Leveraging app features to improve mobile app retrieval

Messaoud Chaa, Omar Nouali, Patrice Bellot

Leveraging app features to improve mobile app retrieval

Messaoud Chaa*

Department of Computer Science,
Faculty of Exact Sciences,
University of Bejaia,
06000 Bejaia, Algeria
and
Research Center on Scientific and Technical Information,
CERIST,
Ben Aknoun 16030, Algiers, Algeria
Email: mchaa@cerist.dz
Email: chaa.messaoud@gmail.com
*Corresponding author

Omar Nouali

Research Center on Scientific and Technical Information,
CERIST,
Ben Aknoun 16030, Algiers, Algeria
Email: onouali@cerist.dz

Patrice Bellot

Aix Marseille University,
Université de Toulon,
CNRS,
LIS,
Campus de Saint Jérôme,
Marseille, France
Email: patrice.bellot@univ-amu.fr

Abstract: The continued increase in the use of smartphones and other mobile devices has led to a substantial increase in the demand for mobile applications. With the growing availability of mobile apps, retrieving the right application from a large set has become difficult. However, the existing term-based search engines tend to retrieve relevant apps based on query terms rather than considering app features really required by users, such as functionalities, technical or user-interface characteristics. The novelty of this paper lies in extracting app features from app description and social users' reviews, extracting user-requested features and matching between them to get the feature-based score. In addition, we propose effective techniques that extract and weight features requested in the query. Finally, we combine Feature-based and term-based scores together to obtain the app

relevance score. The experimental results indicate that the proposed approach is effective and outperforms the state-of-the-art retrieval models for app retrieval.

Keywords: app retrieval; feature extraction; social information retrieval; natural language processing; NLP; feature-based score; term-based score.

Reference to this paper should be made as follows: Chaa, M., Nouali, O. and Bellot, P. (xxxx) 'Leveraging app features to improve mobile app retrieval', *Int. J. Intelligent Information and Database Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Messaoud Chaa is currently a PhD student at the Abderrahmane Mira University, Bejaia, and a research assistant at the Research Center on Scientific and Technical Information (CERIST), Algiers. He was awarded Magister degree from the Abderrahmane Mira University, in 2013. His research interests include information retrieval, machine learning and social networks.

Omar Nouali is currently the Director of Research and the Head of the Security Department at the Research Center on Scientific and Technical Information (CERIST), Algiers, Algeria. He received his PhD in Computer Science from the University of Sciences and Technology Houari Boumedience, Algiers, Algeria, in 2004. His research interests include information filtering and computer security.

Patrice Bellot is a Professor of Computer Science at the Aix-Marseille University since 2011 and a scientific representative at CNRS/INS2I in charge of Text and Data Mining. He is currently the Scientific Director of the OpenEdition Lab which works on text mining for Humanities and Social Sciences and the Head of the DIMAG Team at the Laboratoire d'Informatique et Systèmes (AMU-CNRS). He obtained his PhD in 2000 and his HDR in 2008. His research topic are artificial intelligence, information retrieval, text mining and natural language processing (sentiment analysis, content-based recommendation, question-answering) based on statistical machine learning approaches. He is a co-author of more than 100 publications, member of ACM, IEEE CS, ATALA and the Vice President of ARIA, the French Scientific Society on Information Retrieval.

1 Introduction

The use of smartphones and other mobile devices is on the rise. This has led to increase the demand for mobile applications (apps); there were about 28.7 billion downloads in the second quarter of 2019 combined Global Apple App Store and Google Play Store (Statista, 2020c). On the other side, the number of apps available for download in Google Play Store was about 2.9 million (December 2019) (Statista, 2020b) and was about 2.2 million in Apple App Store (Statista, 2020a) (January 2017). Therefore, an efficient app search system is essential.

The mobile app platforms allow the app's developers to upload their app and make a detailed description to explain the functionalities and features that the app has. They also allow online users to rate apps out of five stars. Online users can also evaluate apps in the form of reviews; they can write about features provided by these apps or point out the difficulties and problems related to use or installation. This important amount of information has drawn the interest of researchers toward mobile applications.

Mobile applications have drawn attention of researchers in information processing fields in order to analyse the description (information provided by developers) and user reviews (social information provided by online users) to extract useful information like features of applications. Kang et al. (1990) defines the feature as the prominent or distinctive visible characteristic or quality of a product. Following Guzman and Maalej (2014), and after looking at app description and reviews, we noticed that app features can be any description of specific app functionality visible to the user (e.g., 'play music', 'send message'), a specific characteristic related to the user interface (e.g., 'large font'), a general quality of the app (e.g., 'real-time'), as well as specific technical characteristics (e.g., 'GPS tracker').

The information extracted from description and reviews can be used to help users to find out the features and advantages of applications or help app developers to update their applications based on the opinion of users with regard to some features. Among these works, the work of Fu et al. (2013) proposes a system that can analyse user ratings and comments in mobile app in order to identify reasons why users like or dislike a given app. At first, the authors applied a linear regression to model the relationship between review text and rating, then latent Dirichlet allocation (LDA) (Blei et al., 2003) was used to discover topics that correspond to the root causes of people's concerns toward apps. Guzman and Maalej (2014) used natural language processing (NLP) techniques to identify fine-grained app features in the reviews. Then they extract the user sentiments about the identified features and give them a general score across all reviews. Mobile app review analyser (MARA) (Iacob and Harrison, 2013) is an automatic retrieval of mobile application features requested by users in their comments. A set of language rules has been defined to facilitate the identification of sentences referring to such requests. Vu et al. (2015) proposed a keyword-based framework (MARK) for semi-automated review analysis. MARK allows an analyst describing his interests in one or several mobile apps by a set of keywords. As a response MARK returns the most relevant reviews to each given keyword. Most recently, Li et al. (2017) adopted NLP methods to identify comparative reviews of similar apps to be used to provide fine-grained app comparisons based on different aspects like performance, stability and usability.

Some research works have paid attention to mobile app retrieval (Jiang et al., 2013; Park et al., 2015; Krishna et al., 2019). We will give more details of these works in Section 3. Their main objective is to retrieve apps that best match a query user using description and reviews as a representation of apps. They rely on term-based search engine that apply term matching technique to retrieve relevant apps based on the keywords issued by user. However, they lack the consideration of app features which are really required by users. To the best of our knowledge, no research has extracted features from reviews and description and used them in a retrieval system.

