Semantic Search

(SemSearch 2008)

International Workshop located at the
5th European Semamntic Web Conference (ESWC 2008)
June 2, 2008, Tenerife, Spain

Published online as
CEUR Workshop Proceedings, ISSN 1613-0073
CEUR-WS.org/Vol-334/

Edited by
Stephan Bloehdorn, Marko Grobelnik, Peter Mika, and Thanh Tran Duc

Supported by the European Research Projects
X-Media, ACTIVE, NEON and the PASCAL Network of Excellence

Organization

Workshop Organizers

Stephan Bloehdorn
Marko Grobelnik
Peter Mika
Thanh Tran Duc

Programme Committee

Bettina Berendt
Paul Buitelaar
Wray Buntine
Pablo Castells
Fabio Ciravegna
Alistair Duke
Blaz Fortuna
Norbert Fuhr
Lise Getoor
Rayid Ghani
Peter Haase
Andreas Hotho
Esther Kaufmann
Christoph Kiefer
Yiannis Kompatsiaris
Alexander Loeser
Eduarda Mendes Rodrigues
Sergej Sizov
Nenad Stojanovic
Raphael Volz
Haofen Wang
Michael Witbrock
Yong Yu
Ilya Zaihrayeu
Hugo Zaragoza

Additional Reviewers

Joachim Kleb
Edgar Meij

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Preface

In recent years, we have witnessed tremendous interest and substantial economic exploitation of search technologies, both at web and enterprise scale. However, the representation of user queries and resource content in existing search appliances is still almost exclusively achieved by simple syntax-based descriptions of the resource content and the information need such as in the predominant keyword-centric paradigm. While systems working on the basis of these rough approximations have shown to work well for topical search, they usually fail to address more complex information needs. Semantic technologies, namely expressive ontology and resource description languages, scalable repositories, reasoning engines and information extraction techniques are now in a mature state such that they can be applied to enable a higher level of semantic underpinning in real-world Information Retrieval (IR) systems. This application of semantic technologies to IR tasks is usually referred to as *Semantic Search* and the field can be roughly organized along three main topic clusters.

Firstly, more expressive descriptions of resources can be achieved through the conceptual representation of the actual resource content and the collaborative annotation of general resource metadata using standard Semantic Web languages. As a result, there is high potential that complex information needs can be supported by the application of Semantic Web technologies to IR, where expressive queries can be matched against expressive resource descriptions. Secondly, in the past year we have also seen the emergence of important results in adapting ideas from IR to the problem of search in RDF/OWL data, folksonomies or micro-format collections. Common to the first two scenarios is that the search is focused not on a document collection, but on metadata (possibly linked to or embedded in textual information). Thirdly, semantic technologies provide powerful tools to complement existing IR systems on classical resource collections, in particular textual documents.

In this context, several challenges arise for Semantic Search systems. These include, among others:

1. How can semantic technologies be exploited to capture the information need of the user?
2. How can the information need of the user be translated to expressive formal queries without enforcing the user to be capable of handling the difficult query syntax?
3. How can expressive resource descriptions be extracted (acquired) from documents (users)?
4. How can expressive resource descriptions be stored and queried efficiently on a large scale?
5. How can vague information needs and incomplete resource descriptions be handled?
6. How can semantic search systems be evaluated and compared with standard IR systems?

We are happy to see that this workshop succeeded in attracting a large number of high quality paper submissions, all of which are targeting one or, most often, multiple of these questions. Overall, the workshop program committee has selected 10 submissions for oral presentation and inclusion in these proceedings.

Furthermore, we are happy to have Michael Witbrock from Cycorp Inc. complementing the main workshop program by discussing the topic of Semantic Search in his invited talk *Large Scale Search Improvement needs Large Scale Knowledge*.

We thank the members of our program committee for their efforts to ensure the quality of accepted papers. We kindly acknowledge the European research projects *X-Media*, *ACTIVE*, *NEON* and the *PASCAL Network of Excellence* that are supporting this workshop. We are looking forward to having interesting presentations and fruitful discussions during the workshop day.

May 2008 - Karlsruhe/Ljubljana/Barcelona

Your SemSearch 2008 Team
Stephan Bloehdorn, Marko Grobelnik, Peter Mika, and Thanh Tran Duc

Table of Contents

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Large Scale Search Improvement needs Large Scale Knowledge (Invited Talk)

Michael Witbrock

Cycorp, Inc., Austin, Texas and
Cycorp.eu, Ljubljana, Slovenia
witbrock@cyc.com

It seems obvious that understanding documents as more than a weighted bag of terms should improve access to the knowledge that they contain, but this has been tremendously hard to demonstrate in practical systems. The fundamental problem is one of scale: the place where semantics matters is not common queries, for those the best document responses can simply be learned; it is in the long tail of rare searches. But these searches are difficult to improve for two reasons:

1) the semantics of less frequent terms tend to be less ambiguous, so more than simple semantic tagging is called for, and
2) that scale is a sine-qua-non: there are many many millions – perhaps many billions – of concepts in the long tail, and only systems that cover a substantial proportion of these can make a difference.

At Cycorp, we've been pushing on the semantic end of improving indexing, and of addressing these problems. In this talk, I'll try to reinforce just how difficult the problem of semantic search really is, and then show some work we've been doing on acquiring both the number of concepts, and the rich relationships between them, that are needed to make a difference when searching on the tail. I'll outline some intermediate, and related, uses of such knowledge bases that can help us bootstrap towards semantic knowledge access. And finally, I'll mention why we haven't yet been concentrating on parsing (and why others should), and, maybe, question answering and the end of search.

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Resolving Lexical Ambiguities in Folksonomy Based Search Systems through Common Sense and Personalization

Mohammad Nauman[1], Shahbaz Khan[2], Muhammad Amin[3], and Fida Hussain[4]

[1] recluze@gmail.com
[2] shazalive@gmail.com
[3] clickforamin@gmail.com
Research Group Security Engineering
Institute of Management Sciences.
[4] fidamsse@gmail.com
City University of Science and Information Technology
Peshawar, Pakistan.

Abstract. Information on Web2.0, generated by users of web based services, is both difficult to organize and organic in nature. Content categorization and search in such situation offers challenging scenarios. The primary means of content categorization in such social services is folksonomy or collaborative tagging. During search in folksonomy, several issues arise due to lexical ambiguities in the way users choose tags to represent content. These are issues of different words representing the same concept, same words representing different concepts and variances in level of expertise of users. Past techniques to address these issues have worked on lexical analysis of term and have thus had only moderate levels of success. We have developed a model in which machine common sense and personalization is used to address these issues. In this paper, we explain our approach in detail, describe a prototype developed for the purpose of demonstrating feasibility of our approach and discuss an effectiveness study conducted to measure the success of our model. The results of the study are analyzed and future directions along this path of research are presented.

Key words: Common Sense, Folksonomy, Search, Web2.0.

1 Introduction

The social web is a collection of services providing user-created content. These are, among others, photo-sharing systems, blogs, wikis and image and map annotation systems. This collection of services is informally termed as Web2.0. Lack of a central organization for this huge amount of information is a significant hurdle that makes searching through Web 2.0 services very difficult. [1]

Categorization in Web2.0 service is based upon tags (or keywords), which make up a user-created organization. This organization of content is termed as folksonomy or more formally collaborative tagging. Tags serve as keywords

attached to a unit of content for the purpose of organization. Due to the reason that users assign tags to content based on their own experience, skill and mental state, several types of ambiguities arise in the categorization. Content retrieval in Web2.0 becomes very difficult in such a situation and several very important pieces of content might not be recalled due to these ambiguities.

Our study focuses on searching techniques for Web2.0 content and addressing the issue of ambiguity in search results. We have proposed a mechanism through which machine common sense can be used to automatically disambiguate tags and return more results which would otherwise be missed by traditional search mechanisms. The second aspect of our model focuses on user personalization in collaboration with machine common sense to increase the relevance of search results based on an individual users' preferences. Unlike some past techniques, our model requires a minimum of effort on the user's part and is thus very effective for system offering services to non-technical users.

The paper is organized as follows: First we describe the problems of lexical ambiguities in folksonomy based systems in detail. Then we discuss some related and background work which is relevant to our proposed model. Section 4 begins with a discussion of our model, describes how machine common sense and personalization can be used for the purpose of disambiguation in folksonomy and describes our model comprehensively. In Section 6 we discuss the effectiveness study conducted. Section 7 includes the results of the study and our thoughts on these results. Finally , we provide a few directions which can be useful in extending our model in the future.

2 Problem Overview

Web 2.0 services deals with huge amount of ever-growing and changing content. These services primarily depend on folksonomy for organization and retrieval of content.

Folksonomy being a very flexible technique also poses some serious drawbacks. The major problem with tagging is that it employs "folk psychology" to textually represent concepts. This problem branches off into two categories, Polysemy (using same word for different concept) and Synonymy (using different words for same concept). These vague variations are encountered due to the difference in inference of different users according to mental constructs such as knowledge and beliefs. To put it simply, this can be the difference of understanding of two or more users and/or different level of understanding of one user at different times. For example a picture of a car's interior can be tagged as "car", "automobile", "steering" or "leather". These problems arise while saving and retrieving of content.

Several strategies have been used to address the issues including those based on synonyms and co-occurrence frequencies. Since all these approaches are based on lexical analysis of terms instead of contextual, they have had only moderate levels of success [2].

Folksonomy is a non-hierarchical and non exclusive ontology. In such knowledge representation techniques, relationships between objects, concepts and other entities are fuzzy and boundaries between them are unclear.

Another problem with folksonomy (which it shares with traditional search systems) is that it does not provide other important sub-processes (facilities) in searching. The user has to examine the results, extract relevant information and take care of reflections and iterations during the search process.

Any search technique targeting folksonomy has to address all these issues. Traditional web search techniques, such as meta-search and/or categorization of contents into hierarchies, cannot be used because of flat ontological structure and loose textual representations. A more effective means of content retrieval might surface if certain non-traditional techniques are used. Our model uses a collaboration of two such techniques: machine common sense and personalization.

3 Related Work

Several techniques have been used for the purpose of solving issues of lexical ambiguities in folksonomy based services. The one closest to our approach of applying machine common sense was proposed in [3] and is called SemKey. It attaches semantics to tags associated with content. The tags are arranged in three relations: *hasAsTopcic, hasAsKind, myOpinionIs*. The user is expected to decide what attribute of the content they're tagging about. The SemKey system also disambiguates tags using WordNet when they're submitted. The issue with SemKey is that it expects users to associate more information with the content than just the tags. The beauty of folksonomy is that the users do not have to learn any formal mechanisms of content arrangement; instead, they can tag content using *freely chosen* words. We believe that whatever the mechanism for solving problems in collaborative tagging systems, this basic freedom should not be sacrificed. Instead, any technique used to address these issues ought to be automatic.

We have identified a technique developed by Liu et al. [4] which uses automated processes for personalization of search results. This basic technique uses search and access history for storing the user profile. The idea behind the approach is this: One user may associate a term, say "apple", with the category "cooking" while another may think of it as a brand. The user's profile and search history can be used to disambiguate the use of terms in such ambiguous cases.

Cat/Term	apple	recipe	pudding	football	soccer	fifa
COOKING	1	0.37	0.37	0	0	0
SOCCER	0	0	0	1	0.37	0.37

Table 1. Example representation of a user profile

User preference is maintained in a user profile matrix of weights, which consists of categories (representing the user's interest) and terms associated with these categories. A larger weight of a term for a category shows that the user normally associates the term with that category. We refer the reader to [4] for details regarding construction of this matrix.

4 Common Sense and Personalization for Folksonomy

Community generated tags are a vast source of information in a Web2.0 service. They are generated by users of the service and are heavily reflective of their own preferences, skill and common sense. This poses some serious problems for search in folksonomy.

We have developed a technique [5, 6] for applying machine common sense on search in folksonomy. The main strength of this technique is that it is based on contextual, not lexical, analysis of terms. The approach is based on query keyword expansion using a common sense knowledge base - the Open Mind Common Sense Project [7] - and a freely available common sense toolkit - ConceptNet[8].

The Open Mind Common Sense Project (OMCS) is a framework developed by Singh [9] for collecting common sense information from the general public using the world wide web as an interface. Since common sense is, by definition, bits of information shared by most people [8], it seems appropriate that everyone should be able to contribute to a common sense knowledge base. OMCS has had a lot of success over the years and has gathered more than 713,000 items of common sense information [10]. Several common sense reasoning tools [8, 11] have been extracted from the OMCS corpus among which ConceptNet [8] is the first. It is composed of more than 250,000 elements of common sense knowledge represented using natural language fragments and has 20 relation-types which include relations such as *PartOf*, *LocationOf*, *MotivationOf* etc. Two types of scores are assigned to each relation – f: number of times the relation occurs in OMCS corpus and i: number of times it was inferred from other fact.

Figure 1 shows an example [5] of concepts and links as used in ConceptNet.

We have identified the lack of contextual information and inference capabilities as the two major problems for search in folksonomy based systems. We believe that machine common sense can be used to address both of these problems. The basic common sense and folksonomy (CS&F) based search technique [5] works through concept expansion and a score function.

The technique expands concepts which are of a user-selected relation-type and have high conceptual similarity to user's search keyword. The value for conceptual similarity is given by:

$$C(x) = f(x) + (10 \cdot i(x)) \tag{1}$$

Search is performed for each expanded concept. Each *result item* may appear as a result item for more than one concepts (along with the associated search engine score S) and for each instance of this appearance, an instance score is calculated using a score function.

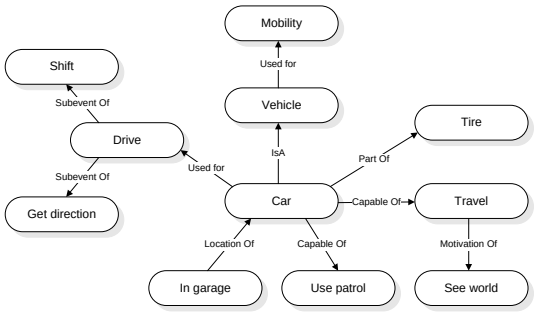


Fig. 1. Concepts related to CAR in ConceptNet

$$inst\_score(x_i) = (G \cdot \sigma(x_i)) + (1 - G) \cdot \gamma(x) \tag{2}$$

The total score of a result item is the sum of all instance scores:

$$score(x) = \sum_{i=1}^{n} inst\_score(x_i) \tag{3}$$

In this technique, two aspects are identified as leading to noise in search results:

- Polysemy: Very similar or even the same words may be used to define completely different concepts. Take for example the brand "Apple" and the fruit apple. Both of these concepts will be considered similar due to the shared lexical representation of the base concepts but for a user they are not similar.
- The score function is rudimentary and only assigns score based on generality and search engine score. Different users may find different results more relevant and therefore the results need some sort of personalization.

One method to address this issue is to use personalized web search for anticipating the user's categories of interest. The expanded concepts and ranked results can be tailored automatically for an individual user based on his/her search and access history. In a past work [6], we have studied this approach in detail.

5 Personalized CS&F Based Search

5.1 Concept Expansion

The personalized technique makes use of the category-term matrix M for concept expansion. Search and access history of a user can be used to personalize the results for individual users. There are two alternatives for using the search history

for concept expansion. One only expands concepts which are in the same category as the original keyword and the other assigns weights to all expanded concepts based on category similarity. The category (Φ_x) associated with a keyword x is that for which the column (T_x) representing the keyword has the highest value.

More precisely, let

Φ_o = Category of the original keyword
T_o = Column representing the original keyword
M_u = Matrix M for user u

then

Φ_o is that row for which

$$M_u(\Phi_o, T_o) = max(M_u(i, T_o)) \tag{4}$$

where i ranges over all rows of matrix M.

For concept expansion:

1. Calculate category for original keyword
2. Expand concepts through ConceptNet
3. Calculate categories for each expanded concept as in 4
4. For each category (k) (returned as result of Step 3), calculate category similarity (Θ) using the function:

$$\Theta_{e_k} = M_u(\Phi_o, T_{e_k}) \tag{5}$$

where

Φ_o is the category of the original keyword and
T_{e_k} is the column representing the concept e_k

5. Calculate *personalized conceptual similarity* by applying category similarity as a weight to the basic conceptual similarity given in 1.

$$C'(e_k) = C(e_k) \cdot \Theta_{e_k} \tag{6}$$

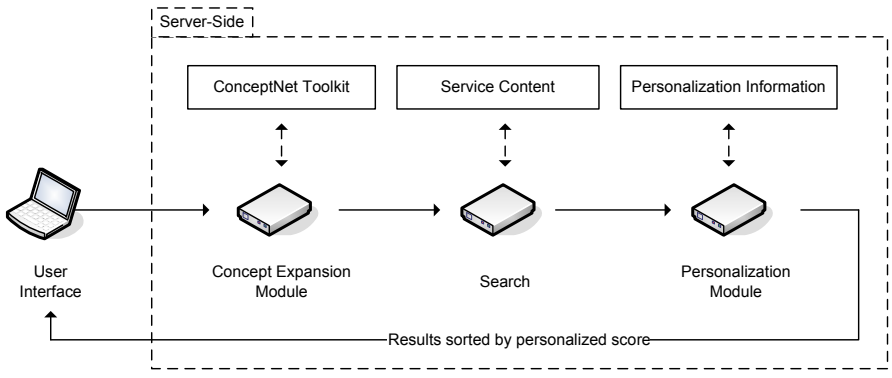


Fig. 2. Basic architecture of common sense and folksonomy based search systems [6]

6. Normalize the conceptual similarity – given as γ':

$$\gamma'(e_k) = \frac{C'(e_k)}{max(C'(e_k))} \tag{7}$$

5.2 Personalized Score Function

Once concepts are expanded, the score of the returned results can be recalculated to give *personalized score*. We note that there are usually more than one tags associated with a single piece of content. Personalized score is designed to take these different tags into account while ranking items. For each of these related tags, category similarity is calculated using the same function as in . We use r for *related* instead of e for *expanded*.

$$\Theta_{r_k}(x) = M_u(\Phi_o, T_{r_k}) \tag{8}$$

Finally, we *personalized score* ($score'$) is calculated as a function of the basic *score* and Θ_{r_k} given as:

$$score'(x) = \frac{score(x) + \sum_{k=1}^{n} \Theta_{r_k}(x)}{n+2} \tag{9}$$

Θ_{r_k} gives preference to those documents which are tagged with keywords belonging to the same category as the original search keyword. It also ensures that if a document is tagged with irrelevant keywords – say, the name of the user – the score is penalized.

5.3 Algorithm

Working of personalized web search in common sense and folksonomy based search systems is summarized in the algorithm described in Figure 3.

6 Effectiveness Study

A prototype of the proposed model showed the feasibility of constructing a search system based on the proposed model. To measure the effectiveness of the approach and the prototype, we conducted an effectiveness study.

The study aimed to gather quantitative results regarding the effectiveness of the search model. Since the intended audience of the system is the general public and not computer science experts, a questionnaire was developed which could be easily filled by non-experts and would provide us with quantitative results for drawing conclusions about the new technique. The sample size of the survey included 8 individuals from different levels of computer expertise. Data was collected through the use of a questionnaire hand-delivered to the participants. The questionnaires were filled by the participants while using the prototype and were returned in the same sitting. The important questions are given below along with their question numbers as given in the questionnaire:

Get search $keyword$ from user
Φ_o := getCategory($keyword$)
e := expandConcepts($keyword$)
$exConcepts$:= {}
for each e_k in e
    Φ_{e_k} := getCategory(e_k)
    $\Theta_{e_k} = M_u(\Phi_o, T_{e_k})$
    $C'(e_k) = C(e_k) \cdot \Theta_{e_k}$
    $\gamma'(e_k) = \frac{C'(e_k)}{max(C'(e_k))}$
    $exConcepts$.add(e_k)
for each e_k in $exConcepts$
    $results$:= performSearch(e_k)
    for each r_i in $results$
        inst_score(r_i) := $G \cdot \sigma(r_i) + (1 - G) \cdot \gamma'(e_k)$
        addtoInstScores($inst\_score(r_i)$)
$scores[x] := \sum_{i=1}^{n} inst\_score(x_i)$
for each x in $scores$
    $relTags$:= getRelatedTags(x)
    for each r_k in $relTags$
        Θ_{r_k} := getCategorySimilarity(Φ_o, r_k)
    $scores'[x] := \frac{score[x] + \sum_{k=1}^{n} \Theta_{r_k}}{n+2}$
Sort by $scores'$ descending

Fig. 3. Algorithm

4. How much do you know about Web2.0 and Tags based web systems?
5. How easy to use, do you think, is the interface of the prototype?
6. Do you understand the concept of relations between concepts?
7. Do you find the concept of generality given in the prototype easy to understand?
8. Are you comfortable with the search system saving your search and/or access history?
9. Do you understand the problem of searching for content tagged with synonymous and/or polysemous words?
10. Have you ever experienced the above mentioned problems while searching for content on the web?
11. Do you understand the concept of common sense, specifically relating different concepts together?
12. Do you understand the technique used in this search system?
13. How would you rate the relevance of the search results to your query?
14. How would you rate the relevance of the search results to your intended target content?
15. Do you think the search results accurately depict your preference in ambiguous words?

16. Were there any irrelevant items in the returned results?
17. How would you rate the overall usefulness of the search results?

7 Results and Analysis

The results to the questionnaire are summarized in Table 2. Here, we briefly analyze the pattern in the results.

Persons	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	Answer Description
Questions									
4	3	1	2	1	1	3	1	2	1–4: Little knowledge – detailed knowledge
5	1	3	2	2	3	1	2	1	1–4: Easy – difficult
6	2	2	2	1	2	3	2	2	1–3: No understanding – complete understanding
7	2	3	3	2	3	2	3	1	1–4: Easy – difficult
8	1	1	1	2	2	2	3	1	1–3: Comfortable – not comfortable
9	2	2	1	2	3	1	3	2	1–3: Complete understanding – no understanding
10	2	3	2	3	3	1	3	3	1–3: Have experienced problems – have not
11	3	2	2	3	2	1	2	2	1–3: Clear – confusing
12	3	3	2	3	3	2	3	2	1–3: Understand – don't understand
13	2	3	2	4	2	2	3	2	1–4: Relevant – not relevant
14	2	3	3	3	2	2	3	2	1–4: Relevant – not relevant
15	3	2	2	3	3	2	3	3	1–3: Personalized – not personalized
16	2	3	2	3	2	2	3	3	1–3: No irrelevant results – many irrelevant results
17	2	3	2	3	2	1	4	2	1–4: Useful – not useful

Table 2. Summary of Results of the Effectiveness Study

Some of the important points to note in these results are the following:

- Answers to Question 7 – "Do you find the concept of *generality* given in the prototype easy to understand?" – suggest that users of the prototype found the concept of generality difficult to grasp. It seems therefore that this variable should be automatically adjusted by any system implementing our model instead of leaving it up to the users to pick its level. We do not think it would be appropriate to embed the value of generality in the model itself because it depends on the context of search and should be left customizable to the individual implementation.
- Several users found the graphical user interface of the prototype a little difficult to understand. While it was not our primary goal to make the prototype easy-to-use, an easier front-end might have shown better results in the effectiveness study. However, this finding does not affect the actual model.
- Many participants, in response to Question 12 – "Do you understand the technique used in this search system?" – answered that they did not understand the technique used in our prototype. In social networks, it is of

immense importance that the users understand the underlying reasoning mechanisms as much as possible. It helps them use the network more effectively. Any service implementing our model needs to put some efforts in educating the users about the working of intelligent search to enable them to utilize it more effectively.
- The issue of noise, according to responses to Question 16 – "Were there any irrelevant items in the returned results?" – was not effectively resolved by our prototype. We believe that the reason for this is that the participants of the survey did not have a detailed *user profile* in our prototype's database. Personalization depends heavily on this profile but it takes a little while to create an effective corpus for each individual user. We believe that with use, the effectiveness of the personalization module would increase. However, a proof of this cannot be obtained without an extensive study conducted over a long period of time on a larger number of constant users.
- It is evident from the answers to Question 8 – "Are you comfortable with the search system saving your search and/or access history?" – that privacy is not an issue in users of our geographical proximity. There seems to be a need to educate the users about privacy being an important issue which should be taken more seriously. However, it is an issue outside the scope of this research and is not our primary concern.

8 Future Work

Search results are, by nature, difficult to analyze and require users' subjective analysis. While the initial tests with the proposed technique of using personalized web search with common sense and folksonomy based search systems has shown positive results, a more detailed usability study is necessary to study the effectiveness of the technique for different users. Future work along this path aims to conduct detailed experimental studies using this new technique using real-world folksonomy based applications such as flickr [12] and Wordpress [13] etc. A comparison with other search techniques is also necessary to determine the full effectiveness of the proposed technique.

This technique still utilizes only three sources of information: tags, user profile and search engine's score. While these are the primary source of content's meta information in a folksonomy based service, other ranking variables, such as links to related content, are still not utilized. This technique may benefit from a more thorough study on how content clustering and relevance feedback techniques may be incorporated in this approach for better ranking of search results.

9 Conclusions

The information overload caused by the coming of user-created data on Web2.0 can only be addressed by utilizing all available resources for search and organization. User created organization of data has produced acceptable levels of results but still has problems because of variances in users creating this organization. A

possible solution to this problem is the application of machine common sense to the problem of search. In this research work we have outlined a framework for using the Open Mind Common Sense project to address the issue. This is done through the application of ConceptNet, a freely available tool kit for machine common sense on folksonomy.

The model, proposed in this research work, uses common sense and folksonomy and offers a different approach towards addressing the issue of search in social networks. However, it also leads to some noise in search results due to polysemy. To overcome this issue of noise, we enhanced the basic technique using a search results personalization technique. A detailed description of a modified approach for utilizing a personalized web search technique for returning more relevant search results in a CS&F based search system is described.

An effectiveness study was developed for measuring the success of the proposed approach. Different users, from different technical and non-technical backgrounds were asked to evaluate the prototype and give their opinions through a questionnaire. The results were collected and analyzed to measure the effectiveness of the prototype. The results have shown that while the prototype was able to demonstrate better recall, it has been prone to some noise in the results. This might be due to the reason that the participants of the study did not have an extensive search and access history in the system and the system was thus unable to perform personalization as effectively as it could have.

References

1. Golder, S., Huberman, B.: The Structure of Collaborative Tagging Systems. Arxiv preprint cs.DL/0508082 (2005)
2. Lieberman, H., Liu, H.: Adaptive Linking between Text and Photos Using Common Sense Reasoning. Conference on Adaptive Hypermedia and Adaptive Web Systems (2002)
3. Marchetti, A., Tesconi, M., Ronzano, F., Rosella, M., Minutoli, S.: SemKey: A Semantic Collaborative Tagging System. Proceedings of 16th International World Wide Web Conference, WWW2007 (2007)
4. Liu, F., Yu, C., Meng, W.: Personalized Web Search by Mapping User Queries to Categories. Proceedings of the eleventh international conference on Information and knowledge management (2002) 558–565
5. Nauman, M., Hussain, F.: Common Sense and Folksonomy: Engineering an Intelligent Search System. In: Proceedings of ICIET'07: International Conference on Information and Emerging Technologies, IEEE (2007)
6. Nauman, M., Khan, S.: Using Personalized Web Search for Enhancing Common Sense and Folksonomy Based Intelligent Search Systems. In: Proceedings of WI'07: IEEE/WIC/ACM International Conference on Web Intelligence. (November 2007)
7. Singh, P., Lin, T., Mueller, E., Lim, G., Perkins, T., Zhu, W.: Open Mind Common Sense: Knowledge acquisition from the general public. Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (2002)
8. Liu, H., Singh, P.: ConceptNet: A Practical Commonsense Reasoning Tool-Kit. BT Technology Journal **22**(4) (2004)

9. Singh, P.: The public acquisition of commonsense knowledge. Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access (2002)
10. OMCS: The Open Mind Common Sence Project. Accessed at: http://openmind. media. mit. edu/
11. Singh, P., Williams, W.: LifeNet: a propositional model of ordinary human activity. Proceedings of the Workshop on Distributed and Collaborative Knowledge Capture (DC-KCAP) at K-CAP (2003)
12. Flickr: About flickr. http://www. flickr. com/ about/ (Retrieved on February 24, 2007)
13. WordPress: Wordpress.com. Accessed at: http://www. wordpress. com/ (Retrieved on November 13, 2007)

Integration of Semantic, Metadata and Image search engines with a text search engine for patent retrieval

Joan Codina[1], Emanuele Pianta[2], Stefanos Vrochidis[3], Symeon Papadopoulos[3]

[1] Fundació Barcelona Media, Ocata 1, 08003 Barcelona Spain
[2] Fondazione Bruno Kessler, via Sommrive 18 38100 Trento, Italy
[3] Aristotle University of Thessaloniki, Thessaloniki, Greece
Joan.codina@barcelonamedia.org, pianta@fbk.eu, {stefanos, papadop}@iti.gr

Abstract. The combination of different search techniques can improve the results given by each one. In the ongoing R&D project PATExpert[1], four different search techniques are combined to perform a patent search. These techniques are: metadata search, keyword-based search, semantic search and image search. In this paper we propose a general architecture based on web services where each tool works in its own domain and provides a set of basic functionalities to perform the retrieval. To be able to combine the results from the four search engines, these must be fuzzy (using a membership function or similarity grade). We focus on how the fuzzy results can be obtained from each technique, and how they can then be combined. This combination must take into account the query, the similarity of the patent to each part of the query, and the confidence on the technique

Keywords: Patent search, semantic search, image search, multimodal, user feedback, similarity search, fuzzy.

1 Introduction

In the field of information retrieval there is an increasing interest in patent retrieval. The legal style of patent documents, where text is obfuscated deliberately and very specific vocabulary is combined with very generic terms, makes patent retrieval a challenging task. Because of the legal implications of a patent invalidity search, it is crucial to get a high recall rate even at the expenses of losing precision. Expert users perform long Boolean queries (having from 5 to 30 statements) where each concept they are searching for is expressed by AND's and OR's of possible synonyms [1].

The use of semantic search allows searching for concepts, instead of words, and for relationships between them. However, semantic search has still to face a number of challenges in order to become the backbone of a search engine. First, it needs an ontology that copes with all the relevant terms. Although several ontologies exist, they do not cover most of the very specific terms found in patents, and the generic terms provide only little information. As illustration, consider the following sentence

1 PATExpert is partially funded by the European Commission in its Sixth Framework Programme (FP6 028116).

from a patent claim of a compact disc reader: "An optical head device for use in combination with an optical source for generating an optical beam along an optical axis from a source point and comprising a lens system". Here, words like "head", "device" or "source" are combined with more specific ones like "axis" or "lens". Additionally, many of these words are combined in multiwords such as "optical head device", "optical axis" or "source point" which may not exist in the ontology.

Another problem arises when disambiguating terms since the most common choices may not apply to patent documents, hence broad-coverage parsers like Minipar [2] may take the wrong decisions. As an example, consider the word "means", which can be either a verb or a noun. In natural language the most common choice would be to consider it a verb. However, this may not be true in patent documents of a given domain, where "means" is often "a noun denoting an instrumentality for accomplishing some end" as in "a transfer means mounted on the frame and operatively associated with the tool means for moving the tool means...".

There is also a complexity problem. A patent can contain thousands of triples, each one composed by a noun, a verb and an object. Triples can be related between them when the same object appears in two triples. For example, the pair "we live in a house", "the house is white" can be equivalent to "we live in a white house" if we know that the house of the first and second triples are the same. Ideally a patent could be represented by a single graph made of thousands of related triples.

In practice, however, all triples and relationships cannot always be determined and one gets a set of unconnected sub-graphs which may fall short to make use of the proper content representation.

Most patents are impossible to understand without the help of drawings. Images are a source of valuable information during search, but can also be a source of confusion since the same object can be drawn in so many different ways. Image search based on image content (and not captions or surrounding text) is still an open research problem, and though results are encouraging they are not reliable enough.

In short, semantic and image search techniques are promising but not yet mature enough to rely exclusively on them. On the other hand, expert patent users feel confident with traditional (but often too short-sighted) text search techniques. A multimodal patent search system may help to circumvent the weakness of the individual techniques. This multimodality characteristic is one of the prominent features in the PATExpert [3] retrieval module.

PATExpert is a European project devoted to the use of linguistic and image analysis tools for patent processing. This includes patent search, but also paraphrasing, summarization, classification, valuing and multilingual search. PATExpert advocates the specification of patent material in terms of techniques that operate on semantic representations rather than on textual ones.

This paper focuses on the search and retrieval module of PATExpert where a multimodal search engine is built from four individual search engines: (1) a metadata information search engine, (2) a keyword-based retrieval engine, (3) a semantic search engine, and (4) an image search engine. The first two allow for keyword-based text search and for metadata search. They are mainly based on classical information retrieval techniques. The third one, namely the semantic search engine, allows for the search of patent documents according to content criteria (e.g., material of which an object is made, availability of a component with a specific functionality, purpose of a

component, etc.). Finally, the image search engine allows for the search of patent material with images similar to images or features provided by the user. The objective of the multimodal search is to improve the performance from the classical retrieval techniques with the inclusion of the results from the advanced search methodologies.

The remainder of the paper is organized as follows. Section 2 first presents the architecture of the multimodal search system and then describes how the individual search modules can be integrated. Section 3 discusses how the results are processed and combined with each other. Finally, conclusions and future directions are given in Section 4.

2 Multimodal Search Engine

As shown in Fig. 1, the multimodal search engine is built upon four independent search engines covering different user needs related to patent search: (1) metadata search, (2) keyword-based search, (3) semantic-criteria search, and (4) image-related search.

Apart from the search engines, the system facilitates a management tool for queries and retrieved patent objects (results); here referred to as *merger*. The merger splits the user query into sub-queries and distributes them to the different search engines. The search engines are independent and use very different approaches to find results and determine scores. Nonetheless, all of the search engines match and retrieve patent objects on the basis of the similarity of their query representation (i.e., similarity-based retrieval). These results are then properly combined by the merger and the final ranked results presented to the user. At this stage, the original query of the user is iteratively refined and adjusted, based on the feedback provided by the user.

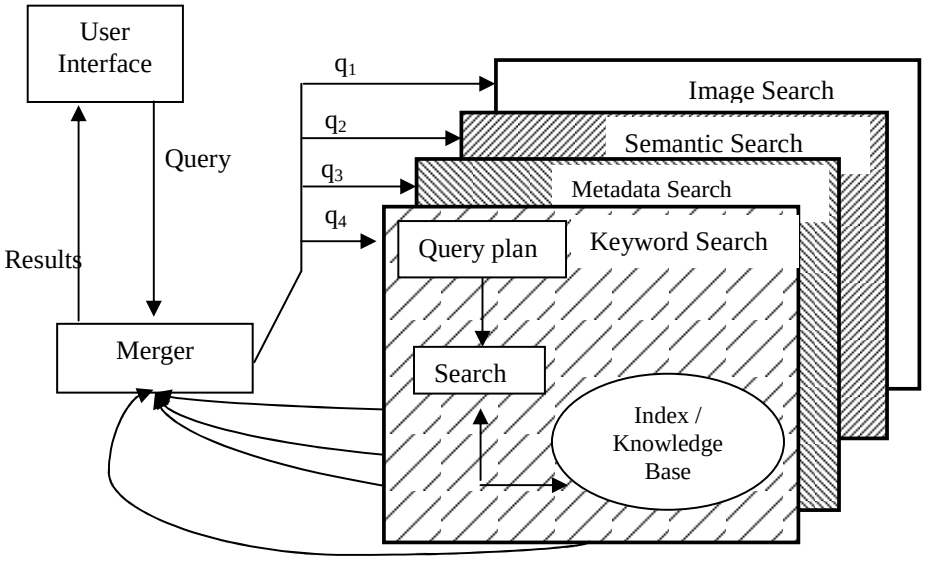


Fig. 1. Multimodal Search Engine

In the following paragraphs, we briefly describe the different search engines, and leave the discussion of the merger module for the next section.

