EDUCAÇÃO CIÊNCIA E SAÚDE, v.2, n.1, Janeiro a Junho de 2015
ISSN 2358-7504 http:// periodicos.ces.ufcg.edu.br

DBFIRE: USING DATABASE QUERIES FOR INFORMATION RETRIEVAL

Vladimir Soares Catão[1], Marcus Costa Sampaio[2], Ulrich Schiel[3]

[1] Education Academic Unit, Federal University of Campina Grande, Cuité-PB, Brazil.
[2,3] Systems and Computing Unit, Federal University of Campina Grande, Campina Grande-PB, Brazil.
Email: vladimirsc@ufcg.edu.br

Resumo

Bancos de dados e documentos são tipicamente controlados por sistemas diferentes, que normalmente não se comunicam entre si: Sistemas Gerenciadores de Bancos de Dados (SGBD) e Sistemas de Recuperação de Informação (SRI), respectivamente. No entanto, é bastante provável que tais sistemas armazenem dados sobre as mesmas entidades, o que constitui um forte apelo para sua integração. Neste artigo, apresentamos uma abordagem para a integração SGBD/SRI, a qual se utiliza dos termos mais bem colocados encontrados numa consulta ao SGBD como sementes de uma busca a documentos no SRI. Estes termos "expandem" um conjunto base de palavras-chave providas pelo usuário, e são ordenados a partir de uma medida de sua difusão ao longo do resultado da consulta ao SGBD. Os experimentos mostram que a abordagem apresenta resultados significativos quando comparada a outros trabalhos.

Palavras-chave: integração da informação, integração SGBD/SRI, expansão de buscas, métodos de ordenação de termos**.**

Abstract

Databases and documents are normally controlled by different systems, which usually do not communicate with each other: Database Management Systems (DBMS) and Information Retrieval Systems (IRS), respectively. Still, it is very likely that they store information about the same entities, which is a strong appeal for their integration. In this paper, we present an approach for DBMS/IRS integration that uses top-ranked terms in a DBMS query as keywords for an IRS search, in order to retrieve documents somehow related to the query. These terms "expand" an initial keyword set provided by the user, and are ranked according to a measure of their diffusion over the database query result. Experiments show that the approach presents significant improvements compared to other integration methods.

Keywords: Information integration, DBMS/IRS integration, query expansion, term ranking methods.

1 Introduction

Databases and documents are the main information storage in most organizations. Examples of structured data, databases follow strict data

organization rules, queried by formal languages, such as SQL. On the other hand, documents are usually written in free text, with no explicit structure, and are retrieved by keyword lists. The structured queries in the database world lead to exact results, while the keyword form in the documents world causes inexact results, frequently retrieving much irrelevant information.

Databases and documents are also controlled by different systems: Database Management Systems (DBMS) and Information Retrieval Systems (IRS), respectively. The different natures of the data managed by these systems makes them usually isolated inside organizations. Nonetheless, integration opportunities are always possible (CHAUDHURI, 2005; WEIKUM, 2007).

We can cite a number of approaches. One of them is the possibility of accessing both systems transparently, in an environment called the “dataspace” (HALEVY, 2006). We can also point proposals for structured querying unstructured information (CAFARELLA, 2007), or for keyword searches over databases (YU, 2010). XML retrieval (AMER-YAHIA, 2006) is also a research field where we can see both DB and IR techniques together.

Unlike these approaches, we view integration in a different way. It is very probable that, in a given organization, both systems (DBMS and IRS) store different information about the same entities. Thus, one possible consequence is to use a database query in order to retrieve documents somehow related to the information need expressed in that query. Some examples: for a query regarding last month’s clients, we could retrieve sales proposals or presentation slides prepared for those clients; for a query returning the list of products made on company’s division X, retrieve user complaints regarding those products.

In order to accomplish this, we present DBFIRe (**D**ata**B**ases for **I**nformation **Re**trieval), a method that uses query expansion concepts usually used in IR (CARPINETO, 2012) to find documents associated to databases queries.

Thus, given a database query and a list of keywords provided by the user, DBFIRe expands these keywords with top-ranked terms selected from the database query result[1]. Once database queries are assumed exact, our term ranking method uses a measure of term spread of the query result: the more

[1] Convention: terms relate to the DBMS world; keywords concern the IRS world.

spread, the better. This measure is based on two probabilities: the probability that the term occurs over the whole term stream of the query result, and the probability that it is found in a single tuple element of the query result. Consider tuple elements as the intersection between rows and columns of the query tuples.

