Towards Keyword-based Pull Recommendation Systems

María del Carmen Rodríguez-Hernández[1], Sergio Ilarri[1], Raquel Trillo-Lado[1] and Francesco Guerra[2]
[1]*Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain*
[2]*University of Modena and Reggio Emilia, Modena, Italy*

Keywords: Keyword-based Search, Recommendation Systems, Mobile Computing, Hidden Markov Model, Information Retrieval.

Abstract: Due to the high availability of data, users are frequently overloaded with a huge amount of alternatives when they need to choose a particular item. This has motivated an increased interest in research on recommendation systems, which filter the options and provide users with suggestions about specific elements (e.g., movies, restaurants, hotels, books, etc.) that are estimated to be potentially relevant for the user.
In this paper, we describe and evaluate two possible solutions to the problem of identification of the type of item (e.g., music, movie, book, etc.) that the user specifies in a pull-based recommendation (i.e., recommendation about certain types of items that are explicitly requested by the user). We evaluate two alternative solutions: one based on the use of the Hidden Markov Model and another one exploiting Information Retrieval techniques. Comparing both proposals experimentally, we can observe that the Hidden Markov Model performs generally better than the Information Retrieval technique in our preliminary experimental setup.

1 INTRODUCTION

Recommender systems (Jannach et al., 2010; Kantor et al., 2011; Adomavicius and Tuzhilin, 2005) suggest (relevant) items to users. The suggestions can help to solve certain decision-making problems which are presented to the users, such as which books to buy, which movies to watch, or which online news to read. They try to adapt the suggestions to each user individually, based on his/her preferences.

Existing pull-based (reactive) recommendation approaches usually assume that the type of item needed by the user is accurately determined by using some external procedure. For example, the user could select an option from a list of predefined types of items. Although this direct selection is very precise and it may be practical in some contexts, we argue that this approach lacks generality and is quite rigid for the user. For example, in a dynamic environment where new data sources could appear or disappear at any time, it may be inconvenient or difficult to have a predefined set of available types of items collected in a static list of options. Moreover, a solution based on a selection among a list of options could be tedious and uncomfortable for the user, who may be forced to use a specific vocabulary and patiently navigate menus.

Therefore, we advocate offering a keyword-based interface to allow users to freely express their needs. For the detection of the type of item requested by a user we consider the existence of a database that contains information about the different types of items available, according to the Entity-Relationship (E/R) schema shown in Figure 1. A part of the schema ("item datasets") focuses on the items available: an item may have a type and can be described through a list of features (name-value pairs). Another part of the schema ("ratings") stores information about the ratings of the items: for each combination user-context-item we may have a specific rating (if available), and each context is characterized by a set of variables. By associating context information to each rating, the E/R schema acknowledges that the context of the user has an impact on the user's perception of the usefulness of different items, as advocated by the so-called *Context-Aware Recommendation Systems* (*CARS*) (Adomavicius and Tuzhilin, 2011); a typical influential context variable is the location of the user (Levandoski et al., 2012). The double rectangles in the E/R diagram indicate weak entity types (Teorey et al., 1986) and the diamonds are oriented towards the regular entity type/s they depend on. In this paper, we focus only on the "item datasets" fragment of the E/R schema.

While intensive research has been performed in the area of keyword-based searching, the use of keyword-based systems as a support for recommen-

Rodríguez-Hernández, M., Ilarri, S., Trillo-Lado, R. and Guerra, F.
Towards Keyword-based Pull Recommendation Systems.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 207-214
ISBN: 978-989-758-187-8

