Journal of Universal Computer Science, vol. 13, no. 12 (2007), 1908-1935
submitted: 15/4/07, accepted: 31/7/07, appeared: 1/12/07 © *J.UCS*

Discovering the Semantics of User Keywords

Raquel Trillo, Jorge Gracia, Mauricio Espinoza and Eduardo Mena
(IIS Department, University of Zaragoza, Spain
{raqueltl, jogracia, mespinoz, emena}@unizar.es)

Abstract: The technology in the field of digital media generates huge amounts of textual information every day, so mechanisms to retrieve relevant information are needed. Under these circumstances, many times current web search engines do not provide users with the information they seek, because these search tools mainly use syntax based techniques. However, search engines based on semantic and context information can help overcome some of the limitations of current alternatives.

In this paper, we propose a system that takes as input a list of plain keywords provided by a user and translates them into a query expressed in a formal language without ambiguity. Our system discovers the semantics of user keywords by consulting the knowledge represented by many (heterogeneous and distributed) ontologies. Then, context information is used to remove ambiguity and build the most probable query. Our experiments indicate that our system discovers the user's information need better than traditional search engines when the semantics of the request is not the most popular on the Web.

Key Words: Querying the Semantic Web, semantic interoperability, information retrieval
Category: H.3.3, H.3.5

1 Introduction

Nowadays, the World Wide Web is an information resource with a virtually unlimited potential. However, this potential is not fully exploited by traditional web search engines, because they only retrieve the documents containing the user keywords, and many documents may convey the desired semantic information without containing those keywords. Besides, current search engines do not consider the context of the user keywords: the same keywords can be used by different users with the purpose of accessing to different information, i.e., keywords can be interpreted differently as they lack explicit semantics. So, it is required to consider the different possible meanings (senses) of the user keywords and to take into account the context of each keyword to improve the retrieval of information from the Web.

As a motivating example, let us suppose that a user wants to find information about the life of famous people and therefore writes the following keywords: *"life of stars"*. In this case, Google returns about 192,000,000 hits[1] when we enter those keywords but, unfortunately, the first tens of hits link to astronomy-related

[1] Obtained on October 15, 2007.

web pages. We see how syntactic-based search engines are very influenced by the enormous amount of information about popular issues on the Web. Similar results are obtained if the keyword is "Java": Java as programming language eclipses the rest of possible senses (the Indonesian island, a coffee plant, different US cities, etc).

In this context, the increasing pools of ontologies (which offer a formal, explicit specification of a shared conceptualization [Gru93]) available on the Web can help to discover the semantics of user keywords, because ontologies provide us with a non-ambiguous description of their terms. Moreover, the more ontologies consulted (each one representing the point of view of their creators), the more chances to find the semantics that the user assigned to the entered keywords.

In this paper, we propose a system that takes as input a list of plain keywords provided by a user, finds out their possible semantics (their meanings or senses) and outputs a formal query which express the user's information need without ambiguity. Firstly, the system discovers the semantics of the user keywords in run-time by accessing to the shared knowledge stored in different ontology pools available on the Web, and so, for each keyword, it obtains a list of possible senses for each keyword. Secondly, the system deals with the possible semantic overlapping among senses and removes the redundancy among them by using a *synonymy probability* measure. Thirdly, a disambiguation method is used to select the most probable intended sense of each user keyword by considering the possible senses of the rest of keywords and a semantic relatedness measure. Finally after the system has selected a sense for each user keyword, those senses are combined to build a query expressed in a knowledge representation language; this query represents what the user is looking for in an unambiguous way. The obtained query could be used to retrieve underlying relevant data indexed by the pool of ontologies [CFV07, MI01] or query expansion in traditional search engines [GMV99, PTBW05], but these tasks are out of the scope of this paper. We have developed a prototype and executed some experiments that indicate that our system behaves better than traditional web search engines in obtaining the intended meaning of user queries when their semantics is not the most popular on the Web.

The rest of this paper is as follows. In Section 2, we overview our approach and describe the main steps performed by the system. In Section 3 we show how the possible senses of each user keyword are obtained. In Section 4 we describe the algorithm that computes the *synonymy probability* between senses to eliminate the possible redundancy. In Section 5, we present how to remove the ambiguity of user keywords. In Section 6 we explain how to build queries and rank them. Some experimental results of the developed prototype are shown in Section 7. Related work can be found in Section 8. Finally, conclusions and future work appear in Section 9.