Two other approaches (LIU, 2006; ROY, 2005) also use DBMS queries to feed IRS searches. In (ROY, 2005), entire tuple elements are ranked, instead of individual terms as in DBFIRe; in (LIU, 2006) the keywords are extracted from the query body, without using the query result.

We can also cite different approaches that could be compared with DBFIRe, concerning our term ranking method. For example, in (AMATI, 2002; CARPINETO, 2001) terms are ranked according to differences in term distribution in the expansion corpus and the whole document collection, while in (LAVRENKO, 2001) terms are ranked due to their co-occurrence with user keywords over the expansion corpus, based on relevance models.

With that in mind, we designed our experiments to investigate the following hypotheses:

- Could a direct keyword search with the original user keywords (that is, without expansion), deliver similar results as those using DBFIRe?
- What is the behavior of DBFIRe comparing with other integration approaches (LIU, 2006; ROY, 2005)?
- And what is the behavior of DBFIRe considering other term ranking methods (AMATI, 2002; CARPINETO, 2001; LAVRENKO, 2001)?

Testing these hypotheses in two different domains, we found that DBFIRe performed significantly better than all other methods.

The remainder of the paper focuses on detailing our method, as well as describing the evaluation performed. We conclude with some suggestions of further improvements.

2 The Method

The basic idea of our method is to expand an initial set of user keywords, extracting terms from database query results; this will take advantage of the assumed exactness of database queries. Therefore, we suppose the query is

known in advance, and the user is able to provide a short description of the information need. The expanded keywords are ultimately sent to the IRS, retrieving documents related to the database query.

Another characteristic of DBFIRe is to focus exclusively on the query result for measures of term's usefulness. Other methods (AMATI, 2002; CARPINETO, 2001) depend on some heuristic involving term distributions over the whole document collection, even those based on relevance models, due to the "zero-probability" problem (LAVRENKO, 2006).

Thus, for DBFIRe the more a term is spread over the query result, the better. We estimate such spread based on two term probabilities: the probability of finding the term in the whole term stream of the query result, and the probability of finding the term in a single tuple element; we consider tuple elements as the intersections of rows and columns of the database query result.

Therefore, a useful expansion term will probably present an overall high frequency over the query result, and occur in multiple tuple elements (that is, in different rows and/or columns).

We estimate the first probability (called stream probability – P_s) through Equation 1, where *s(t)* is the raw frequency of term *t* over the whole term stream *s*, and *#s* is the size in terms of the stream.

$$P_s(t) = \frac{s(t)}{\#s} \quad \textbf{(1)}$$

Similarly, the second probability (called tuple element probability – P_e) is detailed in Equation 2. We assume *e(t)* as the number of tuple elements in which we find term *t* and *#e* as the total number of tuple elements in the query result. To arrive at *s(t)* we consider only the presence of the term in a tuple element, no matter how many occurrences it may present.

$$P_e(t) = \frac{e(t)}{\#e} \quad \textbf{(2)}$$

The joint measure of these probabilities shown in Equation 3 gives a score for estimating term diffusion over the expansion corpus. Thus, terms should be ranked in descending order of their scores.

$$score(t) = P_s(t)P_e(t) \quad \textbf{(3)}$$

Similar to other expansion methods, we can control the number of tuples that should be analyzed, as well as the number of terms to be used for expansion. These are the method's parameters *k* and *n*, respectively.

Lastly, when selecting the *n* best-scored terms, common stopwords (FOX, 1992) are discarded.

2.1 Term Weighting

Expansion methods can be easily affected by the "query drift" problem (MITRA, 1998): expansion terms may drift the IRS, which can return more documents related to the added terms than to the original user keywords. To avoid this, expansion terms should be given lower weights than the user keywords, but keeping the relative differences of their scores *s(t)*.

Thus, inspired by the Rocchio's beta (ROCCHIO, 1971), the weight of each term in DBFIRe normalizes its score *s(t)* with respect to the maximum value of *s*, but limited to a user defined parameter (called *β*); *β* should be set in the interval (0, 1). In DBFIRe, we weight terms as in Equation 4:

$$weight(t) = \beta \frac{score(t)}{\max(score)} \quad \textbf{(4)}$$

The value of *β* adjusts the effect of expansion: for higher values, we give more importance to top-ranked terms; lower values of *β* assign more importance to the user keywords. As a slight difference from the original Rocchio's beta, all user keywords receive the maximum weight (that is, 1.0).