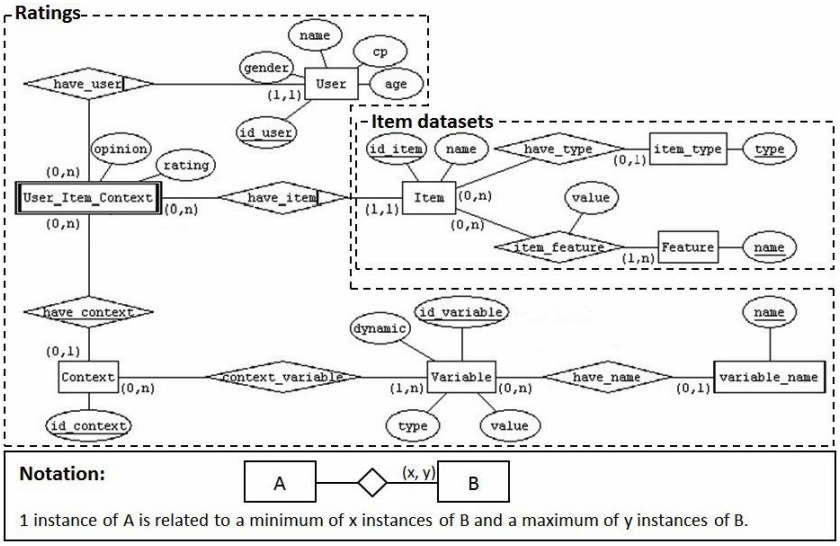

Figure 1: Entity-Relationship diagram modeling data for a pull-based context-aware recommendation system.

dation systems is rather scarce. We believe that keyword-based searching approaches for relational databases cannot be directly applied to help users define their interests for a recommendation system. The reason is that those approaches focus on a different problem; for example, techniques such as those proposed in (Bergamaschi et al., 2010; Bergamaschi et al., 2013) are specialized in queries that retrieve information from several tables at the same time (i.e., join queries), and therefore their adaptation or direct application in scenarios where simpler queries are often needed may result in inefficient solutions. In order to understand the recommendation needs of the user by using keywords, the recommendation system must be able to interpret the semantic meaning of the keywords typed by the user and identify the type of item to recommend. For this purpose, it might need to take into account semantic relations such as synonyms, similar or related keywords (e.g., restaurant – hungry, coffee shop – bar – sleepy), etc.

The current goal of this paper is to start exploring possible approaches for the identification of the type of item requested by a user in a pull-based recommendation process (del Carmen Rodríguez-Hernández and Ilarri, 2015), by using keywords in the user request. For example, if a user introduces in the system the keywords "place to eat" the system must be able to interpret that the user is searching items of the type "restaurant" (or similar types of items, like "bars", if available) without the need to choose the item type from a list previously defined in the system. In this paper, two alternative methods are considered: a solution based on the Hidden Markov Model (HMM) (Rabiner, 1989) and a solution based on the application of traditional Information Retrieval (IR) techniques (Salton and McGill, 1986).

The structure of the rest of this paper is as follows. Section 2 discusses some related work. In Section 3 and 4, we present the HMM and IR approaches, respectively. In Section 5, a set of experiments is conducted to evaluate both proposals. Finally, we conclude the paper and present some lines of future work in Section 6.

2 RELATED WORK

In the area of Information Retrieval (IR) (Salton and McGill, 1986), where generally the data are unstructured (e.g., searching relevant documents in the Web), the problem of keyword-based query answering by using an inverted index (Zobel and Moffat, 2006) has been studied. For structured data, the field of keyword-based search has started to emerge more recently (Chakrabarti et al., 2010). There are several systems that support keyword-based searching over structured data sources, such as BANKS (Aditya et al., 2002), DBXplorer (Agrawal et al., 2002), DISCOVER (Hristidis and Papakonstantinou, 2002), KEYRY (Bergamaschi et al., 2011), QUEST (Bergamaschi et al., 2013), and KEYMANTIC (Bergamaschi et al., 2010). As an example, EASE (Li et al., 2008) is a generic keyword search method which allows to index and query large collections of heterogeneous data (unstructured, semi-structured, and structured data). The authors of EASE extended the tradi-tional inverted index in order to provide keyword-based search, and additionally they proposed a novel

ranking mechanism to improve the search effectiveness.

