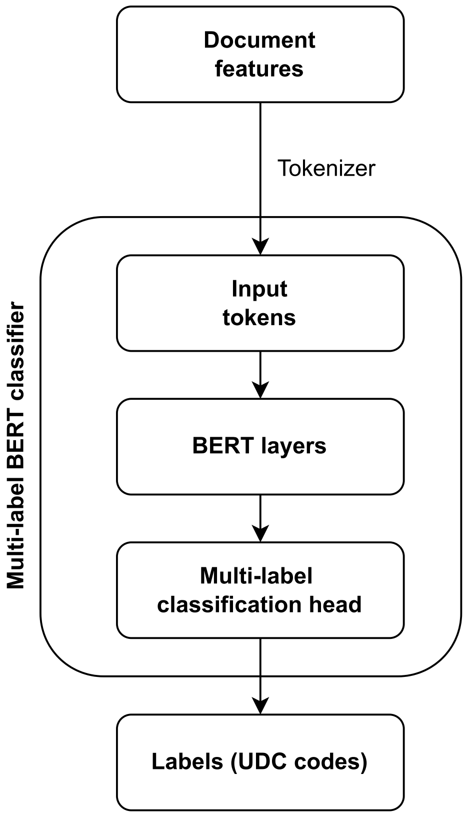

FIGURE 4. Structure of our multi-label BERT-based classifier.

2) MULTI-LABEL BERT CLASSIFIER

A multi-label BERT-based classifier has been trained to be used within the cascading hybridization during the recommendation process. The classifier performs multi-label classification of documents to provide relevant UDC codes depending on the input text. This is a major difference when compared to related work, as all of the approaches performed multi-class classification, where the model is trained to put all weight on a single target class per document. This is important, because their work outputs only one UDC code, which corresponds to a single class within the classifier. As we can observe from examples in Table 4, in real-world scenarios there can be more than one UDC code given to a document during the cataloguing process. To avoid this limitation in our classifier, we redefined the problem of document classification from a multi-class type to a multi-label type. That way, the output of the classifier is a list of scores for UDC codes which correspond to different labels within the classifier. By doing so, we implicitly support recommendation of several UDC codes for a document, which is especially relevant for interdisciplinary documents.

The multi-label BERT-based classifier was trained on all branches of the UDC hierarchy, which covers all fields of expertise within the freely available UDC summary. The idea behind the classifier is to filter relevant UDC codes within the UDC hierarchy with a degree of specificity. That way, more specific UDC codes can be obtained as a result from the classifier, improving the cataloguing process of the document.

The training data consisted of documents from the corpus described in Section IV and their corresponding parsed UDC codes. Each document was labelled with its corresponding UDC code in one of the freely available UDC codes within all branches of the UDC hierarchy [32]. A binary vector was created to represent the UDC codes as target labels for each document. A component in the binary vector was set to value 1 if the corresponding code was part of the document's multi-label identifier hierarchy, and to 0 otherwise. This was done for all hierarchical levels of the UDC hierarchy. For example, a document with the UDC code "004" would have components "0", "00" and "004" in the binary vector set to value 1, and all the rest set to 0. The dataset was split using a 70/15/15 split for training, validation and testing, respectively.

Figure 4 shows the structure of the classifier. First, the document features are tokenized and input tokens are processed with the 12 layers that are part of $\text{BERT}_{\text{Base}}$. Following this, a classification head with sigmoid activation is applied for multi-label classification. Finally, using a threshold value $\theta = 0.5$, the output labels are stored as a list in descending score order.

B. FIRST CASCADE HYBRID STEP

In the first cascade hybrid step (Figure 5), an initial list of recommended UDC codes is produced using the BM25 ranking function. The inputs are document features consisting of titles, keywords and abstracts, which are taken by BM25 as a search query. The BM25 ranking function then outputs all documents that are relevant to the search query, along with their UDC codes. Those UDC codes and their corresponding BM25 scores are taken and stored in a ranked list of initial recommendations R_0 which has a fixed cut-off length of 15 results. Then, the BERT tokenizer is used to transform the document feature into tokens, which are the input to the multi-label hierarchy UDC BERT classifier. The output is a list of output labels L_{hier} representing recommended UDC codes by the classifier. The resulting labels are used with the list of initial recommendations R_0 to re-rank it. The scores of UDC codes within R_0 which are similar to the labels in L_{hier} are boosted using the equation:

$$\sigma_{R_1} = \sigma_{R_0} + \left(\sigma_{R_0} \cdot \text{sim}_{jw}(r, l)\right) \tag{5}$$

where σ_{R_1} is the new score, σ_{R_0} is the old score, and $\text{sim}_{jw}(r, l)$ is the Jaro-Winkler similarity [40] between the UDC code r from R_0 and the label l from L_{hier} representing the UDC code. The Jaro-Winkler similarity is defined on the interval [0, 1], where 0 means no similarity, and 1 means exact match. It gives higher values for similarity to strings that match from the beginning, which suits our task of matching UDC codes by hierarchical depth. Finally, the resulting list of UDC codes and their corresponding scores is sorted by score in descending order to complete the re-ranking process. In the next steps, this list is referenced as R_1.

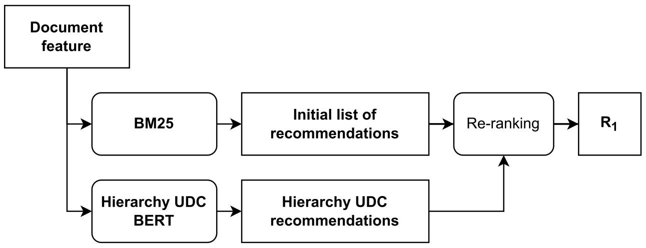
Document feature
BM25
Initial list of recommendations
Re-ranking
R1
Hierarchy UDC BERT
Hierarchy UDC recommendations

FIGURE 5. **Workflow diagram of the first hybrid cascade step.**

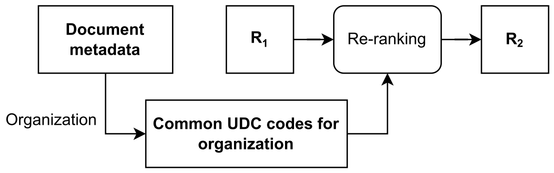

FIGURE 6. **Workflow diagram of the third hybrid cascade step.**

C. SECOND CASCADE HYBRID STEP

As the input, the second cascade hybrid step (Figure 6) takes the ranked list of recommended UDC codes R_1 from the second cascade hybrid step. The document's origin organization, which is part of the document metadata, is extracted and a pre-generated list of the most commonly used UDC codes of documents originating from that organization (L_{org}) is obtained and used with the list R_1 to re-rank it. The scores of UDC codes within R_1 are boosted using the equation:

$$\sigma_{R_2} = \sigma_{R_1} + \left(\frac{\sigma_{\text{org}}}{\sigma_{L_{\max}}} \right) \tag{6}$$

where σ_{R_2} is the new score, σ_{R_1} is the old score, σ_{org} is the score for the UDC code in L_{org} which equals to the number of occurrences of the UDC code for documents within that organization, and $\sigma_{L_{\max}}$ is the maximum score in L_{org}. Once more, the resulting list of UDC codes and their corresponding scores is sorted by score in descending order to complete the re-ranking process. In the next step, this list is referenced as R_2.

D. POST-PROCESSING

Once the list of recommended UDC codes is prepared, post-processing steps follow to further improve recommendations. In general, the post-processing steps are used to improve the user experience by applying logic that is tailored to a specific user. In our recommendation approach, post-processing is used in an attempt to improve the ranking of highly specific UDC codes, as the librarians tend to catalogue the documents in the same way. This can be observed in Figure 2, which shows the distribution of documents in the corpus based on the hierarchical depth of the UDC class. Three different post-processing techniques were applied and evaluated. Top-level boosting was used to improve the rankings of UDC codes depending on their top-level UDC branch. Specificity boosting was used to improve the rankings of highly specific UDC codes. A cut-off was used to remove low-ranking UDC codes from the list of recommendations.

1) TOP-LEVEL BOOSTING

The rankings of recommended UDC codes can be further improved by boosting the scores of all UDC codes in the recommendations that fall under the top-level UDC branches of the top 3 recommended UDC codes. This post-processing step aims to provide a more unified recommendation in the sense, that it will focus on relevant top-level UDC branches and provide some variety of UDC subclasses from that point on. The boosted score is calculated using the equation:

$$\sigma_T = \sigma_{R_2} + \left(\sigma_{R_2} \cdot \frac{\delta}{|R_2|} + \log_2(r_{\text{rank}} + 1) \right) \tag{7}$$

where σ_{R_2} is the score of the recommended UDC code r, r_{rank} is the rank of the recommended UDC code r, δ is the score difference between the first and second ranked UDC codes, representing the dominance of the top result, and $|R_2|$ is the size of the ranked recommendation list R_2. The boost is added to every UDC code score in the list of recommendations. Finally, the list is re-ranked by boosted score in a descending order.