2 Overview of the System

The target of our system is to translate a list of plain keywords provided by a user into a query expressed in a knowledge representation language, which expresses the user's information need without ambiguity. The main steps performed by the system are the following (see Figure 1):

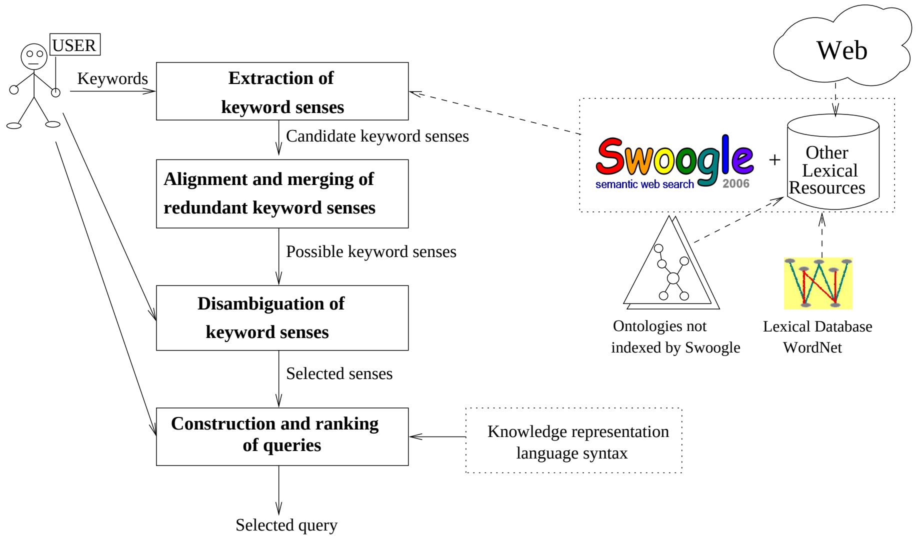

Figure 1: General overview of the system.

1. *Extraction of keyword senses*: Like in traditional search engines, the user must provide the system with a list of keywords $K = \{k_1...k_n\}$. The system queries WordNet [Mil95], Swoogle [FDP$^+$05] and other ontology repositories to find ontological terms that match the normalized keywords. The system builds a sense for each matching obtained and then the extracted senses are semantically enriched with the ontological terms of their synonyms by also searching in the ontology pool. The result is a list of candidate keyword senses for each user keyword. A detailed description of this process is presented in Section 3.

2. *Alignment and merging of redundant keyword senses*: As the obtained senses were built with terms coming from different ontologies, they could represent

the same semantics. An incremental algorithm is used to remove these possible redundancies, aligning the different keyword senses and merging them when they are similar enough: Each input sense is compared with the rest of stored senses in order to decide whether it can be integrated with some other sense. The senses are merged when the estimated *synonymy probability* between them exceeds a certain threshold. Thus, the result is a set of *different* possible senses for each user keyword entered. A more detailed description of this process can be found in Section 4.

3. *Disambiguation of keyword senses*: A disambiguation process is needed to select the most probable intended sense of each user keyword by considering the possible senses of the rest of keywords. The system uses a *semantic relatedness measure* based on information provided by Google, to compute the correlation between the senses of a particular user keyword and the senses of its neighbor keywords. Thus, the best sense for each keyword will be selected according to its context. A more detailed description of this process can be found in Section 5.

4. *Construction and ranking of queries*: From the selected senses in the previous step, the system builds a ranked list of queries that represent the possible semantics intended by the user for the initial keyword set. These queries are expressed in a knowledge representation language (for example in OWL [DS04] and BACK [Pel91], but any other is possible). The system combines the ontological information of the involved senses with the different operators of the chosen language by using parsing techniques and a Description Logics reasoner [BCM$^+$03]. Besides, a weight (*relevance probability*) is associated to each query to rank them. This weight represents the probability that the query expresses the intention of the user. The system can use any language although the specific queries generated depend on the expressivity of the chosen language. A more detailed description of this process can be found in Section 6.

The obtained semantic queries could be used for different purposes: 1) to retrieve underlying relevant data indexed by the pool of ontologies [CFV07, MI01, FB03]; 2) even if the utilized ontologies do not index data, semantic queries are still useful to address query expansion in traditional search engines, as they do in [GMV99, PTBW05], or 3) to select a particular set of results provided by search engines which use clustering or classification techniques (as for example Clusty[2]), because those queries have a well-defined semantics that can be used to find semantic correspondences with the returned clusters.

[2] http://www.clusty.com

3 Extraction of Keyword Senses

In this section we detail our contribution to automatically retrieve candidate keyword senses from a set of user keywords (see Figure 2).