Actually, when a user looks for a particular app, she certainly has in her mind, a list of features that this app must provide. To seek this app in a search engine, she expresses her needs through a set of keywords. For instance, the owner of the query 'simple

notepad large font' requests apps that provide the two features 'simple notepad' and 'large font'. Hence, the retrieval system must return a list of applications that provide these two features. Thus, we are interested to develop a Feature-based app retrieval system that finds applications that provide features requested by user in her query. Our work is the first to consider the app features requested by users, in their queries, in a search system for mobile apps. To achieve this goal, we propose to extract app features from description and reviews, extract features requested by user from her query and develop a retrieval model that matches between query and apps. The main covered research questions are:

- How can we extract app features from the description and the user reviews?
- How can we automatically formulate the user query from a list of keywords to a list of requested features?
- Can the combination of feature-based and term-based models improves the retrieval performance?

The rest of this paper is organised as follows: in Section 2, we present some related works, Sections 3 and 4 devoted respectively to app retrieval and feature extraction. In Section 5 we present our proposed approach, by respectively describing the intuition behind it and explaining terms and features indexing and app retrieving. Section 6 is dedicated to the experimental results and Section 7 provides a brief conclusion and future works.

2 Related works

In this section, we give a brief literature survey of works in the domain of social information retrieval, feature-based recommendation and entity ranking that are very closely related to our work.

Social information retrieval systems (Bouadjenek et al., 2016) are related to our approach since they exploit social information generated by users in online platforms, and we leverage social users' reviews to help users find apps. Several social information like tags, reviews and annotations are used to enhance the retrieval process and improove the ranking results. Zhang et al. (2009) and Chen and Zhang (2009) proposed to add social information to the content of document and reported that adding such information enhances the search quality. Social book search (SBS) (Kumar and Pamula, 2018) is another research domain in which books are represented by a controlled meatadata (isbn, title, publishers, etc.) and enriched with social data (tags from LibraryThing and user reviews from Amazon). The objective of SBS is to help users to search and find relevant books to their requests by exploiting social metadata.

Feature-based recommender systems (Xu et al., 2018; Raja and Pushpa, 2019) that use app features for recommendation are also closely related to our work. These works extract features from apps that the target users have previously installed. Afterwards, the extracted features are used to match features of new applications and generating recommendations. Xu et al. (2018) propose a functionality-based mobile app recommendation architecture. App functional aspects are extracted by applying the part of speech (PoS) tagging method. The approach predicts new functional aspects for the target user based on the functional aspects that were used by this user in order to find

out and recommend the apps that offer those new functionalities. Likewise, DIPMMAR (Raja and Pushpa, 2019) is another work that employs LDA to determine the latent user-specific preferences by extracting features from reviews on installed applications. It also extracts the features for each application and then explores both the user's preferences and application features to provide the top-K recommendation through a context-aware ranking method. These works are different from our approach because they do not need any query given by the user, whereas in our our work, the user must explicitly expresses her needs through a list of keywords.

Since our aim is to retrieve and rank apps which might be considered as entities to be ranked based on their features, entity ranking approaches (De Vries et al., 2007; Balog et al., 2011; Tonon et al., 2012) are also considered as related works as they find and rank entities such as people, organisations, locations and products with respect to a given query. The most related work is Ganesan and Zhai (2012), in which they proposed a different way of leveraging opinionated content, in order to rank interesting entities based on how well the opinions on these entities match a user's preferences on various aspects of an entity. This work differs from our work in three ways. First, the entities that will be ranked have a specific type (cars, hotels, etc.), and therefore they have the same aspects (features), while in our work, apps are not of the same type, and thus each app has a specific features that it could share with other apps. Second, queries submitted to a such system must be structured with user's preferences on different aspects of the entity and have limited support for keyword queries. In contrast, this issue does not arise in our work, since the requested features (user preferences) are automatically extracted from a list of keywords. Third, they used the opinion about each aspect of the entity to match user's preferences, whereas in our work we use only features that apps have without considering the opinions about them.

3 App retrieval

While the general information retrieval task is to retrieve and rank documents for a given query, the app retrieval focuses on apps instead of documents. The problem of apps searching and retrieval is studied for the first time by Jiang et al. (2013) who provide a semantic-based search and ranking method for apps. They propose app topic model (ATM) in order to discover the latent semantics from app descriptions. To evaluate the relevance of an app with respect to a search query, they combined the textual score of the app (relevance of the app with regard to query terms using the traditional IR technique TF-IDF), the semantic search employing semantic similarity between terms and 12 static quality scores from the three perspectives of app popularity, developer reputation and link prestige. Experiments were carried out using a set of 1,000 search queries and 1,000 apps crawled from Google Play, Appgravity and AppBrain. Park et al. (2015) is the first work that rigorously studied the app retrieval problem. For that, the authors build a dataset that contains 43,041 app description enriched by 1,385,607 user reviews from Google Play Store and 56 realistic queries generated by domain experts based on android forums. Reviews were added to represent applications because reviews and queries are written by users, so there may be less vocabulary gap between them. They performed experiments using state-of-the-art information retrieval models including BM25, query likelihood language model and proposed a topic model-based approach, AppLDA. The proposed approach used LDA on both

representations, description and user reviews and identified topics in reviews which were also mentioned in the app description. AppLDA achieved a significant improvement in retrieval performance compared to traditional information retrieval models. The same dataset has been used in Krishna et al. (2019) for mobile app retrieval and categorisation. Word embeddings (RelEmb) are learned with a neural network architecture using app description only. The learning of word embeddings is carried out based on the notion of relevance (for each word, the information contained in the top retrieved documents are utilised when the same word is used as a query to retrieval engine). Thereafter, the learned word embedding was used to expand queries by adding the top five terms closely related to the initial user query. The proposed word embeddings are effective for query expansion task and eventually improve retrieval effectiveness.

4 Feature extraction

Product features extraction from users reviews, has found wide use in recent years (Bakar et al., 2016). It can be defined as the process that identifies and extracts the set of most relevant features of the product that customers have talked about in their reviews. The objective is to help users to know the characteristics of the product before starting their purchase or help product manufacturers to improve the quality of their product. Since NLP methods have shown success in extracting information from text like named entity recognition (Barua and Niyogi, 2019), most of the works that have been performed for product features extraction have used NLP techniques for candidate feature extraction followed by the application of some statistical measures in order to keep only the relevant and discard the irrelevant ones. For more details, we refer the reader to Bakar et al. (2016) which surveys recent works in in the field feature extraction.

One of the most-cited works in this domain is that of Hu and Liu (2004), in which the authors proposed a mining system that, first conducts POS tagging on the corpus, then runs the association rule miner to find all the frequent noun terms and noun phrases as candidate set. Because not all candidates generated by association mining are useful and in order to improve the precision of the extraction system, two types of pruning are applied (compactness pruning and redundancy pruning): the first method to check features that contain at least two words and the second one to remove redundant features that contain single words. Experimental results show that the proposed approach gives good results in term of recall (80%) and precision (70%).