2) SPECIFICITY BOOSTING

Librarians tend to provide very specific UDC codes when cataloguing new documents. Within the UDC hierarchy, this means a range of specific fields of expertise that are related to each other and are a part of the same top-level branch. To achieve more human-like recommendations, this post-processing step boosts the scores for UDC codes that are longer in length (are more specific). The boosted score is calculated using the equation:

$$\sigma_S = \sigma_T + \left(\lambda_r \cdot \frac{\log_2(r_{\text{rank}} + 1)}{\lambda_{\max}} \cdot \phi \right) \tag{8}$$

where λ_r is the length of the UDC code in the list of recommendations, $\lambda_{\max}$ is the length of the longest UDC code in the list of recommendations and $\phi \in [0, 1]$ is a scaling factor regulating the boost value. This post-processing step adds the boost to every UDC code score in the list of recommendations. Lastly, the list is re-ranked by boosted score in a descending order.

3) CUT-OFF

In some cases, an imbalance of recommended UDC code scores can occur. The top recommended UDC codes will have a much larger score than the rest of the recommended UDC codes. A question can be raised whether these low-ranking recommendations should be kept in the recommendation list, or removed from it. Using a cut-off threshold T, this post-processing step aims to exclude the recommendations with significantly lower scores than the top-ranked recommendation. The threshold is calculated using the equation:

$$T = \gamma \cdot \max \sigma_r \tag{9}$$

where $\max \sigma_r$ is the score of the top-ranking UDC code in the list of recommendations and $\gamma \in [0, 1]$ is a free parameter representing the percentage of the top score needed to be kept in the list of recommendations. All UDC codes with scores lower than the value of threshold T are excluded from the recommendations.

VI. EXPERIMENTAL SETUP AND EVALUATION

Due to the different purposes of recommender systems, many different approaches exist for evaluating them. Several useful and popular methods are presented in the studies [41], [42], [43], [44], [45]. With so many diverse options for evaluation, we must first define the end result that we are trying to accomplish with the inclusion of a recommender system [46]. In our case, we are using a hybrid recommendation approach that uses content-based filtering methods to obtain a list of UDC codes that are suitable for the text input. Intuitively, we turn to metrics such as precision and recall, which are well established in use with search engines and information retrieval [47], [48]. Even though these two metrics evaluate how successful a search system is, they are not able to evaluate the user experience, which is arguably more relevant to recommender systems [49]. With our user scenario, the librarians cataloguing the documents encounter a problem of sifting through a large number of UDC codes to select a suitable one. This can be time consuming, and even confusing in some cases. In this scenario, the librarians are satisfied with a smaller set of relevant UDC codes that enables them to catalogue the documents faster. It is also satisfactory if they are provided a top-level UDC code branch from where they can work through the UDC hierarchy manually and select the suitable UDC codes independently.

From an information retrieval perspective, the usual metrics used for evaluation of such systems are precision, recall and a variety of F_β metrics, with the F_1 metric being the one used most commonly. This is because the F_1 metric represents a harmonic mean between precision and recall, weighing them both equally with the idea that a system should perform equally well in precision as well as in recall. However, the recall metric is the preferred metric to measure in our scenario, since the final selection of the most appropriate UDC codes used for classification will always be done manually by the end-user. In the field of recommender systems, the recall metric is also known as hit-ratio (HR). Furthermore, the hit-ratio can be measured at a cut-off value k within the list producing a metric HR@k. Its calculation is given with Equation (10), where $n_{\text{rel},k}$ is the number of correct recommendations in the recommendation list cut-off at k, and $n_{\text{all, k}}$ is the number of all recommendations in the recommendation list cut-off at k.

