Improving Retrieval Experience Exploiting Semantic Representation of Documents

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

Department of Computer Science
University of Bari
70125 Bari, Italy
{basilepp,acaputo,al.gentile,degemmis,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.

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. Finally, main work related to the research presented in this paper is discussed in Section 7. 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 and tagged with a unique Wikipedia URI. A Named Entity Disambiguation (NED) algorithm is adopted to assign Wikipedia URI to words that represent entities.

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 vector for the named entities level might be 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.

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. 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.
- Text Operations - It performs basic and more advanced NLP operations. Basic operations implemented are: *Stop words elimination*, *Stemming*, *POS-tagging*, *Lemmatization* and Named Entity Recognition (NER). Besides basic NLP processing, more advanced procedures were designed for the semantic levels of SENSE: *Named Entity Disambiguation (NED)* and *Word Sense Disambiguation (WSD)*. The core component that performs all the steps (WSD and NED included) needed for building the document representation at the meaning level is META [4].
- 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.

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.

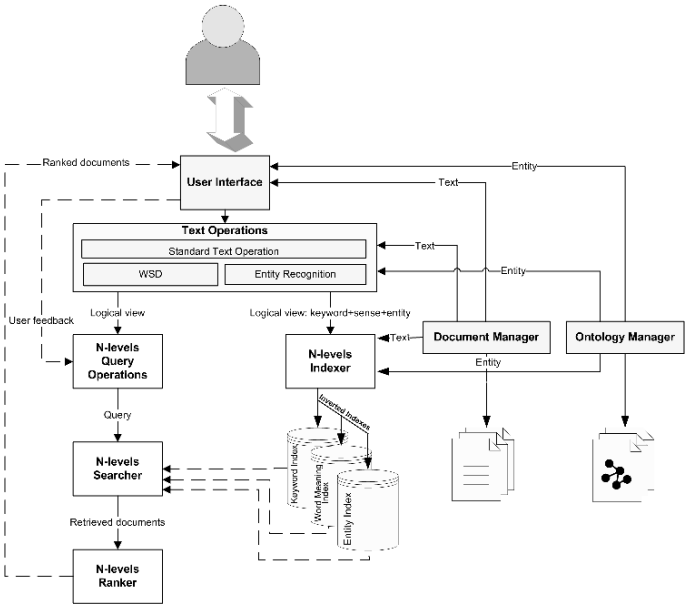

Fig. 1. System Architecture

– 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[1]. 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.

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

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*, 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 [5]. 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.

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 each s_j with the words in the context of w_i. Notice that $k \leq h$ because some words, such as proper names, might not be found in WordNet.

Given the 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 [5, 16]. 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 computed by the WSD procedure, $X = [s_1, s_2, \ldots, s_k]$, the basic idea is to compute a *partial* weight for each $s_j \in X$, and then to improve this weight by finding out some relations among 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:

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

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.

Finally, the synset confidence factor (α) is used to weigh the SFIDF value. Thus, the final local score for synset s_j in d_i is:

$$\text{SFIDF}(s_j, d_i) \cdot (1 + \alpha) \tag{2}$$

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. For the meaning level, both query and document vectors contain synsets instead of keywords. Given a query q and a document d_i, the synset similarity is computed as:

$$synsim(q, d_i) = C(q, d_i) \cdot \sum_{s_j \in q} (\text{SFIDF}(s_j, d_i)(1 + \alpha) \cdot N(d_i)) \tag{3}$$

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 [11]. For the purpose of SENSE a further step is needed and it consists of Entity Disambiguation.

The Named Entity Disambiguation is a specialization of classic WSD. Within this work the task has been performed adapting the Lesk dictionary-based WSD algorithm [1]. The basic assumption is that words in a given neighbourhood will probably share a common topic. Apart from knowledge about the context of a target entity (the immediate surrounding words), the algorithm requires a machine readable dictionary, with an entry for each possible sense for a word. Each token that refers to an Entity in the original document is tagged with the Wikipedia URI that better represents the Entity meaning, thus Wikipedia plays the role of *sense inventory* in the proposed algorithm for NED.

Considering that the words to disambiguate for Named Entity Disambiguation are only those representing an Entity, the algorithm works as follows.