Only few works used the mobile app reviews to extract the features requested by user or extract the existing features of mobile apps. Guzman and Maalej (2014) proposed an approach that produces a fine-grained list of features mentioned in app reviews. The Natural Language Toolkit (NLTK) (Bird and Loper, 2004), was used for identifying and extracting the nouns, verbs, and adjectives from reviews. Afterwards, the features are extracted by applying a collocation finding algorithm provided by the same toolkit before filtering the less frequent. Xu et al. (2018) noticed that most app functionalities are in the form of single nouns, noun phrases (noun + noun), and verb-object phrases (verb + noun, e.g., get direction). Based on this observation, they used the Stanford CoreNLP toolkit (Manning et al., 2014) to perform text preprocessing [tokenisation, POS tagging (select noun, verb and adjective) and lemmatisation]. Finally, they kept only the single nouns, two-gram nouns, and two-gram verb-object that appear more

than ten times, and in less than 80% of the apps as function aspects. MARA is a prototype that has been developed by Iacob and Harrison (2013) in order to mine app reviews and retrieve the app features requested by users. MARA was designed to collect all the reviews available for a given app, mine the content of the reviews for identifying sentences or fragment of sentences that express feature requests, summarise feature requests, and finally present the results in a user-friendly manner. for feature request extraction, the authors of MARA defined a set of linguistic rules to allow the identification of sentences referring to such requests. Then, they apply LDA to identify topics among the feature request extracted. Similarly, SAFE (Johann et al., 2017) is an approach that extracts features from users' reviews and app description based on linguistic rules. The authors manually build 18 part-of-speech patterns and 5 sentence patterns that are frequently used in text referring to app features. Thereafter, these patterns are used to extract features from the app description and reviews.

Liu et al. (2017) proposed an approach to automatically mine domain knowledge from app descriptions. They have defined a set of rules by analysing and summarising the relationships between structures of sentences and features and used them to effectively extract features from app descriptions. Specifically, each sentence from a description text is transformed into a parsing tree using Stanford Parser, then the features are extracted from the tree based on the pre-defined rules, and the result is manually evaluated by domain analysts. In order to experiment and test their approach, they collected descriptions from 140 app products covering 7 main stream classes in Google Play, for a total of 2,245 sentences.

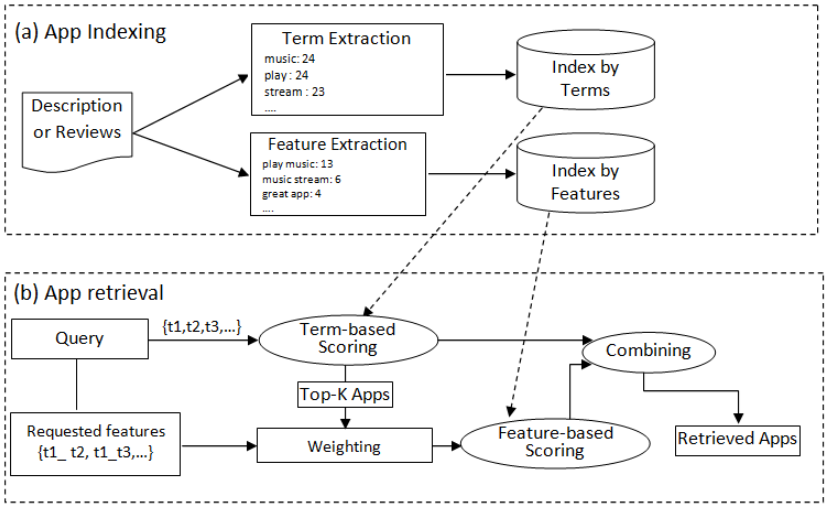
To conclude, the existing works on features extraction can be used in retrieval systems to enhance their performance. However, none of the reported works uses features to retrieve apps. These gaps will be addressed with our proposed method.

5 The proposed approach

We are interested in helping mobile app users to search and retrieve apps in response to their queries and satisfy their needs. In real life, when a user looks for an app, she issues a text query q to a search engine. The keywords of q represent the search intent of the user in terms of functionalities and features. Then, the search engine retrieves the relevant apps that satisfy the user intent, ranks them according to their relevance to q and presents them to the user as a final result to her query. The user usually prefers that the apps returned by the search engine provide all the features that she requested.

Formally, we have M apps $A = \{a_1, ..., a_M\}$. Each app a_i is represented by a description (d_i) given by the app developer and user reviews (r_i) given by online users. Given a user query $q = \{t_1, t_2, ..., t_n\}$, the purpose of the app retrieval system is to rank the apps according to their relevance to q, based on the descriptions and reviews of the apps. Since the users need apps that provide some specific features, our goal in this work is to extract app features from description and reviews to give a feature-based representation to apps, extract the features requested by the user in her query and finally match between query and apps to find relevant apps based on how well their reviews and description match the user's preferences. Thus, our work included two steps: terms and features indexing and app retrieving. Both steps are outlined in Figure 1.

Figure 1 App indexing and retrieval framework based on terms and features



Notes: The same framework will be applied for both representations (description and reviews) separately.

5.1 Terms and features indexing

As shown in Figure 1(a), each app will be indexed using two types of indexes: indexing by terms and indexing by features. In the first indexing type, the terms extracted from description and reviews are used as a basis of the index process. Thus, we first extract tokens from app description and reviews by a classical indexing approach (tokenisation, stop word removal and then stemming). Then, we index the terms with their frequencies for each app into Desc_Term_Index and Review_Term_Index, which are the two term-based indexes of descriptions and reviews, respectively.

In feature indexing, the features extracted from reviews and description are used as a basis of the index. To identify and extract features from description and reviews of applications we followed the same approach as in Guzman and Maalej (2014) study. They reached 91% precision and 73% recall for the whatsapp app. Averaged over all apps used in the experiments, the precision was approximately 60% and the recall was about 50%. The steps of our process are as follows:

- Use POS tags to identify and extract noun, verb and adjective from description and reviews since they are the most likely to be used in natural language to describe features.
- Remove stopwords to eliminate terms that are very common in the English language.
- Use PorterStemmer algorithm to stem the terms in order to give the same root for terms that are semantically equal but syntactically different.
- Use collocation functionality to extract bigram words ($< Noun >< Verb >$, $< Noun >< Noun >$ and $< Adjective >< Noun >$) that frequently occur

together within a window size, regardless of whether these terms are adjacent or not. For example, when we look at the the following three sentences, 'send the auto reply message', 'send out the automatic message' and 'it sends messages', despite the fact that the distance between 'send' and 'message' is different in the three sentences but 'send message' is considered to be a collocation and therefore considered as a feature. More details on the choice of window size will be discussed in the experimental section. We consider also that the word ordering is not important for describing features. For example, we consider that the pairs of words 'send message' and 'message send' reflect the same meaning.

