16th World Congress of the International Fuzzy Systems Association (IFSA)
9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)

Flexibility in Patent Search

Stefania Marrara, Gabriella Pasi

Department of Informatics, Systems and Communication (DISCo)
University of Milano-Bicocca, Building U14,
I-20126, Milano, Italy
{stefania.marrara, pasi}@disco.unimib.it

Abstract

Patent Retrieval is a branch of Information Retrieval (IR) aiming to support people in retrieving patents that satisfy their information needs. Patent Retrieval is a challenging task: usual queries involve a huge number of words - often entire patent claims; moreover the style of writing a patent is often obscure causing a severe amount of vocabulary mismatch between queries and patents. For this reason, despite of the numerous patent search applications already available, we propose in this paper *PatentLight*[1] a search tool that offers novel and flexible functionalities based on both fuzzy logic and IR to help users looking for relevant patents here represented as XML documents.

Keywords: Patent Search, Fuzzy Logic, Information Retrieval, Flexible Query Language, XML

1. Introduction

Traditional Information Retrieval (IR) aims to retrieve documents in response to a user information need from a previously indexed collection of documents. An important specialized branch of IR is Patent Information Retrieval (PIR), which is aimed to support users, often professionals such as patent attorneys, in retrieving patents that satisfy their information needs [2].

In fact a crucial application is *prior-art retrieval*, which is performed by patent searchers to determine the novelty of a new invention. In this scenario patent authors require an exhaustive knowledge of all related patents since overlooking a single important patent could lead to detrimental and very expensive consequences, such as patent infringements and litigation. In other words, *recall* is a very important requirement for any patent search engine.

On the opposite side, today patents are commonly available thanks to collections such as USPTO (United States Patent and Trade Office), EPO (European Patent Office) and WIPO (World Intellectual Property Organization).

Each collection contains several thousands of patents and continues to grow up year by year; this situation poses a serious issue to patent professionals: the cost of filing patents, defining claims and defending a claim of infringement is increasing with time, making the process often too expensive, due to the complexity in finding relevant patent. In 2010 the estimated cost to find relevant patents was $1,500 per patent filing [3].

As a consequence, the amount of patents available for retrieval, combined with the frequent innate difficulty of patent content, create an equally important need for *precision* in retrieval.

For the above reasons Patent Retrieval is achieving a high interest in the scientific community, and it is also considered a complex challenging task:

- the vocabulary used in patents is often obscure since it contains a lot of specialized or technical words; moreover often the obfuscation of content is intentional by writers who wish their patents difficult to retrieve.
- patents contain a laying structure which often include *description*, *claims* or *prior-art* for instance;
- typical queries in patent retrieval include a huge amount of words, often entire claims.

Most approaches presented in the literature, based on *keyword extraction* or *query expansion* techniques, proved to produce poor results (see Section 1.1). Despite that, we believe that a traditional keyword extraction applied to the user query plus our flexible search approach based on both structure and content analysis of XML patents is worth of giving promising results with respect to both recall and precision. The first experimental results produced by [1] promote to further investigate in this direction. To the aim of this work, we start from a user query defined as a traditional keyword based query. We can state that our query approach is the outcome of a traditional keyword extraction engine applied to a usual, huge, patent repository. Moreover, in this paper we propose an approach that relies on the recent outcomes of research in XML Retrieval.

1.1. Related Work

In this section we briefly review the main contributions in the Literature about Patent Search (see section 1.1.1) and XML Retrieval (see section 1.1.2).

1.1.1. Patent Search in a nutshell

Patent IR evolved as a separate branch of IR since it outlines characteristics which drastically reduce

the effectiveness of traditional retrieval techniques.

In the last years, several approaches have been proposed, which can be broadly classified into three categories:

- approaches based on *query expansion* techniques to reduce vocabulary mismatch;
- approaches based on *query extraction* techniques to reduce verbose queries
- approaches based on *query translation* techniques, which include approaches for querying multilingual patent collections, and approaches to query patents section by section instead as a whole document.

Because of the peculiarities of patent retrieval w.r.t traditional retrieval (as described in the Introduction), standard IR techniques such as *query expansion* proved not to work effectively with patent queries due to the presence of *noisy* terms in the typical queries. In real practice however, most patent examiners formulate their queries for invalidating claims by selecting high frequency terms from the query-patent claim text, and hence the first approaches proposed in the literature [4, 5] moved their steps from this practice and were based on keyword extraction to reduce queries dimensions, unfortunately achieving low results. More recently, [6] and [7] showed that using the whole patent text with raw term frequency reduces the job complexity and the best results are obtained when terms are taken from all the fields of the query patent. Other approaches, [8] and [9], used citation extraction to improve the retrieval effectiveness of keyword based IR methods, and this idea is also adopted in [10] which uses also a query expansion technique on segmented queries.