Recommendation Systems (RS) (Jannach et al., 2010; Kantor et al., 2011; Adomavicius and Tuzhilin, 2005) have been a main focus of research, as these systems gradually reduce the existing information overload (information available on the Internet, data provided by devices/sensors of different types or other users, etc.), by recommending to the users personalized items of interest (e.g., movies, music, books, news, images, etc.) based on their preferences. However, in the field of RS, only a few works are marginally related to keyword-based searching. For example, two methods were studied for personalizing and improving the results of a social search engine (Shapira and Zabar, 2011), by using collaborative users' knowledge and integrating information from the user's social network; the proposed engine provides traditional keyword-based search functionalities. That work belongs to the field of IR, thus considering unstructured data sources, and applies recommendations to improve and customize the user experience. For movie recommendations, a hybrid system that alleviates the noise and semantic ambiguity problems present in keyword and tag representations of movies and user preferences was proposed (Stanescu et al., 2013). That proposal combines collaborative filtering and content-based recommendation techniques. As a final example, a study has been developed focused on improving the scalability and efficiency of a Big Data environment (Singam and Srinivasan, 2015). Specifically, the authors exploit keywords to indicate the preferences of users from a keyword candidate list, in order to generate appropriate recommendations based on a hybrid filtering algorithm.

As opposed to previous works, we specifically focus on the problem of correctly identifying the type of item required by a user when using a standard pull-based recommendation system. This first study in that direction represents a preliminary step forward for the development of complete and generic recommendation frameworks (del Carmen Rodríguez-Hernández and Ilarri, 2015).

3 HMM APPROACH

A Hidden Markov Model (Rabiner, 1989) can be defined as a triple A, B, π, where:

- $A = a_{ij}$ are the state transition probabilities.
- $B = [b_j(T)]$ are the observation probabilities (for each observation symbol T) at each state j.
- $\pi = [\pi_i]$ are the initial state probabilities.

There are mainly two basic problems associated to a Hidden Markov Model (Rabiner, 1989) which are relevant for the problem tackled in this paper:

1. Problem 1. Given the observation sequence $O = O_1, O_2, \ldots, O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots, q_T$ with the highest probability $P(Q|O,\lambda)$ (i.e., which best "explains" the observations)?
2. Problem 2. How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

According to the existing literature, the first problem (i.e., the problem of finding the most likely explanation for an observation sequence) can be solved efficiently using the Viterbi algorithm (Forney, 1973; Lou, 1995). To adapt that algorithm for our purposes, we have to define the following structures:

- Q is the set of states, which will be composed of the feature names (that characterize the items) and the item type.
- O is the set of observations, which will be the item types, as well the names and values of the item features.

As an example, we show in Figure 2 a fragment of the HMM proposed for a dataset InCarMusic (Baltrunas et al., 2011). The idea is the same for any other dataset.

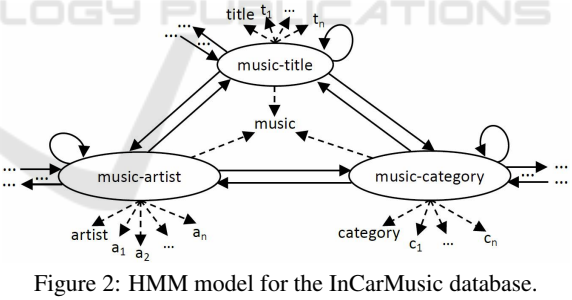

Figure 2: HMM model for the InCarMusic database.

From a reference database containing data on the domain ("item datasets" fragment in Figure 1), we extract a file "observations.txt" that contains the values and the names of the item features (e.g., title, artist and category in the case of Figure 2) and the item types (e.g., music in the case of Figure 2). Moreover, we extract a file "hmm_model.dat" with a specific structure. Considering the example of Figure 2, the structure of the file for three states (e.g., music-title, music-artist and music-category) and several observations (e.g., title, t_1, t_2, artist, a_1, a_2, a_3, category, c_1, c_2, music) is displayed in Figure 3.