This task was applied on descriptions and reviews separately. After the completion of the features extraction process, the features extracted from reviews are filtered by taking into consideration only those that appear in at least two reviews, so as to have a list of features with their frequencies for each app. The obtained result is used to construct the two indexes, Desc_Feat_Index and Review_Feat_Index, respectively, the index of descriptions based on features and the index of reviews based on features. Table 1 shows the features and their frequencies extracted from description and reviews of the application *airplay.android*, one of the most relevant apps of the query '*airplay music stream*'. From this table we can see that the frequency of features is higher in the reviews than in the description. This is the case, because several users can talk about the same feature in their reviews, whereas the app developer does not repeat the same feature several times in the description. We also notice that the features extracted from description and reviews match well with the user requested features ('AirPlay Music' and 'Music Stream'). This observation motivated us to perform a feature-based retrieval system.

Table 1 Features with their frequencies extracted from description and reviews of the app '*airplay.android*'

Representation	*Extracted features*
Reviews	Music Play (13), Google Music (9), Google Play (9), AirPlay Apple (6), Music Stream (6), Airport Express (5), App Bought (5), Give Star (4), App Great (4), App Work (4)
Description	Free Player (2), AirPlay Android (2), Music Play (2), Google Play (2), Music Video (2), Google Support (2), AirPlay Video (2), Speaker Wireless (2), Google Music (2), AirPlay Music (2)

5.2 Apps retrieval framework

The four obtained indexes will be used to retrieve apps based on terms and features extracted from description and reviews. Figure 1(b) shows an app retrieval framework used to retrieve apps that best match the query q. The same framework process will be applied to description and review representations. In order to get the score of apps based on description, we first apply a classical retrieval model using terms only. Then the top-ranked retrieved apps are used to weight the features extracted from query which are used for matching against the apps indexing features. Once term-based and feature-based scores are obtained, we combine them to get for each app the description score $S_d(q, d)$.

As for review representation, the same steps are repeated to compute for each app the review score $S_r(q, r)$. Finally, the description and review scores are combined to get the final score. Thus, the score of each app a corresponding to a query q is calculated as follows:

$$S(q, a) = \alpha S(q, d) + (1 - \alpha) S(q, r) \tag{1}$$

where α is a parameter used to determine the proportions of description score $S(q, d)$ and review score $S(q, r)$ which are calculated as follow:

$$S(q, d) = \beta S_t(q, d) + (1 - \beta) S_f(q, d) \tag{2}$$

$$S(q, r) = \gamma S_t(q, r) + (1 - \gamma) S_f(q, r) \tag{3}$$

where $S_t(q, d)$ and $S_t(q, r)$ are the two term-based scores using, respectively, description and reviews, and $S_f(q, d)$ and $S_f(q, r)$ are the two feature-based scores using, respectively, description and reviews. β and γ are two parameters to determine the proportion of term-based score and feature-based score using description and reviews respectively.

Table 2 Table of notations

Notation	*Description*
a, t, f, q	Application a, term t, feature f, query q
d	Application represented by description
r	Application represented by reviews
$\|D_t\|, \|D_f\|$	Description app lengths in terms of, respectively, terms and features
$\|R_t\|, \|R_f\|$	Reviews app lengths in terms of, respectively, terms and features
C_{td}, C_{fd}	The entire collection using description in terms of, respevtively, terms and features
C_{tr}, C_{fr}	The entire collection using reviews in terms of, respectively, terms an features
$p_{ml}(t, d), p_{ml}(f, d)$	The maximum likelihood estimate (MLE) of t and f in app description
$p_{ml}(t, r), p_{ml}(f, r)$	MLE of t and f respectively in app reviews
$p_{ml}(t, C_{td}), p_{ml}(f, C_{fd})$	MLE of t and f respectively in the description collection
$p_{ml}(t, C_{td}), p_{ml}(f, C_{fd})$	MLE of t and f respectively in the reviews collection
$\|C_{td}\|, \|C_{fd}\|$	Description collection lengths in terms of, respectively, terms and features
$\|C_{tr}\|, \|C_{fr}\|$	Reviews collection lengths in terms of respectively, terms an features
$c(t, d), c(f, d)$	Count of t and f in app description d
$c(t, r), c(f, r)$	Count of t and f in app reviews r
$c(t, C_{td}), c(f, C_{fd})$	Count of t and f in descriptions collections
$c(t, C_{tr}), c(f, C_{fr})$	Count of t and f in reviews collections

5.3 Retrieval model

The language model (Ponte and Croft, 1998; Hiemstra and Kraaij, 1998) was used in several previous works and shown to have a strong performance in ad-hoc retrieval. We therefore use this model, in this paper, to estimate the scores mentioned above. The score of a document d with respect to query q is computed as follows:

$$S(q,d) = \prod_{t \in q} p(t \mid d) \tag{4}$$

where t is a term (unigram), $p(t|d)$ is the probability of t being in d. After applying Dirichlet smoothing technique (Zhai and Lafferty, 2004), in order to avoid zero-frequency problem of the unseen terms in document, the score of document d with respect to a query q is calculated as follows:

$$S(q|d) = \sum_{t \in q \cap d} log\left[1 + \frac{|D| p_{ml}(t,d)}{\mu p_{ml}(t,C)}\right] + nlog\frac{\mu}{|D| + \mu} \tag{5}$$

where n is the number of terms in the query, $|D|$ is the length of the document d, C is the set of all documents, μ is the Dirichlet smoothing parameter, $p_{ml}(t,d) = \frac{c(t,d)}{|D|}$ and $p_{ml}(t,C) = \frac{c(t,C)}{|C|}$ are the maximum likelihood estimate (MLE) of term t in, respectively, the document and in the collection; where $c(t,d)$ and $c(t,C)$ denote the count of term t in, respectively, the document d and the collection C respectively, $|C|$ is the number of terms in the entire collection. Function (5) will be used to calculate the scores of apps. Table 2 depicts the summary of notations.