2.1 Metadata Search

Queries posed to the metadata retrieval system relate to the attribute values of the patents (e.g. the name of the inventor, the publication date or the IPC[2] classification code). The metadata query is intended to offer the possibility to perform a query focused on the contents of a database field.

The standard approach for searching metadata is to perform an exact query based on a Boolean search constraint specified by the user (e.g. "pubdate > 01/01/2000 AND inventor = Y"). The returned results are the set of documents which completely fulfill the constraints. Thus, the result is crisp in the sense that a document either satisfies the query or it does not. This is quite a limitation since it does not allow for partial matching. Moreover, there is no fuzziness or ranking as known from classic information retrieval.

Fuzziness can be introduced in the constraints as well as in the Boolean operators. Fuzzy comparators like >~, <~ , ~ , and !~ are included. As an example consider the query "*pubdate >~01/01/2000*". This fuzzy operator will return all records where "*pubdate > 01/01/20000-FuzzyMargin*". The ones after *01/01/2000* will have a membership grade (ranking) of 1.0, while the documents within the *FuzzyMargin* range are assigned a decreasing membership. The size of the fuzzy margin is user defined.

Fuzziness has also been introduced in the Boolean operators. This means that the user may choose to perform an ORF or ANDF instead of a regular OR/AND. The difference is that the fuzzy operators will give a greater membership if both conditions are true than if only one is true. The Boolean operators for OR/AND over fuzzy values will become the maximum/minimum of the membership grades. The fuzzy operators are the product T-norm (AND) and probabilistic sum for the S-norm (OR).

The drawback of having fuzzy operators is that the FAND becomes an OR when translated to the corresponding SQL query, and then it needs to compute the membership grade for each result.

In the next sample, we show how a fuzzy query is transformed to get a list of patents with the membership:

The Original Query:

```
(appcountry in ('ES', 'FR')) ORF pubdate >~1/2/2002
```

will generate an sql statement in two steps; in the first step the similarity for each condition present in the query is computed, while in the second, the global similarity applying the fuzzy formulas is computed.

```
SELECT id, sim1+sim2-sim1*sim2
FROM
```

2 IPC (International Patent Classification) is a hierarchical classification system providing a common classification for patents.

```
(
 SELECT DISTINCT Patent_id ,
 CASE
    WHEN patents.pubdate>'1/1/2005' THEN 1.0
    WHEN patents.pubdate<'1/1/2004' THEN 0.0
    ELSE  (patents.pubdate-'1/1/2004')/365.0
 END as sim1 ,
 CASE
    WHEN appcountry in ('ES', 'FR') THEN 1.0
    ELSE 0.0
 END as sim2
 FROM patents
 WHERE  (patents.pubdate>'1/1/2004')
    OR appcountry IN   ('ES', 'FR')
)
```

2.2 Keyword-based Search

The majority of the search engines available for patent retrieval are keyword-based. Some include a query pre-processing procedure allowing for the use of wildcards, weighting of query terms, query expansion by using thesaurus relations, proximity search, etc. The vector model is one of the most widely used search techniques as it gives very good results with a rather simple model.

In PATExpert we use Lucene [4], with some adaptations to deal with certain idiosyncratic aspects of patents (such as recognition of patent numbers or IPC codes).

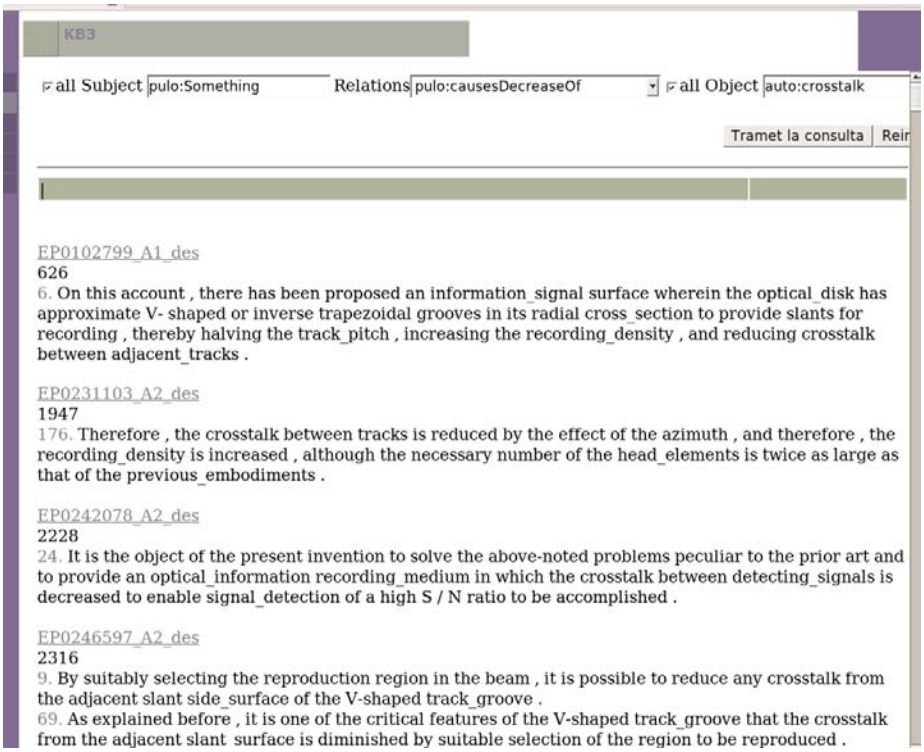
2.3 Semantic Search

State of the art patent processing makes use of the semantic-web formalism based on text labels to extract semantic information. In PATExpert patent documents are first processed with general purpose language processing tools, such as TextPro [5], and MiniPar [2], which carry out PoS tagging, multiword recognition, lemmatization, and dependency parsing. Linguistic annotation are then exploited to recognize frame instances (see FrameNet [6]), and finally concepts and triples.

An ontological framework is needed to work with concepts. In PATExpert[3], the Core Upper Level Ontology (SUMO) with mappings to Wordnet has been employed and several ontologies have been developed: a Patent Upper Level Ontology (PULO), and domain ontologies with concepts of the specific technical fields. As patents are documents where new concepts are forged, PATExpert has the ability to automatically expand existing ontologies with new concepts (marked as *auto*) [7].

In triple-based semantic search the user specifies a target triple by selecting a relation and two concepts filling the subject and object role of the relation. The relation is chosen from a limited list (few tens) of significant relations recognized by the system (e.g. sumo:hasPart, pulo:moves, pulo:hasSubstance). Subject and object are selected from a much larger list (over 30.000) of domain specific concepts. A wizard helps the user to select the KB-concepts matching the concepts he/she has in mind.

In its basic functionality, the search engine will select all sentences in the corpus



3 A detail description of content extraction and developed ontologies in PATExpert can be found in [3].

containing the target triple, whatever the linguistic form in which the triple is expressed (e.g. "An optical head has a prism" or "the prism included in the optical head" or the "prism of the optical head"). However, the user can also choose to expand the search by instructing the system to consider also concepts related to the object or subject of the target relation. For instance instead of searching only for triples having as object the "prism" concept, the user can search also for all kind of more specific "prisms" known by the system according to the domain ontology (hyponyms), e.g. "trapezoidal_prism", "anamorphic_prism", etc. Alternatively, the user can search for concepts generalizing the concept of "prism", like "optical_component" (hypernyms).

If the user chooses one expanded query, the retrieved sentences can be ordered according to their similarity to the base (non-expanded) target triple. The semantic distance between the target and the retrieved triples is measured according to the distance of the retrieved concepts (hypernyms and/or hypopnyms) from the target concepts according to the domain ontology (e.g. "trapezoidal_prism" is closer to "optical_component" than to "engineering_component"). Assuming that the similarity of two equal triples is 1, we multiply this value by a factor $a<1$ for each step down the hyponyms hierarchy, and by a factor $b < a < 1$ for each step up in the hypernyms chain. In this way we obtain a set of patents having a given concept or triple with a similarity value b.

The result of a sample search using semantics is shown in Fig. 2.

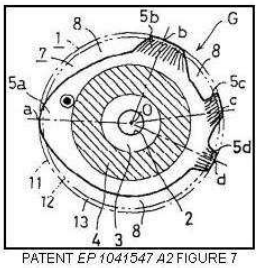
2.4 Image Search

Apart from the textual information, patent documents usually include a number of figures which are descriptive of the overall content. The most important characteristic of these figures is that they are black and white binary images as they mainly represent technical drawings, charts and mathematical relationships. Under these circumstances, image search could be rather challenging as color information, which is one of the main characteristics that content based retrieval relies on, cannot be considered.

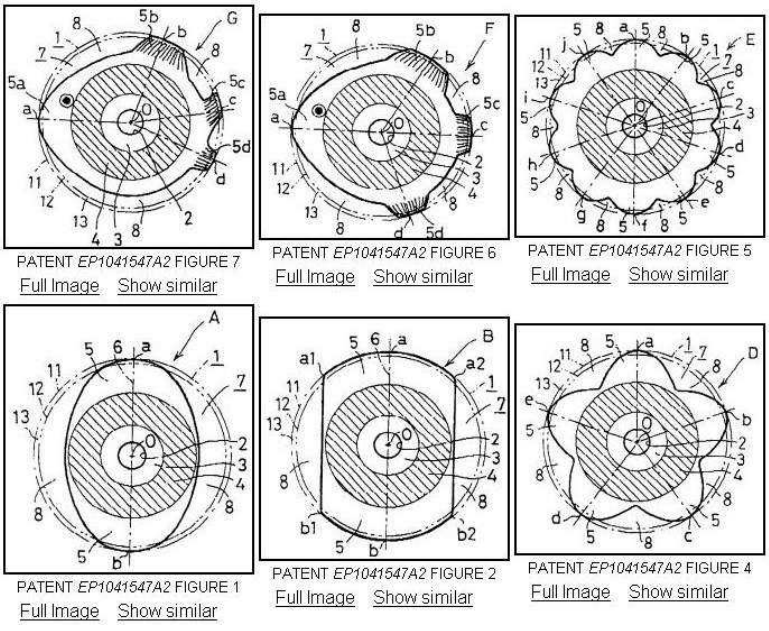
Taking into account the information above, the image similarity search module for patents was implemented in order to allow efficient image retrieval based on visual similarity. This module requires an off line pre-processing step in order to run on-line and provide the desirable results. The whole procedure is described below.

The first step in the off line processing is to detect and extract the pages of the patent that include figures as raster images. Secondly, orientation detection takes place. Connected components regions are extracted from the page (by use of the 8-neighborhood property) and the direction is identified along which the higher number of regions that lie on the same line [8]. Subsequently, individual figures need to be separated as normally such a page may contain more than one figure. The figure separation can be done automatically with an acceptable error[4] while it can be also

[4] The reason for accepting error at this stage has to do with the fact that the figures are placed randomly in the page, some times really close to each other and the labels can be handwritten. In such cases the borders between different figures are very hard to specify.



PATENT *EP 1041547 A2* FIGURE 7



PATENT *EP1041547A2* FIGURE 7 Full Image Show similar

PATENT *EP1041547A2* FIGURE 6 Full Image Show similar

PATENT *EP1041547A2* FIGURE 5 Full Image Show similar

PATENT *EP1041547A2* FIGURE 1 Full Image Show similar

PATENT *EP1041547A2* FIGURE 2 Full Image Show similar

PATENT *EP1041547A2* FIGURE 4 Full Image Show similar

Fig. 2. Examples of retrieved results by image search

manually supported to improve the results. Finally, the extracted images are stored and indexed in a database.

At this stage, the feature extraction takes place. The employed feature extraction method relies on the computation of the Adaptive Hierarchical Geometric Centroids proposed in [9]. The reason for selecting these features was the fact that the majority of the figures are binary so the only useful information could be extracted from the geometry and the shape of the depicted objects.

Assuming that the origin of a 2-d space lies in the first level geometric centroid, we split the image plane into four disjoint quadrants, compute for each one its geometric centroid, and divide it into 4 sub-quadrants in an analogous way. This is recursively performed up to some number of levels *n*. Note that after n levels, there are 4^n disjoint

unequal rectangle areas, i.e., 4^n possible partitions that can be classified in pattern groups. As the feature vector of a binary patent image we use the *n* histograms -one for each level- of the partitions. Consequently, the resulting vector dimension is low in comparison to most standard techniques whose feature vector dimension may reach tens of thousands. Based on this method the feature vectors are extracted and stored in a database in order to be online accessible.

During the online search, we compute the L1 distances of the feature vector of the given image query against every other feature vector from the database. The smaller the distance is between the feature vectors of two images, the more common visual characteristics they share. One specific distance threshold is set in order to distinguish relevant from irrelevant images. High threshold values could result in many results at low precision levels, while lower ones could result in very small or even empty sets of images. For this reason, the threshold was empirically tuned in order to optimize the performance.

A use case for the image retrieval module is presented in Fig. 3. In this, the user selected a figure with cyclic characteristics. The results depicted in Fig. 3 provide an indication of the high relevance achieved by the module.

In order to evaluate the derived results, recall and precision metrics were calculated. The experiments were conducted in a database of 1400 images extracted by European Patents and a 100 of them were arbitrarily chosen to perform the image similarity for each one. By tuning the distance threshold that compromises between these complementary metrics leads to 77% Recall for 49% Precision.

3 Merger

PATExpert does not contain a single language for doing all four kinds of queries, when the user specifies a query, she uses different syntax depending on the search engine that she is writing the query for. Then the user can combine with Boolean or Fuzzy-Boolean operators some queries written for the different query engines, to build a search.

The merger is responsible for the distribution of sub-queries and combines the results back together to produce a single list of results. The query dispatching and collection of results does not have any special feature; the challenge remains on how to combine them. This is done within a fuzzy framework.

PATExpert also provides a similarity search. The similarity search could be the common interface for querying: The user introduces a text she is looking for, and the system returns a list of patents that are similar. This simple approach is not as simple as it seems, first because the task is intrinsically difficult, and from the point of view of the user (and even more: the expert users) there is no control on what the system is doing, or how to control the search process to be sure that the patents retrieved are the correct set. The expert user needs to be able to monitor the process to be sure that the list of patents contains all the patents that could lead to an infringement or invalidation of the patent. For this reason PATExpert provides this functionality in two steps: During the first step the system receives a text of a patent (or portions of it) and produces as output a query, that when executed would provide patents similar to

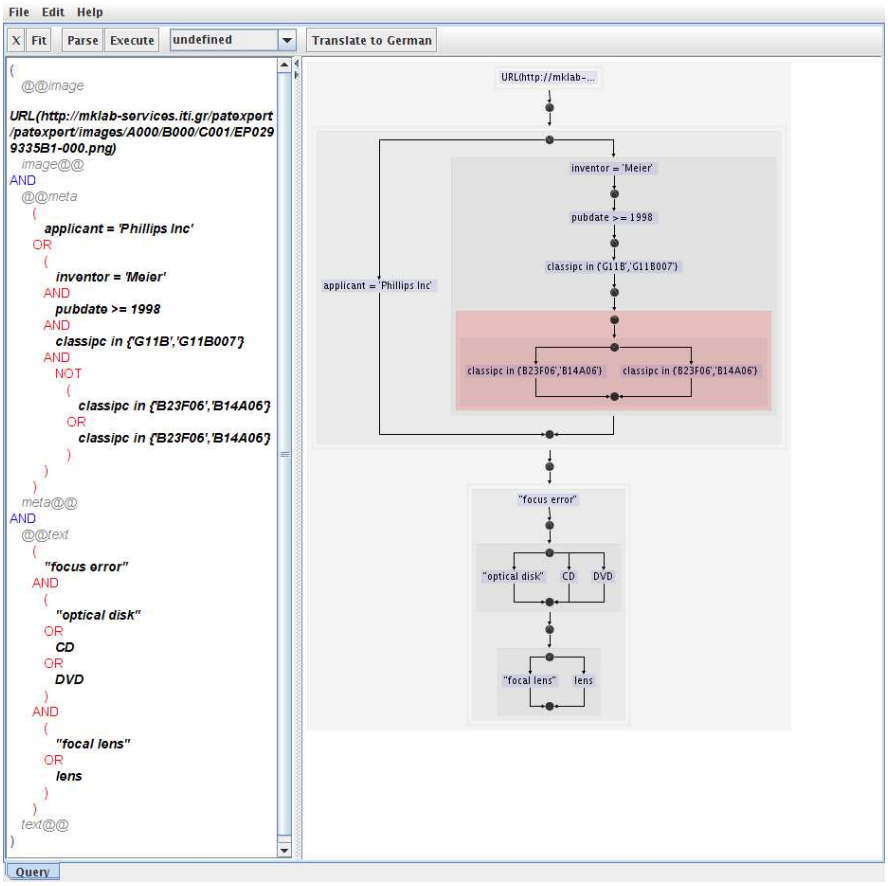


Fig. 3. Sample query as introduced in the user interface: The user specifies a combination of different searches to be performed by the different search engines.

the one provided by the user. The way that the query is generated from the text is out of the scope of this paper.

In the IR literature, the paradigm of a *broker* exists that distributes a query to different search engines, sends the same query to one or all of them, and then merges the result. Usually a broker is associated to distributed systems and the task of the broker is to send the query to the appropriate node that may have the data to answer the question. In PATExpert the role of the merger is different: First it does not send the same query to all the search engines, as each portion of a query is only solved by one search module and secondly when the merger gets the results back, it has to merge them, taking into account the Fuzzy-Boolean operators that combined the original sub-queries. For this reason the "merger" is not called "broker" in PATExpert.

3.1 An overview

We illustrate the merger function through an example. Let us consider the user query as written in the interface, shown in Fig. 3

This query is composed of several sub-queries: the first one is an image query the second one a metadata query and the third one a keyword-based. Each sub-query is sent to the corresponding search engine that will return an ordered list of (patent_id, similarity). The similarity of patent *p* in query *q* is denoted as:

$$\mathrm{Sim}(q,p) = s_{q,p} \in [0,1], \quad \textbf{(1)}$$

and represents the extent to which the patent *p* fulfills query *q*. The larger the grade is, the better the match. In particular, a grade of 1 represents a perfect match while, on the other hand, a grade of 0 implies no match at all. The merger must take these lists and merge them to produce a single list, where for each patent_id the overall similarity grade is computed from the sub-queries similarities. The strategy used to merge the list results takes into account the operator, the similarity and also the degree of confidence on each search technique.

3.2 Combining the results

The combination of results is done within a fuzzy framework in order to overcome the shortcomings of the strict Boolean model (see Section 2.1). This fuzzy framework encompasses most common fuzzy approaches such as fuzzy set models [10] or the extended Boolean models with weights associated to the index terms [11].

Within the classical fuzzy set model, we could define the overall similarity of a patent *p* under the disjunctive and conjunctive queries as

$$\mathrm{Sim}(q_{\mathrm{or}}, p) = \mathrm{Sim}(q_1 \vee q_2 \vee \cdots \vee q_m, p) = max\left(s_{q_1,p}, s_{q_2,p}, \cdots, s_{q_m,p}\right) \quad \textbf{(2)}$$

$$\mathrm{Sim}(q_{\mathrm{and}}, p) = \mathrm{Sim}(q_1 \wedge q_2 \wedge \cdots \wedge q_m, p) = min\left(s_{q_1,p}, s_{q_2,p}, \cdots, s_{q_m,p}\right) \quad \textbf{(3)}$$

However, the use of maximum (or minimum) does not reflect any change in the similarity when values different from the maximum (or minimum) change without becoming bigger (or smaller) than the maximum (or minimum). As an illustration, consider the similarities in Table 1.

Table1: sample similarities to illustrate the differences between operators

Similarity $s_{q,p}$	q_1	q_2
p_1	0.8	0.7
p_2	0.81	0.3

Using the OR boolean operator we get

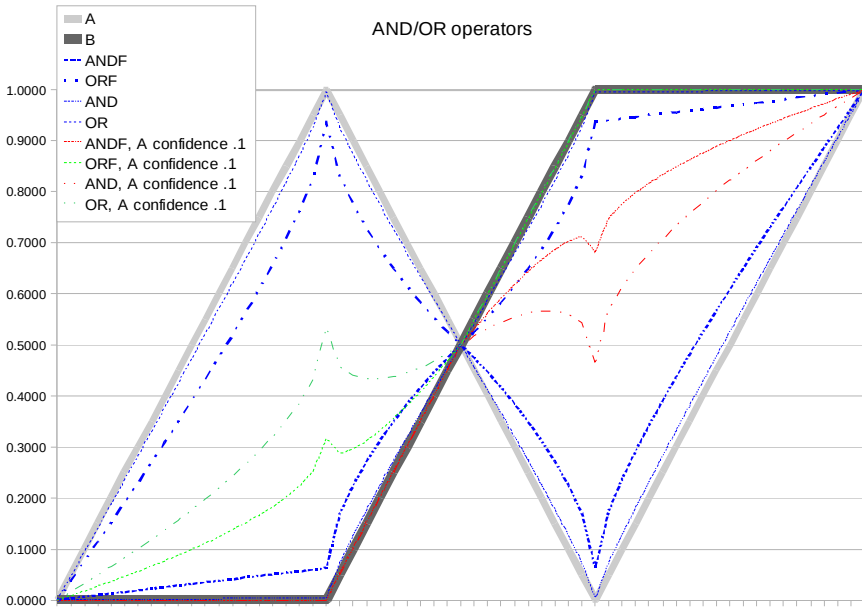


Fig. 4. Graphical representation of the weighted power-mean averaging operators operating as Boolean or Fuzzy and with different confidence levels of the fuzzy variable A. The graph shows for each operator the fuzzy result of the operators depending on the Fuzzy weight of A and B. The operators are the Boolean (AND OR) and fuzzy (ANDF, ORF) . When A has lower confidence level, the result is closer to B.

$$q_{\text{or}} = q_1 \vee q_2; \tag{4}$$

$$S_{q_{\text{or}},1} = max(0.8,0.7) = 0.8; \quad S_{q_{\text{or}},2} = max(0.81,0.3) = 0.81$$

which favors the second one even if one its queries has a very low score, and also despite the fact that both queries scores in the first one are high. To prevent this kind of behaviors, one may use alternatively T-norms (triangular norms) to compute the disjunctive with the corresponding T-conorms for the conjuctive operator [12], [13] but these operators include a non intuitive behavior: $AND(x,x) < x \quad for\ x < 1$. So the use of T-norms was discarded.

In addition to combining the partial similarities, the merger needs also to deal with a belief or confidence factor associated to each result set. For this purpose, a confidence factor (obtained during the training phase) is assigned to each of the search techniques. This process, however, is out of the scope of the paper.

To deal with Boolean/fuzzy operators in a similar way together with the confidence factor, and after testing the result of different fuzzy paradigms, we adopted the weighted power-mean averaging operators [14]. These operators include some nice properties from the user point of view that can be seen in Fig. 4: When both variables have the same value the result coincides with them and also $AND(0,x) \geq 0 \quad for\ x \in [0,1]$, which means that a change on *x*, influences the result. The formula for the *AND* operator is

$$\begin{gathered} q_{\text{and}} = q_1 \wedge q_2 \wedge \ldots q_{\text{m}}; \\ q_1 < q_2 < \ldots < q_{\text{m}} \\ S_{q_{and}} = \left[\frac{1}{m^2} \sum_{k=1}^{m} (u_k \times e_k^r \right]^{\frac{1}{r}} \\ u_k = (m-k)^2 - (m-k-1)^2 \end{gathered} \tag{5}$$

Where the m terms to compute the *AND* operator, are sorted in decreasing order, the weights are then assigned accordingly to that order before being averaged. The different values in the power *r* will make the operator Boolean (value of 0.0001) or fuzzy (when 0.5).

The query may be seen as a tree of AND's/OR's which can be evaluated bottom-up. Each Boolean operator must return a list of documents with the similarity for each document and a belief factor for the full list. This belief is computed as a weighted sum of the beliefs, where the weight is the own belief, giving the formula:

$$w_{\text{and/or}} = \frac{\sum w^2}{\sum w} \tag{6}$$

The weighted operator is computed using the following formula:

$$\begin{gathered} q_{\text{and}} = w_1 q_1 \wedge w_2 q_2 \wedge \ldots w_2 q_{\text{m}}; \\ where \quad q_1 < q_2 < \ldots < q_{\text{m}} \\ and \sum_{k=1}^{m} w_k = 1 \\ S_{q_{and}} = \left[\frac{1}{m^2} \sum_{k=1}^{m} (u_k \times e_k^r \right]^{\frac{1}{r}} \\ u_k = \left(m \sum_{l=k}^{m} w_l \right)^2 - \left(m \sum_{l=k+1}^{m} w_l \right)^2 \end{gathered} \tag{7}$$

The main disadvantage of this approach is that the membership grades (q_i) need to be sorted before performing the computations, and then the weights need to be computed accordingly.

5 Conclusions and Future Work

Combining the different search methods improves over the single-modal search. Moreover, by allowing different search modalities, the users are no longer confined by the text-only interface. They could freely pick a modality to best represent the query element or mix and match several modalities to construct a complex query. The crucial part is the combination of the results obtained by the different search modalities. This combination must take into account the query, the similarity of the patent to each part of the query, and the confidence on the technique.

PATExpert is reaching a stage where expert users can experiment with it. We need them to use the system and give us their feeling about the quality of the results to tune each of the search engines. After this step we plan the use of machine learning tools to automatically adjust the weights based on user feedback.

There is a lack of information share between the different search engines In order to optimize the search process; the search engines should be connected to a global optimizer that could help provide them information to reduce the search space.

The extension of PATExpert to other domains is highly dependent on the ontologies, and makes it difficult to be used as is with standard repositories like TREC or the ones containing patents, as we need ontologies for the specific technical domains, this means that there are no reference queries, and the training has to be done by our users.

References

[1] Homan H.S.: Making the Case for Patent Searchers. Searcher, vol. 12, March 2004
[2] Lin D.: Principle-based parsing without overgeneration. Proceedings of the 31st conference on Association for Computational Linguistics, pp. 112-120, 1993.
[3] PATExpert home page, http://www.patexpert.org
[4] LUCENE, http://lucene.apache.org/java/docs
[5] Pianta, E., Girardi, C. and Zanoli, R.. "The TextPro tool suite", Proc. of LREC 2008, Marrakech, Morocco, May 2008.
[6] Potrich, A, and Pianta, E., "Learning Domain Specific Isa-Relations from the Web", Proc. of LREC 2008, Marrakech, Morocco, May 2008.
[7] FrameNet site: http://framenet.icsi.berkeley.edu/
[8] Hones F., Lichter J.: Layout extraction of mixed mode documents. Machine Vision and Applications, Springer-Verlag 1994
[9] Yang M., Qiu G., Huang Y., Elliman, D.: Near-Duplicate Image Recognition and Content-based Image Retrieval using Adaptive Hierarchical Geometric Centroids. Proceedings of the 18th International Conference on Pattern Recognition, 2006
[10] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, 1999

[11] Salton, G., Fox, E., Wu, H.: Extended Boolean Information Retrieval, CACM 26(11): pp. 1022--1036, 1983
[12] Hájek P.: Mathematics of fuzzy logic. Kluwer, 1998.
[13] Lee, J. H. et al.: On the evaluation of Boolean operators in the extended Boolean retrieval framework. Proceedings of the 16th annual international ACM SIG
[14] W.S. Hong et al., "A new approach for fuzzy information retrieval based on weighted power-mean averaging operators," *Computers and Mathematics with Applications*, vol. 53, 2007, pp. 1800-1819.

Enhancing Semantic Search using N-Levels Document Representation

Pierpaolo Basile[1], Annalina Caputo[1], Marco de Gemmis[1], Anna Lisa Gentile[1], Pasquale Lops[1], and Giovanni Semeraro[1]

Department of Computer Science, University of Bari
70125 Bari, Italy
{basilepp, acaputo, degemmis, al.gentile, lops, semeraro}@di.uniba.it

Abstract. The traditional strategy performed by Information Retrieval (IR) systems is ranked keyword search: For a given query, a list of documents, ordered by *relevance*, is returned. Relevance computation is primarily driven by a basic string-matching operation. To date, several attempts have been made to deviate from the traditional keyword search paradigm, often by introducing some techniques to capture word meanings in documents and queries. The general feeling is that dealing explicitly with *only* semantic information does not improve significantly the performance of text retrieval systems.
This paper presents SENSE (SEmantic N-levels Search Engine), an IR system that tries to overcome the limitations of the ranked keyword approach, by introducing *semantic levels* which integrate (and not simply replace) the lexical level represented by keywords. Semantic levels provide information about word meanings, as described in a reference dictionary, and named entities. We show how SENSE is able to manage documents indexed at three separate levels, keywords, word meanings, and entities, as well as to combine keyword search with semantic information provided by the two other indexing levels.

1 Introduction

Ranked keyword search is quite successful, in spite of its obvious limits basically due to polysemy, the presence of multiple meanings for one word, and synonymy, multiple words having the same meaning. The result is that, due to synonymy, relevant documents can be missed if they do not contain the exact query keywords, while, due to polysemy, wrong documents could be deemed as relevant. These problems call for alternative methods that work not only at the lexical level of the documents, but also at the *meaning* level.

Any attempt to work at the meaning level must solve the problem that, while words occur in a document, meanings do not, since they are often hidden behind words. For example, for the query "apple", some users may be interested in documents dealing with "apple" as a fruit, while other users may want documents related to the company. Some linguistic processing is needed in order to provide a more powerful "interpretation" both of the user needs behind the query

and of the words in the document collection. This linguistic processing may result in the production of *semantic information* that provide machine readable insights into the meaning of the content. As shown by the previous example, named entities (people, organizations, etc.) mentioned in the documents constitute important part of their semantics. Therefore, semantic information could be captured from a text by looking at *word meanings*, as they are described in a reference dictionary (e.g. WordNet [13]), and *named entities*.

This paper proposes an IR system which manages documents indexed at multiple separate levels: keywords, senses (word meanings), and entities. The system is able to combine keyword search with semantic information provided by the two other indexing levels. In particular, for each level:

1. a *local scoring function* weighs elements belonging to that level according to their informative power;
2. a *local similarity function* computes document relevance by exploiting the above-mentioned scores.

Finally, a *global ranking function* is defined in order to combine document relevance computed at each level.

The paper is organized as follows: After a detailed description of the SEmantic N-levels Search Engine model, we sketch its architecture in Section 3. Sections 4 and 5 provide a description of sense and entity levels, respectively. Global ranking strategies are discussed in Section 6. Results of experiments carried out to evaluate the proposed approach are presented in Section 7. Finally, main work related to the research presented in this paper is discussed in Section 8. Conclusions and future work close the paper.

2 N-Levels model

The main idea underlying the definition of an open framework to model different semantic aspects (or levels) pertaining document content is that there are several ways to describe the semantics of a document. Each semantic facet needs specific techniques and ad-hoc similarity functions. To address this problem we propose a framework where a different IR model is defined for each level in the document representation. Each level corresponds to a *logical view* that aims at describing one of the possible semantic spaces in which documents can be represented. The adoption of different levels is intented to guarantee acceptable system performance even when not all semantics representations are available for a document.

We suppose that a keyword level is always present and, when also other levels are available, these ones are used to offer enhanced retrieval capabilities. Furthermore, our framework allows to associate each level with the appropriate representation and similarity measure. The following semantic levels are currently available in the framework:

Keyword level - the entry level in which the document is represented by the words occurring in the text.

Word meaning level - this level is represented through *synsets* obtained by WordNet, a semantic lexicon for the English language. A synset is a set of synonym words (with the same meaning). Word Sense Disambiguation algorithms are adopted to assign synsets to words.

Named entity level - this level consists of entities recognized into the document text. The integration of named entities and domain ontologies permits some reasoning over document content.

Analogously, N different levels of representation are needed for representing queries. The N query levels are not necessarily extracted simultaneously from the original keyword query issued by the user: A query level can be obtained when needed. For example, the ranked list of documents for the query "Apple growth" might contain documents related to both the growing of computer sales by Apple Inc. and the growth stages of apple trees. Then, when the system will collect the user feedback (for instance, a click on a document in which "Apple" has been recognized as a named entity), the query representation for the named entity level is produced.

We also extended the notion of relevance $R(q, d)$, which computes the *degree of similarity* between each document d in the collection and the user query q. The relevance must be evaluated at each level by defining a proper *local similarity function* that computes document relevance according to the weights defined by the corresponding local scoring function. Since the ultimate goal is to obtain a *single* list of documents ranked in decreasing order of relevance, a *global ranking function* is needed to merge all the result lists that come from each level. This function is independent of both the number of levels and the specific local scoring and similarity functions because it takes as input N ranked lists of documents and produces a unique merged list of most relevant documents. Section 6 describes the adopted global ranking function.

3 SENSE System Architecture

SENSE is a semantic IR system based on the N-Levels model described in the previous section. Figure 1 depicts the system architecture and shows the modules involved in the information extraction and retrieval processes.

Some modules are mainly devoted to deal with ontologies, to perform typical Natural Language Processing (NLP) operations, and to manage the interaction with the user. In more detail:

- Document Manager - It manages document collections to be indexed. It is invoked by the User Interface module to display the results of a user query.
- Ontology Manager - It manages ontologies and is mainly accessed by the Entity Recognition module in order to recognize ontology instances (named entities) into the text. It is invoked by the User Interface module to show fragments of ontologies or dictionaries to the user at query time for query refinement or disambiguation.

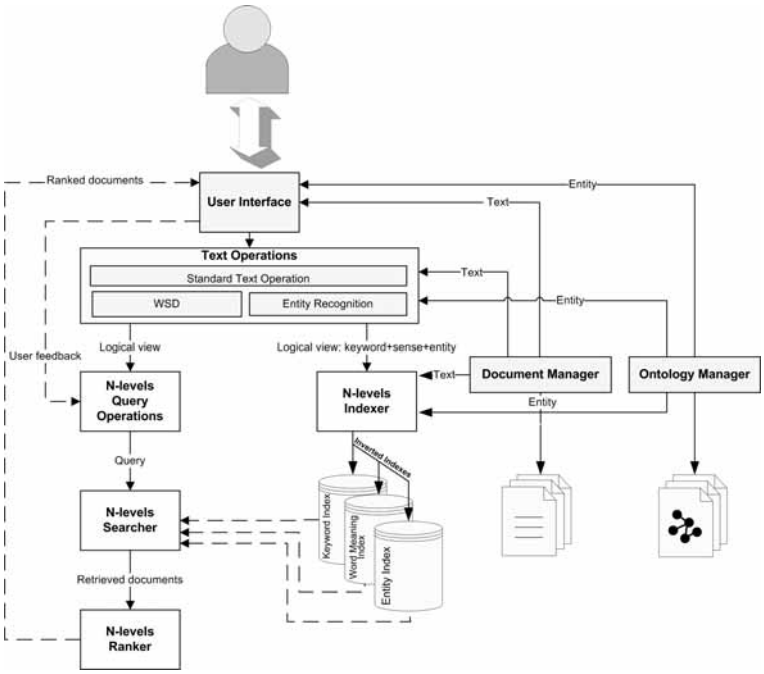


Fig. 1. System Architecture

- Text Operations - It performs basic and more advanced NLP operations. Basic operations implemented are: *Stop words elimination*, *Stemming* (the *Snowball* stemmer is adopted[1]), *POS-tagging* and *Lemmatization*. For POS-tagging, a JAVA version of *ACOPOST tagger*[2] has been implemented; it adopts Trigram Tagger T3 algorithm based on Hidden Markov Models. For lemmatization, the WordNet Default Morphological Processor, as is included in the WordNet 2.0 distribution for English, has been used. Besides basic NLP processing, more advanced procedures were designed for the semantic levels of SENSE: *Named Entity Recognition Driven by Ontologies* and *Word Sense Disambiguation (WSD)*. WSD is the task of selecting a word meaning for a word from a set of predefined possibilities, usually defined in an electronic dictionary or thesaurus. The core component that performs all the steps (WSD included) needed for building the document representation at the meaning level is META [1].
- User Interface - It provides the query interface, which is not just a textbox where keywords can be typed since it allows users to issue queries involving semantic levels.