$$\text{HR@}k = \frac{n_{\text{rel},k}}{n_{\text{all},k}} \tag{10}$$

Another prevalent metric used for recommender systems evaluation is the normalized discounted cumulative gain (NDCG) which measures the ranking quality of the produced recommendation list. It can be measured at a cut-off value k as well, producing a metric NDCG@k. Its calculation is given with Equation (11), where rel_i is the relevance score of the recommendation at position i, IDCG@k is the ideal discounted cumulative gain obtained with the ideal order of recommendations, and $|I(k)|$ is the length of the ideally ordered list of recommendations limited to k elements.

$$\begin{aligned} \text{DCG@}k &= \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{log_2(i+1)} \\ \text{IDCG@}k &= \sum_{i=1}^{|I(k)|} \frac{2^{\text{rel}_i} - 1}{log_2(i+1)} \\ \text{NDCG@}k &= \frac{\text{DCG@}k}{\text{IDCG@}k} \end{aligned} \tag{11}$$

Additionally, metrics such as mean average precision (MAP) and mean reciprocal rank (MRR) are commonly used to evaluate recommender systems. The calculation of MRR is given with Equation (12), where N is the number of all resulting recommendation lists, and rank_i is the rank of the first correct recommendation in the i-th resulting recommendation list.

$$\text{MRR} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{12}$$

The calculation of MAP is given with Equation (13), where AP is the average precision calculated for each resulting recommendation list, n is the length of the resulting recommendation list, P@k is the precision score at cut-off k in the resulting recommendation list, $rel(k)$ is an indicator function that equals 1 if the recommendation at rank k is relevant, and 0 otherwise, and n_{rel} is the number of relevant recommendations. The MAP metric is calculated as the mean of the AP scores over all resulting recommendation lists.

$$\begin{aligned} \text{AP} &= \frac{\sum_{k=1}^{n} \text{P@}k \cdot rel(k)}{n_{\text{rel}}} \\ \text{MAP} &= \frac{1}{N} \cdot \sum_{i=1}^{N} \text{AP}_i \end{aligned} \tag{13}$$

We decided to perform an ablation study using the metrics HR@k, NDCG@k, MAP and MRR to analyze the retrieval and ranking quality of the recommendations. For the parameter k with HR@k and NDCG@k we used values 3, 5, and 10 as they represent a traditional set of choices for the number of results returned by the recommender system.

We compared the performance of different hybrid recommender system configurations. As a baseline, we used a variant of recommendation that included only the BM25 ranking function. We tested the first BM25 variant with parameters at values $k_1 = 1.2$ and $b = 0.75$, and the second BM25 variant with query term frequency using parameter values $k_3 = 7$ and $k_3 = 1,000$. Both variants performed comparably, meaning that the addition of the query term frequency component in the weighing scheme was not significant for performance. This phenomenon has been discussed in related work [50], and we only included the results of the first BM25 variant without the query term frequency component. The baseline

TABLE 5. Tested hybrid recommender system configurations. The checkmark (✓) denotes the use of the functionality and the minus (-) denotes the absence of the functionality.

Method	Cascade step 1	Cascade step 2	Top-level boosting	Specificity boosting	Cut-off
casc-hyb-no-org	✓	-	-	-	-
casc-hyb-full	✓	✓	-	-	-
casc-hyb-full-A	✓	✓	✓	-	-
casc-hyb-full-B	✓	✓	✓	✓	-
casc-hyb-full-C	✓	✓	✓	-	✓
casc-hyb-full-D	✓	✓	✓	✓	✓

TABLE 6. Scores for metrics HR@*k*, NDCG@*k*, MRR and MAP on the tested hybrid recommender system configurations. The method variants marked with an asterisk (*) used the top 3 levels of the UDC hierarchy during top-level boosting, while the methods without it used only the top level of the UDC hierarchy.

Method	HR			NDCG			MRR	MAP
	HR@3	HR@5	HR@10	NDCG@3	NDCG@5	NDCG@10		
bm25 (baseline)	0.766	0.766	0.766	0.831	0.826	0.826	0.740	0.715
casc-hyb-no-org	0.783	0.783	0.783	0.840	0.836	0.836	0.753	0.722
casc-hyb-full	0.787	0.787	0.787	0.842	0.838	0.838	0.765	0.751
casc-hyb-full-A	0.798	0.798	0.798	0.854	0.843	0.843	**0.774**	**0.754**
casc-hyb-full-A*	0.798	0.798	0.798	0.874	0.855	0.855	0.767	0.752
casc-hyb-full-B	**0.812**	**0.812**	**0.812**	0.883	0.868	0.868	0.760	0.739
casc-hyb-full-B*	0.809	0.809	0.809	**0.901**	**0.885**	**0.885**	0.755	0.728
casc-hyb-full-C	0.798	0.798	0.798	0.854	0.843	0.843	0.765	0.748
casc-hyb-full-C*	0.798	0.798	0.798	0.873	0.859	0.857	0.758	0.738
casc-hyb-full-D	**0.812**	**0.812**	**0.812**	0.883	0.868	0.868	0.760	0.736
casc-hyb-full-D*	0.809	0.809	0.809	**0.901**	**0.885**	**0.885**	0.754	0.725