5.4 Term-based app scores

We are given a query $q = \{t_1, t_2, ..., t_n\}$ made of a set of terms and an app a represented by the terms contained in the description d and the terms contained in user reviews r. According to equation (5) the two term-based scores of a corresponding to q are calculated as follows:

$$S_t(q,d) = \sum_{t \in q \cap d} log\left[1 + \frac{|D_t| p_{ml}(t,d)}{\mu_{td} p_{ml}(t,C_{td})}\right] + nlog\frac{\mu_{td}}{|D_t| + \mu_{td}} \tag{6}$$

$$S_t(q,r) = \sum_{t \in q \cap r} log\left[1 + \frac{|R_t| p_{lm}(t,r)}{\mu_{tr} p_{ml}(t,C_{tr})}\right] + nlog\frac{\mu_{tr}}{|R_t| + \mu_{tr}} \tag{7}$$

where $p_{ml}(t,d) = \frac{c(t,d)}{|D_t|}, p_{ml}(t,r) = \frac{c(t,r)}{|D_r|}, p_{ml}(t,C_{td}) = \frac{c(t,C_{td})}{|C_{td}|}$ and $p_{ml}(t,C_{tr}) = \frac{c(t,C_{tr})}{|C_{tr}|}$. μ_{td} and μ_{tr} are the two smoothing parameters used for term-based scores of description and reviews.

5.5 Feature-based app scores

Before introducing the function that calculates feature-based scores of apps, we first explain the extraction of the requested features from the query terms. We mentioned previously, when a user looks for a particular app, he certainly has a list of features and functionalities that the app must provide. To seek this app, she issues a query in the form of terms to a search engine. For instance, the owner of the query 'simple notepad large font' requests apps that provide the two features 'simple notepad' and 'large font'. The challenge is to extract a list of requested features from a set of term given by a user so as to bring the most relevant results to the top ranks. To tackle this problem, we propose three techniques to select the most relevant features from the user query:

5.5.1 All pairs of query terms as requested features

We consider the set of all possible term pairs as the set of requested features. If we have n terms in the query, we will obtain $n(n-1)/2$ requested features. For example, if the query consists of $q = (t_1, t_2, t_3)$, we obtain the following set $F_q = \{f_1, f_2, f_3\}$ of requested features, such as $f_1 = t_1\_t_2$, $f_2 = t_1\_t_3$ and $f_3 = t_2\_t_3$, with the ordering of terms being unimportant (similarly to the indexing process). The disadvantage of this technique is that it considers that all query pairs of terms are requested features and have the same weight $(w(f_i) = 1)$, whereas if we take the query 'simple notepad large font' the term pairs like 'font simple' or 'notepad large' should not be considered as features requested by the user who made the query.

5.5.2 Features in top-k ranked apps as requested features

Pseudo relevance feedback (Rocchio, 1971; Robertson and Jones, 1976), also known as blind relevance feedback, is an important technique that can be used to improve the effectiveness of IR systems. The idea behind this technique is to assume that the initially returned top-k ranked documents are relevant and therefore their terms are used to perform a new query. Intuitively, we assume that the top-k ranked apps returned by the term-based retrieval model, as described in Subsection 5.4, are relevant to the user query, which makes it very likely that the features provided by these apps match well with the features requested by the user in his query. Consequently, the query term pairs that appear for at least one of the top-k ranked apps are considered as requested features and the others are ignored. We give a weight for each term pair as follows:

$$w(f) = \begin{cases} 1, \text{if } \exists\ a_i \in top\text{-}k,\ \ f \text{ appear in } a_i \\ 0, \text{otherwise} \end{cases} \tag{8}$$

where f is the requested feature whose weight we want to calculate and a_i is the i^{th} app in the top-k ranked list represented either by description, or by reviews.

5.5.3 Likelihood ratio to weight requested features

The preceding technique has the disadvantage of being based on the presence of the query term pairs in the top-k ranked applications, giving them the same weight $w(f) = 1$, but ignoring their importance in the description or reviews of application.

However, some of the features extracted from apps as described in Subsection 5.1 are observed due to the high frequency of the involved individual terms. This NLP problem is called *occurrence-by-chance* (Khan et al., 2012). To avoid this problem and determine whether the term pair has some real structural importance and can be considered as a requested feature, we have adopted the 'likelihood ratio' approach for hypothesis testing of independence (Dunning, 1993), which takes into account the description/reviews of the application that has been considered for calculating the frequency of the term pair as well as the frequency of the individual terms. The weight of each term pair will be calculated as follows:

$$w(f = t_1\_t_2) = \sum_{i=1}^{k} LR(f, a_i) \tag{9}$$

where k is the number of top-k ranked apps and LR is the log-likelihood ratio of term pair f in the app a_i and is calculated based on the contingency tables of observed and expected frequencies, shown in Tables 3 and 4, respectively.

$$LR(f, a) = 2 * \sum_{i,j} o_{ij} log\left(\frac{o_{ij}}{e_{ij}}\right) \tag{10}$$

Table 3 Contingency table of observed frequencies of t_2 and t_2

	t_2	*Not* t_2	
t_1	o_{11}	o_{12}	R1
Not t_1	o_{21}	o_{22}	R2
	C1	C2	N

Table 4 Contingency table of expected frequencies of t_1 and t_2

	t_2	*Not* t_2	
t_1	$e_{11} = \frac{R1C1}{N}$	$e_{12} = \frac{R1C2}{N}$	R1
Not t_1	$e_{21} = \frac{R2C1}{N}$	$e_{22} = \frac{R2C2}{N}$	R2
	C1	C2	N

Now that we have calculated the weights of each requested feature of the query, we only need to measure the two feature-based scores of each application using the two indexes Desc_Feat_Index and Review_Feat_Index. We are given a query $q = \{f_1, f_2, ..., f_n\}$ made of a set of requested features weighted using the three previous methods and an app a represented by the features contained in the description d and the features contained in user reviews r. According to equation (5), the two feature-based scores of app a corresponding to q are calculated as follows:

$$S_f(q, d) = \sum_{f \in q \cap d} log\left[1 + \frac{|D_f| p_{ml}(f, d)}{\mu_{fd} p_{ml}(t, C_{fd})}\right] + nlog\frac{\mu_{fd}}{|D_f| + \mu_{fd}} \tag{11}$$

$$S_f(q, r) = \sum_{f \in q \cap r} log\left[1 + \frac{|R_f| p_{lm}(f, r)}{\mu_{fr} p_{ml}(f, C_{tr})}\right] + nlog\frac{\mu_{fr}}{|R_f| + \mu_{fr}} \tag{12}$$

where $p_{ml}(f, C_{td}) = \frac{c(f,C_{fd})}{|C_{fd}|}$ and $p_{ml}(f, C_{fr}) = \frac{c(f,C_{fr})}{|C_{fr}|}$. μ_{fd} and μ_{fr} are the two smoothing parameters used for the feature-based scores of description and reviews.