[1] http://snowball.tartarus.org/
[2] http://acopost.sourceforge.net/

The core of the N-Levels indexing and retrieval processes consists of the following modules:

- N-Levels Indexer - It creates and manages as many inverted indexes as the number of levels into the N-levels model. While the Text Operations component provides the features corresponding to the different levels, the N-Levels Indexer computes the local scoring functions defined for assigning weights to features.
- N-Levels Query Operations - It reformulates user needs so that the query can be executed over the appropriate inverted indexes.
- N-Levels Searcher - It retrieves the set of documents matching the query, for each level identified by Text Operations. It implements the local similarity functions defined in the model.
- N-Levels Ranker - It arranges documents retrieved by the Searcher into a unique list to be shown to the user. For each level involved into the search task, it ranks documents according to the local similarity function and then merges all the local lists into a single list by using the global ranking function.

The core components that perform the N-Levels indexing and retrieval processes are implemented on the Lucene API[3]. Lucene is a full-featured text search engine library that implements the vector space model. We implemented an extension of the Lucene API, the N-Levels Lucene Core, to meet all the requirements of the proposed model.

4 Meaning Level

In SENSE, features at the meaning level are *synsets* obtained from WordNet 2.0. It groups English words into sets of synonyms called *synsets*, provides short general definitions (*glosses*), and records various semantic relations between synonym sets. WordNet distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. Each synset is assigned with a unique identifier and contains a set of synonymous words or collocations; different senses of a word occurs in different synsets.

In order to assign synsets to words, we adopted a WSD strategy. The goal of a WSD algorithm consists in assigning a target word w_i, occurring in a document d, with its appropriate meaning or sense s, by exploiting the *context* C in which w_i occurs. The context C for w_i is defined as a set of words that precede and follow w_i. The sense s is selected from a predefined set of possibilities, usually known as *sense inventory*. The WSD algorithm adopted in SENSE is an improved version of JIGSAW [2]. The basic idea of the algorithm is to combine three different strategies to disambiguate nouns, verbs, adjectives and adverbs respectively. The main motivation behind our approach is that the effectiveness of a WSD algorithm is strongly influenced by the Part of Speech (POS) tag of the target word.

[3] http://lucene.apache.org/

The WSD algorithm takes as input a document $d = [w_1, w_2, \ldots, w_h]$, encoded as a list of words (in order of their appearance), and returns a list of WordNet synsets $X = [s_1, s_2, \ldots, s_k]$ ($k \leq h$), in which each element s_j is obtained by disambiguating the *target word* w_i based on the *similarity* of w_i with the words in its context. Notice that $k \leq h$ because some words, such as proper names, might not be found in WordNet.

Given the target word w_i and the associated sense inventory $S_i = \{s_{i1}, s_{i2}, \ldots, s_{ik}\}$, the algorithm defines a specific (different for each POS) function $\varphi(w_i, s_{ij})$, that computes a real value in $[0, 1]$, representing the confidence with which sense s_{ij} can be associated to w_i. The sense assigned to w_i is the one with the highest confidence. We will not provide further details about the implementation of the WSD procedure because it is not the focus of the paper. More details are reported in [2, 18]. Here we underline that the algorithm achieves about 60% of average precision on the *All-words task*. This result shows that it performs comparably to other state-of-the art knowledge-based WSD algorithms.

The idea behind the adoption of WSD is that each document is represented at the meaning level by the senses conveyed by the words, together with their respective occurrences. The WSD procedure produces a synset-based vector space representation, called bag-of-synsets (BOS). In this model a document is represented by a synset vector, rather than a word vector. Let D be a collection of M documents. The j-th document in D is represented as:

$$d_j = \langle t_{j1}, t_{j2}, \ldots, t_{jn} \rangle, \ j = 1, \ldots, M$$

where t_{jk} is the k-th synset in d_j, n is the total number of synsets in d_j. Document d_j is represented in a $|V|$-dimensional space by a synset-frequency vector, V being the vocabulary for D (the set of distinct synsets recognized by the WSD procedure in the collection):

$$f_j = \langle w_{j1}, w_{j2}, \ldots, w_{j|V|} \rangle, \ j = 1, \ldots, M$$

where w_{jk} is the weight of the synset t_k in d_j, computed according to the local scoring function defined in the next section.

4.1 Synset Scoring Function

Given a document d_i and its synset representation $X = [s_1, s_2, \ldots, s_k]$, the idea is to compute a *partial* weight for each $s_j \in X$, and then to improve this weight by finding out some relations between synsets belonging to X. The partial weight, called SFIDF (synset frequency, inverse document frequency), is computed according to a strategy resembling the tf-idf score for words:

$$\text{SFIDF}(s_j, d_i) = \underbrace{\text{TF}(s_j, d_i)}_{\text{synset frequency}} \cdot \underbrace{log \frac{|\, C \,|}{n_j}}_{\text{IDF}} \tag{1}$$

where $\mid C \mid$ is the total number of documents in the collection and n_j is the number of documents containing the synset s_j. $\text{TF}(s_j, d_i)$ computes the frequency of s_j in document d_i.

The local scoring function for synsets relies on *"Semantic Domains"*, which are areas of human discussion, such as POLITICS, ECONOMY, SPORT, which exhibit their own terminology and lexical coherence. We adopt WordNet Domains [12], an extension of WordNet, in which each synset is annotated with *one or more* domain labels[4]. The domain set of WordNet Domains is composed of about 200 domain labels. The idea of including WordNet Domains in the synset scoring function is based on the *lexical coherence* assumption, claiming that a great percentage of concepts expressed in the same document belongs to the same domain. The availability of WordNet Domains makes it possible to give more weight to synsets belonging to more relevant domains in d. The main advantage of this approach is that WSD errors can be mitigated by domain information. For example, if the noun "bank" was incorrectly disambiguated as "sloping land" (domain: GEOGRAPHY), while its correct sense was "financial institution" (domain: ECONOMY), this error could be recovered by observing that ECONOMY was a common domain in d, while GEOGRAPHY was very rare.

Two different kinds of domain relevance have been taken into account: The relevance of a domain with respect to a specific synset, and the relevance of a domain with respect to the whole set of synsets recognized in a document. In the following, two functions that estimate both kinds of relevance, called *domain relevance* and *document domain relevance*, respectively, are defined. Let $D = \{D_1, D_2, \ldots, D_m\}$ be the set of WordNet Domains. Intuitively, a domain $D_j \in D$ is relevant for a specific synset s if D_j is relevant for the texts in which s usually appears. As an approximation, the information in WordNet Domains can be used to estimate such a function. Let $Dom_j = \{D_{j1}, D_{j2}, \ldots, D_{jh}\}$, $D_j \subseteq D$, be the set of domain labels assigned to synset s_j in WordNet Domains. The domain relevance function is defined as:

$$Rel(D_i, s_j) = \begin{cases} 1/\mid Dom_j \mid & \text{if } D_i \in Dom_j \\ 1/m & \text{if } Dom_j = \text{Factotum} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $m = \mid D \mid$. The domain Factotum covers generic synsets not belonging to a specific domain (they correspond to general language and may appear in any context). Under these settings, generic synsets (Factotum) have low relevance values for each domain, while domain-oriented synsets have high relevance values for a specific domain. $Rel(D_i, s_j)$ can be perceived as an estimated prior probability of the domain given the synset. Given a document d and its synset representation $X = [s_1, s_2, \ldots, s_k]$, the relevance of domain D_i in d is defined as the percentage of synsets in X assigned to D_i. Formally:

[4] Freely available for research at `http://wndomains.itc.it`

$$DocRel(D_i, X) = \begin{cases} \#(s_j, D_i)/\mid X \mid & \text{if } D_i \in Dom_j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $\#(s_j, D_i)$ is the number of $s_j \in X$ for which $D_i \in Dom_j$. For each s_j, the relevance of the domains assigned to s_j is encapsulated into a domain factor α:

$$\alpha = \sum_{D_{jh} \in Dom_j} Rel(D_{jh}, s_j) \cdot DocRel(D_{jh}, X) \quad (4)$$

The domain factor is then exploited to compute the final local score for synset s_j in d_i as $\text{SFIDF}(s_j, d_i) \cdot (1 + \alpha)$.

4.2 Synset Similarity Function

The local similarity functions for both the meaning and the keyword levels are computed using a modified version of the LUCENE default document score. Given a query q and a document d_i, the synset similarity is computed as:

$$synsim(q, d_i) = C(q, d_i) \cdot \sum_{s_j \in q} (\text{SFIDF}(s_j, d_i)(1 + \alpha) \cdot N(d_i)) \quad (5)$$

where: $\text{SFIDF}(s_j, d_i)$ and α are computed as described in the previous section, $C(q, d_i)$ is the number of query terms in d_i, $N(d_i)$ is a factor that takes into account document length normalization.

5 Named Entity Level

The Named Entity Recognition (NER) task has been defined in the context of the Message Understanding Conference (MUC) as the capability of identifying and categorizing entity names, defined as instances of the three types of expressions: entity names, temporal expressions, number expressions [9]. Further specializations of these top level classes have been proposed [16] and general purpose lists of Named Entities are publicly available and incorporated within well-known Text Processing Software, such as GATE (General Architecture for Text Engineering) [4], to give a popular example. However, for the aim of SENSE we cannot rely on general purpose gazetteers to perform the step of NER, due to specificity of categories and instances. For this reason we developed a simple algorithm to recognize entities using a domain ontology as gazetteers. We tag each token in the original document with the ontology class value, if it represents an instance of that class in the domain ontology. Given $C = \{C_1, C_2, \ldots, C_n\}$ the set of classes in the domain ontology, for each class C_k we consider the set $P = \{p_1, p_2, \ldots, p_m\}$ of properties belonging to C_k. Given $T = \{t_1, t_2, \ldots, t_s\}$ the list of tokens obtained from document d, for each token t_j we consider a window of h following tokens. The algorithm checks for each C_k if value of any combination of $t_j, \ldots, t_{j+h}$ matches with the value of any p_m, for all instances of C_k, and assigns to t_j the corresponding label. The search is done beginning

from longer combinations of tokens and in the worst case it ends without any class annotation for the single token t_j.

Similarly to the meaning level, at the entity level documents are represented by using an adaptation of the vector space: The model adopted for this level is indeed a *bag-of-entities* rather than a *bag-of-synsets*. The vocabulary is the set of entities recognized by the NER procedure in the collection, in particular each entity is identified by the URI of the entity instance into the ontology. As first attempt, we adopted a classical tf-idf heuristic to score entities and cosine similarity as local similarity function.

6 Global Ranking

The strategy for defining the *global ranking function* is inspired by prior work on meta-search engines [7], in which algorithms for merging ranked lists are widely used. Formally, we define:

- U: the universe, that is the set containing all the distinct documents in the local lists;
- τ_j=$\{\ x_1 \geq x_2 \geq \ldots \geq x_n\ \}$: the j-th local list, $j = 1, \ldots, N$, defined as an ordered set S of documents, $S \subseteq U$, $\geq$ is the ranking criterion defined by the j-th local similarity function;
- $\tau_j(x_i)$: a function that returns the position of x_i in the list τ_j;
- $s^{\tau_j}(x_i)$: a function that returns the score of x_i in τ_j;
- $w^{\tau_j}(x_i)$: a function that returns the weight of x_i in τ_j.
  Two different strategies can be adopted to obtain $w^{\tau_j}(x_i)$, based on the score or the position of x_i in the list τ_j. Since local similarity functions may produce scores varying in different ranges, and the cardinality of lists can be different, a normalization process (of scores and positions) is necessary in order to produce weights that are comparable.

The aggregation of lists in a single one requires two steps: the first one produces the N normalized lists and the second one merges the N lists in a single one $\hat{\tau}$. In SENSE, we considered both normalization strategies based on scores and positions. Score normalization strategies compute $w^{\tau_j}(x_i)$ by using $s^{\tau_j}(x_i)$, while rank normalization strategies work on $\tau_j(x_i)$. Details are given in Table 1.

In the Score Normalization strategy, min_j is defined as $min_{x_k \in \tau_j} s^{\tau_j}(x_k)$; max_j is defined in an analogous way. While Score Normalization compares $w^{\tau_j}(x_i)$ to the minimum and the maximum scores in τ_j, Z-Score Normalization works on the average of the scores in τ_j, $\mu_{s^{\tau_j}}$, and their variance $\sigma_{s^{\tau_j}}$.

Rank normalization methods work by comparing the position of the document with respect to either the cardinality of the list to be normalized or the cardinality of the universe.

Given N normalized local lists τ_j, the goal of the rank aggregation method is to produce a new list $\hat{\tau}$, containing all documents in τ_j, ordered according to a *rank aggregation function* ψ that combines the normalized weights of local lists in a (hopefully) better ranking. Different strategies can be used to define

Method	Formula
Score Normalization	$w^{\tau_j}(x_i) = \frac{s^{\tau_j}(x_i) - min_j}{max_j - min_j}$
Z-Score Normalization	$w^{\tau_j}(x_i) = \frac{s^{\tau_j}(x_i) - \mu_{s^{\tau_j}}}{\sigma_{s^{\tau_j}}}$
Rank Normalization	$w^{\tau_j}(x_i) = 1 - \frac{\tau_j(x_i)-1}{\lvert\tau_j\rvert}$
Borda	$w^{\tau_j}(x_i) = \begin{cases} 1 - \frac{\tau_j(x_i)-1}{\lvert U\rvert} & \text{if } x_i \in \tau_j \\ \frac{1}{2} + \frac{\lvert\tau_j\rvert - 1}{2 \cdot \lvert U\rvert} & \text{otherwise} \end{cases}$

Table 1. Score and Rank normalization methods

ψ. Some of them are based on the concept of *rank hits* of a document x_i, that is the number of local lists which contain x_i. Let R be the set of all local lists, $R = \{\tau_1, \ldots, \tau_N\}$, $hits(x_i, R) =\mid \{\tau_j \in R : x_i \in \tau_j\} \mid$.

In SENSE, we adopted the following rank aggregation methods:

CombSUM - The score of document x_i in the global list is computed by summing all the normalized scores for x_i:

$$\psi(x_i) = \sum_{\tau_j \in R} w^{\tau_j}(x_i)$$

CombMNZ - It multiplies the CombSUM score by the rank hits, thus increasing the score of documents occurring in more than one local list:

$$\psi(x_i) = hits(x_i, R) \cdot \sum_{\tau_j \in R} w^{\tau_j}(x_i)$$

Weighted Combination - The score of document x_i in the global list is computed similarly to CombMNZ, except for the introduction of a boost factor α_j for each local list, in order to amplify (or reduce) the weight of x_i in each list:

$$\psi(x_i) = hits(x_i, R) \cdot \sum_{\tau_j \in R} \alpha_j \cdot w^{\tau_j}(x_i) \quad \sum \alpha_j = 1, \alpha_j \geq 0$$

where α_j underlines the importance of a local list in the global ranking, i.e. the importance of a level in SENSE. The motivation behind our choice is that CombSUM and CombMNZ operators have proved to perform better than others [11]. Preliminary experiments (not reported here due to space constraints) showed that Z-Score is a good choice, independently of the adopted ranking strategy.

7 Experimental Sessions

Experiments were carried out on a standard test collection. We used the SEMEVAL-1 Task 1[5] dataset derived from the English CLEF data from years 2000-2005, amounting to 169,477 documents (579 MB of raw text, 4.8 GB in XML format) and 300 topics (queries) in English and Spanish. The relevance judgments were taken from CLEF. Due to the size of the document collection, the task organizers decided to take a sixth part of the corpus at random, comprising 29,375 documents (874 MB in XML format). Not all topics had relevant documents in this

[5] `http://ixa2.si.ehu.es/semeval-clir/`

17% sample, and therefore only 201 topics were effectively used for evaluation. In the dataset actually used for the experiments, 923 documents are relevant. All the SENSE components are implemented in JAVA. Experiments were run on a machine with 2 GB of main memory, an Intel Core 2 Quad processor at 2.4 GHz, operating in 32 bit mode, running Linux (UBUNTU 7.10). Performances are evaluated considering three dimensions: the size of index for each level, the indexing times and the query times (Tables 2 and 3).

Level	**Size (MB)**	**Indexing time**
Stemming	23.6 MB	4m:40s
Sense	129.2 MB	22h:40m (6m:32s)

Table 2. Sizes and times for index creation in SENSE

The index for the sense level is larger than the one for the stemming level, because for each synset additional information about WordNet Domains is stored, besides the synset frequency score. That information is stored separately from the synset frequency, by using the Lucene *Payload structure*, thus requiring more space. The time required for building the index for the sense level is higher, compared to the time required for the stemming level. The huge difference is mainly due to the WSD process. If we consider only the time requested for building the index, once all the words in the dataset have been disambiguated, the indexing time remains higher, but still acceptable (6m:32s vs. 4m:40s). The additional time (1m:52s) is due to the computation of WordNet Domains information. Results about query times are reported in Table 3. The first column reports the levels involved in the evaluation (only stemmed keywords, only synsets, both levels), the second column reports the average time required to solve a query, composed by an average number of terms (or synsets) reported in the last column. Queries involving the sense level have been automatically disambiguated by the same WSD procedure adopted for building the inverted index for synsets. Results show that performance is not overmuch affected by time required by the global ranking function to aggregate the results coming from each level. Indeed, the query times for the stemming+sense evaluation is 8% higher than those for senses only, and 40% higher than query times for stemming.

Several experiments were performed in order to evaluate different local scoring functions and different global ranking functions. Options for setting experiments are reported in Table 4. We evaluated the effect of using a simple adapta-

Level	**Time (ms)**	**Avg Terms**
Stemming	1600	24.20
Sense	2080	17.24
Stemming+Sense	2240	-

Table 3. Query times

Setting	Description
LK	Stemming level
LM	Meaning level
TFIDF	Tf-Idf Scoring
SFIDF	Synset Frequency Scoring
SFDOM	SFIDF + WordNet Domains Scoring
NS	Score normalization
NZS	Z-Score normalization
NR	Rank normalization
NB	Borda couting
GS	CombSUM
GM	CombMNZ
GMP_1	Weigthed Combination LK($\alpha_1 = 0.4$) LM($\alpha_2 = 0.6$)
GMP_2	Weigthed Combination LK($\alpha_1 = 0.6$) LM($\alpha_2 = 0.4$)
GMP_3	Weigthed Combination LK($\alpha_1 = 0.8$) LM($\alpha_2 = 0.2$)
GMP_4	Weigthed Combination LK($\alpha_1 = 0.2$) LM($\alpha_2 = 0.8$)

Table 4. Options for Experiments

tion of the TFIDF score for synsets (SFIDF, see Section 4.1) against the use of a more complex scoring function that takes into account WordNet Domains (SFDOM, see Section 4.1). Other options are the type of normalization and the aggregation strategy of results obtained when using both keywords and synsets. All the options described in Section 6 were evaluated and the setting options in Table 4 have been combined, obtaining a total of 22 experiments, with the final aim of evaluating whether keyword search can be improved by the adoption of the *meaning level*, in addition or replacement of the keyword level. Table 5 shows the percentage of total number of relevant documents retrieved over all queries (R) and the MAP (Mean Average Precision) obtained for each experiment.

The first result is that the use of the meaning level alone does not outperform the stemming level (Exp1 vs. Exp2 and Exp3). Even though it was expected, an interesting outcome is that the synset scoring function that takes into account WordNet Domains information achieves a higher recall than the simple adaptation of tf-idf for synsets (Exp2 vs. Exp3). The most interesting result is that the combination of both levels produces better results than the sense level alone (Exp4-22 vs. Exp2 and Exp3). Indeed, in most cases the performance is reasonably comparable to that of the stemming level alone. As regards normalization and global ranking strategies, the best result are obtained by setting Z-Score normalization, independently of the ranking strategy adopted (Exp5, Exp8, Exp11, Exp14, Exp17, Exp20). Finally, from Exp17, it could be noted that a small improvement of R is obtained compared to stemming (Exp1), when a weighted combination strategy is adopted for global ranking, giving a small weight to senses (0.2). This was the only case in which the combination of both levels outperformed keyword search (4 more relevant documents are retrieved by including senses).

Exp	Setting	R	MAP
1	LK+TFIDF	0.5731	0.1498
2	LM+SFIDF	0.5038	0.0782
3	LM+SFDOM	0.5125	0.0795
4	LK+LM+SFDOM+NS+GS	0.5731	0.1187
5	LK+LM+SFDOM+NZS+GS	0.5731	0.1317
6	LK+LM+SFDOM+NR+GS	0.5471	0.0987
7	LK+LM+SFDOM+NS+GM	0.5731	0.1187
8	LK+LM+SFDOM+NZS+GM	0.5731	0.1316
9	LK+LM+SFDOM+NR+GM	0.5471	0.0987
10	LK+LM+SFDOM+NS+GMP_1	0.5710	0.1093
11	LK+LM+SFDOM+NZS+GMP_1	0.5731	0.1209
12	LK+LM+SFDOM+NR+GMP_1	0.5406	0.0967
13	LK+LM+SFDOM+NS+GMP_2	0.5731	0.1309
14	LK+LM+SFDOM+NZS+GMP_2	0.5731	0.1400
15	LK+LM+SFDOM+NR+GMP_2	0.5558	0.1026
16	LK+LM+SFDOM+NS+GMP_3	0.5731	0.1444
17	LK+LM+SFDOM+NZS+GMP_3	0.5742	0.1472
18	LK+LM+SFDOM+NR+GMP_3	0.5601	0.1115
19	LK+LM+SFDOM+NS+GMP_4	0.5547	0.0935
20	LK+LM+SFDOM+NZS+GMP_4	0.5634	0.1016
21	LK+LM+SFDOM+NR+GMP_4	0.5287	0.0888
22	LK+LM+SFDOM+NB+GS	0.5515	0.1007

Table 5. Experimental results

8 Related Work

The general idea of enhancing keyword search by the addition of word meanings is (of course) not new. Many strategies have been used to incorporate semantic information coming from ontologies or electronic dictionaries into search paradigms. Mainly two aspects have been addressed in the past: query expansion with semantically related terms, and the comparison of queries and documents by using semantic similarity measures.

Query expansion with WordNet has shown to potentially improve recall, as it allows matching relevant documents even if they do not contain the exact keywords in the query [19–21]. On the other hand, semantic similarity measures have the potential to redefine the similarity between a document and a user query [3, 10, 15]. The semantic similarity between concepts is useful to understand how similar the meanings of the concepts are. However, computing the degree of relevance of a document with respect to a query means computing the similarity among all the synsets of the document and all the synsets of the user query, thus the matching process could have very high computational costs.

In [8], the authors performed a shift of representation from a lexical space, where each dimension is represented by a term, towards a semantic space, where each dimension is a concept expressed using WordNet synsets. They adapted the Vector Space Model applied to WordNet synsets. The realization of the

semantic tf-idf model was rather simple, because it was sufficient to index the documents or the user-query by using strings representing synsets. The retrieval phase is similar to the classic tf-idf model, with the only difference that matching is carried out between synsets.

While previous methods tried to *replace* the lexical space with *one* semantic space, in SENSE we defined an adaptation of the vector space model that allows the integration of the lexical space with *one or more* semantic spaces. We show how keywords can be integrated with WordNet synsets, but the model can be easily extended by adding more levels, without modifying the whole architecture of the SENSE system. Another remarkable attempt to indexing documents according to WordNet senses which is most similar to our approach is reported in [14]. The authors designed an information retrieval system performing a combined word-based and sense-based indexing and retrieval. They added lexical and semantic information to both the query and the documents during a preprocessing step in which the query and the text are disambiguated. More recent approaches [5, 6] try to combine keyword search with techniques for navigating and querying ontologies. In [5], documents are annotated with concepts in a domain ontology and indexed using classical Bag-Of-Words model, while in [6] it is described a search tool based on ontology assisted query rephrasing and keyword search. The main limitation of the approach is that relevance is computed simply by using a tf-idf score on concepts, instead of keywords.

9 Conclusions and Future Work

We have described SENSE (SEmantic N-levels Search Engine), a semantic N-levels IR system which manages documents indexed at multiple separate levels: keywords, senses, and entities. The system is able to combine keyword search with semantic information provided by the two other indexing levels.

The distinctive feature of the system is that an IR framework is proposed to integrate, rather than simply replace, the lexical space with semantic spaces. We provided a detailed description of the sense level, by defining a WSD algorithm to assign words occurring in a document with senses and an entity recognition method to extract named entities from text. We have defined several global ranking functions describing how to merge rankings produced by different levels. As future work, we plan to perform a more extended experimental session and to investigate new strategies for representing documents both at the synset and at the entity level. An ongoing activity is the integration of the N-Levels IR framework underlying SENSE into a semantic retrieval model based on user profiles described in [17].

Acknowledgments

This research was partially funded by MIUR (Ministero dell'Universit e della Ricerca) under the contract Fondo per le Agevolazioni alla Ricerca, DM19410 "Laboratorio di Bioinformatica per la Biodiversità Molecolare" (2007-2011).

References

1. P. Basile, M. de Gemmis, A. Gentile, L. Iaquinta, P. Lops, and G. Semeraro. META - MultilanguagE Text Analyzer. In *Proc. of the Language and Speech Technnology Conference - LangTech 2008*, pages 137–140, 2008.
2. P. Basile, M. de Gemmis, A. Gentile, P. Lops, and G. Semeraro. Jigsaw algorithm for word sense disambiguation. In *SemEval-2007: 4th Int. Workshop on Semantic Evaluations*, pages 398–401. ACL press, 2007.
3. C. Corley and R. Mihalcea. Measures of text semantic similarity. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence*, 2005.
4. H. Cunningham, Y. Wilks, and R. Gaizauskas. Gate: a general architecture for text engineering. In *Proc. of the 16th Conf. on Computational Linguistics*, pages 1057–1060, Morristown, NJ, USA, 1996. ACL.
5. J. Davies and R. Weeks. QuizRDF: Search technology for the Semantic Web. In *37th Hawaii Int. Conf. on System Sciences*. IEEE Press, 2004.
6. G. Ducatel, Z. Cui, and B. Azvine. Hybrid ontology and keyword matching indexing system. In *Proc. of IntraWebs Workshop at WWW2006*, 2006.
7. M. Farah and D. Vanderpooten. An outranking approach for rank aggregation in information retrieval. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *Proc. of the 30th SIGIR Conf.*, pages 591–598. ACM, 2007.
8. J. Gonzalo, F. Verdejo, I. Chugur, and J. M. Cigarrán. Indexing with wordnet synsets can improve text retrieval. *CoRR*, cmp-lg/9808002, 1998.
9. R. Grishman and B. Sundheim. Message understanding conference- 6: A brief history. In *COLING*, pages 466–471, 1996.
10. J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008, 1997.
11. J.-H. Lee. Analyses of multiple evidence combination. In *Proc. of the 20th SIGIR Conference*, pages 267–276. ACM, 1997.
12. B. Magnini and G. Cavagliá. Integrating subject field codes into wordnet. In *Proc. of the LREC-2000*, pages 1413–1418, 2000.
13. G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
14. D. I. Moldovan and R. Mihalcea. Using wordnet and lexical operators to improve internet searches. *IEEE Internet Computing*, 4(1):34–43, 2000.
15. P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.
16. S. Sekine, K. Sudo, and C. Nobata. Extended named entity hierarchy. In *Proc. of the LREC-2002*, 2002.
17. G. Semeraro. Personalized searching by learning wordnet-based user profiles. *Journal of Digital Information Management*, 5(5):309–322, 2007.
18. G. Semeraro, M. Degemmis, P. Lops, and P. Basile. Combining learning and word sense disambiguation for intelligent user profiling. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence*, pages 2856–2861, 2007. M. Kaufmann.
19. A. Smeaton, F. Kelledy, and R. ODonnell. TREC-4 experiments at Dublin city university: thresholding posting lists, query expansion with WordNet, and POS tagging of Spanish. In *Proc. of TREC-4*, 1995.
20. E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proc. of the 17th SIGIR Conf.*, pages 61–69, 1994.
21. E. M. Voorhees. *WordNet: An Electronic Lexical Database*, chapter 12: Using WordNet for text retrieval, pages 285–304. Cambridge: The MIT Press, 1998.

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

The Interaction Between Automatic Annotation and Query Expansion: a retrieval experiment on a large cultural heritage archive

Véronique Malaisé[1], Laura Hollink[1], and Luit Gazendam[2]

[1] Department of Computer Science
Vrije Universiteit Amsterdam
de Boelelaan 1081 HV
The Netherlands
[2] Telematica Instituut
Brouwerijstraat 1
7523 XC Enschede
The Netherlands

Abstract. Improving a search system for large audiovisual archives can be done in two ways: by enriching the annotations, or by enriching the query mechanism. Both operations possibly benefit from a preliminary terminological enrichment of the controlled vocabulary in use, *i.e.* the thesaurus. In this paper we report on a four-parts experiment in which we evaluate the benefits and drawbacks of both aspects: the added value and pitfalls of automatically generated semantic annotations over classically (*i.e.* manually) assigned keywords and the added value and pitfalls of query expansion over pure keyword matching technique; we then investigate the combination of these operations in the following setup: we create the baseline for our experiments by querying a set of documents annotated by cataloguers with keywords from the thesaurus. We then apply the same querying process on a set of annotations automatically generated from textual resources related to the documents. Thirdly, we apply a querying process enhanced with query expansion functionalities to the first set of manually annotated documents. Finally, we apply the query expansion mechanism on the automatically generated annotations. The results give insight into the interaction between the two approaches.

1 Introduction

Enhancing the search results in large archives is a concern shared by many cultural heritage institutions. The improvement can come from two directions: enhancing the annotations or enhancing the search mechanism. Both directions are active research area's. In this paper we explore the interaction between those two approaches.

Enhancing the annotations can, for example, be done by facilitating manual creation of semantic annotations as in [10] or [4]. As manual annotation due to time constraints inherently leads to a relatively low number of keywords per

document, it can be complemented or even replaced by (semi-)automatically created annotations. In [13], for example, a tool is introduced for semi-automatic semantic annotation, extracted from text resources. Automatically generated annotations, however, seldom reach the quality level of manual annotations.

Another way to enhancing the search mechanism is query expansion: retrieval of not only documents that match the query concept, but also documents that are annotated with concepts that are *related* to the query. Ontology based query expansion is studied, for example, by [2]. The added value of query expansion in a cultural heritage archive has already been shown in [5]. However, the question remains what is the effect of query expansion in the context of automatic annotation? Is query expansion still beneficial when applied to lower-quality automatic annotations? And is it still necessary if a larger number of annotations is generated?

To answer these questions, we perform a study consisting of four experiments:

1. First, we compute a baseline by querying a corpus of hand-made metadata.
2. Second, we query the automatically generated annotations of the same corpus.
3. Third, we query the hand-made metadata using query expansion.
4. Fourth, we query the automatically generated annotations using query expansion.

The experiments that we present in this paper were conducted in collaboration with and on data from the Netherlands Institute for Sound and Vision, the Dutch national Audiovisual Archives. Our use-case consisting of audiovisual documents, we could have taken into account yet another field of research: the extraction of semantic keywords based on the video stream's low level features. As stated in [16], this technology is not really mature yet, and besides no detectors exist so far for the 3800 terms of the thesaurus we are interested in. Usually, the detectors are of hundreds of different types at most, and perform best on one given corpus of documents. For all these reasons, we took only into account so far the extraction based on textual descriptions of the audiovisual programs: extraction of keywords from textual resources gives good results. We did not take into account the transcripts form the videos either because of the numerous errors that these transcriptions contain: no NLP tool performs at an optimal level with syntactically incorrect sentences. Teletext and other resources will be used as input for our process at a later stage but as a first set of experiments we consider textual descriptions at a higher level of abstraction. This is the level that best suited our needs. Indeed, at Sound and Vision, the archived TV programs' *core topics* are described manually by cataloguers and annotated with keywords selected from a thesaurus, the GTAA. Our task is to extract keywords that describe as globally as possible the program's content.

The GTAA thesaurus is subsequently used for searching the archives. Its hierarchical structure is weak. As both query expansion and our automatic annotation mechanism rely on the structure of the thesaurus, we enriched the thesaurus with additional relations between its concepts.

In the remainder of this paper, we first describe the background on which the current paper is based: section 2 describes previous work on conversion of the thesaurus to SKOS, automatic semantic annotation, thesaurus enrichment and query expansion. Section 3 is dedicated to the description of the four experiments and their results. We conclude and propose future work in section 4.

2 Background

2.1 The GTAA thesaurus and its conversion to SKOS

The thesaurus that is used at Sound and Vision for the annotation and retrieval of TV programs stored in the archives is called the GTAA, a Dutch acronym for "Common Thesaurus [for] Audiovisual Archives". It is a faceted thesaurus, in Dutch, and each facet corresponds to at least one field in the document's description scheme. The topic(s) of the TV program is(are) described by terms from the Subject facet, which contains about 3800 Terms and 2000 additional variants of these terms such as so-called Nonpreferred Terms, which are not meant to be used for indexing but which aid in locating the right term. For example *posters* is a Nonpreferred Term that points to the term *affiches*, which is the right term to be used for indexing programs about posters, and is the only term that will enable a user to retrieve these documents. The Subject facet is organised according to hierarchical relationships (Broader Term/Narrower Term, between a term and its more general/specific notion) and associative relationships (Related Terms, such as *schepen* and *scheepvaart*, Dutch for respectively *ships* and *navigation(ship traffic)*). Besides these relationships defined in the ISO and NISO standards, the terms from the Subject facet are also grouped into a set of "topic" categories, like Philosophy, Economy, etc.

In order to use these relationships either in automatic annotation or query expansion processes, we converted the Subject facet to an RDF representation and modeled the relationships as SKOS triples [15]. For details about the conversion see [17].