In the model λ, each state contains the state transition probabilities A, the observation probabilities B,

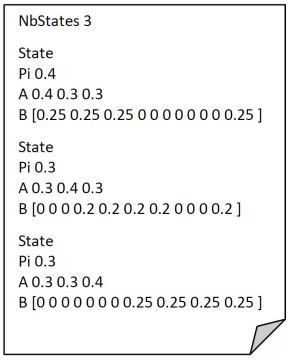

Figure 3: Example of a "hmm_model.dat" file structure.

and the initial state probabilities π. At the moment, by default, the probability values by state of the vector B are equally distributed on all the observations, dividing one by the number of terms related to the current state. Similarly, the state transition probabilities A have the same values for all the states, obtained by dividing one by the number of states. The initial state probabilities π are determined similarly. Nevertheless, our system supports the manual modification of the otherwise-equal values: the developer of a recommendation system can provide higher weights for certain elements that he/she considers more relevant, and the weights of the remaining elements will be readjusted to ensure that the sum of all the probabilities is still equal to one.

The keyword-based pull recommendation process proposed is presented in Figure 4, which summarizes the following sequence of steps:

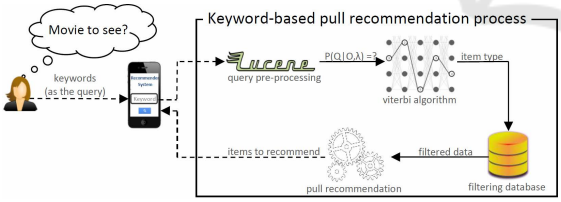

Figure 4: Keyword-based recommendations using HMM.

1. Input of the query: the user introduces the keywords as the input query in the Graphical User Interface (GUI).
2. Query pre-processing: the keywords are preprocessed by using the extension of an analyzer of Lucene, which applies the following filters:
   - Quotation tokenizer: it parses the query respecting the numbers and the double quotes.
   - Standard filter: it applies a standard tokenizer that parses the query into different types based on a grammar; for example, it splits words at punctuation characters, it removes punctuation, it splits words at hyphens (unless there is a number in the token), and it recognizes email addresses and internet host names as a single token.
   - Lower case filter: it normalizes the text of the token by converting it to lower case.
   - Stop filter: it removes stop words from the token streams, by using an input file containing stop words.
   - Snowball filter: it applies a filter that stems words using a Snowball-generated stemmer.
3. Application of the Viterbi algorithm: given the keywords as the observation sequence O and the HMM model λ, it allows determining the state sequence Q with the highest probability (e.g., music-title, music-artist, music-category, book_isbn, book_title, book_author, book_year, book_publisher, etc.).
4. Selection of the type of item: the type of item that the user needs would be determined by the highest-frequency state sequence (obtained in the previous step).
5. Filtering of the database: the database containing the different datasets is filtered by considering the type of item identified in the previous step (e.g., film, music, book, or concert). The data filtered will be used by the pull recommendation algorithm.
6. Application of the pull recommendation algorithm: it allows obtaining items of interest as an answer to the query submitted by the user, by applying any existing recommendation algorithm desired.
7. Display of the items recommended: a list of items recommended are provided to the user.

4 INFORMATION RETRIEVAL APPROACH

A second solution to consider to solve our general problem is the use of Information Retrieval (IR) techniques (Salton and McGill, 1986). In this case, the index of the retrieval engine contains a certain number of documents, whose content is automatically obtained from the databases that store the datasets (see Figure 1). Each document is named with the item type and the feature names. For example, for the dataset InCarMusic (Baltrunas et al., 2011), the document names to index are "music_title", "music_artist", and "music_category". The content of each document is composed of the values of the features (e.g., the artist

names, the music categories, and the music titles), the item type (e.g., music), and the names of the features (e.g., title, artist, and category). The structure of the documents to index is displayed in Figure 5.

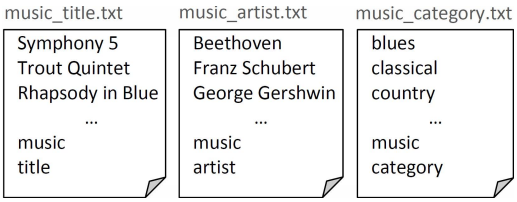

Figure 5: Example of the structure of the documents to index with the IR approach.