Since each requested feature has its weight, the MLE of f in app reviews and description is calculated as follows:

$$p_{ml}(f, d) = \frac{w(f) * c(f, d)}{|D_f|} \tag{13}$$

$$p_{ml}(f, r) = \frac{w(f) * c(f, r)}{|D_r|} \tag{14}$$

where $w(f)$ is the weight of the requested feature f calculated using the above three techniques, based on app description for calculating $S_f(q, d)$ and on app reviews for calculating $S_f(q, r)$.

6 Experiments and results

In this section, we describe the details of the dataset used to perform experiments and the results obtained by our approach using terms and features to index applications.

6.1 Dataset and evaluation metric

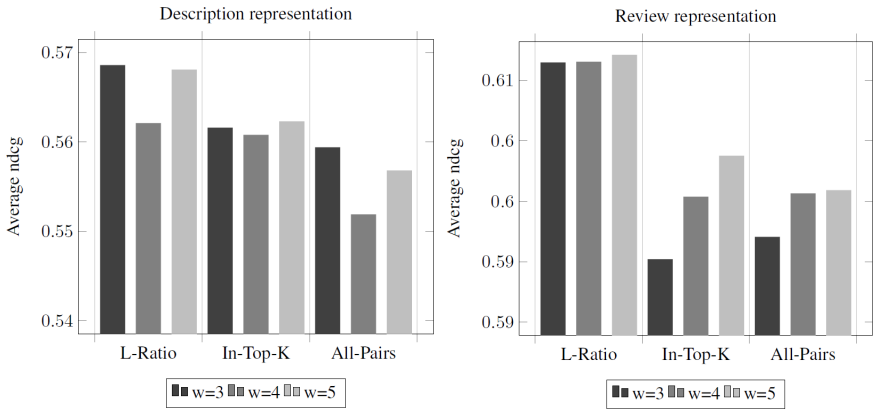
To evaluate our approach, we used the same mobile app retrieval dataset used in Park et al. (2015). This dataset contains 43,041 mobile app and 56 queries with their relevance judgments. Each app is represented by a description given by the app's developer and a set of reviews (max = 50) given by online users. As for queries, each query is a set of keywords generated by domain experts based on android forums. With regard to evaluation metrics, we also employed induced NDCG at 3, 5, 10, 20, used in Park et al. (2015), allowing us to compare the results of our approach with the ones found in their work. The induced NDCG metric ignores unjudged apps from the ranked list and keeps only the judged apps and is calculated in the same way as regular NDCG. More details on this metric can be found in Yilmaz and Aslam (2006).

6.2 Parameter setting

To get the final score of each app for a given query, by using equation (1), we first have to compute several scores, employing the four indexes, before combing them. The computations require the tuning of several parameters. We tuned the parameters using the average of the four Induced NDCG at 3, 5, 10, 20 ($avg - ndcg = \frac{1}{4}\sum_i ndcg@i$). Thereby, we started our experiments by calculating the two term-based scores based on apps descriptions and reviews. for $S_t(q, d)$ and $S_t(q, r)$ we set μ_{td} and μ_{tr} to, respectively, 1,000 and 300, since these values showed the best performance in Park et al. (2015). Once completed the computation of term-based scores, we used the same values of μ_{td} and μ_{tr} to integrate the feature-based scores with equations (2) and (3). We mention here that we tried several window sizes to extract features from description and reviews, as described in Subsection 5.1. Among the tested values (i.e., $w \in$ [3,

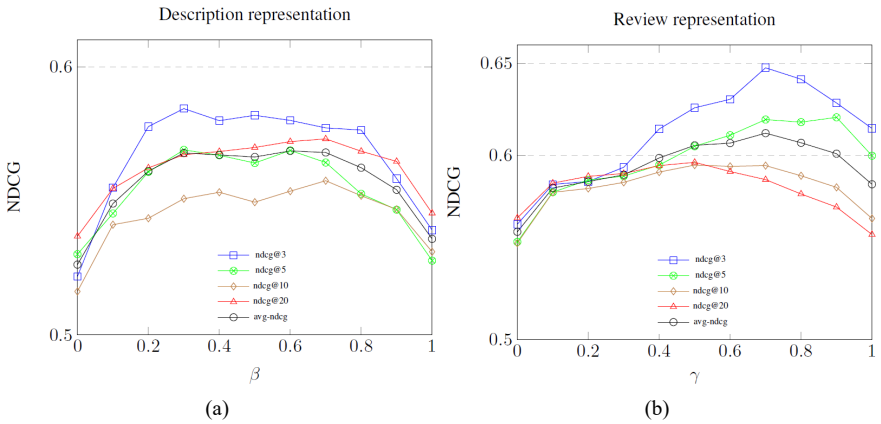
4, 5]), the best results were obtained, respectively, with $w = 3$ for description and $w = 5$ for reviews. For the number of apps used to weight the requested features, we set $k = 10$ and the best results were obtained when we applied the Likelihood ratio to weight the requested features. As for μ_{fd} and μ_{fr}, we followed (Nie et al., 2007; Petkova and Croft, 2008), and set their values to the average collection length in features on description and reviews, respectively. The values β and γ were set to 0.4 and 0.7, respectively. Finally, the two scores $S(q, d)$ and $S(q, r)$ performed to measure the scores of the app using features and terms in description and reviews were combined, and α was set to 0.4 which showed the best results.

Figure 2 Avg-NDCG measures for different values of window size (w) and different weighting methods of requested features (L-ratio, in-top-K, all-pairs) obtained from, (a) description representation (b) review representation



(a) (b)

Figure 3 NDCG measures for different values of β in, (a) description representation (b) different values of γ in review representation to determine the impact of feature-based score and term-based score on both representations (see online version for colours)



(a) (b)

Table 5 NDCG evaluation results obtained by different methods of our approach compared to related works

		ndcg@3	*ndcg@5*	*ndcg@10*	*ndcg@20*
Description	Term-only	0.522	0.530	0.516	0.537
	Terms + features	0.580^{+}	0.569^{+}	0.554^{+}	0.572^{+}
Reviews	Term-only	0.615	0.600	0.566	0.557
	Term + features	0.648	0.620	$0.595^{\bullet}$	$0.587^{\bullet}$
Description and reviews	Term-only	0.649	0.632	0.612	0.615
	Terms + features	*0.680*	*0.665**	*0.635**	0.633
AppLDA (Park et al., 2015)		0.651	0.656	0.627	*0.634*
RelEmb (Krishna et al., 2019)		0.577	0.563	0.556	/
Google Play (Park et al., 2015)		0.589	0.575	0.568	0.566

Figure 4 NDCG measures for different values of α to determine the impact of description score and review score (see online version for colours)

6.3 *Analysis*

In this section we present the results of experiments and show the impact of choosing the widow size to extract features, the impact of requested features weigthing, of combining terms and features and the impact of description and reviews on the model performance. We also compare our results against those obtained in previous works.

