A Methodology for the Automatic Creation of Massive Continuous Query Datasets from Real-Life Corpora

Christos Tryfonopoulos
Dept. of Informatics and Telecommunications
University of the Peloponnese, Tripoli, Greece
trifon@uop.gr

Abstract — **In the information filtering (or publish/ subscribe) paradigm, clients subscribe to a server with continuous queries that express their information needs while information sources publish documents to servers. Whenever a document is published, the continuous queries satisfying this document are found and notifications are sent to appropriate subscribed clients. Although information filtering has been in the research agenda for about half a century, there is a huge paradox when it comes to benchmarking the performance of such systems. There is a striking lack of a benchmarking mechanism (in the form of a large-scale standarised test collection of continuous queries and the relevant document publications) specifically created for evaluating filtering tasks. This work aims at filling this gap by proposing a methodology for automatically creating massive continuous query datasets from available document collections. We intend to publicly release all related material (including the software accompanying the proposed methodology) to the research community after publication.**

continuous queries; dataset construction; information filtering; publish/subscribe; information dissemination; profiles;

I. INTRODUCTION

In recent years, content-based *information filtering* (or *publish/subscribe*) applications, such as news or digital library alerts, have gained popularity to help users cope with the information avalanche problem on the Web. In the information filtering paradigm, users -or services that act on users' behalf- *subscribe* to a server with *continuous queries (*or *profiles)* that are expressed in some well-defined language and capture their information needs. When a document is *published* on the server, the continuous queries satisfying the document are found and notifications are sent to appropriate clients. Publishers may be news feeds, digital libraries, or users who post new blog items. Notice that information filtering is very different from information retrieval (as in search engines), in which a user poses a (one-time) query and the search engine executes it only once to retrieve the currently matching documents.

Since a server may handle millions of clients and continuous queries, the filtering problem needs to be solved efficiently by each server. To this end, a number of systems and algorithms that try to solve the filtering problem efficiently for different data models and query languages have been proposed [1, 6, 10, 17, 11, 12, 16, 18, 19, 20]. However, despite all the research in the area, there is an apparent lack of a *benchmarking mechanism* (in the form of a *large-scale standarised test collection* of continuous queries and the relevant document publications) specifically created for evaluating filtering tasks. From our point of view there exist two major problems to be addressed when trying to experimentally evaluate a filtering algorithm: (i) the *document corpus* to be used as publications and (ii) the set of *continuous queries* relative to that corpus. It may not be difficult to collect data to be used as publications since there is a wide collection of document corpora available. It is however, extremely difficult to find continuous queries relevant to a specific corpus except by obtaining *proprietary data* (e.g., from Google Alerts or CNN's news alert system). Notice also that one-time queries, such as those obtained from public releases of major search engines' query logs (like Google BigQuery, Zeitgeist, or the AOL query set) are *inappropriate* for filtering tasks as they typically express a one-time information need, contrary to continuous queries that are used to express *recurrent* and *long-standing* information needs. Finally, other efforts, such as the TREC Filtering Track, are *insufficient* as they contain only a few dozens of *manually* created and curated continuous queries, and cannot live up to the need of modern benchmarking that is in the order of millions (e.g., as in [12, 16, 18, 19, 20]).

Given the above, it becomes clear that the only viable alternative to this lack of standarised benchmarks is to *artificially generate* sets of continuous queries related to the corpus to be used for the evaluation. To the best of our knowledge *this is the first approach in the literature* to provide a *general-purpose methodology* for artificially generating *realistic continuous query datasets* from *actual document corpora* for benchmarking purposes. To this end, our contributions are the following:

- We formally define the query language and data model named *AWP* supported by our methodology.
- We introduce a new corpus of research papers to be used as publications in the filtering tasks. Notice however, that our methodology is general enough to be used with any (attribute tagged) document corpus; here the new corpus is only used as a proof-of-concept for our continuous query creation process.
- We propose a new methodology for creating synthetic user profiles using words and technical terms extracted automatically from the document corpus. To do so we use the corpus at hand to create realistic continuous queries under query language *AWP*. It should, however, be stressed out that our methodology can be applied to any corpus and any query language similar to *AWP*.