The value of *β* that maximizes the quality of document retrieval will probably be different for different domains, or even from query to query. However, we believe a good general value is *β*=0.5, just in the middle of the whole interval (0, 1). As we will see in section 3, this configuration did show a very reasonable performance on average.

2.2 An Example

One of the domains we tested DBFIRe uses the online movie database available at IMDB[2] (**I**nternet **M**ovie **D**ata**B**ase). Consider a query in such domain, for example, movies directed by Francis Ford Coppola; relevant documents related

[2] http://imdb.com

to this query could be reviews or general comments about the movies on the query result. We show some tuples of the query result in Table 1, highlighting non-stopword terms with more than one occurrence.

Table 1: Fragment of query "Francis Ford Coppola movies"

Title	Plot
apocalypse now (1979)	**vietnam**, 1969. captain willard must find and kill renegade colonel kurtz…
gardens of stone (1987)	a sergeant wants to save the lives of young soldiers being sent to **vietnam**...
the godfather (1972)	vito **corleone**, head of the **corleone** mafia family, sees the clash of his old world values …

Regarding the two highlighted terms, Table 2 shows their stream probability, element probability and the corresponding scores. Consider the stream size $\#s$=52 and the number of tuple elements $\#e$=6.

The definite keywords sent to the IRS are shown as follows, along with their weights according to Equation 4. In this case, we set k=3, n=2 and β=0.5; "Francis Ford Coppola movies" were considered the original user keywords.

1.0 francis 1.0 ford 1.0 coppola 1.0 movies 0.5 vietnam 0.25 corleone

In this very simple example, it is possible to see the main focus of DBFIRe, giving more importance to terms that occur in more than one row or column of the query result. For example, "corleone" has the same stream probability than "vietnam", but receives a higher score due to a higher element probability. The experiments in the next section show this has an important impact in the overall retrieval quality.

Table 2: Scores for highlighted terms in Table 1

t	$P_s(t)$	$P_e(t)$	$score(t)$
corleone	2/52	1/6	0.06
vietnam	2/52	2/6	0.12

3 Experiments

Our method should be judged with respect to the relevance of the documents returned. Therefore, we can follow the same principles applied to the evaluation of an IRS, which is usually based on test collections (SANDERSON, 2010): a set

of documents, a set of topics, and pairs of relevance assessments, indicating the documents that are relevant for each topic.

We used two test collections created under INEX (INitiative for the Evaluation of XML retrieval) workshops (LALMAS, 2007): the 2011 Data Centric Track (WANG, 2012) and the 2013 Linked Data Track (BELLOT, 2013); in both collections, we focused on the ad-hoc search retrieval task. Indri[3] was the chosen IRS, while quality indicators were MAP (Mean Average Precision) and interpolated precision-recall graphs (SANDERSON, 2010).

The tests consisted in comparing DBFIRe with a baseline (representing a keyword search without expansion), with other DBMS/IRS integration methods, and with different expansion methods, in order to evaluate our proposed term ranking approach.

The integration methods compared were SCORE - Symbiotic Context Oriented Information Retrieval (ROY, 2005) - and SEMEX - SEMantic Explorer (LIU, 2006); expansion methods were KLD - Kullback-Leibler Divergence CARPINETO, 2001), DFR - Divergence from Randomness (AMATI, 2002), and a method based on relevance models, which we called RM (LAVRENKO, 2001). Though we acknowledge that the state of the art for expansion in IR uses supervised learning (CAO, 2008), that is not easily accomplished in a typical organization, which is why we excluded these methods from the comparisons.

When comparing with expansion methods we used the general framework of DBFIRe: tuples as expansion corpus, stopword removal, and the same values for k, n and β. For all experiments we made β=0.5.

3.1 Data Centric Environment Setup

The collection is formed by XML files created upon IMDB plain files, with no explicit database. However, we built a database extracting attribute/value pairs from the XML files of the collection. They were then stored in a relational MySQL database. Figure 1 illustrates an example of a file of the collection, showing how it was stored in the database.