2.2 Automatic semantic annotation

In the CHOICE project, we are using the GATE platform [6] for automatic generation of annotations from texts that are related to the TV programs. Other platforms and tool suits exist for generating ontology-based manual, semi-automatic or automatic annotations, like [13], but we chose GATE because we could use our own thesaurus as knowledge resource and tune the platform to our own needs. The idea that we are pursuing is to help cataloguers in their daily work with semi-automatic support. For this purpose, we have co-developed a plug-in called Apolda[3], which takes an ontology and a text as input, and returns an annotated text. The annotations refer to ontology URI (unique identifiers of concepts) and are based on the strings or *labels* that represent the ontology's

[3] Downloadable at the URL:`http://apolda.sourceforge.net/`

concepts for human readers. What we take at first for labels, in our case, are the Terms and Nonpreferred Terms of the GTAA: when they are matched in the text, an annotation is created, specifying the URI of the concept they refer to in the RDF version of the thesaurus. For example, a text containing both the words *posters* and *affiches* gets twice the annotation *GTAA_Subject_Posters*, their common URI. The texts we are using are called *context documents*, and describe the content of TV programs that will be or are stored in the archives: they are online TV guides or broadcaster's Websites for example. Besides the information already present in the thesaurus, we also computed the singular form of the Terms and Nonpreferred Terms based on the Celex lexicon [1], in order to get a better set of possible annotations. The possible annotations are meant as suggestions for annotating the TV programs the texts refer to.

The generated annotations contain sometimes long lists and/or errors due to the ambiguity of terms taken out of their context. In order to solve these two problems[4], we have developed a ranking algorithm. It is based on the structure of the thesaurus and a weighting system to compute the relative importance of the Terms matched in a given text. This algorithm is detailed below.

The semantic annotation pipeline The list of annotations that is extracted by Apolda along with their number of occurrences per textare fed to the CARROT algorithm. CARROT ranks highest the annotations that have direct and indirect thesaurus relationships to other annotations found for the same document, then the Terms that are connected to this group, then the annotations that have only indirect relationships to others, and finally then the rest.

In each of the aforementioned groups (annotations with direct and indirect, only indirect and no relationships to others at all), the annotations are further ordered based on a measure of their weight and their alphabetical order. The weighting of Terms' occurrences that we have experimented so far were pure occurrences counting and tf.idf weighting. For the experiments described in this paper, we also reduced the list of suggestion by taking into account only the first N ones, N being defined as the value of the square root of the list's length. We chose this value based on empirical tests: on average, only the part of the list that we kept are relevant annotation suggestions, the bottom of the list being filled mostly with noise.

For enhancing the search in the archives, a query expansion mechanism was developed in the context of the MUNCH project, aiming at multi-modal search in audiovisual archives.

[4] Having long lists of keywords extracted from texts is seen as a negative point because these lists are made to be shown to cataloguers, in order to speed up and ease their annotation process: showing them lists of more than hundred Terms is not an optimal solution in that respect, given the fact that their rules teach them to use as few of them as possible.

2.3 Query expansion

Like the semantic annotation, the query expansion mechanism is also based on the thesaurus structure. Thesaurus based query expansion requires a richly structured thesaurus. In previous experiments [11], we have show how we could use an anchoring of the GTAA to WordNet to add structure to the weakly structured GTAA. Wordnet is a terminological resource developed at the Princeton University [7], freely available from the Princeton website[5]. In addition, W3C has released a RDF/OWL representation of WordNet version 2.0[6]. For our experiment we use this RDF/OWL version, as it allows us to use Semantic Web tools such as SeRQL to query the WordNet database. We present here briefly the anchoring method that we used and the number of additional relationships inferred back in the original thesaurus, along with the process to infer them. We then go into the details of our query expansion mechanism.

Anchoring GTAA to WordNet As the GTAA is in Dutch, we queried an online dictionary in order to retrieve translations for the terms, along with definitions. Our purpose was to follow the method of [14] and base our anchoring on the lexical overlap between Term's descriptions and WordNet's descriptions: the glosses. The definitions that matched with the WordNet glosses, which was the case for more than 90 % of them, corresponded exactly to WordNet glosses, so the anchoring process was eased.

In total, 1,060 GTAA terms were anchored to WordNet. An evaluation of the correspondences suggests that the number of synsets that is aligned with a particular GTAA term is not an indication of the quality of the match; GTAA terms that are matched to six synsets are equally well matched as GTAA terms that are matched to only one synset.

Inferring additional relations in the GTAA We used the anchoring to WordNet to infer new relations within the GTAA. Using SeRQL [3] queries we related pairs of GTAA subject terms that were not previously related. Figure 1 illustrates how a relation between two terms in the GTAA, t_1 and t_2, is inferred from their correspondence to WordNet synsets w_1 and w_2. If t_1 corresponds to w_1 and t_2 corresponds to w_2, and w_1 and w_2 are closely related, we infer a relation between t_1 and t_2. The inferred relation is symmetric, illustrated by the two-way arrow between t_1 and t_2.

Two WordNet synsets w_1 and w_2 are considered to be 'closely related' if they are connected though either a direct (i.e. one-step) relation without any intermediate synsets or an indirect (i.e. two-step) relation with one intermediate synset. The latter situation is shown in Figure 1. From all WordNet relations, we used only meronym and hyponym relations, which roughly translate to part-of and subclass relations, and their inverses holonym and hypernym. A previous study [12] demonstrated that other types of WordNet relations do not improve

[5] http://wordnet.princeton.edu/
[6] http://www.w3.org/TR/wordnet-rdf/

retrieval results when used for query expansion. Both meronym and hyponym can be considered hierarchical relations in a thesaurus. Only sequences of two relations are included in which each has the same direction, since previous research [12, 9] showed that changing direction, especially in the hyponym/hypernym hierarchy, decreases semantic similarity significantly. For example, w_1 holonym of w_i hyponym of w_2 is not included. At present, all anchoring relations are utilized, also the ones that relate a GTAA term to multiple WordNet terms.

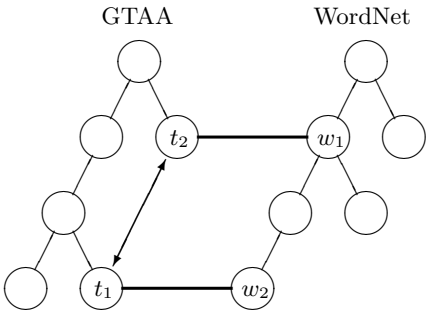


Fig. 1. Using the anchoring to WordNet to infer relations within the GTAA.

A total of 904 pairs of GTAA terms was newly related: 467 with one step between WordNet synsets w_1 and w_2 and 435 with 2 steps between w_1 and w_2. An inspection of the inferred relations reveals that 90 % of the one-step relations were derived from hyponym relations and only 10% from meronym relations. The two-step relations were for 72 % based on sequences of two hyponym relations, for 26 % on combinations of hyponym and meronym and only for 3 % on sequences of two meronym relations.

An informal manual inspection of a portion of the new relations revealed that only very few seem wrong. Based on the original GTAA and the newly inferred relationships, we implemented a query expansion mechanism dedicated to Sound and Vision, but its general mechanism can be applied to any archive using a thesaurus for annotating their data.

The query expansion mechanism Query expansion was done by simply adding concepts to the query that are a fixed number of steps away from the original query concept. All relations were used to walk through the thesaurus: broader, narrower, related, but also the relations inferred from the links to WordNet.

We experimented with expansion to concepts that were only one step away from the query, and with expansion to concepts up to two steps away. As the GTAA has a shallow structure, expanding a query with concepts that are more than two steps away leads too often to concepts that are in an unrelated part of the hierarchy.

2.4 Related work

As we did experiments on both types of methods for enhancing the search process in large archives, we wanted to test how these techniques would interact and what their combination would bring. In the litterature, see [18] for example, either one or the other of the aspects are investigated, namely either improvement based on semantic annotation or on query expansion. We chose to analyze their combination and ran a set of four experiments, described in more details in the following section.

3 Four Experiments

3.1 Material: queries, test corpus and gold standard

In order to be as close as possible from a real-life need, we selected a set of queries from one week of query logs collected at Sound and Vision. We selected the top 44 in the list of most frequently asked keywords, in the keyword search field of the query interface, and stoped the selection with the group of keywords that had only two occurrences in the query log.

The list of the top 44 keywords is: Geschiedenis (history), Kabinetsformaties (forming of parliament), Parlementaire debatten (parliamentary debates), Politici (politicians), Politiek (politics), Politieke partijen (political parties), Politieke programma's (political programmes), Verkiezingen (elections), Verkiezingscampagnes (election campaigns), Gemeenteraden (municipal councils), Asielzoekers (asylum seekers), Islam (islam), Leger (army), Mobilisatie (mobilisation(of army)), Atoombommen (nuclear bombs), Bombardementen (bombardments), Explosies (explosions), Gevaarlijke stoffen (dangerous substances), Gewonden (wounded), Eerste hulp (first aid), Geneesmiddelen (medications), Euthanasie (euthanasia), Dementie (dementia), Broeikaseffect (greenhouse effect), File's (traffic-jams), Snelwegen (highways), Spoorwegongevallen (railway accidents), Autobussen (busses), Alcohol (alcohol), Cafe's (cafe's), Fabrieken (factories), CAO's (collective work agreements), Vulkaanuitbarstingen (vulcano eruptions), Woestijnen (deserts), ‚Zonsondergangen (sunsets), Voetbal (soccer), Zwembaden (swimming pools), Schaatsen (ice skating), Kaartspelen (cardgames), Kermissen (village fairs), Mode (fashion), Opvoeding (education), Dierenhandel (animal trade), Grachten (canals).

These 44 queries are matched against a textual corpus that we had built for previous experiment according to the following rationales:

- The corpus is focused on TV program's description made manually by cataloguers and stored in the previous system for managing the archives at Sound and Vision: Avail. We therefore call these manual catalogue entries "Avail documents"[7];

[7] These can be accessed online at `http://www.beeldengeluid.nl/collecties_zoek_en_vind_tvfilm.jsp`.

– We only selected descriptions of programs which were part of a collection called Academia [8];
– We only selected descriptions of programs for which we could find open accessible context documents: textual descriptions of the TV program's content on broadcaster's Websites or TV-guides, for example;
– We narrowed our selection to documentary programs.

The choice of limiting ourselves to documents related to the academia collection and to documentaires is explained by the fact that, on the one hand, the Academia collection has been cleared from intellectual property rights by Sound and Vision in order to create an open accessible collection for educational and research purposes. Although we do not use this primary audiovisual content in this research, we decided that it would be wise restrict our corpus selection to documents with open accessible AV material.

On the other hand, we narrowed down our selection to documentary programs for multiple reasons: (1) they usually had accessible context information such as web sites, even though some programs could be as old as 7 years. For news items, sport programs or actualities this is not the case. This made the manual selection much more efficient. (2) the information described in their context documents is usually quite extensive. Because we want to gain insight into the process of annotating via context documents, we wanted to have as few content-wise difference with the actual AV document content.

For all the web sites, these textual resources were selected and copied manually. Table 1 details the composition of the corpus.

Series name	Program topic	nb of programs
andere tijden	history	93
beeldenstorm	art	68
de donderdag documentarie	humanities	6
de nieuwe wereld	informative	5
dokument	humanities	6
dokwerk	history or politics	57
Jota!	science	10
Nieuw economisch peil	economy	10
werelden	social	3

Table 1. The composition of our corpus

3.2 Experiment one: the baseline

The baseline experiment consisted in evaluating how many of the Avail documents were annotated with one or more of the "Top 44" keywords. As the assessment of keywords was done by hand and as we evaluate queries consisting in only one keyword, if the keyword is present in the Avail metadata[8], we

[8] The keyword field of the metadata only, to be more specific.

consider that the document is relevant for that keyword. In order to have an idea about the recall, we computed an "estimated recall" by evaluating how many of the documents from the golden standard that we judged relevant to be annotated by one of these 44 keywords were retrieved (column "Estimated recall" from the "Manual Metadata" section in table 2). Our most successful keyword (*geschiedenis*, Dutch for *history*) retrieved 97 documents, but most of the keywords (14) did not retrieve any document in our test corpus. The section "Manual Metadata" of table 2 shows the number of documents retrieved per keyword and the estimated recall, based on our gold standard. The estimated recall is labelled as "Non relevant" (NR) if there were no documents annotated by this keyword in our manually established golden standard.

One first remark that we can derive from this table is the low values for estimated recall. It can be due to two reasons. Firstly, we evaluated whether a set of 44 queries was suitable for annotating documents, whereas the cataloguers have a larger choice: they can select any term from a set of 3800. Therefore the granularity level and the selection can be quite different (for example, they would probably choose *second world war* where we judged that *army* was relevant as a keyword). Secondly, some of the keywords, like *politicians* or *political parties*, can be replaced by a list of names corresponding to the people or parties mentioned in the TV programs. A cataloguer from Sound and Vision would choose this option, as it gives more precise information than the generic Subject keyword. As our experiment focuses only on Subject keywords, and not on the other parts of the metadata, and as there is not built-in relationship between names (of politicians or political parties) and their types in the thesaurus, we could not bridge this gap. But this problem is interesting to keep in mind for providing more relevant automatic semantic annotations in the future, by creating automatically this missing link.

3.3 Experiment two: keyword matching on automatic semantic annotations

After computing the baseline with the first experiment, we applied the same evaluation metrics to the annotations generated automatically by our semantic annotation pipeline: we counted the number of documents that were retrieved for each of the 44 queries, we estimated a recall measure based on the number of documents from our gold standard that were retrieved. We also computed the overlap between the documents that were retrieved based on manual annotation and documents retrieved based on annotations that were generated with the Apolda plugin. This is show in the column called 'overlap' in Table 2.

Queries based on the manually assigned annotations retrieved 142 documents, with an average recall of 22.3 %. Nine queries retrieved documents out of 26 possibilities in our manually established golden standard. The figures are not that good for the queries that were matched against the automatically generated keywords: only 57 documents were retrieved, with an average estimated recall of 9.6% and only 6 keywords out of the 26 possible retrieved documents. Here again, the explanation is twofold. On the first hand, our random sample of documents

	Manual Metadata		Automatic Metadata		Overlap
Query	retrieved	estimated recall	retrieved	estimated recall	
History	97	23/60=38.33	6	1/60=1.66	2
Forming of Parliament	0	NR	0	NR	NR
Parlementary debates	0	NR	0	NR	NR
Politicians	2	0/14=0	3	6/14=42.85	0
Politics	10	1/15=6.66	8	1/15=6.66	3
Political parties	2	NR	0	NR	0
Political programmes	0	0/1=0	0	0/1=0	NR
Elections	1	1/1=100	4	1/1=100	1
Election campaigns	3	1/1=100	0	0/1=0	0
Municipal councils	0	0/2=0	1	1/2=50	0
Asylum seekers	7	0/2=0	2	0/2=0	2
Islam	3	0/4=0	3	0/4=0	2
Army	1	1/7=14.28	9	2/7=28.57	1
Military mobilisation	0	0/1=0	0	0/1=0	NR
Nuclear bombs	1	NR	2	NR	1
Bombardments	2	0/2=0	1	0/2=0	1
Explosions	0	0/1=0	3	0/1=0	0
Dangerous substances	0	0/4=0	1	0/4=0	0
Wounded	1	0/5=0	1	1/5=20	0
First aid	0	NR	0	NR	NR
Medications	2	0/2=0	0	0/2=0	0
Euthanasia	0	NR	0	NR	NR
Dementia	0	0/1=0	0	0/1=0	NR
Greenhouse gas effect	0	NR	0	NR	NR
Traffic jams	0	NR	1	NR	0
Highways	0	0/2=0	1	0/2=0	0
Railway accidents	0	0/1=0	0	0/1=0	NR
Busses	1	NR	2	NR	0
Alcohol	0	NR	1	NR	0
Cafe's	0	NR	0	NR	NR
Factories	0	0/8=0	0	0/8=0	NR
Collective Work Agreement	0	0/3=0	1	0/3=0	0
Volcano eruption	0	NR	0	NR	NR
Deserts	1	1/1=100	0	0/1=0	0
Sunsets	0	NR	1	NR	0
Soccer	3	2/2=100	0	0/2=0	0
Swimming pools	0	NR	0	NR	NR
Ice skating	1	NR	2	NR	0
Card games	0	NR	0	NR	NR
Village fairs	0	0/1=0	0	0/1=0	NR
Fashion	1	1/1=100	0	0/1=0	0
Eduction	3	1/5=20	3	0/5=0	0
Animal trade	0	NR	0	NR	NR
Canals	0	NR	1	NR	0

Table 2. Retrieval results of experiments one and two: keyword search on manually made annotations and automatically generated annotations.

constituting the golden standard contained 97 documents describing the TV series Andere Tijden about history, and the whole collection is annotated by *history*. As all the documents deal with history, the word itself is seldom present in texts describing the content of the individual TV programs of the series, hence our automatic annotation pipeline could not achieve the recall that was obtained by querying on the manual metadata. Here again, this problem shows a point to keep in mind for improving our automatic annotation tool: we need to generate

also keywords that are relevant for the whole series of TV programs and not only for the individual ones.

An interesting point to notice, though, is that the Apolda-based annotations enables us to retrieve a document from the Art documentaries serie that was not annotated with *history* by cataloguers, but was judged relevant in our gold standard. Another possible explanation of the poor performance of the queries ran on Apolda annotations is the fact that they are quite generic, and the Top 44 queries extracted from the query logs are very specific. Thus, they are closer to what cataloguers do as manual annotation than to our automatically generated ones. This distance should be bridged by using a query expansion mechanism, option that we test in the next set of experiments.

Another thing that we can notice is that out of the total number of 199[9] retrieved documents, only 13 were overlapping between the results of the queries based on Avail or Apolda keywords. This number tends to suggest that the two approaches, rather than building one on the other, are complementary and should be run in parallel. A manual check of the retrieved documents that were part of the golden standard shows us that there is also a few overlap in terms of retrieved documents and successful queries, which reinforces our impression of complementary approaches.

3.4 Experiment three: query expansion on manual annotations

While in experiments one and two we retrieved documents based on an exact match between query and annotation concept, in experiments three and four we employ query expansion: we also retrieve documents that are annotated with concepts related to the query concept. We experiment with expansion to concepts that are one or two steps away from the query concept. The results are shown in table 3, agregating the results from experiments 3 and 4. The queries are ordered by decreasing number of hits.

In experiment three, query expansion is done on the manually created annotations. Using one-step expansion, this results in on average 7.6 documents per query. Two-step expansion retrieves four times as many documents: 28.2 on average. As expected, recall is higher than the recall in experiment 1 (37% for one-step and 58% for two-step expansion, compared to 22% in experiment 1), but precision is low (43% and 21% on average). With query expansion, documents are retrieved for 35 (one-step) or 38 (two-step) of the 44 queries. This is considerably more than in experiment 1, where documents were returned for only 19 queries.

3.5 Experiment four: query expansion on automatic semantic annotations

In experiment four, we apply query expansion to automatically generated annotations. One-step query expansion resulted in a mean of 8.6 retrieved documents,

[9] 142+57 documents, by summing up the total amount of the documents retrieved by the queries on the keywords either assigned manually or generated automatically.

two-step expansion in 40.3 documents. The combination of two-step query expansion with automatically generated annotation appears to lead to a strong increase in the number of retrieved docs. Precision is 0.29 for one-step and 0.11 for two-step expansion; recall is 0.30 and 0.48 respectively. A comparison of experiment two to the baseline showed that the Apolda annotations perform worse than the manually assigned annotations. A comparison of experiment three to experiment four paints a similar picture: both precision and recall of experiment four are lower than the query expansion results on manually created annotations in experiment three.

The results further show that where automatic annotations perform poorly when we search for an exact match with a query concept (experiment 2), they do lead to acceptable results when combined with query expansion (experiment 4). This combined strategy returns documents for 41 out of 44 queries.

The overlap between what is found using manual annotations and what is found using the automatically generated annotations is small. If expansion is limited to one step the overlap is 2.3 documents on average. Two-step expansion shows an overlap of 13.8 documents, which is relatively larger but still low. This suggests that it is worthwhile to add automatic annotations also in situations where good manual annotations are available.

The general table (table 3) give rise to some comments: theoretically, broadening the query expansion mechanism by taking into account Terms that ar at a distance 2 from the query Term could lead to one of the following outcomes:

- the query expansion heightens F score (The loss in precision is much lower than the gain in recall);
- the query expansion does not really influences Fscore (a loss in precision is compensated by a rise in recall);
- the query expansion lowers the F score (loss in precision is much larger than the gain in recall);

Interestingly enough, we see all three outcomes in our results. Therefore we cannot make a global conclusion about whether taking one only or the full two steps into account for query expansion is good or not in general, but we can see some properties of the Terms that would enable us to make a choice in some cases. For the Terms that have a high precision and low recall with one step of query expansion, like *education* or *collective work agreement*, one extra step gives a better recall without a big loss in precision. This heuristic holds for both Manual and Automatic metadata. For some Terms, we can observe the inverse: for example for *elections* or *election campaigns*, one step of query expansion already gives a low precision and 100% recall, for both Apolda and Avail. For these Terms, taking into account a second step only lowers the precision. For the third case, we cannot decide on a heuristic, as the F-measure is neither improved or jeopardised. The difference between the two first cases is strongly related to the structure of the thesaurus, which is not homogeneous: some Terms are in broad hierarchies (up to 7 levels down), whereas some Terms are not related to any other in the thesaurus. Thus, it is the results of the narrowest possible query

expansion that gives us the means to decide for the relevance of taking a broader one into account.

4 Conclusion and perspectives

We presented a set of four experiments in this paper, a baseline measurement and three possible ways to improve the retrieval results of this first baseline. One experiment involved automatic annotation and the two other experiments were based on query expansion mechanisms. It turned out that the automatic annotation setting performed worse than the baseline, when looking only at the numbers. But a qualitative look at the results showed us a very nice feature: the few overlap between the retrieved documents and the successful queries in the two settings make them quite complementary. Besides, one of the drawbacks of the automatic annotation is the genericity of the Terms extracted, which can be corrected by the query expansion mechanisms. The results of the fourth experiment confirm this hypothesis: the improvement of the automatic annotation-based setting was greater than the one based on manual annotations, with still a small overlap in the results. The complementarity of the two approaches is thus underlined, and could suggest to use them both in order to improve the search in large archives: adding automatic annotations to existing ones for a large archive could be a way of improving the accessibility of its content at low costs. The query expansion results improved the results, but also showed us the influence of the structure of the thesaurus in its performance: to get better performances by taking into account one or two steps of thesaurus relationships from a Term depends on the richness of the relationships network of that given Term. A two-times approach seems to be better suited to get the best possible results.

These experiments gave us some insights about improvements to add to our automatic annotation pipeline and query expansion mechanisms, and gave us interesting lines for future research: having a closer look at the influence of the relationships' network in the thesaurus and compensating for its non homogeneity in query expansion, using information provided by other metadata values (like the names of the people mentioned in the document) either for query expansion or semantic annotation.

Acknowledgements

This project was done in the context of the CHOICE and MUNCH projects, both part the NWO CATCH program. We would like to thank our colleagues from the Netherlands Institute for Sound and Vision who support us in our research.

References

[1] R. H. Baayen, R. Piepenbrock, and L. Gulikers. *The CELEX Lexical Database*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA:, (release 2) [cd-rom] edition, 1995.

[2] J. Bhogalb, A. Macfarlanea, and P. Smitha. A review of ontology based query expansion. *Information Processing & Management*, 42(4):866–886, July 2007.

[3] Jeen Broekstra and Arjohn Kampman. SeRQL: A second generation RDF query language. In *Proceedings of the SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pages 13–14, Amsterdam, The Netherlands, November 2003.

[4] F. Ciravegna and Y. Wilks. Designing Adaptive Information Extraction for the Semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*. IOS Press, Amsterdam, 2003.

[5] Daniel Cunliffe, Carl Taylor, and Douglas Tudhope. Query-based navigation in semantically indexed hypermedia. In *HYPERTEXT '97: Proceedings of the eighth ACM conference on Hypertext*, pages 87–95, New York, NY, USA, 1997. ACM.

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[7] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[8] Beeld & Geluid. academia collectie. http://www.academia.nl.

[9] Graeme Hirst and David St-Onge. *Lexical chains as representations of context for the detection and correction of malapropisms*, chapter 13, pages 305–332. The MIT Press, Cambridge, MA, USA, 1998.

[10] L. Hollink, A. Th. Schreiber, J. Wielemaker, and B. J. Wielinga. Semantic annotation of image collections. In *Proceedings of the K-Cap 2003 Workshop on Knowledge Markup and Semantic Annotation*, October 2003.

[11] Laura Hollink, Véronique Malaisé, and A. Th. Schreiber. Enriching a thesaurus to improve retrieval of audiovisual material. Submitted for publication.

[12] Laura Hollink, Guus Schreiber, and Bob Wielinga. Patterns of semantic relations to improve image content search. *Journal of Web Semantics*, 5:195–203, 2007.

[13] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, December 2004.

[14] K. Knight and S. Luk. Building a large-scale knowledge base for machine translation. In *the AAAI-94 Conference*, 1994.

[15] Alistair Miles and Dan Brickley. SKOS core guide. W3C working draft, November 2005. Electronic document. Accessed February 2008. Available from: http://www.w3.org/TR/swbp-skos-core-guide/.

[16] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[17] Mark van Assem, Veronique Malaise, Alistair Miles, and Guus Schreiber. A method to convert thesauri to skos. In *Proceedings of the Third European Semantic Web Conference (ESWC'06)*, number 4011 in Lecture Notes in Computer Science, pages 95–109, Budva, Montenegro, June 2006.

[18] M. Volk, B. Ripplinger, S. Vintar, Paul Buitelaar, D. Raileanu, and B. Sacaleanu. Semantic annotation for concept-based cross-language medical information retrieval. *International Journal of Medical Informatics*, 1/3(67):79–112, 2002.

	One-step Query Expanion							Two-step Query Expanion						
	Manual metadata			Automatic metadata			Overlap	Manual metadata			Automatic metadata			Overlap
Query	**retrieved**	**precision**	**estimated recall**	**retrieved**	**precision**	**estimated recall**		**retrieved**	**precision**	**estimated recall**	**retrieved**	**precision**	**estimated recall**	
Politics	17	0.75	3 / 15 = 0.2	23	0.5	4 / 15 = 0.27	9	50	0.42	5 / 15 = 0.33	59	0.35	6 / 15 = 0.4	31
Deserts	3	1	1 / 1 = 1	5	0	0 / 1 = 0	0	59	0.08	1 / 1 = 1	60	0	0 / 1 = 0	37
Education	7	1	1 / 5 = 0.2	11	0.5	1 / 5 = 0.2	3	23	0.67	4 / 5 = 0.8	47	0.43	3 / 5 = 0.6	13
Elections	13	0.33	1 / 1 = 1	18	0.25	1 / 1 = 1	6	22	0.2	1 / 1 = 1	39	0.09	1 / 1 = 1	12
Ice skating	3	NR	0 / 0 = NR	5	0	0 / 0 = NR	1	11	0	0 / 0 = NR	14	0	0 / 0 = NR	5
Army	12	0.67	2 / 7 = 0.29	23	1	4 / 7 = 0.57	10	35	0.5	4 / 7 = 0.57	59	0.4	4 / 7 = 0.57	23
History	96	1	23 / 32 = 0.72	5	1	2 / 32 = 0.06	5	106	0.88	23 / 32 = 0.72	25	0.88	7 / 32 = 0.22	12
Asylum seekers	22	0.4	2 / 2 = 1	7	NR	0 / 2 = 0	5	46	0.17	2 / 2 = 1	21	0	0 / 2 = 0	14
Nuclear bombs	1	NR	0 / 0 = NR	2	NR	0 / 0 = NR	1	7	NR	0 / 0 = NR	27	0	0 / 0 = NR	3
Medications	11	0	0 / 2 = 0	8	0	0 / 2 = 0	1	70	0	0 / 2 = 0	103	0.04	1 / 2 = 0.5	36
Islam	11	1	3 / 5 = 0.6	11	0.33	1 / 5 = 0.2	6	44	0.56	5 / 5 = 1	57	0.08	1 / 5 = 0.2	26
Busses	5	0	0 / 0 = NR	10	0	0 / 0 = NR	4	19	0	0 / 0 = NR	34	0	0 / 0 = NR	15
Fashion	10	0.5	1 / 1 = 1	2	NR	0 / 1 = 0	0	29	0.09	1 / 1 = 1	35	0	0 / 1 = 0	10
Politicians	16	0.25	1 / 9 = 0.11	27	0.5	4 / 9 = 0.44	6	68	0.33	6 / 9 = 0.67	114	0.28	8 / 9 = 0.89	46
Political parties	12	0	0 / 0 = NR	10	0	0 / 0 = NR	5	18	0	0 / 0 = NR	31	0	0 / 0 = NR	9
Factories	12	1	5 / 8 = 0.62	18	0.83	5 / 8 = 0.62	7	49	0.5	7 / 8 = 0.88	71	0.25	5 / 8 = 0.62	29
Bombardements	11	0	0 / 2 = 0	19	0.33	1 / 2 = 0.5	5	50	0.08	1 / 2 = 0.5	73	0.11	2 / 2 = 1	33
Wounded	5	NR	0 / 5 = 0	9	0.25	1 / 5 = 0.2	2	33	0	0 / 5 = 0	28	0.14	1 / 5 = 0.2	12
Soccer	3	1	2 / 3 = 0.67	0	NR	0 / 3 = 0	0	6	0.67	2 / 3 = 0.67	6	0	0 / 3 = 0	1
Election campaigns	5	0.5	1 / 1 = 1	6	1	1 / 1 = 1	2	24	0.2	1 / 1 = 1	28	0.17	1 / 1 = 1	11
Card games	1	0	0 / 0 = NR	2	NR	0 / 0 = NR	0	9	0	0 / 0 = NR	8	0	0 / 0 = NR	2
Swimming pools	2	NR	0 / 0 = NR	2	NR	0 / 0 = NR	1	52	0	0 / 0 = NR	98	0	0 / 0 = NR	28
Village fairs	1	NR	0 / 1 = 0	4	0	0 / 1 = 0	0	15	0	0 / 1 = 0	37	0	0 / 1 = 0	6
Military mobilisation	1	0	0 / 1 = 0	1	NR	0 / 1 = 0	0	20	0	0 / 1 = 0	28	0.17	1 / 1 = 1	9
Traffic jams	1	0	0 / 0 = NR	12	0	0 / 0 = NR	0	30	0	0 / 0 = NR	40	0	0 / 0 = NR	12
Collective Work Agreement	3	0.5	1 / 3 = 0.33	3	1	1 / 3 = 0.33	2	12	0.43	3 / 3 = 1	22	0.29	2 / 3 = 0.67	8
Animal trade	3	0	0 / 0 = NR	8	0	0 / 0 = NR	2	25	0	0 / 0 = NR	60	0	0 / 0 = NR	15
First aid	5	NR	0 / 0 = NR	6	0	0 / 0 = NR	0	32	0	0 / 0 = NR	35	0	0 / 0 = NR	17
Euthanasia	6	0	0 / 0 = NR	5	NR	0 / 0 = NR	1	25	0	0 / 0 = NR	26	0	0 / 0 = NR	8
Cafs	6	NR	0 / 0 = NR	16	0	0 / 0 = NR	4	63	0	0 / 0 = NR	104	0	0 / 0 = NR	36
Forming of Parliament	4	0	0 / 0 = NR	23	0	0 / 0 = NR	2	16	0	0 / 0 = NR	38	0	0 / 0 = NR	11
Political programmes	12	0	0 / 1 = 0	8	0	0 / 1 = 0	3	26	0.14	1 / 1 = 1	51	0.07	1 / 1 = 1	15
Canals	7	0	0 / 0 = NR	17	0	0 / 0 = NR	4	30	0	0 / 0 = NR	62	0	0 / 0 = NR	16
Explosions	3	1	1 / 2 = 0.5	11	0.17	1 / 2 = 0.5	2	7	1	1 / 2 = 0.5	23	0.14	1 / 2 = 0.5	2
Railways accidents	4	1	1 / 2 = 0.5	10	0.2	1 / 2 = 0.5	1	15	0.25	1 / 2 = 0.5	27	0.09	1 / 2 = 0.5	8
Sunsets	0	NR	0 / 0 = NR	1	NR	0 / 0 = NR	0	2	NR	0 / 0 = NR	11	0	0 / 0 = NR	0
Municipal councils	0	NR	0 / 2 = 0	3	1	2 / 2 = 1	0	11	0.5	1 / 2 = 0.5	15	0.5	2 / 2 = 1	3
Highways	0	NR	0 / 2 = 0	12	0.25	1 / 2 = 0.5	0	6	1	1 / 2 = 0.5	25	0.14	1 / 2 = 0.5	2
Parlementary debates	0	NR	0 / 0 = NR	10	0	0 / 0 = NR	0	25	0	0 / 0 = NR	39	0	0 / 0 = NR	9
Alcohol	0	NR	0 / 0 = NR	1	NR	0 / 0 = NR	0	13	0	0 / 0 = NR	16	0	0 / 0 = NR	5
Dementia	0	NR	0 / 1 = 0	0	NR	0 / 1 = 0	0	2	0	0 / 1 = 0	6	0	0 / 1 = 0	0
Greenhouse gas effect	0	NR	0 / 0 = NR	2	0	0 / 0 = NR	0	6	0	0 / 0 = NR	27	0	0 / 0 = NR	3
Dangerous substances	0	NR	0 / 4 = 0	1	NR	0 / 4 = 0	0	16	0	0 / 4 = 0	28	0.17	1 / 4 = 0.25	7
Volcano eruption	0	NR	0 / 0 = NR	0	NR	0 / 0 = NR	0	12	0	0 / 0 = NR	14	0	0 / 0 = NR	6

Table 3. Retrieval results of experiments two and three: query expansion on the manually made metadata and the automatically generated metadata.

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Wikipedia Link Structure and Text Mining for Semantic Relation Extraction

Towards a Huge Scale Global Web Ontology

Kotaro Nakayama, Takahiro Hara and Shojiro Nishio

Dept. of Multimedia Eng., Graduate School of Information Science and Technology
Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
TEL: +81-6-6879-4513 FAX: +81-6-6879-4514
{nakayama.kotaro, hara, nishio}@ist.osaka-u.ac.jp

Abstract. Wikipedia, a collaborative Wiki-based encyclopedia, has become a huge phenomenon among Internet users. It covers huge number of concepts of various fields such as Arts, Geography, History, Science, Sports and Games. Since it is becoming a database storing all human knowledge, Wikipedia mining is a promising approach that bridges the Semantic Web and the Social Web (a. k. a. Web 2.0). In fact, in the previous researches on Wikipedia mining, it is strongly proved that Wikipedia has a remarkable capability as a corpus for knowledge extraction, especially for relatedness measurement among concepts. However, semantic relatedness is just a numerical strength of a relation but does not have an explicit relation type. To extract inferable semantic relations with explicit relation types, we need to analyze not only the link structure but also texts in Wikipedia. In this paper, we propose a consistent approach of semantic relation extraction from Wikipedia. The method consists of three sub-processes highly optimized for Wikipedia mining; 1) fast preprocessing, 2) POS (Part Of Speech) tag tree analysis, and 3) mainstay extraction. Furthermore, our detailed evaluation proved that link structure mining improves both the accuracy and the scalability of semantic relations extraction.

1 Introduction

Wikipedia, a collaborative Wiki-based encyclopedia, has become a huge phenomenon among Internet users. According to statistics of Nature, Wikipedia is about as accurate in covering scientific topics as the Encyclopedia Britannica[1]. It covers concepts of various fields such as Arts, Geography, History, Science, Sports, Games. It contains more than 2 million articles (Oct. 2007, English Wikipedia) and it is becoming larger day by day while the largest paper-based encyclopedia Britannica contains only 65,000 articles.