The rest of the paper is organised as follows. Section II gives a brief overview of related work and Section III hints on the model *AWP*, which is used for specifying profiles and documents. Section IV presents the NN corpus, whereas Section V presents our methodology for the creation of

continuous queries. Finally, Section VI concludes the paper by providing future research directions.

II. Related Work

Our work relates (at a higher level) to the general area of information filtering efficiency as expressed by a number of systems and algorithms that try to provide scalable information filtering solutions for different data models and query languages. Some of these approaches include the systems XFilter [1], YFilter [6], DFA [10], the Boolean version of SIFT [17], and the agent-based DIAS [11]. Other approaches focus more on the algorithmic aspect by providing efficient tree-based data structures such as [12, 16, 18, 19, 20] for dealing with documents that are free text and profiles that are conjunctions of keywords. To the best of our knowledge the only work that is somewhat relevant to ours is [15], where a corpus of documents (but no continuous queries) is built for adaptive filtering tasks.

Interestingly the evaluation of the XFilter [1] and SIFT [17] is based on a synthetic corpus of documents; XFilter creates them using IBM's XML generator [7] and NITF DTD [5], whereas the creation of continuous queries is also synthetic. Contrary, [10] uses Deterministic Finite Automata to parse a corpus of XML documents and the XPath generator used in YFilter [6] to generate the user profiles. However, the main problem with all these approaches is that they are (i) aimed at a single evaluation and cannot be reused, (ii) based in artificial (and not real document) corpora, and (iii) not freely available to use. Contrary, our approach is general enough to cover many filtering tasks, is based on actual documents to create the continuous query dataset, and will be freely available for use after publication.

III. The Data Model *AWP*

In [11] we present the data model *AWP* for specifying continuous queries and textual resource metadata in information filtering systems. *AWP* is based on the concept of named attributes with values of type text. The query language of *AWP* offers Boolean and proximity operators on attribute values as in the work of [4], which is based on the Boolean model of information retrieval.

Syntax. Let Σ be a finite *alphabet*. A *word* is a finite non-empty sequence of letters from Σ. Let V be a (finite or infinite) set of words called the *vocabulary*. A *text value s* of length n over vocabulary V is a total function $s:\{1,2,...,n\}\rightarrow V$.

Let I be a set of *(distance) intervals* $I=\{[l,u]: l,u \in \mathbb{N},\ l \geq 0 \text{ and } l \leq u\} \cup \{[l,\infty): l \in \mathbb{N} \text{ and } l \geq 0\}$. A *proximity formula* is an expression of the form $w_1 \prec_{i\_1} \cdots \prec_{i\_n-1} w_n$ where $w_1,...,w_n$ are words of V and $i_1,...,i_n$ are intervals of I. Operators $\prec_i$ are called *proximity operators* and are generalizations of the traditional information retrieval operators *kW* and *kN* [4]. Proximity operators are used to capture the concepts of *order* and *distance* between words in a text document. The proximity word pattern $w_1 \prec_{[l,u]} w_2$ stands for "word w_1 is before w_2 and is separated by w_2 by *at least l* and *at most u words*". The interpretation of proximity word patterns with more than one operator $\prec_i$ is similar. A *word pattern* over vocabulary V is a conjunction of words and proximity formulas. An example of a word pattern is $applications \wedge efficient \prec_{[0,0]} data \prec_{[0,3]} fusion$.

Let $\mathring{A}$ be a countably infinite set of attributes called the *attribute universe*. In practice attributes will come from *namespaces* appropriate for the application at hand e.g., from the set of Dublin Core Metadata Elements [21].

A *document d* is a set of attribute-value pairs *(A,s)* where $A \in \mathring{A}$, s is a *text value* over V, and all attributes are *distinct*. The following set of pairs is an example document:

{ (AUTHOR, "Christos Tryfonopoulos"),
(TITLE, "Distributed information filtering is ..."),
(ABSTRACT, "In this paper we show that ...") }

A *query* is a conjunction of the form

$$A_1 = s_1 \wedge ... \wedge A_n = s_n \wedge B_1 \sqsupseteq wp_1 \wedge ... \wedge B_m \sqsupseteq wp_m$$

where each $A_i,B_i \in \mathring{A}$, each s_i is a text value and each wp_i is a word pattern. The following formula is an example query:

$$AUTHOR = \text{"Christos Tryfonopoulos"} \wedge$$
$$TITLE \sqsupseteq (distributed \prec_{[0,3]} filtering) \wedge information$$

Semantics. The semantics of *AWP* have been defined in [11] and will not be presented here in detail. It is straightforward to define when a document *d satisfies* an atomic formula of the form $A = s$ or $B \sqsupseteq wp$, and then use this notion to define when *d* satisfies a query [11]. The example document given above satisfies the example query.