[3] http://www.lemurproject.org/indri.php

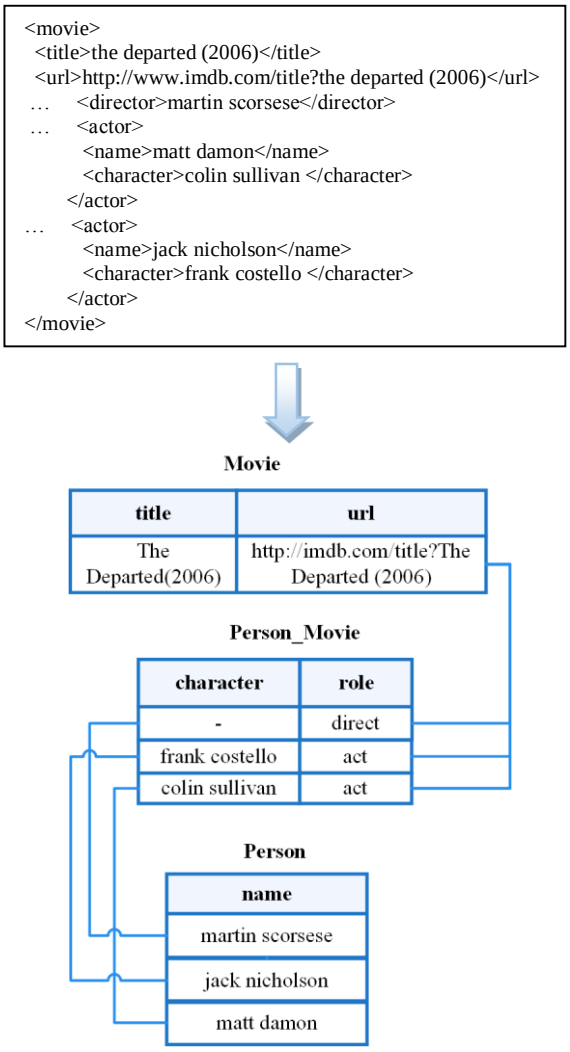

Figure 1. Loading Data Centric XML file into a database

In order to avoid a possible bias due to a database created directly from the documents, the actual document collection is composed of the original XML files plus those files in the TREC 2005 Robust Track (VOORHEES, 2005). This makes the environment a bit similar to that of a common organization, where many documents will have nothing to do with the database queries.

Once all data were loaded, we manually created SQL queries for each collection topic.

3.2 Linked Data Environment Setup

This collection is made of Wikipedia[4] articles, with structured content provided by two ontologies: YAGO (HOFFART, 2013) and DBpedia (LEHMANN, 2014). To build the queries, we used SPARQL[5] , a language similar to SQL. Again, we have an environment with plenty of documents unrelated to the databases (the ontologies do not cover all Wikipedia content).

We used the first 50 Jeopardy topics of the ad-hoc search task (BELLOT, 2013), creating SPARQL queries, that returned triples <resource, property, value>: a resource names a Wikipedia article; a property relates to one of the resource's attribute; and the last component is the property value of the resource.

Resembling a database query result, triples are converted into tuples: resources and values become tuple elements; properties name tuple columns. Figure 2 shows an example of a SPARQL query result, with its conversion into tuples.

3.3 Results

Benchmarks for Data Centric collection are shown in Table 3, and results for Linked Data collection appear in Table 4. Each table shows the absolute values of MAP and the relative differences of each method compared to DBFIRe. Different combinations of parameters k and n were used, with most differences confirmed at the 0.05 level through Wilcoxon signed rank tests (SANDERSON, 2010); only two comparisons were verified at the 0.09 level in the Linked Data tests. These cases appear shadowed in Table 4.

[4] http://en.wikipedia.org
[5] http://www.w3.org/TR/sparql11-query

SPARQL Triples

Guatemala	dbpedia.org/ontology/ capital	dbpedia.org/resource/ Guatemala_City
Guatemala	dbpedia.org/ontology/ governmentType>	dbpedia.org/resource/ Unitary_state
Guatemala	dbpedia.org/ontology/ officialLanguage	dbpedia.org/resource/ Spanish_language
El Salvador	dbpedia.org/ontology/ capital	dbpedia.org/resource/ San_Salvador
El Salvador	http://dbpedia.org/ ontology/language	dbpedia.org/resource/ Castilian

DBFIRe Tuples

Resource	Capital	Language	Government Type
Guatemala	Guatemala City	Spanish Language	Unitary State
El Salvador	San Salvador	Castilian	-