In general, the keyword-based pull recommendation process performs the following steps (see Figure 6):

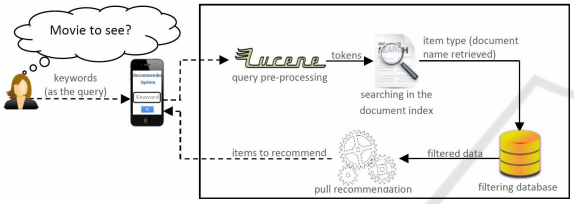

Figure 6: Keyword-based recommendations using IR.

1. Input of the query: the user introduces the keywords as the input query in the Graphical User Interface (GUI).
2. Query pre-processing: the keywords are preprocessed by using the same procedure described in Section 3.
3. Application of the Information Retrieval algorithm: given the input keywords, the system searches in the index the k documents that are most relevant to the query.
4. Selection of the type of item: the item type that the user needs would be the item type corresponding to the most relevant document (of the ranked list obtained).
5. Filtering of the database: the database containing the different datasets is filtered by considering the type of item identified in the previous step (e.g., film, music, book, or concert). The data filtered will be used by the pull recommendation algorithm.
6. Application of the pull recommendation algorithm: it allows obtaining items of interest as an answer to the query submitted by the user, by applying any existing recommendation algorithm desired.
7. Display of the items recommended: a list of items recommended are provided to the user.

5 EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation that we have performed to evaluate the two methods proposed. In Section 5.1, we describe the datasets that we use for the evaluation. In Section 5.2, we present the keyword-based queries that are evaluated. Then, we present the experimental settings and the evaluation results.

5.1 Datasets

We consider the following six datasets: LDOS-CoMoDa (Košir et al., 2011), InCarMusic (Baltrunas et al., 2011), Book-crossing (Ziegler et al., 2005), ConcertTweets (Adamopoulos and Tuzhilin, 2014), RCdata (Vargas-Govea et al., 2011) and Frappe (Baltrunas et al., 2015). In Table 1, some statistics related to the items of these datasets are described.

Table 1: Basic statistics of the datasets.

	LDOS-CoMoDa	InCarMusic	Book-crossing	ConcertTweets	RCdata	Frappe
Number of items	2513	139	271084	50971	130	4082
Number of attributes	8	7	7	8	25	10

In order to represent the HMM model, we used the item types, the feature names, and the feature values of the six datasets considered. However, we only chose the most appropriate features. Specifically, we ignored some features (name and values) of the following datasets: InCarMusic (e.g., album, mp3url, description, and imageurl), Book-crossing (e.g., image-URL-S, image-URL-M, and image-URL-L), ConcertTweets (e.g., URL), RCdata (e.g., fax, URL, and the-geom-meter), and Frappe (e.g., icon, description, and short description). We decided to ignore these features because they do not provide useful information. To experiment with datasets of equal size, we limit the number of instances considered from each dataset to 1000.

Considering the six datasets mentioned, the HMM model λ is composed of 52 states (e.g., film_director, music_artist, book_title, concert_date, restaurant_address, application_category, etc.). These states are the combination of the item types (e.g., film, music, book, concert, restaurant, application) and the feature names (e.g., director, artist, title, date, address, category) of the six datasets.

5.2 Queries

The two methods proposed (the one based on HMM and the one based on traditional IR) were evaluated by using 45 queries (see Table 2). Notice that some queries actually correspond to item types that are not available in the datasets considered. For those

queries, the best possible output is “other”, that represents a type of item not identified. For example, queries with identifiers from 30 to 35 explicitly include elements that are not part of the contents of the datasets.

Table 2: List of queries for evaluation.