Another important work is [11] which also adopted a query expansion technique based on some *structural* properties of patents such as abstract, description and image descriptions.

The last class of patent retrieval approaches tries to take advantage from the multilinguality of most patent collections. Most works are based on natural language processing (NLP) approaches [12, 13, 14]. In particular, the most recent [14] uses NLP, and specifically statistical word alignment to translate patent queries from language to language. More generally, query translation has been a popular research mainstream and it is usually realized by means of dictionaries, machine translation systems, ontologies or combinations of these (see [15] for an overview).

1.1.2. A short presentation of XML Retrieval

In the last ten years, due to the increasing usage of the eXtensible Markup Language (XML) on the Internet, and the resulting development of XML based systems and search engines, XML Retrieval has become one of the most lively research topics for both Information Retrieval and Database research communities. XML documents are not just textual documents, but they also contain information about the logical structure of the documents themselves. The logical structure in fact is a tree-like structure that is encrypted by the XML nodes. In XML Retrieval the retrieved units are not the whole documents but elements and components of the documents. The Information Retrieval community has focused on content based retrieval of XML documents and most research has emerged from the NEXI initiative design [16]. On the opposite side the database research community has focused on the efficient retrieval of information stored in XML documents by means of exploiting in deep the document structure features. This approach has developed structure based query languages, which has given birth to many approaches based on XPath [17], XQuery [18] and, at the end, XPath Full Text and XQuery Full Text [19].

2. FleXy: a Flexible XML Query language

In [20] a flexible extension of the XQuery Full Text language (*FleXy*) by introducing flexible constraints on both XML document structure and content was defined.

A patent search application based on FleXy has been proposed in [1] *PatentLight*.

In PatentLight the structure-based constraints of Flexy named `below` and `near`, and the content-based flexible constraint `around` where employed. In this paper we introduce the constraint `similar` which applies on tag names, and we show how the combination of content-based and structure-based evaluation of results can improve the effectiveness of PatentLight retrieval model. Here below a short explanation of the above flexible constraints is given.

2.1. A short description of `below`, `near` and `around`

The constraint `below` allows to retrieve the fragments of an XML document (in this case a patent) that are closer to the minimum path required by the user's query. The syntax of the `below` constraint follows the standard XQuery axis syntax, and it is specified as: `c/below::t`, where `c` is the *context* node, and `t` is the target node. The best retrieved path is the one in which `t` is direct child of `c`. Others paths, those in which `t` is simply descendant of `c`, will be retrieved but ranked in a lower position w.r.t. the best one. To create the list of results we compute a *path relevance degree* for each retrived fragment, $w_{c,t}$, computed as $w_{c,t} = \frac{1}{|desc\_arc(c,t)|}$ where *desc_arc(c,t)* is a function that returns the set of descending arcs from `c` to `t` if and only if `t` is a descendant node of `c`.

Figure 1 shows an example of the use of the constraint `below`: the user's query `US-patent-Grant/below::p[text() contains`

`text "Bell"]` is evaluated on an XML patent document and allows to retrieve all the paragraphs containing the text *Bell* that are descendant of the node `US-Patent-Grant`. In Figure 1 the black node identifies the *context node* while the grey nodes are the retrieved *target nodes*; the arrows between nodes represent the path connecting each target node with the context node. The query evaluation produces a list with two paragraphs *p* nodes: the first one with a score of $\frac{1}{2} = 0.5$ due to its distance of two arcs from the `US-Patent-Grant` element, and the second with a score of $\frac{1}{3} = 0.33$.

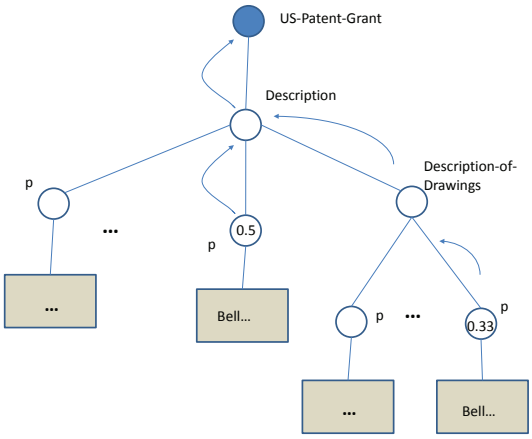

Figure 1: The flexible constraint `below`

The flexible constraint `near` retrieves elements that are connected to the *context node* by any path (not only the *descendant* relationship), i.e., also *ancestor* and *sibling* elements are evaluated. For the `near` constraint, the scoring function is defined as: $w_{c,t} = \frac{1}{|arcs(c,t)|}$ where *arcs(c,t)* is the function that returns the set of arcs that connects the context node `c` to the target node `t` following the shortest path. The `near` constraint syntax is: `c/near::t`, where, as for the constraint `below`, `c` is the context node and `t` is the target node.