method is annotated with "bm25". Table 5 shows the properties of the tested hybrid recommender system configurations and their notations. We tested the full range of values for parameters ϕ and γ which are defined on the interval [0, 1]. We obtained the best results with the scaling factor $\phi = 0.8$ for specificity boosting, and the free parameter $\gamma = 0.5$ for cut-off, which we used in our evaluation. For top-level boosting, we also tested an option where the top-level boosting was not limited to only the top level, but rather to the top 3 levels of the UDC hierarchy. This was because most UDC codes consist of at least 3 levels of the UDC hierarchy and UDC codes of depth less that 3 are only used for certain fields of expertise in real-world scenarios (e.g. Architecture ("72") and History ("93" and "94")). In Table 6, these variants are marked with an asterisk (*).

Table 6 shows the values of metrics HR@k, NDCG@k, MRR and MAP for the tested hybrid recommender system configurations. From the results it can be observed that the cascade hybrid recommender system approaches with different combinations of post-processing steps improve on the reference scores of individual methods. Another observation is, that the scores for HR@k stay consistent no matter whether the limit of the recommendations list is 3, 5 or 10. This indicates that the tested approaches mostly capture the correct UDC codes in the top 3 positions of the resulting recommendation lists. It is worth mentioning here that a catalogued document rarely has more than 2 different UDC codes attached to it. As for NDCG@k, the scores generally decrease as the number of recommendations in the list increases. This is expected, since this metric measures the ranking quality using the relevance score, and, as the length of a recommendation list grows, the scores of lower ranked recommendations get lower values. We also observe that the biggest decrease for this metric is between 3 and 5 given recommendations in the list.

The variant "casc-hyb-no-org", which does not use the second cascade hybrid step, improves the metric scores when compared to the baseline, but is not performing as well as the variant "casc-hyb-full" which uses both cascade hybrid steps. This indicates that the use of the multi-label BERT classifier provides an improvement of results, however the contribution of metadata about the organization is also important. Since these organizations are faculties and research institutions in specific fields of expertise, this metadata maps their fields of expertise to their defined specific UDC codes.

In general, we observe that approaches using post-processing steps further improve metric scores when compared to approaches not using post-processing steps. We can observe that the variant "casc-hyb-full-A", which does not use specificity boosting and cut-off, gives the best values for metrics MRR and MAP. This indicates that the best recommendations are ranked well with this approach, however the values of HR@k and NDCG@k for this method suggest that other good recommendations are either not present or are ranked very low in the final recommendation list. Additionally, the variant "casc-hyb-full-B", which does not use cut-off, and variant "casc-hyb-full-D", which uses all post-processing steps, give the best values for the metric HR@k.

Similarly, the variant "casc-hyb-full-B*", which does not use cut-off, and variant "casc-hyb-full-D*", which uses all post-processing steps, give the best values for the metric NDCG@k. Both of these variants used top-level boosting which was not limited to only the top level, but rather the top 3 levels of the UDC hierarchy. Arguably, this can be interpreted as a good definition of an area of expertise without too much specificity. As observed from the results, this approach gave the best improvement over the NDCG@k scores, but lowered the MRR and MAP scores. We attribute this to the best recommendation being ranked lower or not present in the recommendation list due to specificity boosting and cut-off post-processing steps.

There is no single hybrid approach variant with the best values for all of the metrics. We attribute this to the fact that the metrics treat the recommendation lists differently. MRR focuses on the first relevant item in the recommendation list, but does not evaluate the rest of the recommendations. MAP gives high focus to the top ranked items in the recommendation list, but does not differentiate between the top ranked items as some might still be more relevant than others, despite the ranking order. Finally, NDCG@k also gives high focus to the top ranked items, but uses their graded relevance to obtain the score.