Query Id	Original query
1	films similar to “toy story”
2	a romantic movie of the year 2009
3	videos of the director “walter lang”
4	the english film titled monkeys
5	a movie of the actor “diane ladd”
6	a music of the singer giovanni
7	rock song
8	music titled “für immer”
9	song of a rock artist
10	songs like “potato head blues” by “louis armstrong”
11	books about “seabiscuit”
12	books similar to ”fast women” by the author “jennifer crusie”
13	publications with an isbn number similar to 195153448
14	documents by the publisher scholastic
15	books with title “urban etiquette” and publisher ”wildcat canyon
16	concert of the band “iron maiden”
17	musical group that will play on ”02/04/2014” in Madrid
18	concerts in the venue “twickenham stadium”
19	the band direction in the state germany
20	concerts like “cattle decapitation” in ”cellular center”
21	publications with an isbn
22	place to eat
23	lodging in Modena
24	romantic melody
25	upcoming soccer matches in Barcelona
26	a recent horror movie
27	songs of movies
28	readings about movies
29	books about singers
30	self-help documents
31	romantic movie in 1949
32	policy documents of 1930
33	festivals in the region of the Holguin
34	movies that were premiered in 1927
35	documents of the Antarctica of 1908
36	restaurant with bar and permit smoking
37	place for dinner with an ambience familiar and low price
38	places opened in the hours of “12-00-22-00” to have lunch
39	restaurants with the name “taqueria el amigo”
40	french food and with “MasterCard Eurocard” payment
41	applications of photography
42	mobile applications developed by yahoo
43	chats similars to “whatsapp messenger”
44	“sport game” with many downloads
45	an apk similar to “Angry Birds” and language es

5.3 Implementation and Hardware

For the implementation of the HMM-based method, we used the Hidden Markov Model functionalities provided by the popular library *Apache Mahout* (http://mahout.apache.org/). Similarly, for the indexing of the documents for the IR-based method, we used *Apache Lucene* 2.4.0 (https://lucene.apache.org/). As explained in Sections 4 and 4, Lucene is also used for preprocessing (of input keywords and/or documents) in both methods.

Regarding the hardware, we used a standard standalone computer with the following features: Intel Core i5-2320 processor with 3 GHz and 16 GB of RAM, running Windows 7. We evaluated the performance of the proposals, although we omit the details due to space constraints. The latency is on the order of a few milliseconds (slightly higher for the IR approach) and the average memory consumption is around 9.5 MB (HMM) or 2.85 MB (IR).

5.4 Accuracy

The first proposed solution (based on HMM) computes the most likely state sequence matching an observation sequence given an HMM model. The second proposal (based on IR) searches the keywords in the query in the index of documents and returns a ranked list of hits. According to the values obtained of precision, recall and F1 measure in Tables 3 and 4, the HMM model performs better than the IR model in the experimental setup considered.

Nevertheless, it should be noticed that the IR approach is able to retrieve a ranked list of possible item types, but the HMM approach is only able to return one. Retrieving a top-K list could be interesting, as the user could quickly correct the item type identified as the most likely one if it is not correct. Although the Viterbi algorithm allows querying the probability of the most-likely sequence of states, it is not possible to retrieve the probability of all the possible sequences of states (which would be required in order to obtain a ranking).

Table 3: Evaluation of the HMM model.

Item type	Precision	Recall	F1
film	1.0	0.75	0.86
music	1.0	0.86	0.92
book	1.0	0.64	0.78
concert	0.71	0.83	0.77
restaurant	0.83	1.0	0.91
application	1.0	1.0	1.0
other	0.25	0.67	0.36
Average	0.83	0.82	0.80

Table 4: Evaluation of the IR model.

Item type	Precision	Recall	F1
film	0.67	0.75	0.71
music	1.0	0.43	0.6
book	0.78	0.64	0.7
concert	0.83	0.83	0.83
restaurant	0.71	1.0	0.71
application	0.56	1.0	0.86
other	0.0	0.0	0.0
Average	0.65	0.66	0.63

A possible problem with the HMM model is how to determine suitable probability values when the observation vector size is very large; potentially, it could happen that very small probabilities could be rounded

to zero if the global probability values are shared among many different possible values. Besides, better methods to assign the probabilities (by default we consider a proportional distribution/sharing) could be considered. Despite these concerns, the performance results obtained in the datasets evaluated so far are quite good.