Figure 2 shows an example of how the constraint `near` works on a portion of the XML-structure of USPTO patent related to the definition of people who have different roles in a patent such as examiners, applicants and agents. In the example, the fragments retrieved by the query `//applicants/near::last-name` are the `last-name` elements in grey color, while the context node (`applicants`) is the black node in the figure. The arrows outline the arcs connecting the target node with the context node, while the number inside a target node is its computed score. As an example, the `last-name` element containing the text *Bond* receives a score of 0.33 due to the three arcs that connect such node with the context node `applicants`. In this case the use of the constraint `near` allows to provide higher score to people having a role as *Applicant* for a given patent, and to retrieve with a lower score people having a role of *Agent* or *Examiner*.

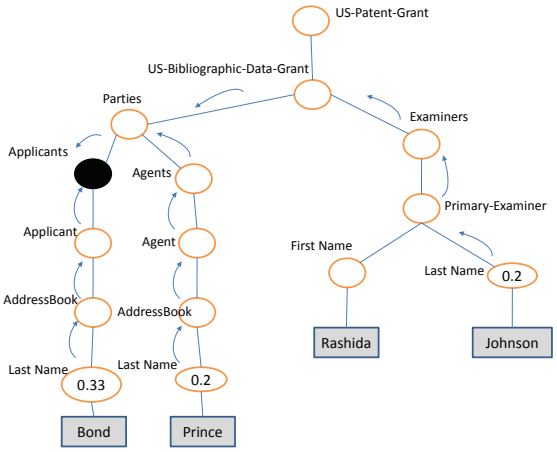

Figure 2: The flexible constraint `near`

`Around` is a flexible constraint which applies to numerical data and its evaluation function is formally defined as the membership function of a fuzzy subset of the considered numerical domain; the membership function expresses the similarity between the retrieved values and the numerical value requested by the user. In the patent domain, the constraint `around` is defined to the aim of analyzing date contents.

The FleXy syntax of `around` is `'tag-date/@date[x around b]'`, where `tag-date` is the attribute having the date value that has to be evaluated, `x` is the date value of the examined patent, and `b` is the date written by the user in his/her query.

The evaluation function of the `around` constraint produces a score in the interval [0,1] based on the date value `b` specified by the user and the date value `x` of the patent. Patents with a date value close to the one specified by the user will receive a higher score (score close to 1) than other patents. The evaluation function of the flexible constraint `around` on the *Date* domain can be defined as fuzzy subset with a triangular membership function centered on *b*.

2.2. The `similar` constraint to assess tag similarity

`Similar` is a flexible constraint defined on tag names that allows to retrieve fragments with a target node name *similar* to the name written in the user's query. `Similar` is defined as a function whose FleXy syntax is `'similar(x)'` `x` is the node name we are looking for. The evaluation of the function returns a list of XML fragments with a target node name similar to `x` where the *similarity degree* is number in the interval [0,1] computed as $ws = \frac{1}{1+ed}$ with `ed = edit distance between the retrieved tag name and x`.

Fig. 3 shows how the `similar` constraint works on the same patent document adopted in the example of Fig 2.

Although the query $Q1$ is looking for a fragment containing the tag name `Address`, the system is able to retrieve the patent fragment containing the tag name `Addressbook` with a *similarity degree* of 0.2.

Note that traditional XML query languages would not retrieve any result in the same situation.

Moreover, to avoid the retrieval of unuseful fragments we can set a threshold value for ws. At present we do not evaluate synonyms since this option would include the use of a dictionary or an ontology.

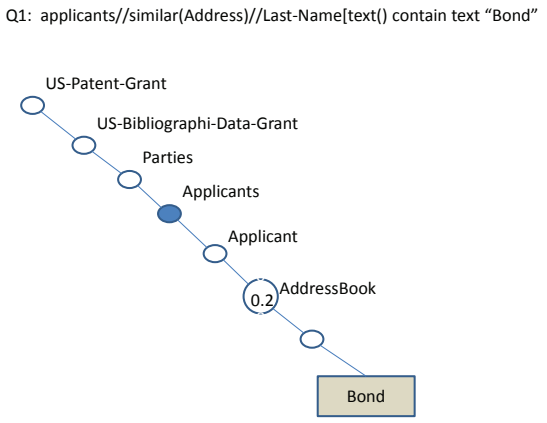

Figure 3: The flexible constraint `similar`

When a query involves more that one flexible constraint, for instance a flexible axis and the `similar` constraint for the target node name, the overall *relevance degree* $wo_{c,t}$ is computed as a combination between the two scores, $w_{c,t}$ and ws. In principle we prefer a conservative evaluation and therefore we use $wo_{c,t} = min(w_{c,t}, ws)$ but different solutions could be tested.