As a corpus for knowledge extraction, Wikipedia's impressive characteristics are not limited to the scale, but also include the dense link structure, sense disambiguation based on URL, brief link texts and well structured sentences. The fact that these characteristics are valuable to extract accurate knowledge from Wikipedia is strongly confirmed by a number of previous researches on

Wikipedia Mining[2–5]. These researches are mainly about semantic relatedness measurements among concepts. Besides, we proposed a scalable link structure mining method to extract a huge scale association thesaurus in a previous research [4]. In that research, we developed a huge scale association thesaurus dictionary extracting a list of related terms from any given term. Further, in a number of detailed experiments, we proved that the accuracy of our association thesaurus achieved notable results. However, association thesaurus construction is just the beginning of the next ambitious research *Wikipedia Ontology*, a huge scale Web ontology automatically constructed from Wikipedia.

Semantic Wikipedia [6] is an impressive solution for developing a huge scale ontology on Wikipedia. Semantic Wikipedia is an extension of Wikipedia which allows editors to add semantic relations manually. Another interesting approach is to use Wikipedia's category tree as an ontology [7–9]. Wikipedia's categories are promising resources for ontology construction, but categories can not be used as an ontology since the structure of Wikipedia category is just a taxonomy and do not provide explicit relation types among concepts.

In contrast to these approaches, we propose a full-automated consistent approach for semantic relation extraction by mining Wikipedia article texts. Since a Wikipedia article is a set of definitive sentences, the article text is yet another valuable resource for ontology construction. The method consists of three subprocesses highly optimized for Wikipedia mining; 1) fast preprocessing, 2) POS (Part Of Speech) tag tree analysis, and 3) mainstay extraction. Furthermore, we show the potential of important sentence analysis for improving both accuracy and scalability of semantic relations extraction.

The rest of this paper is organized as follows. In section 2, we explain a number of researches on Wikipedia Mining for knowledge extraction in order to make our stance clear. In section 3, we describe our proposed integration method based on NLP and link structure mining. We describe the results of our experiments in section 4. Finally, we draw a conclusion in section 5.

2 Related Works

2.1 Wikipedia Mining

As we mentioned before, Wikipedia is an invaluable Web corpus for knowledge extraction. Researches on semantic relatedness measurement are already well conducted[2–5]. WikiRelate [5] is one of the pioneers in this research area. The algorithm finds the shortest path between categories which the concepts belong to in a category graph. As a measurement method for two given concepts, it works well. However, it is impossible to extract all related terms for all concepts because we have to search all combinations of category pairs of all concept pairs (2 million $\times$ 2 million). Furthermore, using the inversed path length as semantic relatedness is a rough method because categories do not represent semantic relations in many cases. For instance, the concept "Rook (chess)" is placed in the category "Persian loanwords" together with "Pagoda," but the relation is not semantical, it is just a navigational relation. Therefore, in our previous research, we

proposed $pfibf$ (Path Frequency - Inversed Backward Link Frequency), a scalable association thesaurus construction method to measure relatedness among concepts in Wikipedia.

2.2 Wikipedia and Web Ontology

Semantic Wikipedia[6] is an impressive predecessor of this research area. It allows editors to put additional tags to define explicit relations between concepts. For example, assume that there is a sentence written in Wiki format like this;

```
'London' is the capital city of [[England]]
```

"[[...]]" is a hyperlink tag to another article (concept) and will be translated into a hyperlink when it is shown to readers, so the readers can understand that "London" is the capital of "England." However, obviously, machines can not understand the relation type if no NLP techniques are used because the relation is written in natural language. To solve this problem, Semantic Wikipedia allows users to add special annotations like this;

```
'London' is the capital city of [[capitalof::England]]
```

Semantic Wikipedia is a promising approach for a huge scale Web ontology construction but we wish an automated approach without any additional human-effort since a Wikipedia article already includes rich semantic relations.

3 Proposed method

To achieve full-automated Web ontology construction from Wikipedia, we propose a consistent approach for semantic relation extraction by mining Wikipedia article text. Basically, the proposed method extracts semantic relations by parsing texts and analyzing the structure tree generated by a POS parser. However, parsing all sentences in an article is not efficient since an article contains both valuable sentences and non-valuable sentences by mixture. Our assumption is that it is possible to improve accuracy and scalability by analyzing only important sentences for the topic.

In this section, we describe our proposed method for semantic relation extraction from Wikipedia. The whole flow of the proposed method is performed in the following three phases;

1. Preprocessing
   (Trimming, chunking and partial tagging)
2. Parsing and POS structure tree analysis
3. Mainstay extraction.

These phases are described in detail in the following subsections.

3.1 Preprocessing

Before we parse sentences, we need to trim, chunk and segment the sentences in order to make them processable for the parser. For this aim, we usually use statical NLP tools, however these tools cannot process the Wikipedia articles correctly since the articles are written in a special syntax composed of HTML tags and special Wiki command tags such as triple quotations, brackets for hyperlinks and tables. That is why we developed our own Preprocessor for this aim. Preprocessing is accomplished in three sub steps; 1) Trimming, 2) Chunking and 3) Partial tagging.

First, the preprocessor trims a Wikipedia article to remove unnecessary information such as HTML tags and special Wiki commands. We also remove table tags because table contents are usually not sentences. However, we do not remove link tags ("[[...]]") because links in Wikipedia are explicit relations to other pages and we use this link information in the following steps.

Second, the preprocessor separates the article into sentences. Basically, an article is separated into sentences by periods ("."). However, abbreviations etc. also use ".", so the preprocessor does not separate a sentence if the following character is a small letter. This simple strategy works very well in almost all cases (Over 99%) for Wikipedia articles. Furthermore, since it is based on neither semantic nor statistic methods, the process is much faster than ordinary chunkers. After separating an article into sentences, each sentence is separated into semantic chunks (phrases). Basically, terms are separated by white space (" "), but terms are bounded if these terms are placed in quotations or link tags.

Finally, phrases in quotations and link tags are tagged as nouns to help the following parsing phase. Bounding and partial tagging are helpful information for the parsing process because one of the most difficult technical issues in parsing natural language is chunking and bounding. Especially for domain specific terms or new terms, parsers often cannot parse the sentence structure correctly.

3.2 Parsing and Structure Tree Analysis

After the preprocessing, partially tagged and chunked sentences are given. In this phase, we parse each sentence to get a structure tree and analyze that structure tree to extract relations between concepts. To parse sentences, we adopted an unlexicalized PCFG (Probabilistic Context-Free Grammars) parsing method based on the factored product model. We used the Stanford NLP parser[10] for this purpose. It can parse a sentence accurately if the sentence is trimmed, chunked and tagged correctly, even if the sentence contains hyperlink tags ("[[...]]"). Figure 1 shows the detailed illustration of this phase.

"/NN" is a special POS tag for nouns, which is added in the partial tagging process. A list of main POS (Part Of Speech) tags used in this research is shown in Table 1.

The parser gets a partially tagged sentence and constructs a structure tree for the given sentence. For instance, assume that there is a semi-tagged sentence like this: "[[Madrid]]/NN is the [[capital]]/NN and largest city of [[Spain]]/NN .

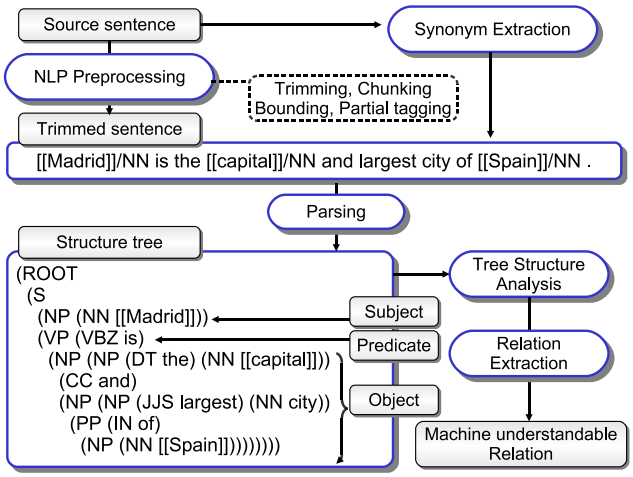


Fig. 1. Overview of the content mining process.

". The parser generates a structure tree like Figure 1. After that, the structure tree is analyzed in order to extract semantic relations in the following steps:

1. Extract "(NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))" pattern from the parsed sentence.
2. Co-reference resolution
3. For both NP, split the NP into two NP parts if the NP contains CC. After that, perform step 1 again.
4. Finally, extract the 1st NP part as a subject, VB part as a predicate, the 2nd NP part as an object.

In the first step, we extract "(NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))" and assume that the 1st NP part is the subject, the VB part is the predicate, the 2nd NP part is the object respectively. In the second step, the parser determines whether the subject is a co-reference of the topic of the article. To do that, we used two strategies mentioned in Nguyen's work[11]. The first strategy is to use the article title. If the all terms appeared in subject part are contained in the title of the article, the subject is determined as a co-reference to the topic. The second strategy is to use the most frequently used pronoun in the article. In the third step, NP will be separated if it contains CC such as "and" and "or". In the fourth step, if the 1st NP is a literal and a synonym of the concept representing the article, then the NP is replaced by the concept of the article. Finally, the first NP part is extracted as a subject, the VB part as a predicate, the 2nd NP part as an object.

The first step's POS tag pattern can be replaced by other alternatives. Currently, we prepared following three patterns for the first step.

1. (NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))
   Normal pattern. E. g. "is-a"

Table 1. POS tags.

Tag	Description
NN	singular or mass noun
NNS	Plural noun
NNP	Singular proper noun
NNPS	plural proper noun
NP	Noun phrase
VB	Base form verb
VBD	Past tense
VBZ	3rd person singular
VBP	Non 3rd person singular present
VP	Verb phrase
JJ	Adjective
CC	Conjunction, coordinating
IN	Conjunction, subordinating

2. (NP ...) (VP (NP (NP ...) (PP (IN ...) ...))
   Subordinating pattern. E. g. "is-a-part-of"
3. (NP ...) (VP (VBZ ...) (VP (VPN ...) ...))
   Passive pattern. E. g. "was-born-in"

We can prepare further POS tag patterns to improve the coverage of semantic relation extraction. However, in this research, we applied only these three basic patterns to confirm the capability of this research direction. We also extract a relation even if the object part does not contain any hyperlinks to other pages. We call it *literal object*. For example, assume that there is a sentence with the following structure tree;

```
Brescia is a city.

(S (NP (NNP [[Brescia]]))
    (VP (VBZ is)
        (NP (DT a) (NN city))))
```

The object part is "a city" but it is not a hyperlink to an article about "city" but it is just a literal. Literal object is not machine understandable but the literal information is useful depending on the application even if the meaning of the term can not be specified uniquely.

3.3 Mainstay extraction for object

By performing the process described above, we distinguish subject part and object part. After that, we need to extract mainstays for both subject part and object part respectively. A mainstay is a semantic central term (or phrase) in the part. For instance, assume that there is a following sentence and structure tree. In this phase, for the 2nd NP (object part), replace the NP by the last NN/NNS in the NP if the NP parts consist of JJ and NN/NNS. So in the case

shown below, the parser obtains "[[astronomer]]" as the mainstay of the object part.

```
Lutz_D._Schmadel is [[Germany|German]] [[astronomer]].

(S (NP (NN Lutz_D._Schmadel)
    (VP (VBZ is)
        (NP (NN [[Germany|German]]) (NN [[astronomer]])
))))
```

The 2st NP consists of two NN and both of them have a hyperlink to other pages. The 1st NN has a link to a country "Germany" but it is used as a adjective, so it can not be a mainstay of the object part. So in this case, we have to obtain "[[astronomer]]" as the subject.

3.4 A Parsing Strategy: ISP

We conducted a small number of experiments using the above algorithms and realized that parsing all sentences is quite time consuming work and sometimes returns irrelevant results. More detailed preliminary investigation and experiment showed that it is possible to reduce the calculation and improve the accuracy of semantic relation extraction by filtering non important sentences in the article.

The Important Sentence Parsing (ISP) method parses sentences that seem important for an article (concept). We used $pfibf$ (Path Frequency - Inversed Backward link Frequency), an association thesaurus construction method we proposed in a previous research[4][1]. An association thesaurus is a set of terms and association relationships among them. The ISP method uses the association thesaurus to detect whether a sentence is important to an article or not. In this section, we describe the essentials of the method.

Basic Strategy of pfibf Wikipedia consist of a set of articles (concepts) and hyperlinks among them, thus they can be expressed by a graph $G = \{V, E\}$ (V: set of articles, E: set of hyperlinks). Let us consider how we can measure the relatedness between any pair of articles (v_i, v_j). The relatedness is assumed to be strongly affected by the following two factors:

- the number of paths from article v_i to v_j,
- the length of each path from article v_i to v_j.

The relatedness is strong if there are many paths (sharing of many intermediate articles) between two articles. In addition, the relatedness is affected by the path length. In other words, if the articles are placed closely together in the graph G and share hyperlinks to same articles, the relatedness is estimated to be higher than between farther ones. Therefore, if all paths from v_i to v_j are given

[1] The method name was $lfibf$ in the past and was changed to $pfibf$

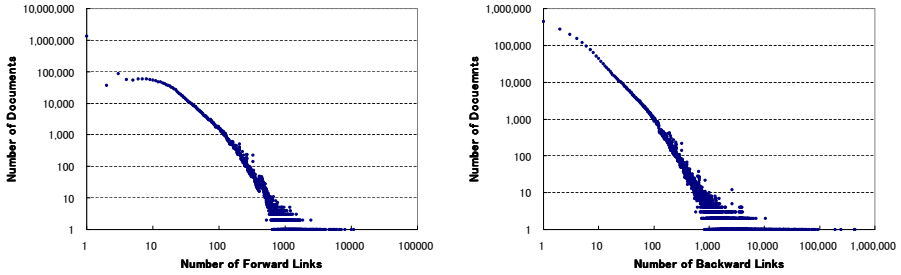


Fig. 2. Zipf distribution of the Wikipedia link structure.

as $T = \{t_1, t_2, ..., t_n\}$, we define the relatedness pf (path frequency) between them as follows:

$$pf(v_i, v_j) = \sum_{k=1}^{n} \frac{1}{d(|t_k|)}. \tag{1}$$

$d()$ denotes a function which increases the value according to the length of path t_k. A monotonous increasing function such as a the logarithm function can be used for $d()$.

In addition, the number of links between individual articles is also estimated as a factor of relatedness because the dense link structure is one of the most interesting characteristics of Wikipedia. *Dense* means that Wikipedia has a lot of *inner links*, links from pages in Wikipedia to other pages in Wikipedia. This means that articles are strongly connected by many hyperlinks. Let us show statistics of link structure analysis for Wikipedia that we investigated. Figure 2 shows the distribution of both backward links and forward links. Our statistics unveiled that both forward links and backward links have typical Zipf distribution, containing a few nodes that have a very high degree and many with low degree.

The statistics shows that we need to consider the characteristics to design algorithms for analyzing the Wikipedia link structure. For instance, assume that there is an article which is referred to from many other articles. This article would have a lot of short paths from many articles. This means that it has a strong relatedness to many articles if we used only pf. However, this kind of articles must be considered as a general concepts, and the importance of general concepts is not high in most cases. Therefore, we must consider the inversed backward link frequency ibf as follows in addition to the two factors above. We therefore define the algorithm $pfibf$ as follows:

$$ibf(v_j) = \log \frac{N}{bf(v_j)}, \tag{2}$$

$$pfibf(v_i, v_j) = pf(v_i, v_j) \quad ibf(v_j). \tag{3}$$

N denotes the total number of articles and $bf(v_j)$ denotes the number of backward links of the page v_j. This means a page which shares forward/backward links with a specific page but not does not share with other pages, has a high $pfibf$.

Dual Binary Tree (DBT) The counting of all paths between all pairs of articles in a huge graph is a computational resource consuming work. Thus, making it efficient is a serious issue on Wikipedia mining. Using adjacency matrices and multiplication is not a clever idea because of the low scalability. Wikipedia has more than 2 million articles, thus we need several terabytes just for storing data. Further, we need unimaginably much time to calculate the multiplication because the order is $O(N^3)$. However, a large number of elements in the adjacency matrix of a Web site are zero, thus effective compression data structures and analysis methods are the key to achieve high scalability on Wikipedia mining. Therefore, we propose an efficient data structure named *Dual binary tree* (DBT) and a multiplication algorithm for the DBT.

Since the adjacency matrix of a Web site link structure is a sparse matrix (almost all elements are zero), the DBT stores only the non-zero elements for data compression. The DBT consists of two types of binary trees; i-tree and j-tree. Each element in the i-tree corresponds to a row in the adjacency matrix and each i-tree element stores a pointer to the root of a j-tree. This means that the DBT consists of totally $N+1$ (1 i-tree and N j-trees) binary trees. The point is that operations for both getting and storing data are very fast because the number of steps is in both cases $O(logN)$.
The function $j\text{-}Tree(i)$ extracts all elements in the ith row of the adjacency matrix A. $a_{j,k}$ denotes the element in the jth row and kth column of the matrix. The first loop will be executed N times, but the numbers of cycles of the second and third loop depend on the average link number M. Thus the total number of steps is $O(NlogN)$ $O(M^2)$. Further, our statistics unveiled that M is constantly 20 to 40 in Wikipedia in spite of the evolvement of the matrix size N. Finally, the result is stored in another DBT R.

We conducted a benchmark test for the DBT and the multiplication algorithm compared with conventional methods. We used GNU Octave (with ATLAS library), one of the most effective numerical algebra implementations, as a base line method because a study[12] has proved that the performance for sparse matrix operations on Octave is better than that of Matlab, the most popular and well tuned numeric computation environment all over the world. In [12], it is described that "Octave implements a polymorphic solver for sparse matrices, where the exact solver used to factorize the matrix, depends on the properties of the sparse matrix itself." Table 2 shows the result of the performance comparison of $N \times N$ matrix multiplication with the density D.

N is the number of rows, equivalent to the number of columns in the adjacency matrix. Density D is the rate of non-zero elements in the matrix. It can be calculated by the following formula:

$$D = \frac{Number\ of\ non\ zero\ elements}{N^2}. \tag{4}$$

Table 2. Benchmark of multiplication.

Order (N)	Density (D)	Avg. link (M)	Octave	DBT
10,000	1.e-5	0.1	0.62	0.01
10,000	1.e-4	1	0.63	0.09
10,000	1.e-3	10	0.87	5.18
15,000	1.e-5	0.15	1.39	0.01
15,000	1.e-4	1.5	1.42	0.28
15,000	1.e-3	15	2.15	17.74
20,000	1.e-5	0.2	2.49	0.02
20,000	1.e-4	2	2.55	0.62
20,000	1.e-3	20	4.72	42.72
50,000	1.e-5	0.5	74.94	0.14
50,000	1.e-4	5	75.25	6.24

(Unit: sec.)

The result of the benchmark test proved that the DBT is very beneficial for multiplication on a matrix whose density is less than 1.e-4. Further, as the size of N increases, it also advantages the performance.

English Wikipedia has 3.8 million pages (Sept. 2006, including redirect pages), 73.3 million links and the density is about 5.e-6. This means that the adjacency matrix of Wikipedia is a typical sparse matrix with a huge number of rows and columns. Therefore, the DBT is more suitable for Wikipedia Mining than other numerical algebra implementations such as Octave. What we should consider, however, is that the DBT is suitable only while the matrix is sparse enough. Repeated multiplication makes the matrix dense, thus after each multiplication, all elements except top k ranked elements in each row should be removed to keep the sparsity of the matrix.

pfibf with DBT In this section, we describe the concrete flow of $pfibf$ calculation using a DBT. Since $pfibf$ analyzes both forward and backward links of the articles, first we calculate A' by adding A and the transpose matrix A^T as follows:

$$A' = A + A^T. \tag{5}$$

By calculating the power of A', we can extract the number of paths for any pair of articles in n-hop range. An element $a'^n_{i,j}$ in matrix A'^n denotes the number of paths from article v_i to article v_j whose length is n. However, before calculating A'^n, each element in A should be replaced by the following formula to approximate ibf (Formula (2)):

$$a'_{i,j} \leftarrow a'_{i,j} \quad log\frac{N}{|B_{v_j}|}. \tag{6}$$

$|B_{v_j}|$ denotes the number of backward links of article v_j. Finally, we can extract the $pfibf$ for any pair by adding the matrices A'^1, A'^2, ... , A'^n as follows:

$$pfibf(i,j) = \sum_{l=1}^{n} \frac{1}{d(n)} \quad a'^{l}_{i,j}. \tag{7}$$

$d()$ denotes a monotonically increasing function such as a logarithm function which increases the value according to the length of path n.

FB Weighting After a number of experiments to evaluate the accuracy of $pfibf$, we realized that the accuracy decreased in particular situations. Then, after conducting further experiments in order to detect the cause, we finally realized that the accuracy of general term analysis is worse than the accuracy of domain specific terms. General terms have the following characteristics:

- They have a lot of backward links,
- They are referred to from various topic-ranges,
- The content is trustful because it is usually edited by many authorities.

General terms, such as "United states," "Marriage" and "World War II," are referred to from various articles in various topic ranges. This means that the backward link analysis cannot be converged because the topic locality is weaker than in domain-specific terms such as "Microsoft" and "iPod." Although the backward link analysis is not convergent, the forward link analysis is effective because the contents are trustful and usually edited by many authorities.

In contrast to this, domain-specific terms have a much stronger topic locality. Although they have less links from other pages and the contents are sometimes not trustful, each link from other pages is topically related to the content. Therefore, we developed the *FB weighting* method which flexibly changes the weight of the forward link analysis and backward link analysis as follows:

$$W_b(|B_d|) = 0.5/(|B_d|^{\alpha}), \tag{8}$$

$$W_f(|B_d|) = 1 - W_b(|B_d|). \tag{9}$$

$|B_d|$ is the backward link number of document d. The constant must be optimized according to the environment. After a number of experiments, an value of about 0.05 was recognized to be suitable for the link structure of Wikipedia. The weight W_b is multiplied for each element on A and W_f for A^T as well. Thus formula (5) must be modified into the following formula (10):

$$A' = W_f \quad A + W_b \quad A^T. \tag{10}$$

Table 3 shows an example of an association thesaurus constructed by $pfibf$ with FB weighting. For example, when analyzing the article "Google," associated concepts such as "Search engine", "PageRank" and "Google search" are extracted from the association thesaurus.

We also conducted several experiments in the previous research[4] and the results proved that the FB Weighting method is significantly more effective

Table 3. Sample of queries and terms extracted by pfibf with FB weighting.

Query	Extracted association terms		
Sports	Basketball	Baseball	Volleyball
Microsoft	MS Windows	OS	MS Office
Apple Inc.	Macintosh	Mac OS X	iPod
iPod	Apple Inc.	iPod mini	iTunes
Book	Library	Diamond Sutra	Printing
Google	Search engine	PageRank	Google search
Horse	Rodeo	Cowboy	Horse-racing
Film	Actor	Television	United States
DNA	RNA	Protein	Genetics
Canada	Ontario	Quebec	Toronto

for association thesaurus construction than other traditional methods such as link co-occurrence analysis and TF-IDF. Especially for domain-specific terms, it achieved remarkable accuracy.

Important Sentence Detection By using $pfibf$, a set of important links for each article (concept) in Wikipedia can be extracted. ISP detects important sentences in a page from sentences containing important words/phrases for the page. It crawls all sentences in the article to extract sentences containing links to the associated concepts. The extracted sentences are then parsed as the important sentences in the article. For each links in a sentence, the parser calculates $pfibf$ and the max value denotes the importance of the sentence. The importance can be used for filtering unimportant sentences by specifying thresholds.

4 Experiments and discussion

First, we analyzed the whole Wikipedia link structure and gathered 65,391 articles (pages) that have more than 100 backward links, to filter noisy pages. After that, we randomly selected about 100 articles as a test set. We applied Preprocessing, parsing and structure tree analysis proposed in Section 3.

Table 4 shows some examples of explicit relations extracted by our method. We can see that it extracts various relations such as “borders,” “hosted” and “separates”. However, machines cannot understand the meaning “borders” without any instruction from humans. So, in order to make the predicate part machine understandable, we have to define the relation between predicates. For example, “is” and “was” have the same meaning but the tense is different. By giving this kind of knowledge, it will be inferable relations. We believe that the amount of relations among verbs are limited compared with relation between nouns.

Table 5 shows examples of literal relations extracted by our method. We realized that literal objects are often extracted when the object part is a too common word such as “city” or “town.” We believe that the reason for this lack

Table 4. Examples of extracted explicit relations.

Subject	Predicate	Object
Apple	is	Fruit
Bird	is	Homeothermic
Bird	is	Biped
Cat	is	Mammal
Computer	is	Machine
Isola_d'Asti	is	Comune
Jimmy_Snuka	is	Professional_wrestler
Karwasra	is	Gotra
Mineral_County,_Colorado	is	County
Nava_de_Francia	is	municipality
Sharon_Stone	is	Model
Sharon_Stone	is	Film_producer
Al Capone	was	gangster
Gaius Valerius Catullus	was founded by	Vladimir Lenin
Colorado	is one of	U.S. states
Quartz	is one of	mineral
Djibouti	is bordered by	Eritrea
Djibouti	is bordered by	Ethiopia
Djibouti	is bordered by	Somaliland

of links is just because of the difficulty for making links for a lot of common words. To make these literal relations machine understandable, we have to specify the meaning of these too common words.

Turning now to the accuracy of our proposed method. We realized that some irrelevant semantic relations have been extracted. For example, the semantic relation "[[Niagara Falls]] (carry) vehicles" is extracted from a sentence "it carries vehicles, trains, and pedestrians between Canada." However, the main subject of this sentence is "Whirlpool Rapids Bridge" that appeared in the previous sentence. This is due to the limitation of the co-reference resolution method based on frequent pronouns. Sometimes, "it" or "she/he" are most frequent pronouns but they are not used for the main topic of the article. To confirm the capability of ISP to filter irrelevant semantic relations, we evaluated the precision by specifying thresholds to filter unimportant sentences. Figure 3 shows the result of this evaluation. It is clear that the importance of sentence affects accuracy of semantic relation extraction. This means that our conviction that the sentence importance calculated by link structure analysis is helpful information to filter inaccurate semantic relations is strongly confirmed.

5 Conclusion

In this paper, we proved that Wikipedia is an invaluable corpus for semantic relation extraction by showing both detailed characteristics of Wikipedia and the effectiveness of our proposed method. Furthermore, the results showed that the

Table 5. Examples of extracted literal relations.

Subject	Predicate	Object
Taranto	is	Coastal city
The_Isley_Brothers	is	Black music group
Toronto_Islands	is	Chain
Mauritania	is	Country
Mauritania	is	Country
Ilirska_Bistrica	is	Town
Ilirska_Bistrica	is	Municipality
Brescia	is	City
Bolsheviks	were	Faction
Gaius Valerius Catullus	was	poet

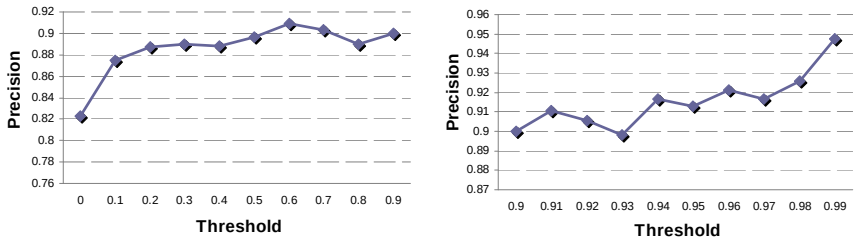


Fig. 3. Precision of ISP by filtering thresholds.

parsing strategies can improve the accuracy and scalability of semantic relation extraction.

More than anything else, the important thing this paper is trying to show is the possibility and capability of semantic relation extraction using Wikipedia knowledge. We believe that this direction will be an influential approach for Semantic Web in near future since it has great capability for constructing a global ontology. The extracted association thesaurus and semantic relations are available on our Web site.

- Wikipedia Lab.
  http://wikipedia-lab.org
- Wikipedia Thesaurus
  http://wikipedia-lab.org:8080/WikipediaThesaurusV2
- Wikipedia Ontology
  http://wikipedia-lab.org:8080/WikipediaOntology

The concrete results will be a strong evidence of the capability of this approach since other Wikipedia mining researches do not provide concrete results on the WWW in most cases. Our next step is to apply the extracted semantic relations to Semantic Web applications (Esp. Semantic Web search). To do that, we need further coverage of relations by enhancing the POS tag analysis patterns and mappings among relations.

6 Acknowledgment

This research was supported in part by Grant-in-Aid on Priority Areas (18049050), and by the Microsoft Research IJARC Core Project. We appreciate helpful comments and advices from Prof. Yutaka Matsuo, the University of Tokyo.

References

1. J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, pp. 900–901, 2005.
2. E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis.," in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 1606–1611, 2007.
3. D. Milne, O. Medelyan, and I. H. Witten, "Mining domain-specific thesauri from wikipedia: A case study," in *Proc. of ACM International Conference on Web Intelligence (WI'06)*, pp. 442–448, 2006.
4. K. Nakayama, T. Hara, and S. Nishio, "Wikipedia mining for an association web thesaurus construction," in *Proc. of IEEE International Conference on Web Information Systems Engineering (WISE 2007)*, pp. 322–334, 2007.
5. M. Strube and S. Ponzetto, "WikiRelate! Computing semantic relatedness using Wikipedia," in *Proc. of National Conference on Artificial Intelligence (AAAI-06)*, pp. 1419–1424, July 2006.
6. M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proc. of International Conference on World Wide Web (WWW 2006)*, pp. 585–594, 2006.
7. S. Chernov, T. Iofciu, W. Nejdl, and X. Zhou, "Extracting semantics relationships between wikipedia categories," in *Proc. of Workshop on Semantic Wikis (SemWiki 2006)*, 2006.
8. D. N. Milne, O. Medelyan, and I. H. Witten, "Mining domain-specific thesauri from wikipedia: A case study," in *Web Intelligence*, pp. 442–448, 2006.
9. F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 697–706, ACM, 2007.
10. D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proc. of Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 423–430, 2003.
11. D. P. T. Nguyen, Y. Matsuo, and M. Ishizuka, "Relation extraction from wikipedia using subtree mining," in *Proc. of National Conference on Artificial Intelligence (AAAI-07)*, pp. 1414–1420, 2007.
12. D. Bateman and A. Adler, "Sparse matrix implementation in octave," 2006.

QuiKey – a Demo

Heiko Haller

Forschungszentrum Informatik (FZI), Germany
heiko.haller@fzi.de

Abstract. QuiKey is a light-weight tool that can act as an interactive command-line for a semantic knowledge base. It focuses on highest interaction-efficiency to browse, query and author graph-based knowledge bases in a step-by-step manner. It combines ideas of simple interaction techniques like auto-completion, command interpreters and faceted browsing and integrates them to a new interaction concept. It is being developed in the Semantic Desktop project nepomuk [1]. Despite its versatility, QuiKey needs very little screen space, which also makes it a candidate for future mobile use.

1 Idea

QuiKey is inspired by *quicksilver* [2], a kind of advanced application launcher for the Mac that has gained a lot of popularity due to its versatility and efficiency. With very few keystrokes, quicksilver can open files and applications and trigger a large variety of common actions not only on any files but also on specific information objects: Depending on the plug-ins installed, it can e. g. manage play-lists in iTunes, send files via e-mail or dial a contact's phone number.

In knowledge bases like a semantic desktop, knowledge is typically be modelled in a formal and fine granular way. QuiKey provides a light-weight generic UI for browsing and editing them in such fine-granular ways. It also brings simple ways of constructing structured queries to not-so-technically-advanced users.

2 Examples / Interaction

Fig. 1. Mock-up showing how both a new statement and relation are added.

[1] http://nepomuk.semanticdesktop.org/
[2] http://blacktree.com/?quicksilver

QuiKey is organised around the notion of *parts*. A part can be an existing item, a relation, a new text string or a command. Depending on the number, order and types of parts entered, it is decided what action to take.

Authoring To add a new text item to the knowledge base, it is enough to just type the text into the QuiKey console and press enter. To make statements about existing items, the statement can just be entered in a subject-predicate-object fashion, separated by tab-keys. So. e. g.

```
Claudia Stern→works for→SAP Research[enter]
```

would just add that statement. Only that the user would not even have to type in the whole labels because parts that are already known can be chosen from a list an auto-completion manner with the best fitting NameItem pre-selected. So for this example it is actually enough to type in

```
Ster→wor→SAP R[enter]
```

If not all three parts in such a statement are known strings, the respective items or relations are automatically added to the knowledge base– c. f. Fig. 1. Like this, a knowledge graph can be woven in single simple steps in an ad-hoc fashion. Apart from requiring the user to think in triple patterns, cognitive overhead is reduced to a minimum since additional actions and decisions that are not part of the actual content, like starting an application, opening a new document, finding the right place to add or change content, choosing a file name and location, are not necessary anymore.

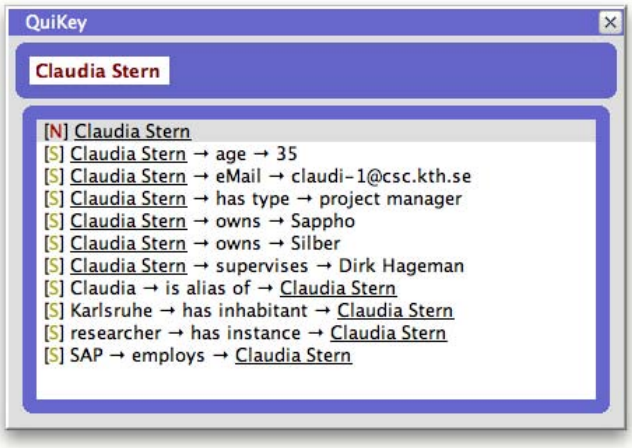


Fig. 2. Screen shot of the current QuiKey implementation showing a list of statements about "Claudia Stern".

Browsing Simply navigating the knowledge base through its graph structure is done with QuiKey without even changing into a different mode: when a part has been selected, before the user types anything new to select the next part, existing contents that fit the part pattern are already displayed in the suggestion area and can be browsed in a way similar to faceted browsing (s. Fig. 2).

Queries Constructing complex, possibly nested queries is difficult for non-expert users and every slight error in the syntax makes the whole query fail or return unintended results. QuiKey tackles two common problems:

a) *Misspellings and syntax errors* are largely avoided because instead of requiring the user to write a whole query in some complicated syntax which is parsed later on, in QuiKey the query is constructed interactively, selecting from existing items and without the need of syntactical characters.

b) To facilitate modular construction of complex queries in a step-by-step manner, each query can be saved and referred to as a special query item. Simple query items can be constructed with the easy pattern shown in the two examples in Fig. 3:

```
Dirk→knows→?DirksFriends[enter]
```

creates a new query item that represents a query about everyone that 'Dirk' 'knows'.

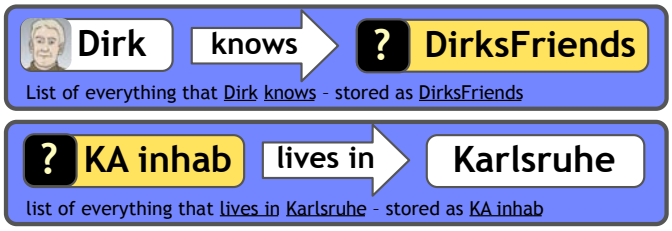


Fig. 3. Mock-up of simple elementary queries including generic descriptions of their meaning.