5.5 Impact of the Number of Instances

We conducted another experiment with the aim of analyzing the impact of increasing the number of instances in the datasets on the performance of the two methods analyzed. For this experiment, we specifically selected the datasets Book-crossing, Concert-Tweets, and Frappe, which contain the larger number of instances (see Table 1). For each dataset, we considered four different versions (subsets of the original datasets) with an increasing number of instances (1000, 2000, 3000, and 4000 instances, respectively).

Then, we evaluated the performance (precision, recall, and F1-measure) of both models, as shown in Figures 7, 8, and 9. As shown in the figures, increasing the number of instances in the datasets in general leads to a decrease in the performance of both methods, but it is quite moderate and not very significant beyond 2000 instances per dataset. Again, in general the HMM-based method performs better than the IR-based approach.

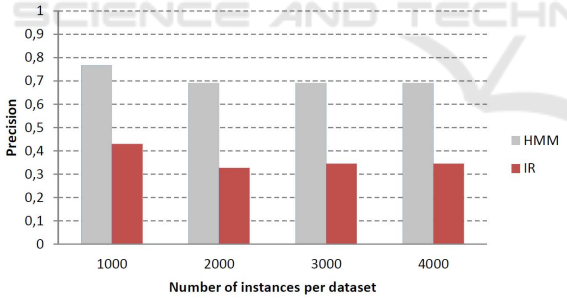

Figure 7: Average precision of both approaches.

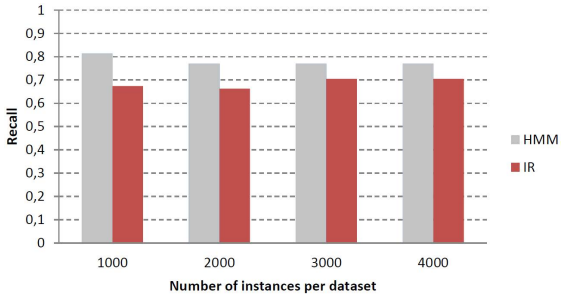

Figure 8: Average recall of both approaches.

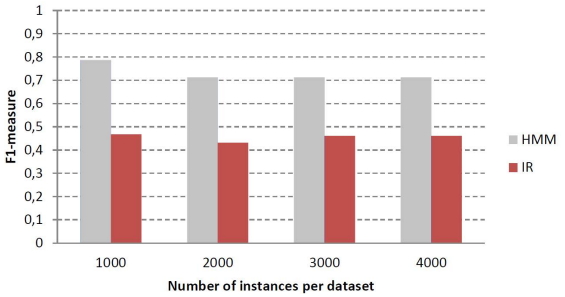

Figure 9: Average F1-measure of both approaches.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented two methods for the identification of the type of item required by a user in a pull-based recommendation process. Up to the authors' knowledge, this is the first work that focuses on the problem of explicitly applying keyword-based techniques in a recommendation system scenario.

Despite the interest of the results obtained, this study is still preliminary and we can therefore envision several avenues of improvement. For example, a thesaurus can be used to obtain synonyms of keywords provided by the user. Similarly, key-phrases could be considered rather than only keywords. Besides, more experiments should be performed with a larger number of datasets (types of items) and a larger number of items. In particular, the problem that may arise if the sizes of the datasets are very different from each other should be analyzed, as a bias in favor of larger datasets may appear in the identification of the type of item.

ACKNOWLEDGEMENTS

This work has been supported by the CICYT project TIN2013-46238-C4-4-R, DGA-FSE, and the Keystone COST Action IC1302.

REFERENCES

Adamopoulos, P. and Tuzhilin, A. (2014). Estimating the value of multi-dimensional data sets in context-based recommender systems. In *Eighth ACM Conference on Recommender Systems (RecSys)*. CEUR.

Aditya, B., Bhalotia, G., Chakrabarti, S., Hulgeri, A., Nakhe, C., Parag, P., and Sudarshan, S. (2002). BANKS: Browsing and keyword searching in relational databases. In *28th International Conference*

on Very Large Data Bases (VLDB), pages 1083–1086. VLDB Endowment.