3. How the flexible constraints works in patent search

This section explains how the flexible constraints introduced in the previous section are used to exploit the patent search task. In particular, the proposed approach has been defined and tested in [1] on the USPTO patent collection that can be freely downloaded on the internet. USPTO is the corpus adopted by most patent search applications such as Google Patents and PatentsSearcher. In any case it was noted that also EPO and WIPO patent documents show more or less the same structure, even if with different tags. We expect that the `similar` constraint will be helpful in querying a cross collections domain since it will capture the same information stored in elements that do not share exactly the same tag name. The proposed approach allows users to search patents using the formulation of a keyword based query, in addition the user can also choose the *similar* tag option to extend the search also to tag names similar to the standard ones.

Subsection 3.1 describes how search results are categorized according to keyword-based queries, while subsection 3.2 shows how the *similar* constraint evaluation changes the original approach in [1].

3.1. Keyword-based Query: the approach

An important functionality of FlexSearch [1] is to categorize patents by exploiting their XML structure. In this way the engine organizes the XML patents into meaningful semantic XML elements covering the main patent information. In this way the categorization process described below can easily capture what the user topical search intent is by identifying the possible interpretations associated with a patent. By analyzing the patents in the USPTO collection, four categories were identified in [1]: *People*, *Title*, *Description*, and *Claims*. Formally, let E be the set of XML elements defined in a patent collection, and Cat be the set of categories, then one or more elements $e_i \in E$ are mapped into each category $c \in Cat$, i.e $\{e_1, ..., e_m\} \rightarrow c$. In the application the four identified categories along with the corresponding XML elements are: *People* (the associated elements are `Applicants`, `Agents`, `Assignee`, `Examiners`), *Title* (`title`), *Description* (`Description`), *Claims* (`claims`).

A user specified keyword based query (here below "`query terms`") is automatically rewritten into four distinct FleXy queries, one for each of the four categories. The structure of each query is predefined in order to search the query terms in pre-established elements as follows:

```
People:
applicants/near::Last-Name[
text() contains text "query terms"]

Title:
invention-title[
text() contains text "query terms"]

Descriptions:
Description/below::p[
text() contains text "query terms"]

Claims:
claims/below::claim-text[
text() contains text "query terms"]
```

The proposed query translation process uses the `near` constraint in the FleXy query related to the category *People*, and the context node is the tag `applicants`; this means that we assume that the

applicant role (i.e., the inventor) has more importance in the search with respect to the other roles defined in the patent such as *Agent*, *Examiner*, etc as reported in the example of Figure 2. This choice has been motivated to be coherent with respect to standard patent search applications (i.e., Google Patents, PatentSearcher, etc.).

In case of user queries formulated by the standard textual search area, where a user writes a *name* of a person it is supposed that he/she is interested in finding inventors of patents. However, it is important to notice that by the approach also patents containing the *name* with a different role will be retrieved.

3.2. The evaluation of `similar` in FlexSearch

In this work we improve the FlexSearch engine by introducing the possibility to retrieve also fragments with different tag names w.r.t. those expressed by the query. This feature is useful when we interrogate collections for which we roughly know the internal structure (tag names and node positions) or we want to apply the same query to a composition of patent collections which contain more or less the same information stored in different tag nodes (for instance the node `Orgname` instead of the node `Last-Name`).

In the engine, if the user chose to add the *similar* tag evaluation (in the prototype the user just flags the option in the interface), the set of FleXy queries would change accordingly as shown below:

```
People:
similar(applicants)/near::Last-Name[
text() contains text "query terms"]

applicants/near::similar(Last-Name)[
text() contains text "query terms"]

Title:
similar(invention-title)[
text() contains text "query terms"]

Descriptions:
similar(Description)/below::p[
text() contains text "query terms"]

Claims:
similar(claims)/below::claim-text[
text() contains text "query terms"]

claims/below::similar(claim-text)[
text() contains text "query terms"]
```

Note that the `similar` constraint is applied to the relevant nodes of each query and therefore the categories *People* and *Claims* will contain two queries instead of one. The retrieved fragments are ranked according to two values: the degree of structural relevance based on the evaluation of FleXy constraints ($wo_{c,t}$), and the degree of relevance obtained by the full-text scoring of the XQuery Full Text language (the prototype in [1] uses the BaseX system [21]). The approach privileges the structural ranking w.r.t. the content based relevance since it was observed that the paragraphs most related to the invention are usually structurally closer to the tag `Description`.