IV. The Neural Network Corpus

The proof-of-concept corpus we use (called *NN corpus*) consists of a fraction of research papers from ResearchIndex [14,13] having Neural Networks as a subject. ResearchIndex, formerly known as Citeseer, is a digital library that targets the improvement in the dissemination of scientific literature. ResearchIndex indexes research articles in various formats and provides a variety of free services, such as full-text and citation indexing as well as paper statistics.

TABLE I. Some characteristics of the NN corpus

Description	***Value***
Number of documents	10,426
Document vocabulary size	641,242
Maximum document size (words)	110,452
Minimum/maximum word size	1/35

The NN corpus consists of 10,426 scientific papers in English. Some important values for this corpus are summarised in Table I above. The documents were downloaded from the ResearchIndex site as postscript files and were converted to text files. Then all references and equations were removed and each word in the document was assigned a grammatical tag (e.g. noun, verb etc.) using a simple rule-based part of speech (POS) tagger [3]. This processing was necessary as a first step for the extraction of multi-word terms by the C-value/NC-value method described briefly in Section V.A and also in [9]. To use the corpus for our continuous query creation we also utilised the full citation graph of ResearchIndex.

Initially, we removed all the POS tags from all the documents. We then used the information from the full citation graph of ResearchIndex to extract the title, authors, abstract, and year of the publication. This information was not extracted from the actual corpus since the flat form of the documents contained considerable noise even after several rule-based filters were applied to it. The next step was to process the abstracts as POS-tagged text files, extracted from the original postscript files. After processing the abstracts we were able to identify the body of the document by excluding the information we already had in hand. When the processing phase was completed, we merged the different attributes extracted, along with the appropriate attribute tags. We then had at our disposal an attribute-tagged corpus with five fields: title, authors, abstract, body and year.

At this point we have to stress out that the information obtained from the citation graph was incomplete, resulting in documents without all the attribute fields filled in. This is actually not a problem in an experimental setting since in an information dissemination scenario users may post documents with only some of the attributes filled in. Table II gives some interesting measures of the fraction of documents out of the document corpus that contain each attribute, and summarises the fraction of documents that contain a specific number of attributes.

TABLE II. ATTRIBUTE STATISTICS

Attribute	*% fraction of documents*
title	63%
authors	58%
abstract	88%
body	86%
year	63%

Number of attributes	*% fraction of documents*
1	7.4%
2	28.0%
3	1.9%
4	16.0%
5	45.0%

V. CONTINUOUS QUERY GENERATION METHODOLOGY

The main construct in our profile creation process is that of a unit. Units in our context represent different entities that can be used to create a profile. The first two unit sets consist of proximity formulas created from multi-word terms, that were extracted from the NN corpus using the C-value/NC-value method described below. The third one is the set of all the nouns extracted from document abstracts, and the fourth one is the set of all author last names in the NN corpus documents. Combining units from these four sets in a well-defined way, allows us to create realistic profile databases in order to conduct our experiments.

A. Automatic Term Extraction

The multi-word terms used in the profiles for our experiments are extracted from the NN corpus using the C-value/NC-value approach of [9]. The process of identification of *terms* or *technical terms* or *terminological phrases* from a collection of documents belongs to the research area called *automatic term recognition*. The C-value/NC-value approach of [9] specifies the "termhood" of a candidate multi-word term as the probability (co-location value) to be a real term. The C-value of a term is an enhancement of the common statistical measure of frequency of occurrence, incorporating information about nested terms, whereas NC-value embodies information form words that appear in the vicinity of terms in texts. Both methods have been shown to perform better than the classical frequency of occurrence measure in terms of precision and recall [8]. For details on the method the reader is invited to see [9, 8].