Table 5 shows the evaluation performance, in terms of ndcg@3, 5, 10, 20 and the average NDCG of the suggested methods compared with previous works (Park et al., 2015; Krishna et al., 2019) as well as with Google Play's app search engine. We use symbols +, • and * to mark if the improvement for integrating features is statistically significant (paired t-test with $p \leq 0.05$) in each measure over term-based models using description, reviews and both description and reviews, respectively.

As expected, Table 5 shows that when integrating features in the score function has an influence on the retrieval performance. The models that combine terms and features outperform term only model in all measures when either using description, reviews or

combining the two representations to represent apps; indicating that apps may be better represented by terms and features than terms only. The same table also shows that the model that use reviews only perform better than the model that use app descriptions only; that means that social users' reviews are a good resource for app search compared to descriptions. For example when using terms only, ndcg@3 = 0.522 for description, whereas it equals 0.615 for reviews and when combining feature and terms, ndcg@3 = 0.580 for description, whereas it equals 0.648 for reviews. We also noticed that the impact of combining features and terms for description is more significant than for reviews. For example, ndcg@3 increases from 0.522 to 0.580 (11.11%) for description, whereas it increases from 0.615 to 0.648 (5.36%) and it is not statistically significant in terms of this measure as shown in the table. In addition, Table 5 shows the good results obtained when window size is fixed to, respectively, 3 and 5 for description and reviews in the process of extracting features and using Likelihood-ratio to weight requested features.

Figure 2 shows the effects of weighting methods and the parameter w on the retrieval performance for the two representations description and reviews. We noticed that whatever the representation of the apps, the best results in terms of Avg-NDCG are obtained when likelihood-ratio (L-ratio) is present as a weighting method compared to other methods. However, the worst results are obtained when all term pairs of the query terms are used as requested features. It is also clear that the best results are obtained when $w = 3$ for description and $w = 5$ for reviews regardless of the used weighting method. This difference may be due to the difference between the language styles used by the app developer to write the description and by online users to describe their opinion in reviews. Consider as an example the app 'airplay.android' and the feature 'AirPlay Music'. When we look at the description of this app, we find that the two terms of this feature are close to each other ("Magicplay brings *AirPlay Music* and video support to Android", "*AirPlay Music* from Android to Wi-Fi speakers and receivers") but when we look at reviews, these terms are far from each other ("Latest update kinda broke *AirPlay* in Google Play *music*", "Show me the *AirPlay* code as it does when trying *music* broadcast") which explains the obtained results.

In order to understand the effects of features and terms, we further investigate performance when different values of β (descrription score) and γ (review score) are used. NDCG measures for different values of these parameters are shown in Figure 3. Here, when $\beta = 0$ or $\gamma = 0$ the models exploit terms only, whereas they exploit features only when $\beta = 1$ or $\gamma = 1$. From the figure, we notice that using features and terms together yields even better performance than using terms only or features only for both representations. We notice also that for description ($\beta = 0.4$), the effect of feature score is stronger than the term score but the opposite is true for reviews ($\gamma = 0.7$).

Figure 4 shows the performance of our model when different values of α are used. Here, when $\alpha = 0$, the model exploits description only while it exploits reviews only when $\alpha = 1$. From the figure, we notice that using description and reviews together yields even better performance than using description only or reviews only. The best results are obtained when $\alpha = 0.4$, which means that the two representations are important and each complements the other.

Making a comparison with the results of related works, AppLDA (Park et al., 2015) (exploits description and reviews), RelEmb (Krishna et al., 2019) (exploits description only) and Google Play's app search engine (unavailable information), We can see from Table 5 that, when using description and reviews together, our approach outperforms

AppLDA in terms of all NDCG measures except for ndcg@20 when the results are almost the same (0.634 for AppLDA and 0.633 for our approach. As for the case of using description only, our model outperforms RelEmb model in ndcg@3 and ndcg@5 and gives almost the same results as this approach in terms of ndcg@10.

7 Conclusions

In this paper, we conducted a study to build an app retrieval model based on terms and features extracted from the description given by the app developer and the social reviews given by online users. We started first by extracting and indexing terms and features from description and reviews using NLP techniques. Then, and in order to make the matching between the query and feature indexes, we proposed three techniques to extract and weight the requested features from the query terms. Finally, several scores using description and reviews as well as terms and features were calculated and combined to measure the relevance of each app to a given query. Evaluation results show that the models that incorporate features with terms outperform the models that use terms only regardless of whether reviews or descriptions are used, meaning that integrating features in the retrieval model effectively helps app retrieval. Using reviews only gives better results than using descriptions only, meaning that the vocabulary gap between social reviews and users queries is smaller than the gap between descriptions and users queries. Extracting features from reviews requires the window size of co-occurrence to be larger than the one used to extract features from description. This is probably due to the difference between the language styles used by the app developer to write description and by online users to describe their opinion in reviews. Evaluation results show also that statistical association measures, in our case likelihood-ratio, can be used to weight the requested features extracted from query terms.

Our proposed approach provides interesting directions for future work. Firstly, not all the features mentioned by users in their reviews are actual features provided by the application, some of them can be features requested by users because the application does not provided them. This could potentially skew the results. Therefore, it is necessary to, first use a review analyser system that can distinguish between features provided by the app and features requested by the users before incorporating them into the retrieval model. Secondly, the application of a topic modeling algorithm like LDA to group related features in the same topic could possibly improve the performance by bridging the existing gap between the language present in description and reviews and the language present in queries.

References

Bakar, N.H., Kasirun, Z.M., Salleh, N. and Jalab, H.A. (2016) 'Extracting features from online software reviews to aid requirements reuse', *Applied Soft Computing*, Vol. 49, pp.1297–1315.

Balog, K., Serdyukov, P. and de Vries, A.P. (2011) 'Overview of the TREC 2011 entity track', in *TREC*, Vol. 2011, p.11.

Barua, J. and Niyogi, R. (2019) 'Improving named entity recognition and disambiguation in news headlines', *International Journal of Intelligent Information and Database Systems*, Vol. 12, No. 4, pp.279–303.

Bird, S. and Loper, E. (2004) 'NLTK: the Natural Language Toolkit', in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, Association for Computational Linguistics, Barcelona, Spain, pp.214–217.

Blei, D.M., Ng, A.Y. and Jordan, M.I. (2003) 'Latent Dirichlet allocation', *Journal of Machine Learning Research*, January, Vol. 3, pp.993–1022.