Overall, variants "casc-hyb-full-A" and "casc-hyb-full-B" seem to perform the best in the sense of a balanced trade-off between the metric scores. Both of these variants do not use cut-off, which indicates that some relevant UDC codes are being removed from the recommendations. It is therefore arguably worth reconsidering using cut-off, however further experiments with different approaches to cut-off could provide more insight.

VII. CONCLUSION

In this paper we have presented a hybrid approach to recommending UDC codes during the document cataloguing process. We have described the used text corpus and text preprocessing techniques. The cascade hybrid steps, as well as post-processing steps were explained in detail. We performed an evaluation of a baseline BM25 recommender and different hybrid recommender system configurations using the HR@k, NDCG@k, MRR and MAP metrics. We found that the cascade hybrid approaches improved on the metric scores of a baseline BM25 recommender.

Our hybrid approach can be modified and tuned in different ways and at different points in the workflow. One such modification would be the use of a fully licensed version of UDC codes, as it would enable us to parse the already catalogued documents with better domain specificity. This would be especially useful at deeper levels of the UDC hierarchy. Another improvement could be a fine-tuning optimization step [51], [52] that would find the optimal values for the BM25 parameters k_1 and b. This would be a periodic process that would be performed after a certain number of documents have been added to the corpus. Furthermore, a similar hybrid recommender system architecture could be implemented, where domain-specific classifiers would be paired with a top-level classifier in cascade. The role of the top-level classifier would be to choose which domain-specific classifiers are to be used in the cascade step. This approach would require several domain-specific classifiers to be trained. Finally, in the future, we wish to experiment with inclusion of collaborative filtering methods and other deep neural network architectures, as the latter have recently advanced in the fields of text mining and natural language processing.

REFERENCES

[1] A. Slavic, "UDC implementation: From library shelves to a structured indexing language," *Int. Cataloguing Bibliographic Control*, vol. 33, no. 3, pp. 60–65, Jul./Sep. 2004. [Online]. Available: https://archive.ifla.org/ubcim/admin/content.htm

[2] J. Wang, "An extensive study on automated Dewey decimal classification," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 60, no. 11, pp. 2269–2286, Nov. 2009.

[3] C. J. Godby and J. Stuler, "The library of congress classification as a knowledge base for automatic subject categorization," in *Subject Retrieval in a Networked Environment*. Berlin, Germany: K. G. Saur, 2003, pp. 163–169. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/9783110964912.163/html

[4] E. Frank and G. W. Paynter, "Predicting library of congress classifications from library of congress subject headings," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 55, no. 3, pp. 214–227, Feb. 2004.

[5] P. Melville and V. Sindhwani, "Recommender systems," in *Encyclopedia of Machine Learning and Data Mining*. Boston, MA, USA: Springer, 2017, pp. 1056–1066.

[6] P. Lops, M. D. Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 73–105.

[7] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Model. User-Adapted Interact.*, vol. 12, no. 4, pp. 331–370, Nov. 2002.

[8] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Inform. J.*, vol. 16, no. 3, pp. 261–273, 2015.

[9] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intell. Data Anal.*, vol. 21, no. 6, pp. 1487–1524, Nov. 2017.

[10] L. M. D. Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks," *Int. J. Approx. Reasoning*, vol. 51, no. 7, pp. 785–799, Sep. 2010.

[11] D. Billsus, M. J. Pazzani, and J. Chen, "A learning agent for wireless news access," in *Proc. 5th Int. Conf. Intell. User Interfaces*. New York, NY, USA: Association for Computing Machinery, 2000, pp. 33–36.

[12] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf. Sci.*, vol. 180, no. 22, pp. 4290–4311, 2010.

[13] P. Bedi, P. Vashisth, P. Khurana, and Preeti, "Modeling user preferences in a hybrid recommender system using type-2 fuzzy sets," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2013, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/author/37086744975 and https://ieeexplore.ieee.org/author/37086744975

[14] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proc. 5th ACM Conf. Digit. Libraries*. New York, NY, USA: Association for Computing Machinery, 2000, pp. 195–204.

[15] A. S. Lampropoulos, P. S. Lampropoulou, and G. A. Tsihrintzis, "A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis," *Multimedia Tools Appl.*, vol. 59, no. 1, pp. 241–258, Jul. 2012.