B. Creation of the Different Unit Sets

The creation of the first two unit sets was based on the extraction of multi-word terms from the corpus. To create these sets, a ranked list of multi-word terms was extracted from the corpus documents. We then, excluded from this list all terms that contained more than five words since they were noise produced by the C-/NC-value methods. Additionally, we specified an upper and lower NC-value cut-off threshold for the terms remaining in the list. These cut-off thresholds were used to increase the discriminating power of the set of terms. The upper cut-off threshold was used to exclude top ranked terms, that is terms that appear very often in corpus documents. Such an example is the 2-word term "neural networks" that is contained in most of our documents. Moreover the bottom ranked terms are also excluded from the list of the useful terms since they are mostly noise created from the procedure of transforming the original postscript files to simple text files. This processing resulted in a list containing 2-, 3-, 4- and 5-word terms, which was then used to create two different sets as follows.

Let $a_1 a_2 ... a_n$, where each a_i is a word, be a multi-word term from the aforementioned list, containing n words. A proximity formula is created out of this term in the following two ways:

1. $a1 \prec_{[0,0]} a2 \prec_{[0,0]} ... \prec_{[0,0]} a_n$. For each multi-word term in the list we introduce the proximity operator $\prec_{[0,0]}$ between the words of the multi-word term in order to create proximity formulas that represent strings. All the proximity formulas that are created this way form the first set of units named PF_0, which stands for *proximity formulas with word distance zero*. The number of operands in these proximity formulas varies according to the number of words contained in the multi-word terms. The minimum number of words in a multi-word term is obviously two, whereas the maximum is five. An example of a unit in this set, which was produced from the term "inverse dynamic function", is *inverse* $\prec_{[0,0]}$ *dynamic* $\prec_{[0,0]}$ *function*.
2. $a_1 \prec_{[0,k]} a_n$, where $1 \leq k \leq 10$. From each term in the list of multi-word terms we create proximity formulas with exactly two operands. These proximity formulas are created as follows. We replace *all* the middle words of the 3-, 4-, 5-word terms with the proximity operator $\prec_{[0,k]}$, specifying k to be a natural number drawn uniformly between 1 and 10. The choice of using a relatively small upper bound in the distance between two operands is inspired by the implementation of

operator '*' and '*NEAR'* in Google and Yahoo! respectively. All the proximity formulas created this way form the second set of units named PF_k, since they are proximity formulas with word distance k. An example of a unit in this set could be *rbf* $\prec_{[0,6]}$ *networks*, which could be created from the term "rbf dynamic decay adjustment networks".

The second set of units used in the creation of our profiles database is the set of *nouns* that were extracted from document abstracts. The choice of nouns taken from document abstracts as opposed to the whole document can be justified by the argument that the abstracts are expected to be a brief description of the work carried out in the paper thus, very appropriate to describe the content of a paper. The procedure of creating the set of nouns is as follows. First, we identified all the nouns in singular and plural form using the POS-tagged abstracts that were available to us. After that, we created a frequency-ranked list of these words and specified an upper/lower cut-off threshold to cut the most/least frequent words. The set of units that resulted from this procedure is denoted by *NS*, which stands for nouns.

The last set of units created is that of the authors' last names. We extracted all the names of the authors that were available to us from the corpus documents to obtain an author vocabulary V_{author} of 8,833 last names. Please notice that using this author vocabulary to *uniformly* draw author names for continuous queries is not a good choice, since authors that are more active or produce more important papers than others are expected to be used heavier in continuous queries. The criterion for identifying the more important authors is how many citations they get from papers written by other researchers. In the citation graph of the NN corpus this is captured by the in-degree of the papers as explained in [2]. The highest the in-degree for the papers of a specific author, the highest the probability for this author to appear in a profile. We define N_a to be the number of papers in the corpus that refer to at least one document of author a, and V_{author} the author vocabulary. N_a can easily be extracted from the full citation graph, and the author vocabulary is available to us from the NN corpus documents. Thus, the probability of author a to be used in a continuous query is:

$$P(a) = \frac{N_a}{\sum_{k \in V_{author}} N_k}$$

The above formula associates an author with the popularity of his writings and thus, with a probability of another researcher being interested in his work. To capture the probability distribution of the author surnames, we used a *multi-set* that contains an author surname N_a times, and presents a power-law distribution (Figure 1). This can be explained by taking into account the general observation that in every scientific domain there exist a few heavily cited authors, while the rest receive less visibility (in terms of citations of their work). The unit multi-set described above is denoted by AS (author surnames).