Figure 2: Converting SPARQL triples into DBFIRe tuples

Table 3: Benchmark for Data Centric test collection

k and n	*DBFIRe*	*BASELINE*	*SCORE*	*SEMEX*	*KLD*	*DFR*	*RM*
k=10, n=10	0.3518	0.3119 **12.7%**	0.1871 **88.0%**	0.2125 **65.5%**	0.3199 **9.9%**	0.3168 **11.0%**	0.3230 **8.9%**
k=20, n=10	0.3536	0.3119 **13.3%**	0.1871 **88.9%**	0.2125 **66.4%**	0.3182 **11.1%**	0.3126 **13.2%**	0.3268 **8.1%**
k=10, n=20	0.3504	0.3119 **12.3%**	0.1871 **87.2%**	0.2125 **64.8%**	0.3324 **5.3%**	0.3297 **6.2%**	0.3107 **12.7%**
k=20, n=20	0.3542	0.3119 **13.5%**	0.1871 **89.2%**	0.2125 **51.6%**	0.3334 **6.2%**	0.3275 **8.1%**	0.3166 **11.8%**

Table 4: Benchmark for Linked Data test collection

k and n	*DBFIRe*	*BASELINE*	*SCORE*	*SEMEX*	*KLD*	*DFR*	*RM*
k=10, n=10	0.3240	0.2923 **10.8%**	0.1286 **>100%**	0.1946 **66.4%**	0.3091 **4.8%**	0.3029 **6.9%**	0.3041 6.5%
k=20, n=10	0.3231	0.2923 **10.6%**	0.1286 **>100%**	0.1946 **66.0%**	0.3082 **4.8%**	0.2999 **7.7%**	0.3055 **5.7%**
k=10, n=20	0.3224	0.2923 **10.2%**	0.1286 **>100%**	0.1946 **65.6%**	0.3020 **6.7%**	0.2904 **10.9%**	0.3023 **6.6%**
k=20, n=20	0.3231	0.2923 **10.5%**	0.1286 **>100%**	0.1946 **66.0%**	0.3005 **7.5%**	0.2884 **12.0%**	0.3048 **6.0%**

As we saw very short differences among all combinations of *k* and *n*, the interpolated precision-recall graphs were built with the lowest overhead, that is, with *k*=10 and *n*=10. The graph for Data Centric collection is shown in Figure 3 and the Linked Data graph is presented in Figure 4.

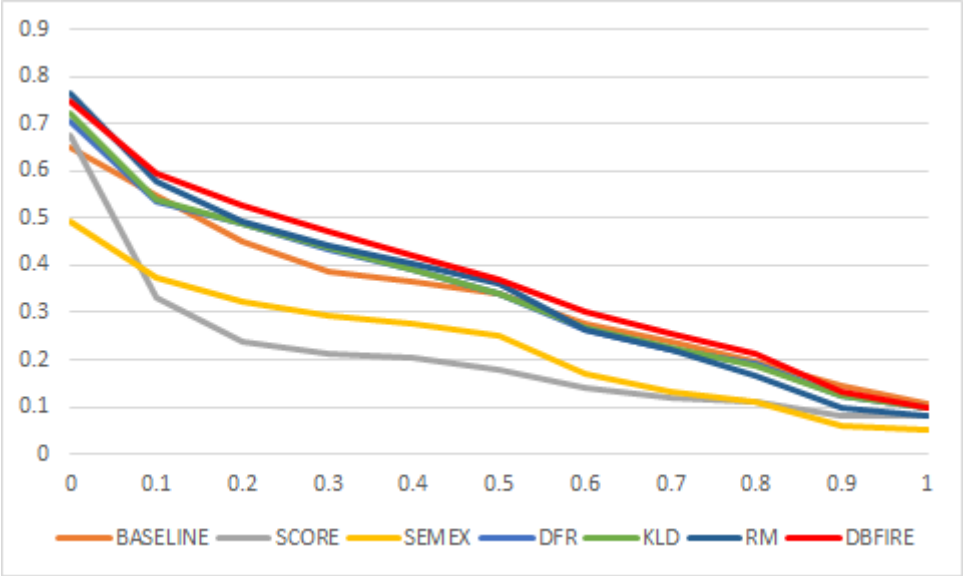

Figure 3: Interpolated precision-recall graph (Data Centric).