Given an input document $d = [w_1, \ldots, w_{j-1}, w_j = e_j^1, w_{j+1}, \ldots, w_{h-1}, w_h = e_h^2, w_{h+1}, \ldots]$, where $[e_j^1, e_j^2, \ldots, e_j^k]$ are k entities and $[w_1, w_2, \ldots]$ are tokens occurring in the document d, the algorithm returns a list of Wikipedia URIs $X_d = [s_1, s_2, \ldots, s_k]$. Each element s_i is obtained by disambiguating the *target entity* e_i on the ground of the information obtained from Wikipedia for each candidate URI (Wikipedia page content of the URI) and words in the *context* C of e_i. The list of Wikipedia candidate URIs comes from the Wikipedia Search page for e_i and only the first ten results are exploited. The *context* C of the target entity e_i is defined as a window of n words to the left and another n words to the right, for a total of $2n$ words surrounding e_i. In the current version of the algorithm, if other entities occur in the context of the target entity, they are considered as words and not as entities.

Similarly to the meaning level, documents are represented at the entity level by using an adaptation of the vector space model, the rapresentation adopted for this level is a *bag-of-entities* rather than a *bag-of-synsets*. The vocabulary is the set of entities recognized by the NER text operation in the collection; specifically each entity is identified by the URI of the entity instance (borrowed from Wikipedia). As first attempt, a classical tf-idf heuristic has been adopted to score entities and cosine similarity has been implemented as local similarity function.

6 Global Ranking

Given a query q, each local similarity function produces a local ranked list of relevant documents. All the local lists must be merged in order to give a single ranked list to the user. The global ranking function is devoted to this task.

Algorithms for merging ranked lists are widely used by meta-search engines, which send user requests to several search engines and aggregate results into a single list [9]. Our strategy for defining the *global ranking function* is thus inspired by prior work on meta-search engines.

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 denoted by $\hat{\tau}$. The two steps are thoroughly described in [2]. We choose to adopt Z-Score normalization and ComSUM respectively as score normalization and rank aggregation function. In particular the Z-Score normalization is computed using the following formula:

$$w^{\tau_j}(x_i) = \frac{s^{\tau_j}(x_i) - \mu_{s^{\tau_j}}}{\sigma_{s^{\tau_j}}}$$

Regarding ComSUM list aggregation method, the score of document x_i in the global list is computed by summing all the normalized scores for x_i:

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

where R is the set of all local list.

7 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 [17–19]. On the other hand, semantic similarity measures have the potential to redefine the similarity between a document and a user query [6, 12, 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 [10], 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 [7, 8] try to combine keyword search with techniques for navigating and querying ontologies. In [7], documents are annotated with concepts in a domain ontology and indexed using classical Bag-Of-Words model, while in [8] 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.

8 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 disambiguation method to identify entities within text. We defined a global ranking functions describing how to merge rankings produced by different levels. A preliminary evaluation involving both the keyword and the word meaning level has been performed at CLEF 2008 [3]. As future work, we plan to perform an extended experimental session and to investigate new strategies for representing documents both at the synset and at the entity level.

References

1. S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 136–145, London, UK, 2002. Springer-Verlag.
2. P. Basile, A. Caputo, A. L. Gentile, M. Degemmis, P. Lops, and G. Semeraro. Enhancing semantic search using n-levels document representation. In S. Bloehdorn, M. Grobelnik, P. Mika, and D. T. Tran, editors, *SemSearch*, volume 334 of *CEUR Workshop Proceedings*, pages 29–43. CEUR-WS.org, 2008.
3. P. Basile, A. Caputo, and G. Semeraro. UNIBA-SENSE at CLEF 2008: SEmantic N-levels Search Engine. *CLEF 2008: Ad Hoc Track Overview*, 2008. CLEF 2008 Working Notes.
4. 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.
5. 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.
6. C. Corley and R. Mihalcea. Measures of text semantic similarity. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence*, 2005.
7. J. Davies and R. Weeks. QuizRDF: Search technology for the Semantic Web. In *37th Hawaii Int. Conf. on System Sciences*. IEEE Press, 2004.
8. G. Ducatel, Z. Cui, and B. Azvine. Hybrid ontology and keyword matching indexing system. In *Proc. of IntraWebs Workshop at WWW2006*, 2006.
9. 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.
10. J. Gonzalo, F. Verdejo, I. Chugur, and J. M. Cigarrán. Indexing with wordnet synsets can improve text retrieval. *CoRR*, cmp-lg/9808002, 1998.
11. R. Grishman and B. Sundheim. Message understanding conference- 6: A brief history. In *COLING*, pages 466–471, 1996.
12. J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008, 1997.
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. 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.
17. 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.
18. E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proc. of the 17th SIGIR Conf.*, pages 61–69, 1994.
19. E. M. Voorhees. *WordNet: An Electronic Lexical Database*, chapter 12: Using WordNet for text retrieval, pages 285–304. Cambridge: The MIT Press, 1998.