4. Conclusions and Future Work

In this paper we have introduced the flexible constraint on tag names `similar` in the definition of PatentSearch, a patent search application and we have shown how different flexible constraints can be combined to compute an overall relevance degree of a document fragment. Future work will include the development of the full prototype and two sets of experiments: one on the collection USPTO to compare the results with those obtained in [1] and the second one with a collection composed by USPTO, EPO and WIPO to fully test the potentiality of the constraint `similar`. Furthermore we could also study the evaluation of synonyms for the tag names used in the queries of the four categories.

References

[1] S. Calegari, E. Panzeri, G. Pasi, *PatentLight: a Patent Search Application*, Proceedings of lliX 2012, Nijmegen, The Netherlands.

[2] Mihai Lupu, Allan Hanbury. Patent Retrieval Foundations and Trends in Information Retrieval. 7(1): 1-97 (2013)

[3] B. Wiens. *Understanding Patents.* http://www.benwiens.com/patents.html, 2010.

[4] H. Itoh, H. Mano, and Y. Ogawa. Term distillation in patent retrieval. In Proceedings of the ACL-2003 worshop on Patent corpus processing - Volume 20, pages 41-45, Stroudsburg, PA, USA, 2003.

[5] T. Takaki, Query terms extraction from patent document for invalidity search. In NTCIR-5, 2005.

[6] X. Xue and W. B. Croft. Transforming patents into prior-art queries. In SIGIR, pages 808-809, 2009.

[7] M. Z. Wanagiri amd M. Adriani. Prior art retrieval using various patent document fields contents. In CLEF-2010 (Notebook Papers/LABs/Workshops), 2010.

[8] A. Fujii, Enhancing patent retrieval by citation analysis. In SIGIR, pages 793-794, ACM, 2007.

[9] W. Magdy, P. Lopez, and G. J. F. Jones. Simple vs. sophisticated approaches for patent prior-art search. In ECIR, pages 725-728, 2011.

[10] D. Ganguly, J. Leveling, and G. J. F. Jones. United We Fall, Divided WWe Stand: A Study of Query Segmantation and PRF for Patent

Prior Art Search. In Proceedings of PaIR'11, pages 13-17, Glasgow, UK, ACM, 2011.
[11] V. Hristidis, E. Ruiz, A. Hernandez, F. Fanfan, and R. Varadaraian. Patentssearcher: a novel portal to search and explore patents. In PaIR'10, pages 33 - 38, New York, NY, USA, ACM, 2010.
[12] L. S. Larkey. A patent search and classification system. In ACM DL, pages 179 - 187, ACM, 1999.
[13] M. Osborn, T. Strzalkowski, and M. Marinescu. Evaluating document retrieval in patent database: A preliminary report. In F. Golshani and K. Makki, editors, CIKM, pages 216 - 221, ACM, 1997.
[14] C. Jochim, C. Lioma, H. Schutze, S. Koch, and T. Ertl. Preliminary Study into Query Translation for Patent Retrieval. In PaIR'10, Toronto, Ontario, Canada, pages 57 - 66, ACM, 2010.
[15] D. W. Oard, D. He, and J. Wang. User-assisted query translation for interactive cross-language information retrieval. Inf. Process. Manage., 44(1):181 - 211, 2008
[16] A. Trotman and B. Sigurbjornsson. NEXI, Now and Next. In Advances in XML Information Retrieval, vol. 3493, N. Fuhr, M. Lalmas, S. Malik, and Z. SZlavik, Eds., ed: Sprienger Berlin Heidelberg, 2005, pages 16-40.
[17] W3C. XML Path Language 1.0. http://www.w3.org/TR/xpath/
[18] W3C. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/
[19] W3C. XQuery and XPath Full Text 1.0. http://www.w3.org/TR/xpath-full-text-10/
[20] E. Damiani, S. Marrara, and G. Pasi, A flexible extension ox XPath to improve XML querying. In SIGIR, pages 849-850, Singapore, 2008, ACM.
[21] C. Grun, S. Gath, A. Holupirek, and M. H. Scholl. XQuery Full Text Implementation in BaseX. In XSym '09, pages 114-128, Berlin, Heidelberg, 2009. Springer-Verlag.