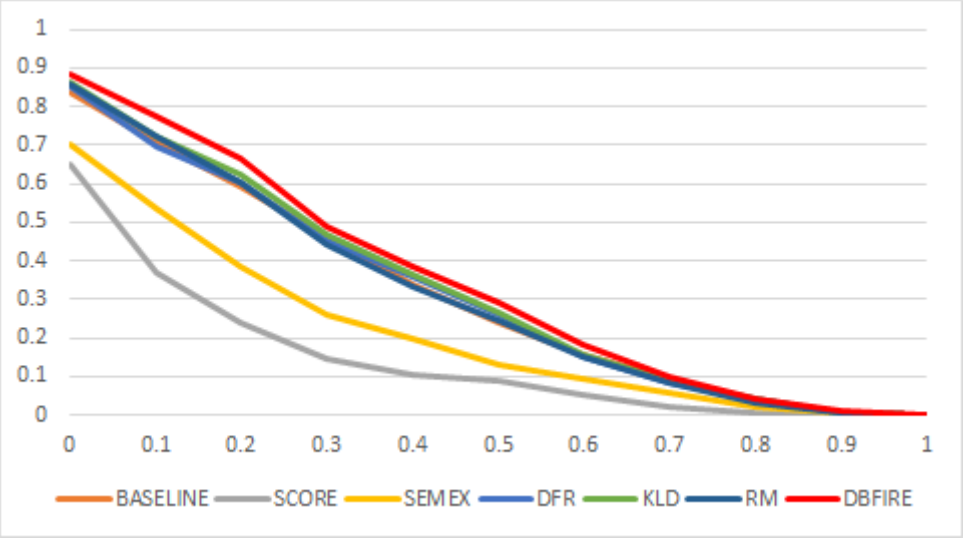

Figure 4: Interpolated precision-recall graph (Linked Data).

3.4 Discussion

DBFIRe is very effective when we consider its overall performance comparing to the baseline and with other integration methods (SCORE/SEMEX).

The approach taken by SCORE, ranking entire tuple elements, makes it difficult to separate terms that help increase the quality of document retrieval from those that do not. Moreover, the fact that it is does not use the original keywords can be also considered a negative bias. Nevertheless, even if we use SCORE to

expand the user keywords, its results are still below those of integration method SEMEX.

Regarding SEMEX, it does present an approximation of the user information need, but as we saw from its numbers, it seems much better for the user to describe it directly: it presents better results than SCORE, but does not overcome the baseline.

Given this picture, it seems wise to expand the user keywords with terms in database queries. However, would any term ranking method be equally good? The results support our claim that DBFIRe performs better.

We see significant differences pro-DBFIRe in almost all comparisons with traditional methods KLD, DFR and RM; only two of them had a p-value above the usual “standard” level 0.05, but still below 0.09. Even though, a significance test with these methods and the baseline report *p*-values above 0.2 for all methods, in all configurations of *k*, *n*, and in both test collections. Therefore, we cannot say they beat the baseline, but we can truly say that about DBFIRe.

At last, the red line represented by DBFIRe in both precision-recall graphs of Figures 1 and 2, appears above all other methods in most recall levels. This is another evidence of the quality of our method.

4 Conclusions and Future Work

In this paper, we presented DBFIRe, a method for retrieving relevant documents related to database queries. It presented two innovations: expanding the user keywords with terms found in database query results, and a new term ranking method focused exclusively on term probabilities over the expansion corpus. These characteristics proved very effective when compared with other methods and in different domains. Moreover, it can be easily implemented, not only at application level but also as a built in resource at DBMS level.

However, we can still improve the method. By now, DBFIRe completely neglects the presence of the user keywords over the query result. Their presence can be used in at least two different scenarios.

First, as a way to better adjust the importance of expansion terms: the more user keywords in the query result, the more we can “trust” it, applying a higher weight to top-ranked terms.

Finally, the presence of user keywords can also rank the best tuples to be used in expansion. For a DBMS, all returned tuples are equally relevant, but they will probably have different terms, with different sizes. Thus, tuples can be ranked in terms of their expansion usefulness, according to how much they contain the original user keywords.

5 References

AMATI, Gianni et al. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. **ACM Transactions on Information Systems**, v. 20, n. 4, p. 357–389, Oct./2002.

AMER-YAHIA, Sihem et al. XML Search: Languages, INEX and Scoring. **ACM SIGMOD Record**, v. 35, n. 4, p. 16–23, Dec./2006.