Chained queries like "Who works on a project funded by the EU?" can be asked as shown in Fig. 4. Note that a node or variable between *works on* and *is funded by*, like it is necessary e. g. in SPARQL, can be omitted here since the meaning is clear from the pattern of two relation names after each other. Furthermore, it is consistent with reaching the same query by browsing:

```
EU→funds→
```

would result in a list of everything funded by the EU. Continuing this pattern with

```
EU→funds→has member→
```

would result in a query of all members of these things funded by the EU. Like this browsing and constructing queries becomes the same.

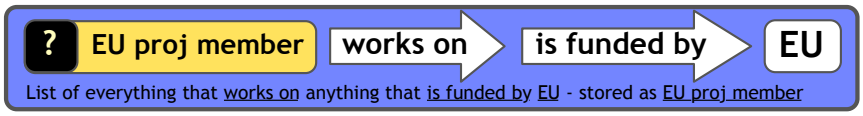


Fig. 4. Mock-up of a chained query including generic description of its meaning.

More complex queries can be constructed by combining existing query items like the examples in Fig. 5.

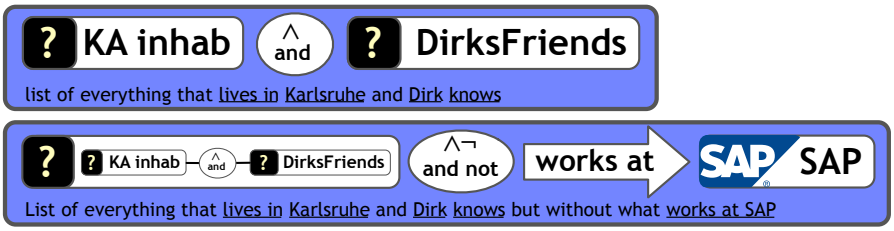


Fig. 5. Mock-up modular queries combining previously existing query items including generic descriptions of their meaning.

3 Technical Background

The current implementation of QuiKey is built on top of CDS ("Conceptual Data Structures"), a lightweight top-level ontology designed to bridge the gap between unstructured content like informal notes and formal semantics like ontologies. CDS allows the use of vague semantics by subsuming arbitrary specific relation types under more general ones. CDS is described in [1] and [2]. The CDS-*framework* which we use as a back-end is a CDS-API in Java, which is designed to serve as a back-end for semantic personal knowledge management tools. It is described in detail in [3].

In CDS there are four basic kinds of items that can be freely added, edited and queried:

ContentItems that can hold html-like content
NameItems that are characterised by a unique, typically short string – comparable to e. g. a file name or a wiki page name
relations i. e. types of relations that can be stated between items (plus, in CDS every relation type has an inverse assigned)

statements in the form of subject–predicate–object or rather item–relation–item(in CDS statements are addressable as first-order citizens)

While the general QuiKey approach could be used with any kind of graph-based knowledge base, the CDS framework is especially suited for QuiKey, since NameItems can be used to easily identify items with a unique string using auto-completion mechanisms. And since every relation has an inverse relation defined, any statement can be made and browsed / queried in both directions.

QuiKey will soon also be integrated into the *visual knowledge workbench* of the nepomuk project, e.g. to open existing items directly in the visual iMapping browser [4] or to 'summon' an existing item into a specific place in a map.

The currently used CDS back-end converts the queries to SPARQL. However, since the expressiveness of QuiKey's queries does not exceed EL++[5], there could also be optimised implementations that scale to large knowledge bases without slowing down user experience.

Acknowledgments:

Research reported in this paper has been financed by the EU in the Social Semantic Desktop project NEPOMUK (IST-FP6-027705).

References

1. Völkel, M., Haller, H.: Conceptual data structures (cds) – towards an ontology for semi-formal articulation of personal knowledge. In: Proc. of the 14th International Conference on Conceptual Structures 2006, Aalborg University - Denmark (2006)
2. Völkel, M., Haller, H., Abecker, A.: Modelling higher-level thought structures - method and tool. In: Proceedings of Workshop on Foundations and Applications of the Social Semantic Desktop. (2007)
3. Völkel, M., Haller, H., Bolinder, W., Davis, B., Edlund, H., Groth, K., Gudjonsdottir, R., Kotelnikov, M., Lannerö, P., Lundquist, S., Sogrin, M., Sundblad, Y., Westerlund, B.: Conceptual data structure tools. Deliverable 1.2, nepomuk consortium (2008)
4. Haller, H.: imapping – a graphical approach to semi-structured knowledge modelling. In Rutledge, L., ed.: Proceedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI2006). (2006) Poster and extended abstract presented at the The 3rd International Semantic Web User Interaction Workshop.
5. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for horn description logics. In: Proceedings of the 22nd AAAI Conference on Artficial Intelligence, Vancouver, British Columbia, Canada, AAAI Press (2007) 452–457

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Microsearch: An Interface for Semantic Search

Peter Mika

Yahoo! Research
Ocata 1, 08003 Barcelona, Spain
pmika@yahoo-inc.com

Abstract. In this paper we discuss the potential for semantic search and focus on the most immediate problem toward its realization: the problem of the sparsity and relatively low quality of embedded metadata. We suggest that a part of the solution is to expose users to embedded metadata as part of their daily activity of searching the Web. We present the publicly available microsearch system which enriches search result presentation with metadata extracted from search results and report on some of the early feedback we have received.

1 Introduction

The current generation of search engines is severely limited in its understanding of the user's intent and the Web's content and consequently in matching the needs for information with the vast supply of resources on the Web.

For Information Retrieval purposes, both queries and documents are typically treated at a word or gram level, with minimal language processing involved. In other words, the search engine is missing a semantic-level understanding of the query or the content: it is as if one would try to understand the content of a document by picking out the most commonly occurring or underlined words.

The fact that search is still considered as a technology that largely 'works' has to do with a number of factors. First, a number of queries are easy in the sense that they belong to the class of navigational queries, where there is a single known item sought, e.g. 'air france'. At the other end of the scale, in answering very broad queries (such as 'hotel paris') there are typically a vast array of similarly relevant documents.

Second, search engines have managed to mask their limitations by a number of techniques. Foremost, the unit of retrieval is limited to individual documents, as the statistical methods applied degrade quickly when considering smaller units such as paragraphs or sentences. Situations of ambiguity are solved by applying measures such as PageRank which automatically zoom in on the most common interpretation of a query. (For example, the query 'George Bush' returns results related to the famous politician, irrespective of the number of persons named George Bush.) Further, users are aided in refining their query, although not on the basis of an explicit understanding of a query, but on the basis of the refinements made by other users starting with the same query.

Yet there are a number of situations where one can clearly see the limits of a syntax-based approach to search. Here we list but some of the examples. Interestingly, users have adapted to the limitations of search engines to the extent that some of these queries are rarely entered anymore.

- The *ambiguous queries* mentioned above are the most straightforward examples, in that it becomes almost impossible to find an object that relates to the secondary sense of a term, in case a dominant sense exists. In the example, consider searching for George Bush, the beer brewer. Note also that in widely scoped information spaces nearly all terms are ambiguous.
- The capabilities of *computational advertising*, which is largely also an information retrieval problem (i.e. the retrieval of the matching ads from a fixed inventory), are clearly impacted because of the greater sparsity of advertizements.
- Search engines are also unable to perform *queries on descriptions of objects*, where no clear key exists. For example, one might want try to search for the author of this paper as "semantic web researcher working for yahoo". A typical, and much important example of this category is product search. For example, search engines are unable to look for music players with at least 4GB of RAM without understanding what a music player is, what it's characteristics are, etc.
- Current search technology is also unable to satisfy any complex queries requiring *information integration* such as analysis, prediction, scheduling etc. An example of such integration-based tasks is opinion mining regarding products or services. (While there have been some successes in opinion mining with pure sentiment analysis, it is often the case that one would like to know what specific aspects of a product or service are being described in positive or negative terms.) Information integration is not possible without structured representations of content.
- Lastly, *multimedia queries* are also difficult to answer as multimedia objects are typically described with only a few keywords (tagging) or sentences. This is typically too little text for the statistical methods of IR to be effective.

Clearly, these problems cannot be addressed without moving toward *semantic search*, which we define as information retrieval with the capabilities to understand the user's intent and the Web's content at a much deeper, conceptual level. We believe that building on the results from Information Retrieval and the Semantic Web, with important contributions from the field of Natural Language Processing, semantic search could become a reality in the coming years [2]. However, before we could move to consider methods for semantic search we have to face the problems related to the *sparsity and low quality of metadata* on the Semantic Web.

Even after ten years of the publishing of the first Semantic Web standards, the technology has largely failed to impact the way information is encoded on the Web. In fact, in recent years the focus has shifted from a vision of the Annotated Web that characterized early Semantic Web research to one that is

focused almost exclusively on Linked Data, i.e. on databases instead of documents. Interestingly, at the point where Semantic Web researchers have almost but given up on the idea of an annotated web, significant advances have been made in this area by the Web 2.0 movement, in particular through the introduction of microformats. Microformats lower the barrier for manually authoring metadata or implementing metadata production by simplifying the knowledge representation paradigm and reducing choice. (In particular, each microformat is a fixed vocabulary designed to describe one information type without possibilities of extension. From the user's perspective this makes it almost trivial to choose and follow a format.) Microformats have also earned the support of major participants in the Web industry with Yahoo! alone publishing over one billion microformat enabled pages. Encouraged by this development, the W3C has also moved forward rapidly with the standardization of RDFa, a format for embedding RDF into XML (including XHTML) in a similar way that microformats are encoded in HTML. Yet we can still consider metadata sparse when considering the fraction of metadata-enabled web pages.

The quality of embedded metadata is also of concern as it will have significant impact on any semantic search effort. While Linked Data is typically exposed in fully automated ways and thus it is no lower quality than the original data, manually created metadata suffers a loss of quality at the point of encoding. Unfortunately, users expect that the same way browsers tolerate errors in HTML markup, mistakes made during microformat authoring would also be easily corrected automatically by the processing agent. However, while forgetting to close an angled bracket in HTML is relatively easy to correct, incorrect microformat markup is much harder and often impossible to spot by automated means, e.g. in cases where the wrong class is applied to a particular information as a result of forgetting to close a DIV or SPAN element.[1] This situation is likely to be worsened by further complexity introduced in RDFa.

In our judgment the problems of sparsity and data quality on the Semantic Web are tied together by a common solution: bringing metadata to the surface of the Web. At the moment the Semantic Web is what many refer to as a *shadow web* where users almost never see metadata displayed in any shape or form. This means that users no see incentive to create new metadata. Just as importantly, users have no ways to correct incorrect metadata as this would require the mistakes to be visible. Last, to unleash collaborative effects it should be possible to correct erroneous metadata by any user not just the user who created and maintains the page with the incorrect metadata.

In this paper we present microsearch, a research prototype that demonstrates ways to bring metadata to the surface by incorporating it in the result display of a search engine. Microsearch also showcases some of the early benefits of

[1] In practice, auto-correction of microformat data is not even attempted: both microformat and RDFa data are typically processed by means of XSLT typically after running Tidy on the page. While Tidy corrects HTML markup it is not concerned with microformats and the XSLT stylesheets used are engineered for correct markup.

metadata-enabled search engines when it comes to information integration and spatial-temporal visualization.

2 The microsearch system

The microsearch system enriches the search experience by visualizing embedded metadata. First, for result pages that contain embedded metadata a summary of the data is presented as part of the abstract ('snippet'). Further, the user can take direct actions based on the semantics of the information, such as adding an address to his/her local address book, starting to compose an email or directly dialling a telephone number. Second, it is often possible to relate pages through metadata in which case the related pages can be visually grouped together. Figure 1 illustrates these features using the query 'ivan herman'. (Ivan Herman is W3C's Semantic Web Activity Lead.)

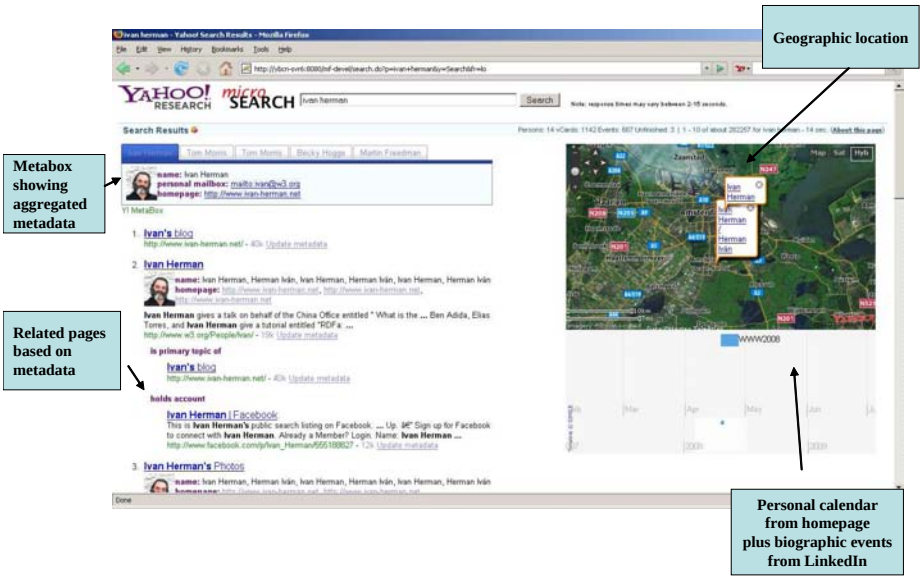


Fig. 1. Result display for the query *ivan herman*.

Microsearch also demonstrates the promise of semantic search when it comes to the aggregation of information across result pages. A Yahoo! Map shows resources which have a geographic relevance and for which a location is given (and this location can be successfully geocoded). At the moment this is limited to foaf:Person instances with geographic coordinates and vCards for persons and organizations in which case the address is geocoded using the Yahoo! Maps API itself. Figure 2 shows this feature for the query 'peter site:flickr.com', i.e. for

all the users named Peter on the Flickr web site. The map zooms and pans automatically in order to include all the nodes being visualized. Similarly, a timeline shows event information when available using the SIMILE Timeline API. The timeline can show both points in time as well as periods in time such as biographical information from profile sites such as LinkedIn. The scale of the timeline is fixed, but two bands are shown to allow scrolling by month and by year. Also, the timeline is centered on the last event displayed (which may be in the future). Figure 3 shows this feature for the query 'san francisco conference'. At the moment the map and the timeline are shown for all queries, but it would be easy to change this behaviour in a way that only relevant modules are shown.

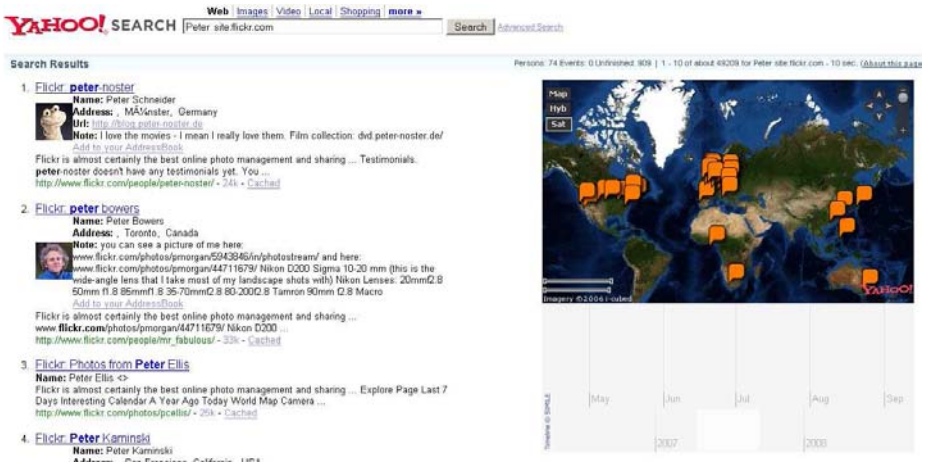


Fig. 2. Result display for the query *peter site:flickr.com*.

Figure 4 shows an overview of the architecture of the microsearch system. The dynamic behaviour of the system is as follows. On the microsearch website [2], users initiate a search the same way they would with Yahoo!'s main search engine. The query is issued against the search engine and the top results are retrieved for display. Besides retrieving regular search results, we also retrieve the top results that are known to contain certain types of microformat data. In a next step, the metadata is extracted from the displayed results and the pages that are known to microformat results. (The reason we process the display pages is that not all forms of embedded metadata are available from the search index.) After running Tidy on the pages, the extractor (known as the sponger) extracts popular microformats, linked RDF and RDFa data. (Support for GRDDL is among the future work.)

[2] `http://yr-bcn.es/demos/microsearch/`

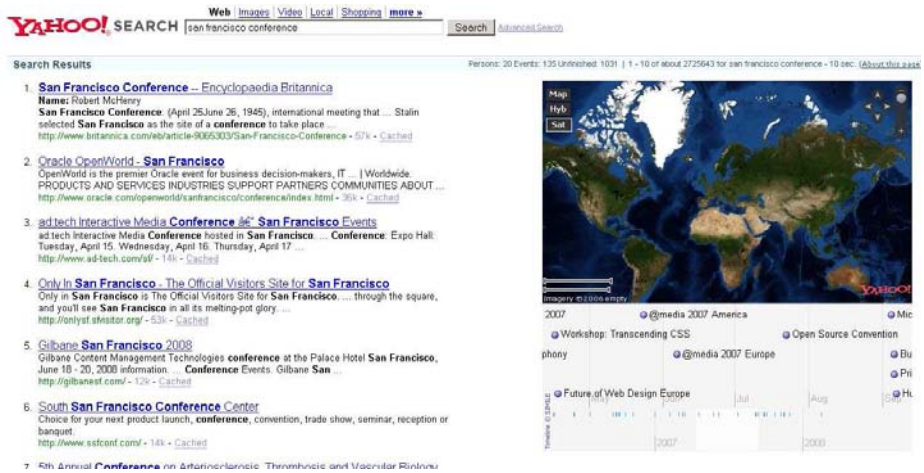


Fig. 3. Result display for the query *san francisco conference*.

Next, the metadata is aggregated and stored in a temporary Sesame[3] repository as well as cached to speed up further queries. We perform entity reconciliation on the aggregated data although this is not used in the current version of the system. Next, the result display is generated by using the Elmo API to populate a Java object model from the RDF data. The Fresnel API[4] developed by the SIMILE project is used to generate snippets from metadata. Transformations in Fresnel are described in declarative manner, providing among others what properties to display for certain classes of objects, which properties should be visualized as links or images etc. These descriptions known as Fresnel lenses are written in RDF using the Fresnel vocabulary. Using RDF provides the flexibility to create visualizations by inheriting from existing descriptions. Further, in principle the system could discover and reuse Fresnel lenses created by external developers to visualize resource types unknown to the current system. However, this possibility is not yet exploited.

3 Discussion

The microsearch demo has been made available online only recently and therefore long term statistics are not available yet. Although the prototype was not widely advertized, in the first week of its availability 7848 queries have been issued from 1037 unique IP addresses.

Figure 5 shows the distribution of unique queries according to the number of displayed results that contained metadata and thus resulted in metadata-based snippets. These statistics show that in 53.6% per cent of the unique queries at

[3] `http://www.openrdf.org`
[4] `http://simile.mit.edu/wiki/Fresnel`

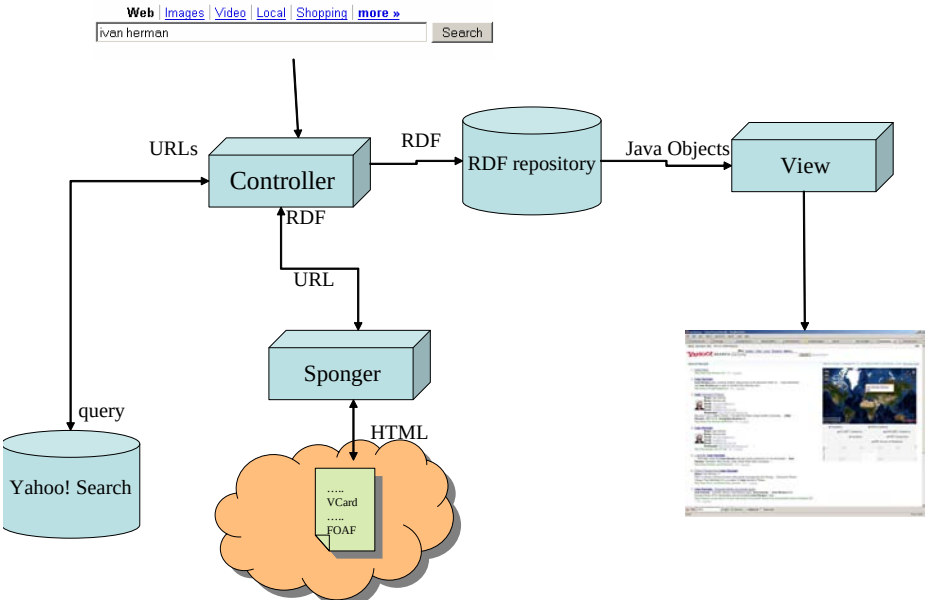


Fig. 4. The architecture of the microsearch system.

least one of the top 10 displayed results contained some metadata.(Note that the map and timeline may show metadata extracted from results below rank ten.)

The population of those who have tried the demo is hardly indicative of the general web population (mostly Semantic Web researchers and developers) and the queries issued are also a-typical (mostly person names). Thus the only observation we can make for now is that a metadata-enriched search engine can bring benefits to this particular community and the kind of queries issued, with no extra cost on the user's side. (When no metadata is present, microsearch simply behaves as the main search engine except for latency). We plan to investigate the shape of this distribution using a query log from Yahoo!'s main search engine. The advantage of using a live search engine or a query log for this analysis is that one is able to measure the metadata content of the pages that are likely to be useful for users. (While the Web is large, only a fragment of it is ever accessed through search.)

Based on the feedback we received the experience was also positive for the users with the obvious drawback of the increased query time. (However, by extracting and storing metadata as part of an offline process this delay can be significantly reduced.) Some of the expected benefits of exposing metadata were immediately visible: the present author, for example, discovered that his FOAF profile links to his old geographic address in the Netherlands. After being exposed to the interface, some users have also asked for ways in which they could metadata to their own pages. To help them, we have created a simple FAQ with short descriptions of how to add common types of metadata to HTML using

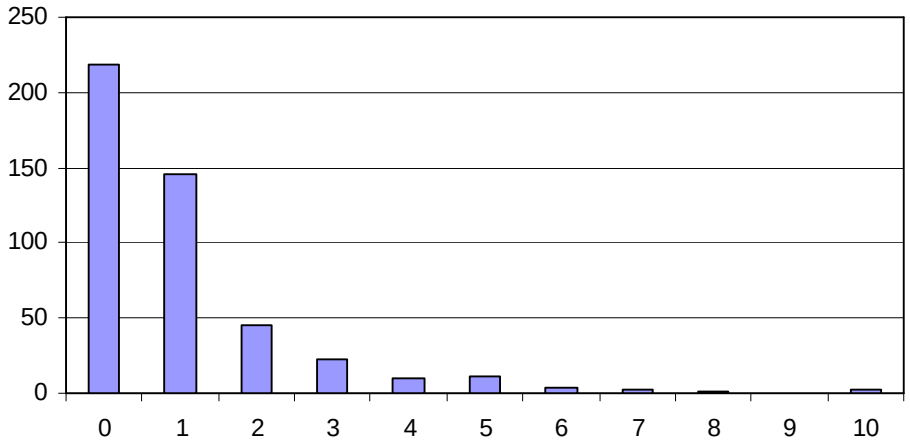


Fig. 5. Histogram showing the number of queries (y-axis) with 0, 1, ... 10 metadata-enabled pages (x-axis) within the top ten results.

microformats or RDFa. We have also included an "Update metadata" button next to each search result so that users can immediate see the results after adding or updating metadata to a particular page. Semantic Web developers have also asked for ways in which they could build other kinds of interfaces using the aggregated metadata produced, which prompted us to expose the metadata as a feed. Their reaction also confirmed our expectation that on the long run semantic search is likely to impact both query input and results presentation, reshaping the ways users interact with search engines.

Some of the ideas behind microsearch are also reflected in the design of Yahoo!'s Open Search Platform, also known as Search Monkey. Search Monkey will enable for any developer to create similar experiences in a highly scalable fashion. Search Monkey divides up the process of developing semantic search applications in two steps: metadata extraction and result presentation. (These are a single step in the microsearch process.) First, developers will have the possibility to create their own extraction modules as well as provided with metadata automatically extracted during the crawling process. The metadata resulting from running such extraction modules on webpages will be stored in the search index and made publicly available. Second, developers can also write visualization modules that create metadata-based snippets using the extracted metadata. Users of the search engine will be able to pick and choose the visualization modules they would like to use to enhance their search results.

4 Conclusions

Current methods of bringing semantics to Web search rely mostly on large editorial efforts, where web pages are classified manually or semi-automatically into

semantic classes. This method, for example, allows to display custom content on both Yahoo! and Google Search: see for example the Yahoo! Shortcut to Yahoo! News for the query 'britney spears'[5] and the similar shortcut to Yahoo! Shopping for the query 'apple ipod touch 8gb'[6]. Once the query intent is identified in terms of a taxonomy, web search engines are also able to provide much better help in breaking down the results, as shown among others by Google for the query 'ritalin'[7] and Hakia for the query 'george bush' [8].

This classification effort runs into two kinds of scaling problems when applied to Web search. First, there are a vast number of pages on the Web, which is fed by an endless production pipeline. This problem is addressed by harnessing the human effort of Web users as it has been done in Google Co-op[9] which lets users tag certain categories of Web sites (e.g. *health*) with predefined labels (e.g. *side effects*, *overdose*, *clinical trials* etc.)

However, there is another, potentially more difficult challenge related to the breadth of the information needs of Web users. The long tail of information needs is longer than most of us realize: Baeza-Yates et al. report that in the one year query log they studied 88% of the unique queries are singleton queries, and 44% are singleton queries out of the whole volume, which means that the vast majority of Web queries are only seen once, even when looking at a full year of query production [1]. This means that systems that rely on a fixed taxonomy of information needs (as all of the Web examples do) will certainly run into limitations when covering more than just the most common classes of objects and their most common aspects.

Microsearch and SearchMonkey bring semantics to long tail queries by relying on Semantic Web technology. Relying on standard semantic technology enables the system to aggregate information provided by users (manually annotating their web pages), and in the case of SearchMonkey, also information submitted to the system in the form of data feeds or extracted from Web pages. The application of semantic technology to vocabulary management (RDF, OWL) also means that the system is not limited to a fixed hierarchy of information types and a limited set of aspects when it comes to understanding query intent.

These systems in their present forms are still far away from exploiting all the possibilities offered by semantic search and tackling many of the challenges described in Section 1. However, by relying on open Semantic Web standards in metadata representation we believe that these systems have the potential to bring semantics to search in a way that scales to both the size and breadth of the Web.

[5] `http://search.yahoo.com/search?p=britney+spears`
[6] `http://search.yahoo.com/search?p=apple+ipod+touch+8gb`
[7] `http://www.google.com/search?q=ritalin`
[8] `http://www.hakia.com/search.aspx?q=george+bush`
[9] `http://www.google.com/coop/`

References

1. Ricardo Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. The impact of caching on search engines. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, New York, NY, USA, 2007. ACM.
2. V. Richard Benjamins, John Davies, Ricardo Baeza-Yates, Peter Mika, Hugo Zaragoza, Mark Greaves, Jose Manuel Gomez-Perez, Jesus Contreras, John Domingue, and Dieter Fensel. Near-term prospects for semantic technologies. *Intelligent Systems*, 23(1):76–88, 2008.

Exploring the knowledge in Semi Structured Data Sets with Rich Queries

Jürgen Umbrich and Sebastian Blohm

Institut AIFB, Universität Karlsruhe(TH), D-76128 Karlsruhe, Germany
{juum, blohm}@aifb.uni-karlsruhe.de

Abstract. Semantics can be integrated in to search processing during both document analysis and querying stages. We describe a system that incorporates both, semantic annotations of Wikipedia articles into the search process and allows for rich annotation search, enabling users to formulate queries based on their knowledge about how entities relate to one another while simultaneously retaining the freedom of free text search where appropriate. The outcome of this work is an application consisting of semantic annotators, an extended search engine and an interactive user interface.

1 Introduction

Currently, there is a vast amount of data available on the Web, mostly encoded in unstructured formats, such as plain text or HTML pages. Users are investing a substantial amount of effort in an attempt to organise and structure the unstructured information within their respective knowledge bases. A classic way for managing information in a semi structured way is the use of encyclopedias. Encyclopedias are compendiums containing information about branches of knowledge. Depending on the scope of the encyclopedia, each knowledge branches try to capture the information of a particular knowledge field or of a group, like of a community, of a nation or ideally of the whole mankind.

In the online encyclopedia Wikipedia[1], articles are organised as follows:

- **Article titles cover the subject**
  Each subject in the encyclopedia is covered by one article and is identifiable by the article title. Usually, these articles can be accessed by the list of the article titles, which are ordered in alphabetical manner.
- **Articles belong to categories**
  Articles can also belong to one or more categories, which pre-existing or created by the author manually. Encyclopedia users can access the knowledge base by exploring the articles within a category.
- **Article can link to other articles**
  If articles refer to other articles or subjects the author can express this relationship via a link. By reading articles contained within Wikipedia, the users can navigate to other articles following the links.

[1] http://www.wikipedia.org/

While this provides a good structure for manual browsing, it does not directly facilitate search nor does it support machine-understandable information about document content. At this stage, we would like to introduce a specific user scenario to motivate our study.

A user wants to find famous scientists born in Germany, specifically scientists which have received a degree at the University of Karlsruhe.

In our example user scenario, the user knows that he is looking only for information about persons, especially about scientists, beyond this, the user knows that the entities in his query are related to each other; the wanted scientists are *born in* in Germany, and have *received a* degree at a University in the city Karlsruhe. When entering a keyword-based query, most of this information is not conveyed and thus cannot be used to exploit Wikipedia's rich structure to increase the retrieval quality.

Depending on the structure and on the search possibilities of the encyclopedia the users are not able to use their background information to explore the knowledge base. Usually, the users have two ways to get access to the knowledge: 1) They can use a keyword -based search interface and 2) browse and navigate through the data set by articles, links or by categories. The problem, with the keyword-based search interfaces is, that people can neither express the meaning of words nor the relation between words, nor can they specify the category of the search results. For data sets of a manageable size, browsing might be a good way to explore the knowledge, but for huge knowledge repositories browsing can result in a very time consuming task, one which is not guaranteed to find the required results. Another disadvantage of browsing through categories and articles is that users have to inspect each article to decide if the subject matter presented is a suitable answer to the query.

Our approach is based on the idea to extract the implicit knowledge encoded in the category system and furthermore make the knowledge explicitly searchable via the annotation of articles, thus enables structured query functionalities over the knowledge base. With new query options, users can express the meaning of words with annotations. Moreover, they can describe and model relationships between entities through the combination of both annotation- and free text search. Thus, users are able to apply their background knowledge about the search term and the expected results to ask more specific queries and receive higher quality results.

We will show how to improve search functionalities for semi structured information sources by using annotation search combined with rich query functionalities. Therefore, we use the online encyclopedia Wikipedia and annotate the articles with meta information encoded in Wikipedia's category structure and with information from a external knowledge base. More specifically, we exploit Wikipedia's category and link structure for capturing semantics in keyword-base search. Pages are annotated with Wikipedia's category information, which is semantically grounded by using the Yago ontology[1]. References to other en-

tities within a document for which Wikipedia holds further information are also annotated with categories and Yago concepts.

The remainder of the paper is organised as follows: In Section 3 we give an overview about Semantic Search across annotated text. Section 4 describes our architecture and the functionality of each individual component. In Section 5 we introduce the semantic query syntax used by the components of the architecture. In Section 6 we present a user-interface, that hides the syntactic complexity of the extended notion of queries from the end-user. Finally, in Section 7 we conclude with an outlook of an approach for supporting end-users in creating structured queries.

2 Related Work

A simple but appealing definition of Semantic Search has been given by Soumen Chakrabarti [2] which states that queries "must enable schema-free searches but reward schema knowledge". Schema knowledge thereby can be integrated in various positions in the Semantic Search process. Generally speaking, information retrieval processing consists of an *indexing time (offline)* phase and a *query time* phase. At indexing time, documents are collected and pre-processed. This includes data normalisation, identification of relevant content (e.g. text tokens) and may include higher level processing like the extraction of relevant metadata and information as well as deriving a semantic document representation. At query time, the user query is taken to construct a query that is interpretable by the query processor. The output may be a (weighted) Boolean or bag-of-words query, a SPARQL expression or a request otherwise formalised according to the requirements of the query processor. After triggering query processing, the results are ranked and presented to the user. A feedback processing component may then allow a user to refine the request and re-trigger the query time process.

The vision of Semantic Search has inspired work in various directions. Different parts of the information retrieval process have been augmented with semantic information. We discuss briefly several Semantic Search systems which have in common that users may be unaware of (parts of) the ontology and that query language enables schema-free searches but allows improving retrieval when further knowledge is incorporated.

Guha et al. [3] introduce Semantic Search as the idea of using information from the Semantic Web for search. Applications are presented that add to classical search results with search results from RDF knowledge bases traversed using graph search, into classical search results. Thus, the semantics are captured in an additional query processor and then integrated during result presentation. A mechanism to capture rich ontological structure during query construction for parallel query processing has been presented by Tran et al. [4].

In the field of XML retrieval, methods from text search are being integrated into the structured XML retrieval paradigm. The XXL-engine [5] allows for the retrieval of objects, which have similar semantic names as the search term

given. The similarity operator is defined using semantic distance measures in an ontology graph. This work thus approaches Semantic Search from the side of structured retrieval.

Bonino et al. [6] transfer the standard term-index-based retrieval to an ontology-based paradigm by mapping the term by term to concepts in an ontology and then applying tf-idf like similarity search on "conceptual vectors". The query is also mapped to such conceptual vectors with the help of query-refinement techniques.

Chakrabarti [7] presents a search system that operates on both, the plain corpus and annotations. While annotations are defined as (probabilistic) connections to one or more ontologies and queries may involve ontology elements as well as uninterpreted strings. Among those discussed here, this work is closest to ours as it works on a shallowly annotated corpus relying on an extended standard information retrieval index.

The RelSE system[2] uses the category information in Wikipedia to allow precise search. Keyword search is made possible on a set of pages restricted by selecting Wikipedia categories. Our work extends this principle by indexing not only articles with their categories but also providing category informations for words mentioned within the pages.

Semantic content for Wikipedia has been derived in various ways and for various purposes. We employ the Yago ontology[1] which connects the Wikipedia category system to the WordNet lexical taxonomy and thereby creates an ontology with a large coverage within Wikipedia. The DBPedia project [8] provides relational information as captured in the Wikipedia Infoboxes. Information Extraction techniques can further extend Wikipedia annotations [9, 10].

3 Semantic Search on Annotated Text

Assuming that added value of Semantic Search is "rewarding schema knowledge" and based on our review of related work, one can observe that different approaches to Semantic Search differ in the amount and type of knowledge that is integrated as well as where and how the knowledge comes into play. Semantic technologies in an Information Retrieval context can be applied in two ways:

- **Interpretation of the query:** Allow the system/user to relate the contents of the query to formalised concepts and relations.
- **Interpretation of the content:** Allow the system/user to relate the contents of the documents to formalised concepts and relations.