[16] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 2, pp. 435–447, Feb. 2008.

[17] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific paper recommendation: A survey," *IEEE Access*, vol. 7, pp. 9324–9339, 2019.

[18] I. Liao, W. Hsu, M. Cheng, and L. Chen, "A library recommender system based on a personal ontology model and collaborative filtering technique for English collections," *Electron. Library*, vol. 28, no. 3, pp. 386–400, Jun. 2010.

[19] J. Beel, A. Aizawa, C. Breitinger, and B. Gipp, "Mr. DLib: Recommendations-as-a-service (RaaS) for academia," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries (JCDL)*, Jun. 2017, pp. 1–2.

[20] C. Porcel, J. M. Moreno, and E. Herrera-Viedma, "A multi-disciplinar recommender system to advice research resources in university digital libraries," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12520–12528, Dec. 2009.

[21] S. Vargas, M. Hristakeva, and K. Jack, "Mendeley: Recommendations for researchers," in *Proc. 10th ACM Conf. Recommender Syst.*, Boston, MA, USA, Sep. 2016, p. 365.

[22] M. Ojsteršek, J. Brezovnik, M. Kotar, M. Ferme, G. Hrovat, A. Bregant, and M. Borovič, "Establishing of a Slovenian open access infrastructure: A technical point of view," *Program*, vol. 48, no. 4, pp. 394–412, Aug. 2014.

[23] K. Yi, "Automated text classification using library classification schemes: Trends, issues, and challenges," in *International Cataloguing and Bibliographic Control*, vol. 36. London, U.K.: International Federation of Library Associations, 2007, pp. 78–82. [Online]. Available: https://pascal-francis.inist.fr/vibad/index.php?action=getRecordDetail&idt=20041463

[24] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *J. Adv. Inf. Technol.*, vol. 1, no. 1, pp. 4–20, 2010.

[25] K. Golub, J. Hagelbäck, and A. Ardö, "Automatic classification of Swedish metadata using dewey decimal classification: A comparison of approaches," *J. Data Inf. Sci.*, vol. 5, no. 1, pp. 18–38, Feb. 2020.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[27] J. Schrumpf, F. Weber, and T. Thelen, "A neural natural language processing system for educational resource knowledge domain classification," in *Proc. DELFI*, A. Kienle, A. Harrer, J. M. Haake, and A. Lingnau, Eds. Bonn, Germany: Gesellschaft für Informatik, 2021, pp. 283–288.

[28] O. Nevzorova and D. Almukhametov, "Towards a recommender system for the choice of UDC code for mathematical articles," in *Proc. DAMDID/RCDL*, A. Pozanenko, S. Stupnikov, B. Thalheim, E. Mendez, and N. Kiselyova, Eds., Moscow, Russia, 2021, pp. 54–62.

[29] M. Kragelj and M. K. Borštnar, "Automatic classification of older electronic texts into the universal decimal classification–UDC," *J. Documentation*, vol. 77, no. 3, pp. 755–776, Apr. 2021.

[30] M. Ojsteršek, J. Brezovnik, M. Ferme, G. Hrovat, A. Bregant, and M. Borovič. (2014). *OpenScience Slovenia Dataset*. Accessed: Aug. 15, 2021. [Online]. Available: http://www.openscience.si/OpenData.aspx

[31] M. Borovič, M. Ferme, J. Brezovnik, S. Majninger, A. Bregant, G. Hrovat, and M. Ojsteršek, "The OpenScience Slovenia metadata dataset," *Data Brief*, vol. 28, Feb. 2020, Art. no. 104942.

[32] *Multilingual Universal Decimal Classification Summary (UDCC Publication No. 088)*, UDC Consortium (UDCC), The Hague, The Netherlands, 2012.

[33] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009.

[34] A. Trotman, A. Puurula, and B. Burgess, "Improvements to BM25 and language models examined," in *Proc. Australas. Document Comput. Symp. (ADCS)*, New York, NY, USA, 2014, pp. 58:58–58:65.

[35] Y. Lv and C. Zhai, "Adaptive term frequency normalization for BM25," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag. (CIKM)*. New York, NY, USA, 2011, pp. 1985–1988.

[36] Y. Lv and C. Zhai, "Lower-bounding term frequency normalization," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag. (CIKM)*, New York, NY, USA, 2011, pp. 7–16.