Bouadjenek, M.R., Hacid, H. and Bouzeghoub, M. (2016) 'Social networks and information retrieval, how are they converging? A survey, a taxonomy and an analysis of social information retrieval approaches and platforms', *Information Systems*, Vol. 56, pp.1–18.

Chen, S-Y. and Zhang, Y. (2009) 'Improve web search ranking with social tagging', in *1st International Workshop on Mining Social Media*.

De Vries, A.P., Vercoustre, A-M., Thom, J.A., Craswell, N. and Lalmas, M. (2007) 'Overview of the INEX 2007 entity ranking track', in *International Workshop of the Initiative for the Evaluation of XML Retrieval*, Springer, pp.245–251.

Dunning, T. (1993) 'Accurate methods for the statistics of surprise and coincidence', *Computational Linguistics*, Vol. 19, No. 1, pp.61–74.

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J. and Sadeh, N. (2013) 'Why people hate your app: making sense of user feedback in a mobile app store', in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, pp.1276–1284.

Ganesan, K. and Zhai, C. (2012) 'Opinion-based entity ranking', *Information Retrieval*, Vol. 15, No. 2, pp.116–150.

Guzman, E. and Maalej, W. (2014) 'How do users like this feature? A fine grained sentiment analysis of app reviews', in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, IEEE, pp.153–162.

Hiemstra, D. and Kraaij, W. (1998) 'Twenty-one at TREC-7: ad-hoc and cross-language track', in *TREC*, pp.174–185.

Hu, M. and Liu, B. (2004) 'Mining opinion features in customer reviews', in *AAAI*, No. 4, pp.755–760.

Iacob, C. and Harrison, R. (2013) 'Retrieving and analyzing mobile apps feature requests from online reviews', in *Proceedings of the 10th Working Conference on Mining Software Repositories*, IEEE Press, pp.41–44.

Jiang, D., Vosecky, J., Leung, K.W-T. and Ng, W. (2013) 'Panorama: a semantic-aware application search framework', in *Proceedings of the 16th International Conference on Extending Database Technology*, ACM, New York, NY, USA, pp.371–382.

Johann, T., Stanik, C., Maalej, W. et al. (2017) 'SAFE: a simple approach for feature extraction from app descriptions and app reviews', in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, IEEE, pp.21–30.

Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E. and Peterson, A.S. (1990) *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Khan, M.A.H., Iwai, M. and Sezaki, K. (2012) 'A robust and scalable framework for detecting self-reported illness from Twitter', in *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, pp.303–308.

Krishna, S., Bajaj, A., Rungta, M., Vala, V. and Tiwari, H. (2019) 'RelEmb: a relevance-based application embedding for mobile app retrieval and categorization', *Computación y Sistemas*, Vol. 23, No. 3, pp.969–978.

Kumar, R. and Pamula, R. (2018) 'Social book search: a survey', *Artificial Intelligence Review*, Vol. 53, No. 1, pp.95–139.

Li, Y., Jia, B., Guo,Y. and Chen, X. (2017) 'Mining user reviews for mobile app comparisons', *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, Vol. 1, No. 3, pp.1–15.

Liu, Y., Liu, L., Liu, H.,Wang, X. and Yang, H. (2017) 'Mining domain knowledge from app descriptions', *Journal of Systems and Software*, Vol. 133, pp.126–144.

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S. and McClosky, D. (2014) 'The Stanford CoreNLP natural language processing toolkit', in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp.55–60.

Nie, Z., Ma, Y., Shi, S., Wen, J-R. and Ma, W-Y. (2007) 'Web object retrieval', in *Proceedings of the 16th International Conference on World Wide Web*, pp.81–90.

Park, D.H., Liu, M., Zhai, C. and Wang, H. (2015) 'Leveraging user reviews to improve accuracy for mobile app retrieval', in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, Association for Computing Machinery, New York, NY, USA, pp.533–542.

Petkova, D. and Croft, W.B. (2008) 'Hierarchical language models for expert finding in enterprise corpora', *International Journal on Artificial Intelligence Tools*, Vol. 17, No. 1, pp.5–18.

Ponte, J.M. and Croft, W.B. (1998) *A Language Modeling Approach to Information Retrieval*, PhD thesis, University of Massachusetts, Amherst.

Raja, D.K. and Pushpa, S. (2019) 'Diversifying personalized mobile multimedia application recommendations through the latent Dirichlet allocation and clustering optimization', *Multimedia Tools and Applications*, Vol. 78, No. 17, pp.24047–24066.

Robertson, S.E. and Jones, K.S. (1976) 'Relevance weighting of search terms', *Journal of the American Society for Information Science*, Vol. 27, No. 3, pp.129–146.

Rocchio, J. (1971) 'Relevance feedback in information retrieval', *The Smart Retrieval System-Experiments in Automatic Document Processing*, pp.313–323.

Statista (2020a) *Number of Available Applications in Apple App Store* [online] https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-appstore/ (accessed 2 February 2020).

Statista (2020b) *Number of Available Applications in the Google Play Store* [online] https://www.statista.com/statistics/266210/number-of-available-applications-inthe-google-play-store/ (accessed 2 February 2020).

Statista (2020c) *Number of Downloads of Applications* [online] https://www.statista.com/statistics/604343/number-of-apple-app-store-and-google-playapp-downloads-worldwide/ (accessed 2 February 2020).

Tonon, A., Demartini, G. and Cudré-Mauroux, P. (2012) 'Combining inverted indices and structured search for ad-hoc object retrieval', in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp.125–134.

Vu, P.M., Nguyen, T.T., Pham, H.V. and Nguyen, T.T. (2015) 'Mining user opinions in mobile app reviews: a keyword-based approach (t)', in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, Lincoln, NE, USA, pp.749–759.

Xu, X., Dutta, K., Datta, A. and Ge, C. (2018) 'Identifying functional aspects from user reviews for functionality-based mobile app recommendation', *Journal of the Association for Information Science and Technology*, Vol. 69, No. 2, pp.242–255.

Yilmaz, E. and Aslam, J.A. (2006) 'Estimating average precision with incomplete and imperfect judgments', in *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, ACM, pp.102–111.

Zhai, C. and Lafferty, J. (2004) 'A study of smoothing methods for language models applied to information retrieval', *ACM Transactions on Information Systems (TOIS)*, Vol. 22, No. 2, pp.179–214.

Zhang, X., Yang, L., Wu, X., Guo, H., Guo, Z., Bao, S., Yu, Y. and Su, Z. (2009) 'SDOC: exploring social wisdom for document enhancement in web mining', in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ACM, pp.395–404.