A semantically enabled retrieval system can employ either of them or both. These aspects therefore constitute two key dimensions in which Semantic Search systems can differ. Classical text search systems (no interpretation of query no interpretation of content) and fully formalised knowledge bases with formal query language (full semantic access to both content and query) form the corners of the

[2] http://relse.apexlab.org/

space spanned by these dimensions. As opposed to mere KB lookups semantic search does not operate on any kind of formalised knowledge but on structured information derived from text. Due to the imperfection and incompleteness of this derivation, the text itself cannot be discarded during search.

Annotating semantic information within text is a challenging and error-prone task. Search systems must be prepared to handle a large amount of queries from various domains from users with completely different information needs. Information Extraction tools are either focused on a limited domain or can handle only a generic set of semantic concepts or properties Furthermore, the cost of engineering Language Resources either a Machine Learning or Knowledge Based Approach is time-consuming and expensive, requiring either specialist knowledge or large volumes of quality training data, which may be difficult to obtain. Our approach attempts to leverage pre-existing metadata into the IE process to aid retrieval thus viewing Wikipedia as a preannotated semantic corpus to be exploited. We thus build our present work on what we call the *Annotation/Query trade-off* hypothesis: A lack of fine-grained annotation can be compensated by incorporating more knowledge in the querying process and conversely, richer annotations allow semantic retrieval with less effort on the side of the user.

In this study, we produce annotations based on knowledge that is present in the Wikipedia. This knowledge consist of conceptual annotations for articles and words mentioned in the articles. Yet, we allow the user to query for relational information by providing a query mechanism that allows him to formulate his knowledge of the relations (e.g. domain and range) into the query.

4 Setup

In this section we describe our architecture and present each individual component. Figure 1 gives an overview of our system, consisting of the following five components (the dotted components are future work):

- A **semi structured data set** serves documents as input for the annotation engine.
- Various text analsys engines within the **annotation engine** parse the documents and extract the implicit knowledge and simultaneously anchoring it to the document.
- The **search engine** stores and index the document and the anchored annotation set and enables access to the indexed data via keyword, annotation and structured queries.
- The **pattern extraction** module will use the information from the annotation set and the document content to extract relation patterns between entities, like *born_in*, *studied_at*, *capital_of*. It also will supports the query interface with these extracted patterns. The extracted patterns are used to support the query creation process and can give recommendation for the most frequent relation patterns between certain entities. (*The implementation of this component is future work*)

– With the **query interface** users can create and query the index of the search engine in a user friendly way.

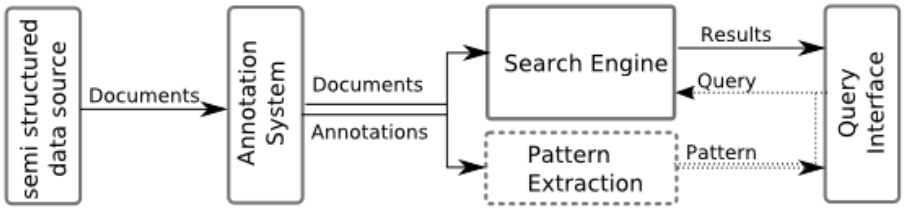


Fig. 1. Overview of the architecture

For the five components of the architecture we choose the following setup:

4.1 Data Source: English Wikipedia

We use a dump of the English Wikipdia from December 17th 2006 containing 1.6 million articles. Wikipedia is a multilingual, web-based, free content encyclopedia project and the biggest collaboratively edited knowledge source on the internet. More than 75.000 contributors have published over nine million articles in around 250 languages. Furthermore the knowledge is not restricted to a particular domain, the Wikipedia data set contains articles about a various of different domains and topics. Our motivation to use this data set was, that Wikipedia articles provide a lot of meta-information like the Wikipedia-categories or links to other articles.

4.2 Annotation Engine : Apache UIMA Framework and Text Analytic engines

The annotation engine processes documents and annotates new discovered knowledge to the documents. These information can be obtained directly from the content of the document or can be added from external meta data sources. Various text analysis engines, like word and sentence tokenizer, named entity recogniser or part of speech taggers, can be developed and plugged in the processing pipeline of a annotation engine. We use the Unstructured Information Management Architecture (UIMA)[3] to drive the annotation engine. UIMA provides from scratch some useful text analysis engines like a tokeniser, sentence and paragraph splitter, moreover, the API allows annotator developers to focus on writing the annotation logic (in Java) while a common data structure and a workflow engine are provided. Hence, we developed some text analysis engines expose the implicit knowledge in Wikipedia articles and annotate the articles

[3] http://incubator.apache.org/uima/

with additional information derived from the structure. We extract the following knowledge from the Wikipedia articles and pass them to the search engine together with meta information from the Yago knowledge base.

- **Discovered knowledge from the document corpus**
  Wikipedia articles contain already meta information about the covered subject and about other relevant articles or subjects, encoded in the article categories and in links. We annotate explicit the title and the Wikipedia categories of a document. Also, we parse the document structure for other occurrences of the title string to obtain more information of the page. For outgoing links to other related articles we anchor the link title and the categories of the targeted article to the hyperlinks in the input document. Furthermore, we identify year, month and day information from various date formats and annotate the original document by this date information.
- **Additional added knowledge from the Yago knowledge base**
  Beside extracting knowledge from the document itself, we annotate the articles with additional information derived from the Yago ontology, a huge semantic knowledge base, containing the unification of Wikipdia and WordNet[4] and knows around 14m facts about entities(e.g. person, city, organisation). The Yago data contains for each Wikipedia category a hierarchy of abstract concepts, e.g. the category `american_tennis_player` has the following hierarchy: `american_tennis_player < player < person < causal_agent`. For each Wikipedia category discovered in an article, resulting in link and page categories, we attach the corresponding Yago category and its hierarchical ancestor categories.

Figure 2 shows for parts of the Wikipedia article of Robert Cailliau, one of the inventors of the WWW, and what kind of information we extract and annotate for each article.

4.3 Index-based Search Engine

The search engine stores and indexes the document content and output from the annotation engine, further it offers search, browsing and navigation functionalities over the indexed data. We use the enterprise search platform of IBM, OmniFind, as the search engine in our architecture. OmniFind provides a UIMA compliant processing engine and offers beside the keyword search over document content also search capabilities for the annotate knowledge from the UIMA annotation engine. With the semantic query syntax of OmniFind the users can create structured queries and can efficiently exploit the indexed knowledge.

4.4 Pattern Extraction

With the new discovered knowledge from the annotation engine we can extract the word patterns between two annotated entities. The idea is to obtain information about the relationship between two entities from the tokens of these

[4] http://wordnet.princeton.edu/

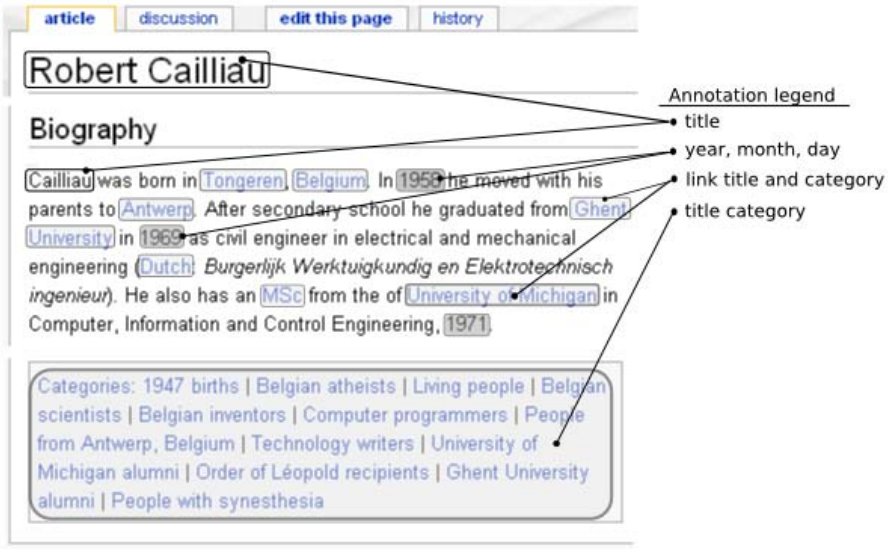


Fig. 2. Annotations for a Wikipedia article.

patterns. But, exposing knowledge about the relation between two entities is not a trivial task and will be addressed in future work.

4.5 Query Interface

The query interface use the search and index API (SIAPI) of the OmniFind search engine to get access to the knowledge base and to execute structured queries. The user friendly query interface, written in Java using Swing components, is described in detail in Section 6.

5 Structured Queries

5.1 OmniFind Query-Interface

OmniFind's search interface supports the same standard query operators as most common search engines, like free text and word phrase search as well as search operators like `AND`, `OR`, `NOT` and `WILDCARDS`. In addition, it provides two functionally equivalent types of query syntax, *XML fragments* and a subset of *XPath*. We use the *XML fragment* syntax in our work.

5.2 Structured Queries

XML Fragments provides a wide variety of additional query functionalities[11]. An XML Fragments query consists of an underspecified XML structure and thus

combines keyword queries with queries for annotated information. This enables search for more specified concepts, like searching for person's names. With the help of their domain knowledge, users can express relationships between object like "'the person and the city must occur in the same sentence"' or more specific like "'a persons *lives in* the city"' or "'a person *died in* a city"'.

The following query shows the OmniFind XML fragment semantic query syntax and how free text search and annotation search can be combined.

```
@xmlf2::'
  <page category="scientist" /> OR <page category="person" />
 +<sentence>
    </title> * "born in" * <link category="country">Germany</link>
  </sentence>
 +<sentence>
    </title> * "studied at" * <link category="University"/>
             * <link category="city">Karlsruhe</link>
             * "received" * <link category="Degree"/>
  </sentence>
```

The first line restricts the results to articles about persons or scientists. Next, we describe some further restrictions to the results. The sought after person and different entities have to occur in the same sentences and between these annotated entities certain word sequences have to show up. Searching for annotations combined with keywords and several query operators, like wildcards, is one way to express the relationships between entities.

6 End-User Query Interface

The end user query interface allows users to access and search the data indexed by the search engine. The whole query interface is a stand-alone software, written in JAVA 1.5 and adapted especially for the annotations from our annotation engine. The main focus of the end-user query interface is to allow users to create complex queries in a user friendly and understandable way. The queries are converted into the OmniFind query syntax and executed using the search and Index API (SIAPI) of OmniFind. The standard query interface, provided by OmniFind, can only process semantic queries encoded in the OmniFind query syntax. As can be seen from the example query in the previous section, the query syntax can be hard to use and understand for the users. There are two query creation modes available, a very simple version of creating the queries and a advance version, that allows to create complex structured queries.

6.1 Simple Query Interface

The simple query creation interface, enables basic query functionalities. As Figure 3 shows, the simple query interface uses common query concepts like, text

fields or drop-down menus. The same concept can be found in other well-known search interfaces, like by Ebay[5] or Amazon[6]. People can use the *"keyword search"* field for very simple keyword queries, optional with operators like AND, OR, NOT, WILDCARDS and word phrases. In the *"Page Title"* query field people specify the search for page titles or word snippets in page title. Using the *"Page Category"* query field, people can filter the result set for pages of a special page category. Below this is the *"Result Pages Containing"* query field, where people can search for labels or categories of the outgoing links. The search functionalities of this query interface go far beyond the search functionalities offered by the original encyclopedia page.

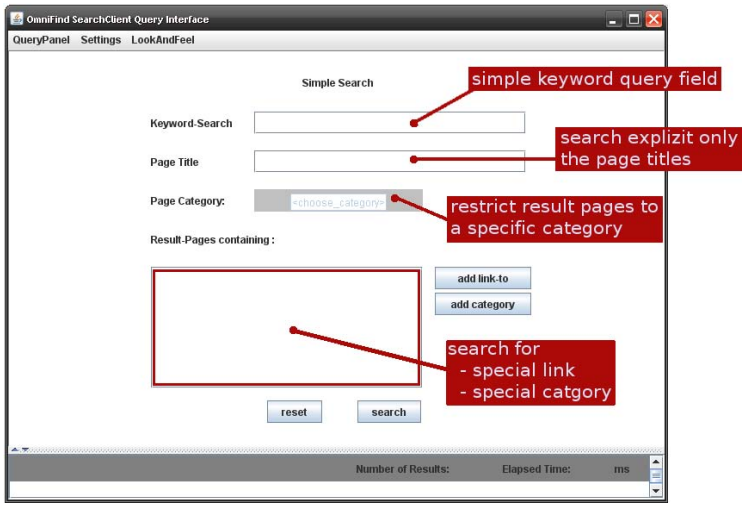


Fig. 3. Simple Search Interface

6.2 Advanced Query Interface

Users, more familiar with the query interface or structured queries, can use the advanced query interface to create complex queries and combine keyword and annotation search. Figure 4 shows the advanced query interface with its core component, the query creation module.

A query is a combination of various query patterns with following query operators:

- keywords, word phrases and keyword query operators.
- queries for links and their Wikipedia or Yago category.

[5] http://www.ebay.com/
[6] http://www.amazon.com/

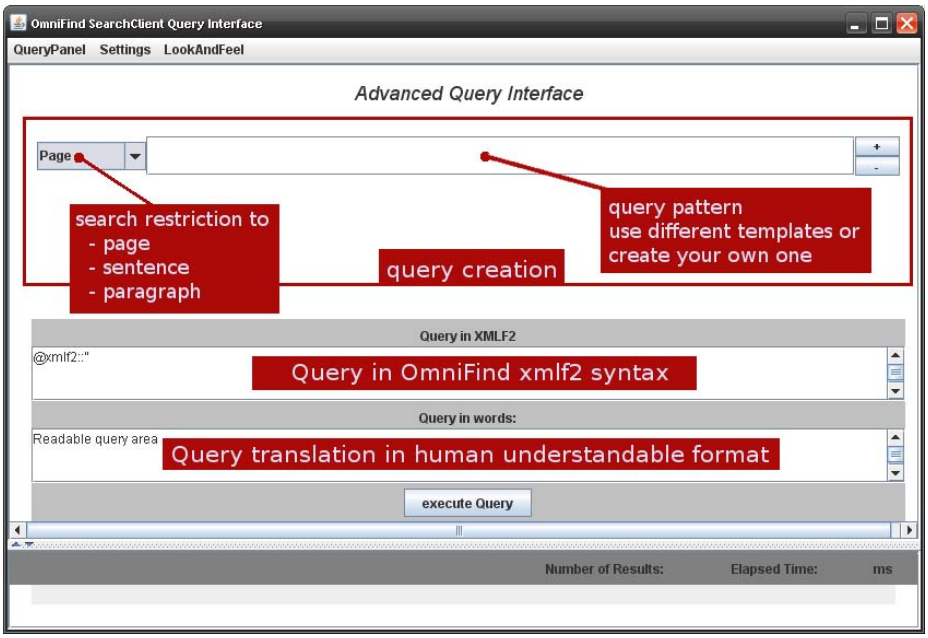


Fig. 4. Advanced Query Interface

– queries for page titles and page categories, as Wikipedia categories and Yago categories.

Furthermore, the user can specified, where the query pattern has to occur in the document, either looking for a match in the whole document, or in a subset of the document content, like a paragraph or a sentence. The query interface offers additionally the translation of the query patterns into the OmniFind query syntax and into a more human readable representation. Figure 5 shows the advanced query interface with out running example query.

6.3 Resultset and Ranking

Generally speaking, the approach allows using the full ranking and result presentation capabilities of the employed search engine. The results returned by the API contain the document URI, title of a summary or short description of the textual content. We show the title and the hyperlink to the Wikipedia article in our result panel of the user interface. Highlighting the semantic annotations is easily possible. Users can then open the documents in a separate browser window. If the indexed collection contains Web documents, like in our case, the ranking of the results also contains link analysis, based on the in-link counts of Omnifind's crawler.

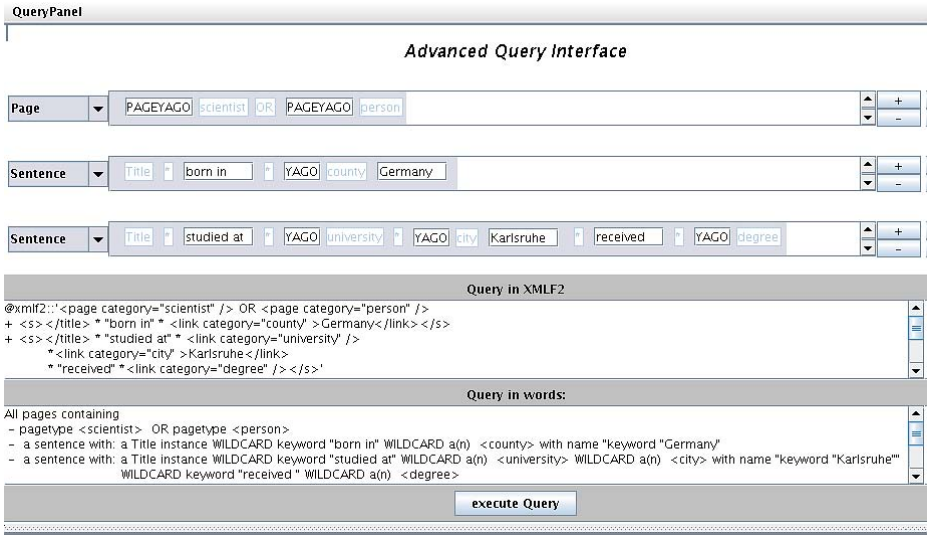


Fig. 5. Advanced Query Interface shows the running example

7 Future Work

Future work includes the implementation of the pattern extraction module and the investigation of discovering information about the relation between two entities. Therefore, we will extract the word tokens between two annotated entities and try to expose relation patterns out of them. With these patterns we can model the semantic relation between entities, beyond this, we can support the query creation task for the end-users by recommending relationships between two entities. The users do not have to know what kind of keyphrases are used in the knowledge base to describe relation, e.g the word tokens *"studied at"*, *"was a student at"* and *"completed a degree at"* are describing the relation between a person and a university. The semantic patterns and their corresponding word tokens can help the end users to model relations between entities in their query, without knowing what keyphrases describe these relations in the knowledge base. For example, in our user scenario, we know that the person we are looking for received a degree at the University in Karlsruhe, but we do not know the exact keyphrases between these entities. A semantic relation pattern for this scenario can be *entity:*`[PERSON]` *relation:*`received` *entity:*`[DEGREE]`.

8 Conclusion

We presented an architecture to exploit Wikipedia's category and link structure for capturing semantics in keyword-base search. Pages are annotated with Wikipedia's category information which is semantically grounded by using the Yago ontology. References to other entities within a document for which Wikipedia

holds further information are also annotated with categories and Yago concepts. This allows extended structured queries which can be posed through a dedicated search interface. Future work contains to use the extracted relation patterns to help users in creating their queries and the corresponding results.

Acknowledgements

This work has been supported by MFG Stiftung, Baden-Würrtemberg and by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) program under EC grant number IST-FP6- 026978.

References

1. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web (WWW), ACM Press (2007) 697 – 706
2. Chakrabarti, S.: Building blocks for semantic search engines: Ranking and compact indexing in entity-relation graphs. Keynote talk at the International Workshop on Intelligent Information Access (IIIA-2006) (2006)
3. Guha, R., McCool, R., Miller, E.: Semantic search. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press (2003) 700–709
4. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-based interpretation of keywords for semantic search. In: Proceedings of the 6th 6th International Semantic Web Conference, Busan, Korea (2007) 523–536
5. Schenkel, R., Theobald, A., Weikum, G.: Semantic similarity search on semistructured data with the xxl search engine. Information Retrieval **8**(4) (2005) 521–545
6. Bonino, D., Corno, F., Farinetti, L., Bosca, A.: Ontology driven semantic search. SIGIR Forum **1**(6) (2004) 1597–1605
7. Chakrabarti, S., Puniyani, K., Das, S.: Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM Press (2006) 717–726
8. Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: In: Proceedings of ISWC 2007. (2007)
9. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In: Natural Language Processing and Information Systems. Springer, Berlin / Heidelberg (May 2005)
10. Blohm, S., Cimiano, P.: Using the web to reduce data sparseness in pattern-based information extraction. In: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Warsaw, Poland, Springer (SEP 2007) 18–29
11. Hampp, T., Lang, A.: Semantic search in websphere information integrator omnifind edition: The case for semantic search. IBM Developer Works (2005)

Search, Natural Language Generation and Record Display Configuration: Research Directions Stemming From a Digital Library Application Development Experience (Discussion Paper)

Andrew Russell Green and José Antonio Villarreal Martínez

Instituto Mora (National Council for Science and Technology, Mexico)
and Instituto de Investigaciones Estéticas (National Autonomous University of Mexico)
ahg@servidor.unam.mx,quetzal1910@gmail.com

Abstract. Digital libraries and archives stand to benefit greatly from the Semantic Web (SW), which may provide a basis for novel end-user functions targeted at research and teaching. The project "Image Preservation, Information Systems, Access and Research" seeks to develop an adaptable digital library application based on a back-end of semantically modeled data. By "adaptable" we mean able to adapt to diverse library and archive scenarios, especially involving the integration of different types of material (photographic prints, negatives, drawings, periodicals, books, etc.) in a single system. A problem we have encountered is: the design of algorithms for processing information as it moves from the model to the user interface, and, following user input, from the interface back into the model. In this paper we discuss two specific issues that are encompassed by this general problem: full-text search mechanisms and record display configuration.

Key words: Semantic Web search, record display configuration, natural language generation, digital libraries

1 Introduction

Digital libraries and archives stand to benefit greatly from the Semantic Web (SW). Semantically modeled catalogues should provide a basis for new functions to help users sift through large and diverse repositories, discover patterns, explore associations among objects, find relevant information, and create and share descriptions of objects in a structured, flexible manner. This is the promise the SW holds for knowledge repositories, and one can hardly underestimate its potential impact in History and other Social Sciences: archives are primary sources—essential deposits of partially processed information, used for research in these disciplines—and despite the high degree of interrelation among data in different

archives, catalogues are often isolated and employ divergent record formats that are hard to align using standard information technology.[1]

This issue is one of the reasons the project Image Preservation, Information Systems, Access and Research (IPISAR) set out to build a SW-based digital library application. The project investigates the dissemination, study and management of heritage resources, and attempts to provide solutions to common problems in these areas.

The application being built, called "Pescador", will store catalogue data in a persistent triple store (whose function will be similar to that of a relational database in traditional systems). The requirements for the application include the ability to integrate data in various catalogue formats and adapt to the cataloguing needs of diverse archives. In this paper, the terms "catalogue format" and "record format" refer to the selection, organization and meaning of fields used to describe objects in an archive or library catalogue, as well as other conventions related to catalogue creation. Since Pescador will use the SW to model catalogues, each record format will correspond to a distinct kind of graph structure, often requiring specialized vocabulary and rules, and related to specialized application logic.

The application will have three main types of user: (1) regular users (or "patrons") who will consult the material provided by the digital library, (2) cataloguers, who will provide and manage the library's materials and metadata, and (3) catalogue designers/modelers/programmers, who will select or create the catalogue record formats and corresponding ontologies, and adapt the system to the needs of a given scenario. Pescador will provide a Web interface for the first two kinds of users; on this level, numerous functions targeted at research, teaching and cataloguing are planned [5]. When these users view data from the catalogue, they will see a user-friendly organization of information extracted from the SW graph; similarly, when cataloguers modify elements in the catalogue, they will employ easy-to-use forms, and the SW graph will be changed according to their input. The third type of user, the catalogue designer/modeler/programmer, will use a programming interface.

A problem we have encountered is: the design of algorithms for processing information as it moves from the model to the user interface, and, following user input, from the interface back into the model. In this paper we discuss two specific issues that are encompassed by this general problem: full-text search mechanisms and record display configuration. We conclude that record display, natural language generation and other text generation logic, text fragment caching mechanisms, and full-text search algorithms must be studied and designed together.

To date, two incomplete versions Pescador have been created. Both are currently used for Web sites that offer simple consultation functions for on-line archives (available at [8] and [3]). Our proposals stem from the experience of developing these versions of the application. Though the project IPISAR may

[1] The situation of historical archives varies greatly from one archive to another. Other recurring difficulties include access restrictions and insufficient funding; the first of these is also a major focus of the project described in this article. See [6] and [1].

yet generate new archival Web sites using the second version, it is clear that to implement all proposed features, a major rewrite is unavoidable. It should be noted that work on the rewrite has yet to begin. The general nature of the proposals outlined here is a reflection of this.

All versions of Pescador are provided under the terms of the free GNU GPL license.

2 Display Templates

There exist several general systems for record display specification, which we will call "display template systems", and many SW applications use internal template mechanisms. We agree with the definition of the problem given by the authors of Fresnel (an important proposal in this area), who state that "presenting Semantic Web content in a human-readable way consists in addressing two issues: specifying what information contained in an RDF graph should be presented and how this information should be presented." [2] However, this definition is deceptively simple, as both parts of the problem—the selection of information from the model and its transformation into a presentable format—can be quite complex.

Clearly there is a need for templates in SW applications: models often do not contain all the information required to create user-friendly descriptions, and even when they do, it is not always be desirable to show users all available information. The most basic kind of SW template involves a selection and ordering of properties; when the template is "applied" to a resource, label-value pairs are created from the properties' labels (often set using `rdfs:label`) and values for that resource. On this foundation, numerous advanced features may be built, such as:

- Facilities for creating sections, subsections and similar structures within records. This is often required for lengthy description; see, for example, full records in [3] and [8].
- Ways of including additional elements in records, such as images and text that is not part of a label-value pair.
- Facilities for defining short, human-readable labels for resources—normally used for values in label-value pairs, to refer to resources that are the objects of the properties displayed.
- Ways of setting special, context-appropriate property labels. (Consider, for example, a property with the `rdfs:label` "Location Photographed". In a photograph's catalogue record, one might wish to call the property "Location", since in this context, the full label would be needlessly long.)
- Means of embedding the result of one template in the result of another one.
- Means of retrieving information from diverse parts of the model—not just over the direct properties of the resource being described. This may be accomplished using path definitions.
- A hierarchy of templates and inheritance of templates' characteristics over the hierarchy.

- Media-agnostic template definitions, or a separation of record content specifications from media-specific formatting details.
- Facilities for embedding arbitrary logic—in other words, executable code—in templates, in a manner similar to languages for creating dynamic Web pages (JSP, ASP, PHP, RHTML, etc.). This allows templates to run loops, generate text and modify their output on the basis of conditions described in the executable code.
- Programmatic template creation and modification. For example, at runtime, a search component may create temporary templates that display only fields containing hits.
- Vocabulary and conventions for modeling the templates themselves.

Fresnel, Pescador 0.1 and Pescador 0.2 all implement different subsets of these possible features. A challenge for the next version of Pescador is to determine which features are required, and how to integrate them with our system while maintaining support for encapsulation and separation of concerns. In additional, we must take into account a lesson learned in work on Pescador 0.2, namely: that the scope of a templating system is wider than record display itself. This is because numerous elements of a user interface must be coordinated with record display. To illustrate this, let us consider a catalogue in which photographs are described with the fields "photographer", "title", "date", "location" and "topics". A user interface that provides access to such a catalogue would refer to these fields in several places, not just when displaying records. For example, a menu might offer the option of listing items ordered by date or title. Another might offer the possibility of grouping items by photographer, location or topic. An advanced search interface could include options for searching only within one or more of these fields. On the screen displaying the catalogue records themselves, diverse functions may be available in fields' context menus. Last but not least, the interface for adding, deleting and modifying items in the catalogue will mention fields in various ways. In all these parts of the interface, references to fields must be consistent, clear and appropriate for their respective contexts. To achieve this, specialized display specification mechanisms are required, and it makes sense to integrate these mechanisms with the template system.

3 Natural Language Generation and Searching

In this section we review two related issues: natural language generation (NLG) and full-text search. Like display templates, these problems fall under the broad category of algorithms for processing information as it moves back and forth between the model and the user interface.

NLG is "the subfield of artificial intelligence and computational linguistics that focuses on computer systems that can produce understandable texts in English or other human languages" [9]. Typically SW applications have not used complex, human language-aware subsystems to create text output, opting instead for simpler mechanisms that extract strings from the model and place

them in slots established by an interface generation subsystem (which may use a display template mechanism, as described above). Though in many cases this is sufficient, in developing Pescador we have come across several scenarios that call for a more elaborate language generation mechanism, able to create, from subgraphs, understandable fragments of natural language, taking into account language features such as pluralization, gender and the chaining of adjective phrases.

We demonstrate the problem with the hypothetical results of a full-text search in a mixed archive (Fig. 1). In this mock-up, items are grouped according to their relationship to nodes that produced full-text hits. Of course, these relationships would exist as paths in the graph. Only an NLG system would be able to produce concise, correct and easy-to-read descriptions of associations such as those shown in the mock-up. (For many languages the generation of descriptions like these is more complicated than it is for English—an example is Spanish, in which adjectives must agree both in number and gender with the nouns they describe.) In other processes that might employ NLG, its potential benefits are similar though perhaps less notorious.

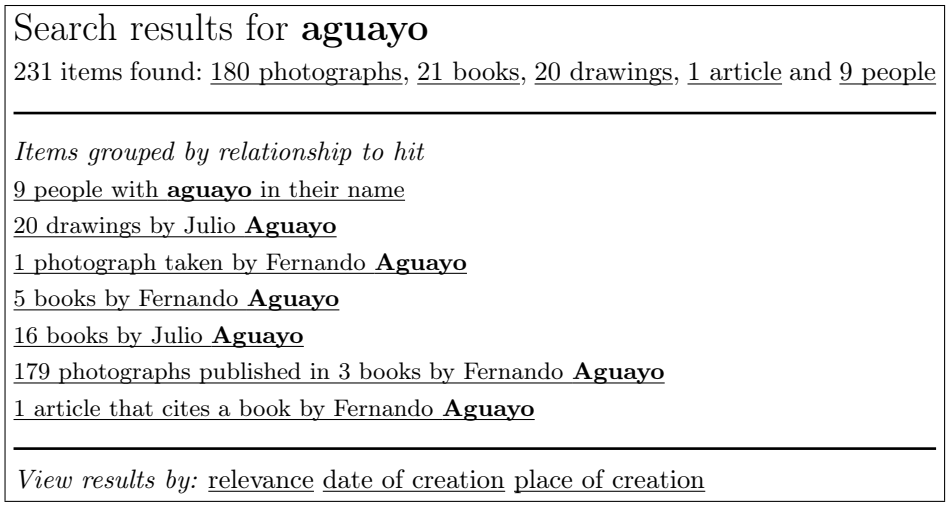


Fig. 1. Mock-Up of Search Results
Underlined elements are hyperlinks.

Thus we identify NLG processes—including, but not limited to, the translation of paths into natural language descriptions—as an issue to be studied for the next version of Pescador. In general, we view NLG as a low-level process in user interface generation, since it creates small text fragments—as opposed to a display template system, which operates at a higher level, organizing much larger segments of the interface. Note that at this low level of text fragment generation, the NLG system, once implemented, will not be alone; many text

fragments will still be easier to create using more traditional sorts of text concatenation logic (for example, a string with a person's family names and given names). Note also that we can distinguish two types of text fragment generation processes: (1) those that create transient text fragments, not cacheable for later reuse (for example, the relationship descriptions in Fig. 1); and (2) those that generate more stable fragments, which might be retained in a cache and inserted repeatedly into the user interface.

We mention the distinction between transient and cacheable text fragments because, to explain the issues we are facing in full-text searching, we must first review the functioning of these stable, cacheable fragments. In Pescador 0.2, cached text fragments are mainly low-level building blocks of catalogue records. That version of the system caches them not only to speed record generation, but also to allow full-text search within them. This is important because of the way users expect full-text search to work. To illustrate briefly: suppose that a model contains resources that refer to people, and that those resources may have three properties: `hasFamilyNames` and `hasGivenNames`, which point to literals, and `hasTitle`, which points to resources that represent titles. The resources for titles, in turn, have two properties: `hasAbbreviation` and `hasFullName`. Human-readable labels for people are constructed with literals; for example, the label "Smith, Dr. George" would aggregate literals that represent the abbreviation for "doctor" and Dr. Smith's family and given names. In a full-text search, to correctly locate resources associated with "Smith", "George" or "Dr", the search component would need only look at strings contained in the model. But what if a user searches for the *exact phrase*, "Smith Dr George"—that is to say, those words together, in precisely that order? Nowhere in the model do they appear in that manner, but the end user does not know that, and if s/he has seen such text fragments in the catalogue, s/he will expect such a search to produce results. A possible solution is for the system to cache this text fragment, make it available to the search component, and associate it with the resource that refers to Dr. Smith; then searches may find that phrase and return a hit on the correct resource.

Despite the apparent feasibility of such a mechanism, there are unsolved issues related to how cached, generated text fragments may be treated in searches. In the preceding example, clearly it is fine for searches for "Smith", "George" or "Smith, Dr. George" to located the generated fragment and thus return Dr. Smith as a search result. But searches for "Dr" should not do the same—rather than finding all the generated text fragments that include that string, they should provide more meaningful results, for example "**Dr**, abbreviation for the title 'doctor', borne by 20 people in the knowledge base". The precise algorithm needed here remains to be flushed out.

4 Conclusion

In this paper we have considered two issues related to the algorithms for processing information as it flows between the model and the user interface: record display templates and full-text search algorithms. We conclude that template mech-

anisms, NLG and other text generation logic, text fragment caching mechanisms, and full-text search algorithms must be studied and designed together, in order to construct friendly, easy-to-understand and meaningful catalogue records, interfaces and search results. This is in part because most users will not have technical knowledge of the SW, or of how data is modelled and transformed to construct catalogue records, and they will expect searches to be performed on records as displayed. As a result, the application's search component must consider these same data transformation algorithms when it moves from hits in the SW graph to user-consumible seach results. This realization provides starting points for further work towards the creation of the application we envisage.

References

1. Aguayo, F., Roca, L.: Estudio introductorio. In: Aguayo, F., Roca, L. (eds.): Imágenes e investigación social. Instituto Mora, México (2005) 9-28 http://durito.nongnu.org/docs/Aguayo_Roca_2.html
2. Bizer, C., Lee, R., Pietriga, E.: Fresnel Display Vocabulary for RDF: User's Manual. World Wide Web Consortium (2005) http://www.w3.org/2005/04/fresnel-info/manual-20050726/
3. Fototeca Digital: Fotógrafos y Editores Franceses en México. Siglo XIX. Instituto Mora and Instituto de Investigaciones Estéticas, National Autónomous University of Mexico (2007) http://afmt.esteticas.unam.mx
4. Green, A.: Logic and a Little Language for Heritage Resource on the Semantic Web. Poster accompanying a system demonstration, presented at the 4th European Semantic Web Conference (June, 2007) http://durito.nongnu.org/docs/innsbruck2.pdf
5. Green, A. R.: Metadatos transformados: Archivos digitales, la Web Semántica y el nuevo paradigma de la catalogación. In: Amador C., P., Robledano A., J., Ruiz F., R. (eds): Quintas Jornadas: Imagen, Cultura y Tecnología. Universidad Carlos III de Madrid: Madrid (2007) 11-22 http://durito.nongnu.org/docs/metadatos_transformados_green.pdf
6. Green, A. R.: Rescate de la memoria. Ciencia y Desarrollo (Sept. 2006). Consejo Nacional de Ciencia y Tecnología, Mexico
7. Kochut, K. and Janik, M., SPARQLeR: Extended Sparql for Semantic Association Discovery (2007) http://www.eswc2007.org/pdf/eswc07-kochut.pdf
8. Marcas de Fuego de la Biblioteca "José María Lafragua" de la BUAP. Autonomous University of Puebla (2006) http://www.marcasdefuego.buap.mx/
9. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press: Cambridge, UK (2000)

SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Concept Search: Semantics Enabled Syntactic Search

Fausto Giunchiglia, Uladzimir Kharkevich, and Ilya Zaihrayeu

Department of Information Engineering and Computer Science
University of Trento, Italy
{fausto,kharkevi,ilya}@disi.unitn.it

Abstract. Historically, information retrieval (IR) has followed two principally different paths that we call syntactic IR and semantic IR. In syntactic IR, terms are represented as arbitrary sequences of characters and IR is performed through the computation of string similarity. In semantic IR, instead, terms are represented as concepts and IR is performed through the computation of semantic relatedness between concepts. Semantic IR, in general, demonstrates lower recall and higher precision than syntactic IR. However, so far the latter has definitely been the winner in practical applications. In this paper we present a novel approach which allows it to extend syntactic IR with semantics, thus leverage the advantages of both syntactic and semantic IR. First experimental results, reported in the paper, show that the combined approach performs at least as good as syntactic IR, often improving results where semantics can be exploited.