[37] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge Univ. Press, 2008.

[38] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retr.*, vol. 13, no. 4, pp. 346–374, 2010.

[39] K. S. Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: Development and comparative experiments: Part 2," *Inf. Process. Manag.*, vol. 36, no. 6, pp. 809–840, Nov. 2000.

[40] W. E. Winkler, "String comparator metrics and enhanced decision rules in the Fellegi–Sunter model of record linkage," in *Proc. Sect. Surv. Res.*, 1990, pp. 354–359.

[41] P. Pu, L. Chen, and R. Hu, "A user-centric evaluation framework for recommender systems," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, New York, NY, USA, 2011, pp. 157–164.

[42] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 257–297.

[43] D. Monti, E. Palumbo, G. Rizzo, and M. Morisio, "Sequeval: An offline evaluation framework for sequence-based recommender systems," *Information*, vol. 10, no. 5, p. 174, May 2019.

[44] M. Bogaert, J. Lootens, D. Van den Poel, and M. Ballings, "Evaluating multi-label classifiers and recommender systems in the financial service sector," *Eur. J. Oper. Res.*, vol. 279, no. 2, pp. 620–634, Dec. 2019.

[45] C. Krauss, A. Merceron, and S. Arbanowski, "The timeliness deviation: A novel approach to evaluate educational recommender systems for closed-courses," in *Proc. 9th Int. Conf. Learn. Anal. Knowl. (LAK)*, New York, NY, USA, Mar. 2019, pp. 195–204.

[46] S. Rendle, L. Zhang, and Y. Koren, "On the difficulty of evaluating baselines: A study on recommender systems," 2019, *arXiv:1905.01395*.

[47] D. Hand and P. Christen, "A note on using the F-measure for evaluating record linkage algorithms," *Statist. Comput.*, vol. 28, no. 3, pp. 539–547, May 2018.

[48] L. Derczynski, "Complementarity, F-score, and NLP evaluation," in *Proc. 10th Int. Conf. Lang. Resour. Eval.* Portorož, Slovenia: European Language Resources Association, May 2016, pp. 261–266.

[49] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *Proc. CHI Extended Abstr. Hum. Factors Comput. Syst. (CHI EA)*. New York, NY, USA: Association for Computing Machinery, Apr. 2006, pp. 1097–1101.

[50] A. Schuth, F. Sietsma, S. Whiteson, and M. D. Rijke, "Optimizing base rankers using clicks," in *Advances in Information Retrieval*, M. de Rijke, T. Kenter, A. P. de Vries, C. Zhai, F. de Jong, K. Radinsky, and K. Hofmann, Eds. Cham, Switzerland: Springer, 2014, pp. 75–87.

[51] B. He and I. Ounis, "A study of parameter tuning for term frequency normalization," in *Proc. 12th Int. Conf. Inf. Knowl. Manag. (CIKM)*, New York, NY, USA, 2003, pp. 10–16.

[52] B. He and I. Ounis, "Term frequency normalisation tuning for BM25 and DFR models," in *Advances in Information Retrieval*, D. E. Losada and J. M. Fernández-Luna, Eds. Berlin, Germany: Springer, 2005, pp. 200–214.

MLADEN BOROVIČ received the M.Sc. degree in computer science from the University of Maribor, in 2012, where he is currently pursuing the Ph.D. degree in computer science. He is associated with the Laboratory for Heterogeneous Computing Systems, Faculty of Electrical Engineering and Computer Science, University of Maribor, where he works as a Teaching Assistant. His research interests include recommender systems, information retrieval, text mining, artificial intelligence, and natural language processing.

MILAN OJSTERŠEK received the Ph.D. degree in computer science from the University of Maribor, in 1994. He is currently associated with the Laboratory for Heterogeneous Computing Systems, Faculty of Electrical Engineering and Computer Science, University of Maribor, where he works as the Head of the Laboratory. His research interests include heterogeneous computing systems, digital libraries, semantic web, and service-oriented architecture.

DAMJAN STRNAD received the M.Sc. and Ph.D. degrees in computer science from the University of Maribor, Slovenia, in 2000 and 2006, respectively. He is currently associated with the Laboratory for Geospatial Modelling, Multimedia and Artificial Intelligence, Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include computer graphics, artificial intelligence, and optimization algorithms.