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

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.

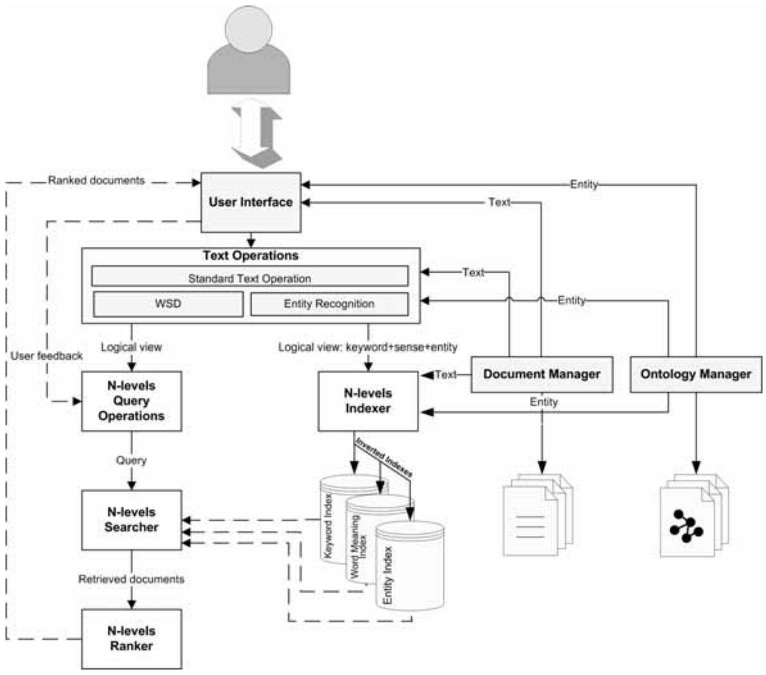

Fig. 1. System Architecture

– 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:

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

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:

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

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`

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

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:

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

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:

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

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	$w^{\tau_j}(x_i) = \begin{matrix} 1 - \frac{\tau_j(x_i)-1}{\lvert U\rvert} & \text{if } x_i \in \tau_j \\ \frac{1}{2} + \frac{\lvert\tau_j\rvert - 1}{2 \cdot \lvert U\rvert} & \text{otherwise} \end{matrix}$

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

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

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.