1 Introduction

The goal of information retrieval (IR) is to map a natural language query, which specifies the user information needs, to a set of objects in a given collection, which meet these needs. Most existing systems also compute a numeric score on how relevant each retrieved object is to the query, and order these objects according to the degree of relevance.

Historically, there has been two major approaches to IR that we calls syntactic IR and semantic IR. In syntactic IR, search engines use words or multi-words phrases that occur in documents and queries as atomic elements in document and query representations. The search procedure, used by these search engines, is principally based on the *syntactic matching* of document and query representations. These search engines are known to suffer in general from low precision while being good at recall.

Semantic IR is based on fetching document and query representations through semantic analysis of their contents using natural language processing techniques and then retrieving documents by matching these semantic representations. Differently from syntactic IR, in this approach the *meaning* of words is analyzed and not only their syntactic representations. Semantics-based approaches, in general, allow to reach a higher precision but lower recall than syntactic approaches [11].

In practice, results of semantic IR are inferior to that of syntactic one. In fact, most of the state of the art search engines are based on syntactic IR. There are many reasons for this, where one of them is that techniques based on semantics, to be used properly, need a lot of background knowledge which is in general not available [6].

In this paper we propose a novel approach to IR which extends syntactic IR with semantics, thus addressing the problem of low precision of syntactic IR. We call it *Concept Search* (*C-Search* in short). The main idea is to keep the same machinery which has made syntactic IR so successful, but to modify it so that, whenever possible, syntactic IR is substituted by semantic search, thus improving the system performance. This is why we say that *C-Search* is semantics enabled syntactic search. In principle, our approach allows it to scale on the continuum from purely syntactic search to purely semantic search, performing at least as well as syntactic search and improving over it by taking advantage of semantics when and where possible. Our approach scales as much as syntactic IR can scale because semantics is seamlessly integrated in the syntactic search technology.

The remainder of the paper is organized as follows. In Section 2, we first discuss IR in general and then we discuss syntactic search approach to IR. In Section 3, we discuss semantic IR and introduce semantics enabled syntactic search. In Section 4, we describe how semantic matching of (complex) concepts, the core of semantic search algorithm, can be efficiently implemented using inverted index technology. Section 5 presents some preliminary experimental results. In Section 6, we discuss the state-of-the-art in semantic search and compare our approach with other related approaches. Section 7 summarizes the achieved results and concludes the paper.

2 Syntactic Search

The goal of an information retrieval system is to map a natural language queries Q, which specify user information needs, to a set of documents in the document collection D, which meet these needs, and (optionally) to order these documents according to the degree of relevance. The search S in general can be represented as a mapping function:

$$S : Q \rightarrow D \tag{1}$$

In order to implement an IR System we need to decide (i) what is an atomic element (*Term*) in document and query representations, (ii) which matching techniques (*Match*) are used for matching of document and query terms, (iii) which models (*Model*) are used for document and query representations, for computing query answers and relevance ranking, and (iv) which data structures (*Data_Structure*) are used for document indexing and retrieval. Thus, the IR System is a 4-tuple:

$$IR\_System =< Model, Data\_Structure, Term, Match > \tag{2}$$

The Bag of words model, i.e., the model in which the ordering of words in a document is not considered, is the most widely used model for document representation. The Boolean Model, the Vector Space Model, and the Probabilistic Model are the classical examples of models used for computing query answers and relevance ranking [1].

Various index structures, such as Signature File and Inverted Index, are used for efficient retrieval. Inverted Index, which stores a mapping from terms to their locations in documents, is the most popular solution [1].

In syntactic IR, *Term* and *Match* are instantiated as follows:

- *Term* - a word or a multi-words phrase,
- *Match* - a syntactic matching of words or phrases.

In the simplest case, syntactic matching is computed through search for equivalent (possibly stemmed [14]) words. Some systems approximate matching by search for words with common prefixes or words within a certain edit distance with a given word.

Let us consider the document collection shown in Figure 1.

D1 :	A small baby dog runs after a huge white cat. ...
D2 :	A laptop computer is on a coffee table. ...
D3 :	A little dog or a huge cat left a paw mark on a computer table. ...

Fig. 1. A document collection

In Figure 2, we show examples of four queries, which are submitted to this document collection.

Q1 :	Babies and dogs	Q3 :	Table computer
Q2 :	Paw print	Q4 :	Carnivores

Fig. 2. Queries

An example of syntactic IR using Inverted Index technology is given in Figure 3. The two parts of an Inverted Index are: *Dictionary*, i.e., a list of terms used for document indexing; and posting lists (*Postings*), where every posing list is associated with a term and consists of documents in which this term occur. The query processing in Inverted Index is separated into two main steps: (i) to locate terms in dictionary which match query terms, and (ii) to search Inverted Index with these terms. Consider, for example, processing a query *table computer*. First, for each query term we identify those terms in dictionary that match this term ($table \rightarrow \{table\}$ and $computer \rightarrow \{computer\}$). Second, we search inverted

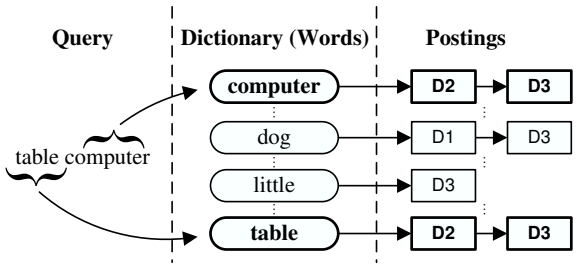


Fig. 3. Inverted Index in classical syntactic search

index with computed dictionary terms ($table \rightarrow \{D_2, D_3\}$ and $computer \rightarrow \{D_2, D_3\}$). And finally, we take the intersection of document sets, found for every query terms, as an answer to the query (D_2 and D_3 in our example).

There are several problems which negatively affect the performance of syntactic search. These problems are discussed bellow:

Polysemy. The same word may have multiple meanings (see Figure 4) and, therefore, in syntactic search, query results may contain documents where the query word is used in a meaning which is different from what the user had in mind. For instance, a document which talks about *baby* in the sense of a very young mammal is irrelevant if the user looks for documents about *baby* in the sense of a human child who has not yet begun to walk or talk. An answer for query $Q1$, computed by syntactic search engine, includes document $D1$, while the correct answer is an empty set.

Synonymy. Two different words can express the same meaning in a given context, i.e., they can be synonyms (see Figure 5). Syntactic search approaches do not explicitly take synonymous words into account. For instance, words *mark* and *print* are synonymous when used in the sense of a visible indication made on a surface, however, only documents using word *print* will be returned if the user query was exactly this word. An answer for query $Q2$, computed by syntactic search engine, is an empty set, while the correct answer includes document $D3$.

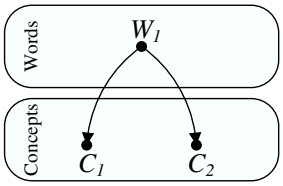


Fig. 4. Polysemy

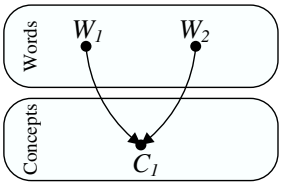


Fig. 5. Synonymy

Complex concepts. State-of-the-art syntactic search engines fall short in taking into account complex concepts formed by natural language phrases and in discriminating among them (see Figure 6). For instance, phrases *computer table* and *table computer* denote two quite different concepts, whereas a conventional search engine is very likely to return similar results if they are submitted as queries. Moreover, the results of these queries may contain documents irrelevant to both of them, e.g., a document, containing a sentence *A laptop computer is on a coffee table*, being irrelevant to both of our queries, is likely to be found as an answer to these queries. An answer for query $Q3$, computed by syntactic search engine, includes documents $D2$ and $D3$, while the correct answer is an empty set.

Related concepts. Syntactic search does not take into account concepts which are closely related to the query concept (see Figure 7). For instance, a user looking for *carnivores* might not only be interested in documents which talk about carnivores but also in those which talk about the various kinds of carnivores such as *dogs* and *cats*. An answer for query $Q4$, computed by syntactic search engine, is an empty set, while the correct answer includes documents $D1$ and $D3$.

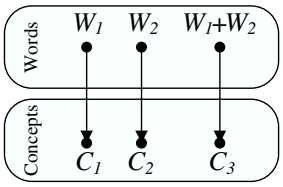


Fig. 6. Complex concepts

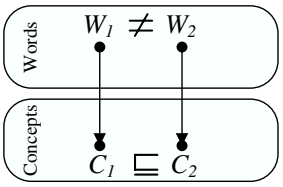


Fig. 7. Related concepts

3 Semantics Enabled Syntactic Search

In semantic search, *Term* and *Match* elements of the model, described in Formula 2, are instantiated as follows:

- *Term* - an atomic or a complex concept,
- *Match* - semantic matching of concepts.

Where concepts are computed, for example, by mapping words to concepts in a lexical database such as WordNet [13]. Semantic matching can be implemented by using semantic matching approach described in [7–9]. The main idea of semantic matching is to compare meanings (concepts) and not words, as in syntactic matching. For example, phrase *A little dog or a huge cat* syntactically is very different from a word *carnivores* but semantically they denote related concepts.

Because we build on top of standard syntactic search technology, classical information retrieval models and data structures can be fully reused in semantic

search with the difference in that now words (W) are replaced with concepts (C) and syntactic matching of words ($WMatch$) is replaced with semantic matching of concepts ($SMatch$).

$$\boxed{\textit{Syntatic Search} \xrightarrow{(W \rightarrow C),\ (WMatch \rightarrow SMatch)} \textit{Semantic Search}}$$

Note that semantic search can solve the problems related to the ambiguity of natural language, namely, the problems of polysemy and synonymy, because concepts are unambiguous by definition.

In this paper we propose an approach in which semantic search is build on top of syntactic search. We call it semantics enabled syntactic search (*C-Search*). In our approach, we extend the classical syntactic search approach with semantics as follows:

- Indexing and searching documents is done using complex concepts. Complex concepts are computed by extracting multi-word phrases (that function as a single unit in the syntax of a sentence) and then by analyzing the meaning of these phrases. For example, phrase *A little dog or a huge cat* is converted into concept $C(A\ little\ dog\ or\ a\ huge\ cat)$ which then is used as a single term during document indexing and retrieval. Note that because we analyze multi-word phrases we solve the problem related to complex concepts discussed in Section 2.
- The notion of complex concepts allows us to represent uncertainty (partial information) coming from the coordination conjunction "OR" in natural language. For instance, phrase *A little dog or a huge cat* represents a concept which encodes the fact that it is unknown if *a little dog* or *a huge cat* is actually described in the document. Note that classical syntactic search is not capable of representing this kind of uncertainty and, therefore, of taking it into account during indexing and retrieval.
- Searching for documents describing concepts which are semantically related to query concepts. We assume that when a user is searching for a concept she is also interested in more specific concepts. For example, the extension of concept $C(A\ little\ dog\ or\ a\ huge\ cat)$ is a subset of the extension of concept $C(carnivores)$. Therefore, documents describing the former concept should be returned as answers to the query describing the later concept. In our approach, semantic matching is used in order to implement a search for related (complex) concepts. It allows us to solve the problem with related concepts discussed in Section 2.
- Semantic continuum. When we move from words to concepts in semantic search it is not always possible to find a concept which corresponds to a given word. The main reason for this problem is lack of background knowledge, i.e., a concept corresponding to a given word may not exist in the lexical database. In this case, in our approach, semantic search is reduced to an underlying syntactic search, i.e., we index and retrieve by words and not by concepts. This means that *C-Search* should perform at least as good as classical syntactic search.

An example of semantics enabled syntactic search using Inverted Index technology is given in Figure 8. Analogously to syntactic search, the query processing in semantics enabled Inverted Index is separated into two main steps: (i) to locate terms (which can be concepts or words) in dictionary which match query terms, and (ii) to search Inverted Index with these terms. Note that the second step is identical to that of syntactic search. First step may require semantic matching of (complex) concepts in a query to (complex) document concepts stored in the Inverted Index dictionary (see Section 4). Consider, for example, processing a query *mark of canine or feline*.

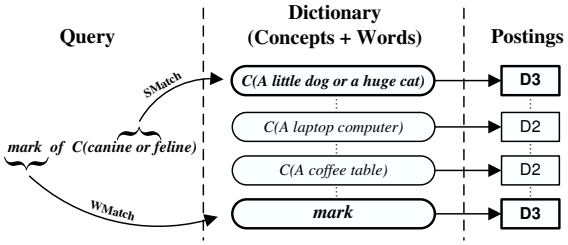


Fig. 8. Inverted Index in *C-Search*

Assume that words *canine* and *feline* present in our lexical database and word *mark* does not. In this case, phrase *canine or feline* will be converted into a complex concept $C(canine\ or\ feline)$ defined as a set of all fissiped mammals with non-retractile claws and typically long muzzles, or lithe-bodied roundheaded fissiped mammals with retractile claws, and word *mark* will not be changed. Modified query is processed as follows. First, for each query term, i.e., for word *mark* and for concept $C(canine\ or\ feline)$, we identify those terms in dictionary that match these query terms ($mark \xrightarrow{WMatch} \{mark\}$ and $C(canine\ or\ feline) \xrightarrow{SMatch} \{C(A\ little\ dog\ or\ a\ huge\ cat)\}$). Second, we search inverted index with computed dictionary terms ($mark \rightarrow \{D_3\}$ and $C(A\ little\ dog\ or\ a\ huge\ cat) \rightarrow \{D_3\}$). And finally, we take the intersection of document sets, found for every query term, as an answer to the given query (D_3 in our example).

4 Concept Indexing

In this section, we discuss how we implement the semantic matching of (complex) query concepts C^q to related (complex) document concepts C^d stored in the Inverted Index dictionary. Let $C_{ms}(C^q)$ be a set of all (complex) document concepts C^d matching (complex) query concept C^q, i.e., a set of all C^d, which are equivalent or more specific (ms) than the given C^q.

$$C_{ms}(C^q) = \{C^d \mid C^d \sqsubseteq C^q\} \tag{3}$$

During the query processing we need to compute set $C_{ms}(C^q)$ for every query concept C^q in the query. One approach to computing this set is to sequentially iterate through each concept C^d, compare it to the given query concept C^q by using semantic matching [7–9] technique, and collect those concepts for which semantic matching returns *more specific* ($\sqsubseteq$) relation. This approach may become prohibitory expensive as there may be thousands and millions of concepts stored in the document index dictionary. In this section we show how Inverted Index technology can be used in order to allow search for concepts in $C_{ms}(C^q)$, as efficient as Inverted Index technology can allow.

It is known, that in natural language, concepts are expressed as noun phrases [17]. In order to support complex concepts which encode uncertainty (see Section 3), we introduce the notion of descriptive phrase, where descriptive phrase is a set of noun-phrases, representing alternative concepts, connected by coordinating conjunction "OR":

$$descriptive\_phrase ::= noun\_phrase \ \{OR \ noun\_phrase\} \tag{4}$$

Descriptive phrases are converted into concepts expressed in Propositional Description Logic language L^C by following the approach described in [5]. Complex document concepts extracted from descriptive phrases are DNF formulas of atomic concepts representing words

$$C^d = \sqcup \sqcap A^d \tag{5}$$

For instance, descriptive phrase *A little dog or a huge cat* is converted into the following complex concept.

$$C_1^d(A \ little \ dog \ or \ a \ huge \ cat) = (A(little) \sqcap A(dog)) \sqcup (A(huge) \sqcap A(cat))$$

where $A(w)$ is an atomic concept corresponding to the word w.

Let $\mathbf{C^{DNF}}$ be the set of all complex document concepts and $\mathbf{C}^{\sqcap}$ be the set of conjunctive clauses from which concepts in $\mathbf{C^{DNF}}$ are composed. For instance, concept C_1^d belongs to $\mathbf{C^{DNF}}$ and its conjunctive clauses, i.e., concepts $C_2 = A(little) \sqcap A(dog)$ and $C_3 = A(huge) \sqcap A(cat)$, belong to $\mathbf{C}^{\sqcap}$.

Assume that query concept is converted into CNF

$$C^q = \sqcap \sqcup A^q \tag{6}$$

Recall also that if A,B, and C are concepts, then:

$$\begin{array}{l} (A \sqcup B) \sqsubseteq C \iff A \sqsubseteq C \text{ and } B \sqsubseteq C \\ A \sqsubseteq (B \sqcap C) \iff A \sqsubseteq B \text{ and } A \sqsubseteq C \end{array} \tag{7}$$

Given 5, 6, and 7, Formula 3 can be rewritten as follows:

$$
\begin{aligned}
C_{ms}(C^q) &= \{C^d \in \mathbf{C^{DNF}} \mid (\sqcup \sqcap A^d) \sqsubseteq (\sqcap \sqcup A^q)\} \\
&= \{C^d \in \mathbf{C^{DNF}} \mid \forall(\sqcup A^q) \in C^q, \forall(\sqcap A^d) \in C^d, (\sqcap A^d) \sqsubseteq (\sqcup A^q)\} \\
&= \bigcap_{\sqcup A^q \in C^q} \{C^d \in \mathbf{C^{DNF}} \mid \forall(\sqcap A^d) \in C^d, (\sqcap A^d) \sqsubseteq (\sqcup A^q)\} \qquad (8) \\
&= \bigcap_{\sqcup A^q \in C^q} C_{ms}(\sqcup A^q)
\end{aligned}
$$

where by $C_{ms}(\sqcup A^q)$ we denote the set of all concepts in $\mathbf{C^{DNF}}$ which are equivalent to or more specific than disjunctive clause $\sqcup A^q$:

$$C_{ms}(\sqcup A^q) = \{C^d \in \mathbf{C^{DNF}} \mid \forall(\sqcap A^d) \in C^d, (\sqcap A^d) \sqsubseteq (\sqcup A^q)\} \qquad (9)$$

Formula 9 can be rewritten as follows:

$$C_{ms}(\sqcup A^q) = \{C^d \in \mathbf{C^{DNF}} \mid \forall(\sqcap A^d) \in C^d, (\sqcap A^d) \in C^{\sqcap}_{ms}(\sqcup A^q)\} \qquad (10)$$

where by $C^{\sqcap}_{ms}(\sqcup A^q)$ we denote the set of all conjunctive clauses in $\mathbf{C}^{\sqcap}$ which are equivalent to or more specific than the given disjunctive clause $(\sqcup A^q)$:

$$C^{\sqcap}_{ms}(\sqcup A^q) = \{\sqcap A^d \in \mathbf{C}^{\sqcap} \mid (\sqcap A^d) \sqsubseteq (\sqcup A^q)\} \qquad (11)$$

Set $C_{ms}(\sqcup A^q)$ (see Formula 10) consists of complex concepts $C^d \in \mathbf{C^{DNF}}$ which have all its conjunctive clauses $\sqcap A^d$ in $C^{\sqcap}_{ms}(\sqcup A^q)$. In order to allow fast computation of $C_{ms}(\sqcup A^q)$ at query time, every concept $C^d \in \mathbf{C^{DNF}}$ containing more than one conjunctive clause is indexed (at indexing time) by its conjunctive clauses in the index which we call the concept $\sqcup$-index. Concept $\sqcup$-index stores a mapping from each conjunctive clause to a set of all concepts $C^d \in \mathbf{C^{DNF}}$ which contain this conjunctive clause ($conjunctive\_clause \rightarrow \{dnf\_concept\}$). In Figure 9 we show a fragment of a concept $\sqcup$-index for concept C^d_1.

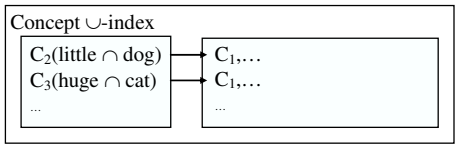


Fig. 9. Concept $\sqcup$-index

Now let us consider set $C^{\sqcap}_{ms}(\sqcup A^q)$ (see Formula 11). Notice that from WordNet we can extract only relations between atomic concepts (e.g., $A \sqsubseteq B$).

Therefore, using WordNet as our background knowledge, we can prove that $(\sqcap A^d) \sqsubseteq (\sqcup A^q)$ only if $\exists A^q, \exists A^d$, such that $A^d \sqsubseteq A^q$. Taking this into account, Formula 11 can be rewritten as follows:

$$C^{\sqcap}_{ms}(\sqcup A^q) = \{\sqcap A^d \in \mathbf{C}^{\sqcap} \mid \exists A^q, \exists A^d, \text{s.t. } A^d \sqsubseteq A^q\} = \bigcup_{A^q \in \sqcup A^q} \{\sqcap A^d \in \mathbf{C}^{\sqcap} \mid \exists A^d, \text{s.t. } A^d \sqsubseteq A^q\} = \bigcup_{A^q \in \sqcup A^q} C^{\sqcap}_{ms}(A^q) \quad (12)$$

where by $C^{\sqcap}_{ms}(A^q)$ we denote the set of all conjunctive clauses $\sqcap A^d \in \mathbf{C}^{\sqcap}$ which are equivalent to or more specific than the given atomic concept A^q:

$$C^{\sqcap}_{ms}(A^q) = \{\sqcap A^d \in \mathbf{C}^{\sqcap} \mid \exists A^d, \text{s.t. } A^d \sqsubseteq A^q\} \quad (13)$$

Formula 13 can be rewritten as follows:

$$C^{\sqcap}_{ms}(A^q) = \{\sqcap A^d \in \mathbf{C}^{\sqcap} \mid \exists A^d, \text{s.t. } A^d \in A_{ms}(A^q)\} \quad (14)$$

where by $A_{ms}(A^q)$ we denote a set of all atomic concepts A^d which are equivalent to or more specific than the given atomic concept A^q:

$$A_{ms}(A^q) = \{A^d \mid A^d \sqsubseteq A^q\} \quad (15)$$

Set $C^{\sqcap}_{ms}(A^q)$ (see Formula 14) consists of conjunctive clauses $\sqcap A^d \in \mathbf{C}^{\sqcap}$ with at least one its atomic concept A^d in $A_{ms}(A^q)$. In order to allow fast computation of $C^{\sqcap}_{ms}(A^q)$ at query time, conjunctive clauses $C^{\sqcap}$, containing more than one atomic concept, are indexed (at indexing time) by them in the index which we call the concept $\sqcap$-index. Concept $\sqcap$-index stores a mapping from each atomic concept to a set of all conjunctive clauses in $\mathbf{C}^{\sqcap}$ which contains this concept ($atomic\_concept \rightarrow \{conjunctive\_clause\}$). In Figure 10, we show a fragment of a concept $\sqcap$-index which indexes conjunctive clauses of concept C^d_1, i.e., it indexes concepts C_2 and C_3.

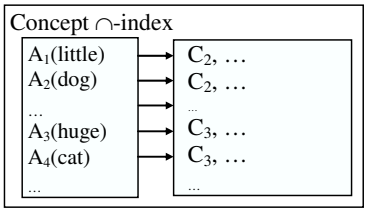


Fig. 10. Concept $\sqcap$-index

Now we will describe how concept retrieval, i.e., computation of $C_{ms}(C^q)$, can be performed given that concept $\sqcap$- and $\sqcup$- indices were constructed. As an example of query concept we will consider the following concept.

$$C_1^q \equiv A(canine) \sqcup A(feline)$$

Set $C_{ms}(C^q)$ is computed in the following six steps:

1. Query concept is converted into CNF. For example, concept C_1^q is already in CNF, so it will not be changed.

2. For every atomic concept A^q we search the lexical database for all atomic concepts which are equivalent to or more specific than A^q, i.e., we compute set $A_{ms}(A^q)$ (see Formula 15). For example, $A_{ms}(A(canine)) = \{A(dog), A(wolf), \dots\}$ and $A_{ms}(A(feline)) = \{A(cat), A(lion), \dots\}$.

3. For every atomic concept A^q we compute set $C_{ms}^{\sqcap}(A^q)$ (see Formula 14), i.e., a set of all conjunctive clauses which are equivalent to or more specific than A^q. Sets $C_{ms}^{\sqcap}(A^q)$ are computed by searching concept $\sqcap$-index with atomic concepts in $A_{ms}(A^q)$. For example, $C_{ms}^{\sqcap}(A(canine)) = \{C_2, \dots\}$ and $C_{ms}^{\sqcap}(A(feline)) = \{C_3, \dots\}$.

4. For every disjunctive clause $\sqcup A^q$ we compute a set $C_{ms}^{\sqcap}(\sqcup A^q)$ (see Formula 12), i.e., a set of all conjunctive clauses which are equivalent to or more specific than disjunctive clause $\sqcup A^q$. We compute $C_{ms}^{\sqcap}(\sqcup A^q)$ by taking the union of all the sets $C_{ms}^{\sqcap}(A^q)$:

$$C_{ms}^{\sqcap}(\sqcup A^q) = \bigcup_{A^q \in \sqcup A^q} C_{ms}^{\sqcap}(A^q) \tag{16}$$

For example, $C_{ms}^{\sqcap}(A(canine) \sqcup A(feline)) = \{C_2, C_3, \dots\}$.

5. For every disjunctive clause $\sqcup A^q$ we compute set $C_{ms}(\sqcup A^q)$ (see Formula 10), i.e., a set of all complex document concepts in $\mathbf{C^{DNF}}$ which are equivalent to or more specific than disjunctive clause $\sqcup A^q$. Sets $C_{ms}(\sqcup A^q)$ are computed by searching concept $\sqcup$-index with conjunctive clauses in $C_{ms}^{\sqcap}(\sqcup A^q)$. Note that we search only for concepts C^d which have all its conjunctive clauses in $C_{ms}^{\sqcap}(\sqcup A^q)$, and discard other concepts. For example, $C_{ms}(A(canine) \sqcup A(feline)) = \{C_1^d, \dots\}$.

6. We compute $C_{ms}(C^q)$ (see Formula 8) by taking the intersection of all the sets $C_{ms}(\sqcup A^q)$:

$$C_{ms}(C^q) = \bigcap_{\sqcup A^q \in C^q} C_{ms}(\sqcup A^q) \tag{17}$$

For example, concept C_1^q has only one disjunctive clause, therefore, set $C_{ms}(C_1^q)$ is equal to set $C_{ms}(A(canine) \sqcup A(feline))$, i.e., $C_{ms}(C_1^q) = \{C_1^d, \dots\}$.

Note that steps described above require searching the lexical database, searching inverted indices, computing union and intersection of sets. All these operations are fast in practice and, therefore, the computation of $C_{ms}(C^q)$ is also time efficient.

5 Evaluation

The data-set, used for the evaluation of our approach, was generated from *Home*[1] subtree of DMoz web directory. Documents classified to nodes in the sub-tree are used as a document set, labels of nodes are used as a query set[2], and node-document links represents relevance of documents to queries. The data-set consists of 29506 documents and 890 queries.

To locate descriptive phrases in documents and queries we, first, follow a standard NLP pipeline to locate noun phrases, i.e., we perform sentence detection, tokenization, part-of-speech (POS) tagging, and noun phrase chunking and after that we perform addition step which we call descriptive phrase chunking, where the goal of this step is to locate descriptive phrases, satisfying Formula 4, given that noun phrases are already identified. In particular, we use the GATE [3] infrastructure and resources. Queries usually are short phrases and, as shown in [20], standard NLP technology, primarily designed to be applied on full-fledged sentences, is not effective enough in its application on such phrases. Therefore, for query processing we use a POS-tagger from [20], which is specifically trained on short phrases.

The conversion of descriptive phrases into formulas in L^C was performed as follows. First, for each token in a descriptive phrase, we looked up and enumerated its meaning(s) in WordNet [13]. Next, we performed word sense filtering, i.e., we discard word senses which are not relevant in the given context. In order to do this, we followed the approach presented in [20], which exploits POS tagging information and WordNet lexical database for WSD in short noun phrases. Differently from [20] we did not use the filtering technique which leaves only the most probable sense of the word, because of its low accuracy. Finally, for every descriptive phrase we build a complex concept which encodes the meaning of this phrase. Each word is represented as an atomic concept, noun phrases are translated into logical conjunction of atomic concepts, and descriptive phrases are translated into logical disjunction of formulas for noun phrases.

In order to evaluate our approach, we built two inverted indices. First index was build by using Lucene[3]. Second index was build by using semantics enabled version of Lucene, which was implemented following the methodology described in Sections 3 and 4. Evaluation results for both indexes are reported in Table 1.

Table 1. Evaluation results

	Precision (%)	Recall (%)
Lucene	7.72	20.43
C-Search	8.40	24.69

[1] http://www.dmoz.org/Home/.

[2] Queries were created by concatenation of node's and its parent's labels adding "AND" in between. Queries created from nodes which contained less than 10 or more than 100 documents were eliminated from the query set.

[3] http://lucene.apache.org/java/docs/index.html

After manual inspection of the results, we concluded that the main reason for low precision and recall, achieved by Lucene and C-Search, is low quality of the data-set. Documents in our collection represent web-sites with many interconnected pages, whereas we indexed only root page for each web-site. This leads to low recall because relevant information can be stored on pages other than the root page. Queries in the used data-set are also not always good, for instance, query *purchasing AND new* (created from node New[4]) was associated only with documents about automobiles because nodes *purchasing* and *new* are children of the node *automobiles*, whereas, obviously, information about purchasing of something new can be found in documents from other subtrees. The problem with queries leads to low precision. Nevertheless, in this particular data set, *C-Search* performed better than purely syntactic search, which supports the underlying assumption of our approach.

6 Related work

The fact that the syntactic nature of the classical IR leads to problems with precision was recognized in the IR community long ago (e.g., see [18]). There were two major approaches to addressing this problem: one is based on natural language processing and machine learning techniques in which (noun) phrases in a document corpus are identified and organized in a subsumption hierarchy which is then used to improve the precision of retrieval (e.g., see [19]); and the other is based on a linguistic database which is used to associate words in a document corpus with atomic lexical concepts in the database and then to index these documents by the associated concepts (e.g., see [16]). Our approach is different from these two because the former approach is still essentially syntactic (and semantics is only implicitly derived with no guarantee of correctness) and in the latter approach only atomic concepts are indexed, wherein *C-Search* allows for indexing of complex concepts and explicitly take into account possible relations between them which allows it to compute more accurate query results. More importantly, our approach *extends* syntactic search and not replaces it as it is the case in the latter approach. Therefore, our approach supports a continuum from purely syntactic to fully semantic IR in which indexing and retrieval can be performed at any point of the continuum depending on how much semantic data are available.

In the Semantic Web community, semantic search is primarily seen as the task of querying an RDF graph based on the mapping of terms appearing in the input natural language query to the elements of the graph. An analysis of existing semantic search systems is provided in [10]. Our approach is principally different because, like in classical IR, input query is mapped to document contents and not to elements of a knowledge representation structure. Document retrieval approaches developed in the context of the Semantic Web are surveyed in [12]. Matching of document and query representations, in these approaches, is based on query expansion (e.g., see [2]), graph traversal (e.g., see [15]), and

[4] http://www.dmoz.org/Home/Consumer_Information/Automobiles/Purchasing/New/.

RDF reasoning (e.g., see [4]). Differently from these approaches, in *C-Search*, document and query representations are matched via semantic matching [7–9] of complex concepts, which is implemented by using Inverted Index technology.

7 Conclusions

In this paper we presented an approach in which syntactic IR is extended with a semantics layer which allows it to improve over results of a purely syntactic search. The proposed approach performs as good as syntactic search while allowing for an improvement where semantics is properly integrated. In principle, our approach supports a continuum from purely syntactic to fully semantic IR in which indexing and retrieval can be performed at any point of the continuum depending on how much semantic data are available. The reported experimental results demonstrate the proof of concept of the proposed solution. Future work includes: (i) development of document relevance metrics based on both syntactic and semantic similarity of query and document descriptions; (ii) integration of more accurate algorithms for concept identification during indexing; (iii) comparing the performance of the proposed solution with the state-of-the-art syntactic IR systems using a syntactic IR benchmark; and, (iv) providing support for queries in which concepts can be associated with a semantic scope such as equivalence, more/less general, disjoint.

References

1. Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
2. Irene Celino, Emanuele Della Valle, Dario Cerizza, and Andrea Turati. Squiggle: a semantic search engine for indexing and retrieval of multimedia content. In *SEMPS*, pages 20–34, 2006.
3. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.
4. John Davies and Richard Weeks. QuizRDF: Search technology for the semantic web. In *HICSS '04: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4*, page 40112, Washington, DC, USA, 2004. IEEE Computer Society.
5. Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. In *Journal on Data Semantics (JoDS) VIII*, Winter 2006.
6. Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In *Proc. of ECAI*, 2006.
7. Fausto Giunchiglia and Mikalai Yatskevich. Element level semantic matching. In *Meaning Coordination and Negotiation workshop, ISWC*, 2004.
8. Fausto Giunchiglia, Mikalai Yatskevich, and Enrico Giunchiglia. Efficient semantic matching. In *Proc. of ESWC*, Lecture Notes in Computer Science. Springer, 2005.

9. Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics (JoDS)*, 9:1–38, 2007.
10. M. Hildebrand, J. van Ossenbruggen, and L. Hardman. An analysis of search-based user interaction on the semantic web. Technical Report INS-E0706, Centrum voor Wiskunde en Informatica, MAY 2007.
11. Bernardo Magnini, Manuela Speranza, and Christian Girardi. A semantic-based approach to interoperability of classification hierarchies: evaluation of linguistic techniques. *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 11–33, 2004.
12. Christoph Mangold. A survey and classification of semantic search approaches. *Int. J. Metadata Semantics and Ontology*, 2(1):23–34, 2007.
13. George Miller. *WordNet: An electronic Lexical Database*. MIT Press, 1998.
14. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
15. C. Rocha, D. Schwabe, and M. de Aragao. A hybrid approach for searching in the semantic web. In *Proceedings of the 13th International World Wide Web Conference*, 2004.
16. Hinrich Schutze and Jan O. Pedersen. Information retrieval based on word senses. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, 1995.
17. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
18. Christopher Stokoe, Michael P. Oakes, and John Tait. Word sense disambiguation in information retrieval revisited. pages 159–166, 2003.
19. William A. Woods. Conceptual indexing: A better way to organize knowledge. 1997.
20. I. Zaihrayeu, L. Sun, F. Giunchiglia, W. Pan, Q. Ju, M. Chi, and X. Huang. From web directories to ontologies: Natural language processing challenges. In *6th International Semantic Web Conference (ISWC 2007)*. Springer, 2007.