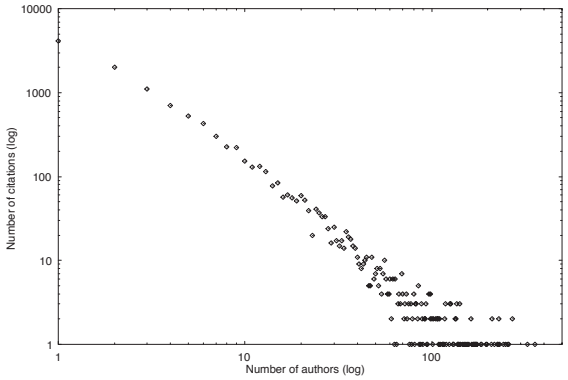

Figure 1. Distribution of citations among authors

C. Details

In this section, we provide details of how all the above extracted information is combined to create realistic continuous queries. A continuous query under the subset of query language *AWP* consists of a conjunction of atomic queries. These atomic queries can only be of the form $A \sqsupseteq wp$, where A is an attribute and wp is a conjunction of words and proximity formulas. In the rest of this section we will examine the different types of atomic queries that can be created according to the attributes that are available to us.

TABLE III. SPECIFICS FOR THE CREATION OF ATOMIC FORMULAS

Attribute	***Participating unit sets***	***Indicatory value of σ***
title	PF0	0.4
	PFk	0.4
	NS	0.2
abstract	PF0	0.4
	PFk	0.4
	NS	0.2
body	PF0	0.4
	PFk	0.4
	NS	0.2
author	AS	1.0

In our context, creating a continuous query can be viewed as the problem of choosing with a probability distribution between units contained in different sets. Not all the sets of units participate in the creation of an atomic formula of a specific attribute. Moreover, different unit sets that participate in an atomic formula may have different *selection probabilities (σ)* in being chosen to participate in a profile. The unit sets that participate in the creation of an atomic formula, along with an indicatory value for the σ of each unit are summarised in Table III. Notice that these values may vary depending on the properties of the continuous query database to be generated.

In general, a creation of an atomic query is a 3-step process that can be described as follows:

1. Choose the number of units (or the *size* of an atomic query) S. This value is drawn uniformly from $[1,S_{max}]$, where S_{max} is the maximum number of units in an atomic query. S_{max} is defined to be 2 for atomic formulas of author and title attributes, whereas it is set to 3 for the abstract and body attributes. This differentiation in S_{max}

is due to the different number of words contained typically in the different attributes of a document.

2. Taking into account the units that may participate in a specific atomic formula, we pick *S* units from these sets according to the selection probabilities summarised in Table III.
3. Having chosen these units, we take their conjunction to create the atomic formula.

Thus, an atomic formula for the title attribute may be:

$$title \supseteq (rbf \prec_{[0,6]} networks) \wedge java$$

which contains two units (remember that this is the maximum number of units allowed for the title attribute): unit ($rbf \prec_{[0,6]} networks$) drawn from unit set PF_k and unit *java* drawn from unit set *NS*. Modifiying σ in the different unit sets results in controlling how often units of a specific set will appear in atomic queries of the corresponding attribute. Thus, other possible atomic formulas could be:

$$title \supseteq implementation \wedge (dynamic \prec_{[0,0]} functions)$$
$$title \supseteq real \prec_{[0,0]} world \prec_{[0,0]} application$$
$$title \supseteq algorithm \wedge implementation$$

Atomic queries for abstract and body attributes are created in a similar way. The only differentiation between atomic formulas of different attributes is the value of σ for the unit sets and the maximum atomic query sizes.

At the same time creating atomic queries for attribute author is somewhat different since it may contain either one unit or a conjunction of two units from *AS*. Note that for the case of an atomic query for the author attribute using more words in conjunction would make the profile very specific, thus not suitable for an information alert setting. Note also, that proximity operations may also be used in these atomic queries (e.g., $John \prec_{[0,0]} Brown$). However, the authors' first names were not available from the corpus documents so this option was not adopted. Some examples of such atomic queries are $author \supseteq Brown$ or $author \supseteq Smith \wedge Johnson$.