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 217–253. Springer.

Agrawal, S., Chaudhuri, S., and Das, G. (2002). DBXplorer: A system for keyword-based search over relational databases. In *18th International Conference on Data Engineering (ICDE)*, pages 5–16. IEEE.

Baltrunas, L., Church, K., Karatzoglou, A., and Oliver, N. (2015). Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR*, abs/1505.03014.

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.-H., and Schwaiger, R. (2011). InCarMusic: Context-aware music recommendations in a car. In *EC-Web*, volume 11, pages 89–100. Springer.

Bergamaschi, S., Domnori, E., Guerra, F., Orsini, M., Lado, R. T., and Velegrakis, Y. (2010). Keymantic: Semantic keyword-based searching in data integration systems. *Proceedings of the VLDB Endowment*, 3(1–2):1637–1640.

Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R., and Velegrakis, Y. (2013). QUEST: A keyword search system for relational data based on semantic and machine learning techniques. *Proceedings of the VLDB Endowment*, 6(12):1222–1225.

Bergamaschi, S., Guerra, F., Rota, S., and Velegrakis, Y. (2011). A Hidden Markov Model approach to keyword-based search over relational databases. In Jeusfeld, M., Delcambre, L., and Ling, T.-W., editors, *Conceptual Modeling–ER 2011*, volume 6998, pages 411–420. Springer.

Chakrabarti, S., Sarawagi, S., and Sudarshan, S. (2010). Enhancing search with structure. *IEEE Data Engineering Bulletin*, 33(1):3–24.

del Carmen Rodríguez-Hernández, M. and Ilarri, S. (2015). Pull-based recommendations in mobile environments. *Computer Standards & Interfaces*.

Forney, J. G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Hristidis, V. and Papakonstantinou, Y. (2002). DISCOVER: Keyword search in relational databases. In *28th International Conference on Very Large Data Bases (VLDB)*, pages 670–681. VLDB Endowment.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, first edition.

Kantor, P. B., Rokach, L., Ricci, F., and Shapira, B. (2011). *Recommender Systems Handbook*. Springer, New York, USA.

Košir, A., Odic, A., Kunaver, M., Tkalcic, M., and Tasic, J. F. (2011). Database for contextual personalization. *Elektrotehniški vestnik*, 78(5):270–274.

Levandoski, J. J., Sarwat, M., Eldawy, A., and Mokbel, M. F. (2012). LARS: A location-aware recommender system. In *28th International Conference on Data Engineering (ICDE)*, pages 450–461. IEEE.

Li, G., Ooi, B. C., Feng, J., Wang, J., and Zhou, L. (2008). EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 903–914. ACM.

Lou, H.-L. (1995). Implementing the Viterbi algorithm. *IEEE Signal Processing Magazine*, 12(5):42–52.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Shapira, B. and Zabar, B. (2011). Personalized search: Integrating collaboration and social networks. *Journal of the American Society for Information Science and Technology*, 62(1):146–160.

Singam, J. A. and Srinivasan, S. (2015). Optimal keyword search for recommender system in Big Data application. *ARPN Journal of Engineering and Applied Sciences (ARPN-JEAS)*, 10(7):3243–3247.

Stanescu, A., Nagar, S., and Caragea, D. (2013). A hybrid recommender system: User profiling from keywords and ratings. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 73–80. IEEE.

Teorey, T. J., Yang, D., and Fry, J. P. (1986). A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2):197–222.

Vargas-Govea, B., González-Serna, G., and Ponce-Medellın, R. (2011). Effects of relevant contextual features in the performance of a restaurant recommender system. In *Fifth ACM Conference on Recommender Systems (RecSys): Third Workshop on Context-Aware Recommender Systems (CARS)*, volume 791. CEUR.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *14th International Conference on World Wide Web (WWW)*, pages 22–32. ACM.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM Computing Surveys (CSUR)*, 38(2):6.