CAFARELLA, Michael J. et al. *Structured Querying of Web Text: A Technical Challenge*. In: Third Biennial Conference on Innovative Data Systems Research - CIDR 2007 (Asilomar, USA, 2007). **Proceedings…** p. 225–234.

CAO, Guihong et al. *Selecting Good Expansion Terms for Pseudo-Relevance Feedback*. In: Thirty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR’08 (Singapore, 2008). **Proceedings…** p. 243–250.

CARPINETO, Claudio et al. An information-theoretic approach to automatic query expansion. **ACM Transactions on Information Systems**, v. 19, n. 1, p. 1–27, Jan./2001.

CARPINETO, Claudio et al. A Survey of Automatic Query Expansion in Information Retrieval. **ACM Computing Surveys**, v. 44, n. 1, p. 1–50, Jan./2012.

CHAUDHURI, Surajit et al. Integrating DB and IR Technologies: What is the Sound of One Hand Clapping? In: Second Biennial Conference on Innovative Data Systems Research - CIDR'05 (Asilomar, USA, 2005). **Proceedings…** p. 1–12.

FOX, Christopher. Lexical analysis and stoplists. In: **Information Retrieval: Data Structures and Algorithms**. William B. Frakes and R. Baeza-Yates, eds. Prentice Hall, 1992. p. 102–130.

BELLOT, Patrice et al. Report on INEX 2013. **SIGIR Forum**, v. 47, n. 2, p. 21-32, Dec./2013.

HALEVY, Alon et al. Principles of dataspace systems. In: Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems - PODS '06 (Chicago, USA, 2006). **Proceedings…** p.1–9.

HOFFART, Johannes et al. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. **Artificial Intelligence**, v. 194, p. 28–61, Jan./2013.

LALMAS, Mounia et al. INEX 2002 - 2006: Understanding XML Retrieval Evaluation. In: First international conference on Digital libraries: research and development-DELOS'07 (Pisa, Italy, 2007). **Proceedings…** p. 187–196.

LAVRENKO, Victor et al. Real-time Query Expansion in Relevance Models. **Technical Report**, Center for Intelligent Information Retrieval, University of Massachussets, 2006.

LAVRENKO, Victor et al. Relevance-Based Language Models. In: Twenty-fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '01 (New Orleans, USA, 2001). **Proceedings…** p. 120–127.

LEHMANN, Jens et al. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. **Semantic Web Journal**, v.6, n. 2, p. 167-195, 2015.

LIU, Jing et al. Answering Structured Queries on Unstructured Data. In: Ninth International Workshop on the Web and Databases - WebDB 2006. (Chicago, USA, 2006). **Proceeding…** p. 25–30.

MITRA, Mandar et al. Improving Automatic Query Expansion. In: Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'98 (Melbourne, Australia, 1998). **Proceedings…** p. 206 – 214.

ROCCHIO, J.J. Relevance feedback in information retrieval. In: **SMART Retrieval System - Experiments in Automatic Document Processing**. G. Salton, ed. Prentice Hall, 1971. P. 313–323.

ROY, Prasan et al. Towards Automatic Association of Relevant Unstructured Content with Structured Query Results. In: Fourteenth ACM Conference on Information and Knowledge Management - CIKM '05 (Bremen, Germany, 2005). **Proceedings**… p. 405–412.

SANDERSON, Mark. Test Collection Based Evaluation of Information Retrieval Systems. **Foundations and Trends in Information Retrieval**, v. 4, n. 4, Jun./2010, p. 247–375.

VOORHEES, Ellen. The TREC Robust Retrieval Track. **SIGIR Forum**, v. 39, n. 1, p. 11–20, Jun./2005.

WANG, Qiuyue et al. Overview of the INEX 2011 Data-Centric Track. In: **Focused Retrieval of Content and Structure – LNCS 7424**. Shlomo Geva et al., eds. Springer-Verlag. p. 118–137, 2012.

WEIKUM, Gerard. DB & IR: Both Sides Now. In: ACM SIGMOD International Conference on Management of Data-SIGMOD '07 (Beijing, China, 2007). **Proceedings…** p. 25–30.

YU, Jeffrey. et al. Keyword Search in Relational Databases: A Survey. **IEEE Data Engineering Bulletin**, v. 33, n. 1, p. 67–78, Mar./2010.