Finally, to decide which atomic queries will be introduced as conjuncts in each continuous query we assign selection probabilities to each one of the four types of atomic queries and according to this selection probabilities we include or exclude atomic queries. Each type of atomic query is (or is not) included in a profile independently of the rest of the types. For example, for a specific profile generation scenario if the selection probability of all four types of atomic queries is 85% then atomic queries for the author attribute will appear in the 85% of the profiles in the profile database. The same holds for the rest of the attribute types (title, abstract and body). At this point we should stress that in this way all possible combinations of atomic queries may appear in the generated continuous queries, and that a simple probability calculation allows us to control or exclude certain types of atomic queries.

VI. Conclusions and Future Work

In this work we presented a methodology for creating realistic artificial continuous query databases from any real-life (attribute-tagged) corpus, and as a proof-of-concept we applied it to the NN corpus. The robustness of the proposed methodology is highlighted not only by the publications in top-class venues that utilize it (e.g., [11, 12, 16, 18]), but also by the different document corpora it was applied on (TREC *.gov, TREC ClueWeb09*, *OHSUMED*, *NN,* and others). Upon publication of this work we plan to publicly release all code (i.e., for corpus preprocessing and query generation) and the proof-of-concept NN corpus, in an effort to assist researchers in creating their own benchmarks for the evaluation of filtering tasks and give visibility to the hosting venue.

Finally, interesting directions for future work include the design and implementation of modules for creating realistic vector space and semi-structured continuous queries.

References

[1] M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In VLDB, 2000.

[2] Y. An, J. Janssen, and E. Milios. Characterizing and Mining the Citation Graph of the Computer Science Literature. Technical Report CS-2001-02, Dalhousie University, Canada, 26 September 2001.

[3] Eric Brill. A simple rule-based part-of-speech tagger. In ANLP, 1992.

[4] C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. ACM TODS, 17(1):1–39, 1999.

[5] R. Cover. The SGML/XML Web Page. http://www.oasis-open.org/cover/sgml-xml.html

[6] Y. Diao, M. Altinel, M.J. Franklin, H. Zhang, and P. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. ACM TODS, 2003.

[7] A.L. Diaz and D. Lovell. XML Generator. http://www.alphaworks.ibm.com/tech/xmlgenerator

[8] L. Dong. Automatic term extraction and similarity assessment in a domain specific document corpus. MSc thesis, 2002.

[9] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms:the C-value/NC-value method. IJDL, 5(2), 2000.

[10] T. J. Green, G. Miklau, M. Onizuka, and D. Suciu. Processing XML Streams with Deterministic Automata. In ICDT, 2003.

[11] M. Koubarakis, T. Koutris, C. Tryfonopoulos, and P. Raftopoulou. Information Alert in Distributed Digital Libraries: The Models, Languages and Architecture of DIAS. In ECDL, 2002.

[12] L. Zervakis, C. Tryfonopoulos, S. Skiadopoulos, and M. Koubarakis. Query Reorganisation Algorithms for Efficient Boolean Information Filtering. IEEE TKDE 29(2): 418-432, 2017.

[13] S. Lawrence, C. Lee Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. IEEE Computer, 32(6):67–71, 1999.

[14] ResearchIndex. The NEC Research Institute scientific literature digital library. http://www.researchindex.org

[15] I. Soboroff and S.E. Robertson. Building a filtering test collection for TREC 2002. In ACM SIGIR, 2003.

[16] C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. In ACM SIGIR, 2004.

[17] 17. T.W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. ACM TODS, 19(2):332–364, 1994.

[18] C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Information filtering and query indexing for an information retrieval model. *ACM TOIS*, 2009.

[19] W. Rao, L. Chen, S. Chen, and S. Tarkoma. Evaluating continuous top-k queries over document streams. WWW, 2014.

[20] M. Sadoghi and H. Jacobsen. Analysis and optimization for boolean expression indexing. *ACM TODS*, 2013.

[21] http://dublincore.org/documents/dcmi-terms/