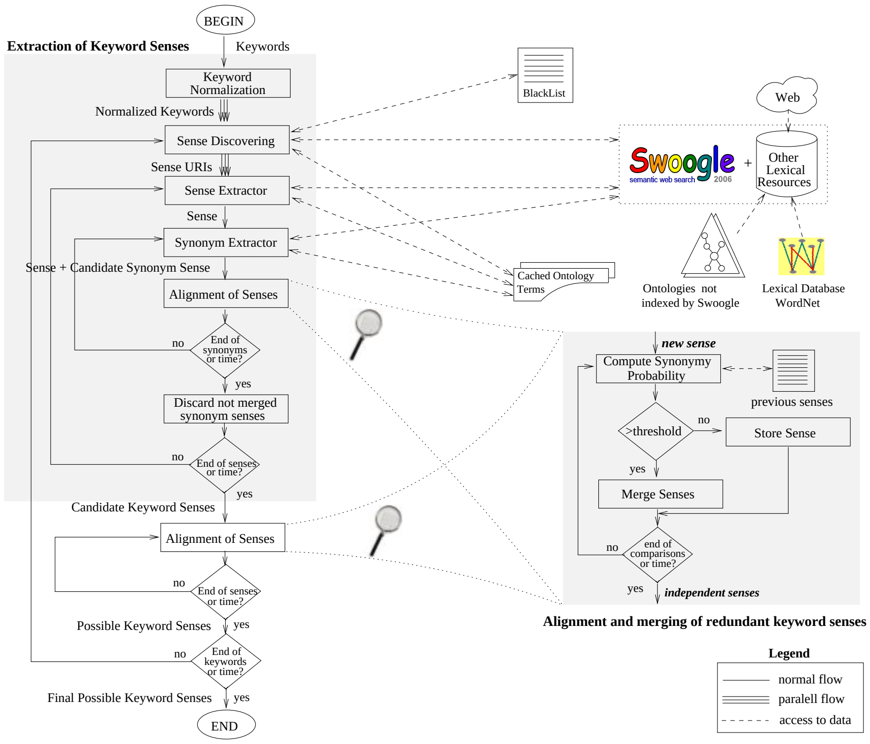

Figure 2: Obtaining the possible senses of user keywords.

Initially, the user keywords are normalized (e.g., rewriting them in lowercase, removing hyphens, etc.) generating a list of normalized keywords $NK = \{nk_1...nk_n\}$. Then, the *sense discovering* module queries different third-party knowledge pools: Swoogle [FDP$^+$05], which indexes many ontologies available on the Web, remote lexical resources as Wordnet [Mil95], and other ontologies not indexed by Swoogle to find ontological terms that match those normalized keywords. A *black list* of ontologies is managed to avoid known low-quality ontologies, and a *buffer* stores previously parsed ontology terms to avoid accessing the same remote data twice. We advocate using a pool of ontologies instead of just a single one, like WordNet (as many works do [LDKG04]), because many technical or subject-specific senses of a keyword cannot be found in just one ontology. For example, for the keyword “developer”, WordNet does not provide the sense “someone who programs software”. However this sense can be obtained from the Software Project Ontology[3], so it can be added to the senses retrieved from WordNet (and others) to be considered by our system.

Secondly, the *sense extractor* module builds a sense for each URI (Uniform Resource Identifier) obtained in the previous step. In our approach, a sense (meaning) of a normalized keyword nk, denoted by s_{nk}, is represented as a tuple s_{nk} = <*s, grph, descr, pop, syndgr*>, where s is the list of synonym names[4] of normalized keyword nk, *grph* describes the sense s_{nk} by means of the hierarchical graph of hypernyms and hyponyms of synonym terms found in one or more ontologies, *descr* is a description in natural language of such a sense, and *pop* and *syndgr* measure the degree of popularity of this sense; *pop* is the number of times it appears in the ontology pool and *syndgr* represents a percentage of synonymy degree of different ontological terms integrated in the sense (see section 4 for more details). Thus, senses are built with the information retrieved from matching terms in the ontology pool [EGTM06]. Notice that the more ontologies or knowledge bases accessed, the more chances to find the semantics that the user is looking for. As matching terms could be ontology classes, properties or individuals, three lists of candidate keyword senses are associated with each normalized keyword nk: S_{nk}^{class}, S_{nk}^{prop} and S_{nk}^{indv}.

Thirdly, each keyword sense extracted is enhanced incrementally with their synonym senses by the *synonym extractor* module (which also searches the ontology pool). The module that aligns senses integrates the keyword sense with those synonym senses representing the same semantics, and discards the synonym senses that do not enrich the keyword sense. So the result is a list of possible senses for each keyword. We will provide more details later in this section.

Notice that the whole process can be limited in time. Also, senses obtention is executed in parallel for each keyword; within that task, the semantic enrichment of each keyword sense with its synonym senses is performed in parallel too.

In Figure 3, we show the first three steps of the extraction of keyword senses when user keywords are “life of stars”. Our prototype finds 16 matches denoted by URIs for the keyword “life” and 13 for “star”. For simplicity we only show three. Notice that the second sense of keyword “star”, $s2_{star}^{class}$, has the semantics assigned by the user (“famous people”). However $s3_{star}^{prop}$ corresponds to a different interpretation (it represents a property sense with the class “hotel” as domain).

[3] http://keg.cs.tsinghua.edu.cn/persons/tj/ontology/software.owl

[4] The system extracts the synonym names of a term by consulting the synonym relationships defined in the ontology of such a term. For example the relations *equivalentClass* and *equivalentProperty* of OWL.

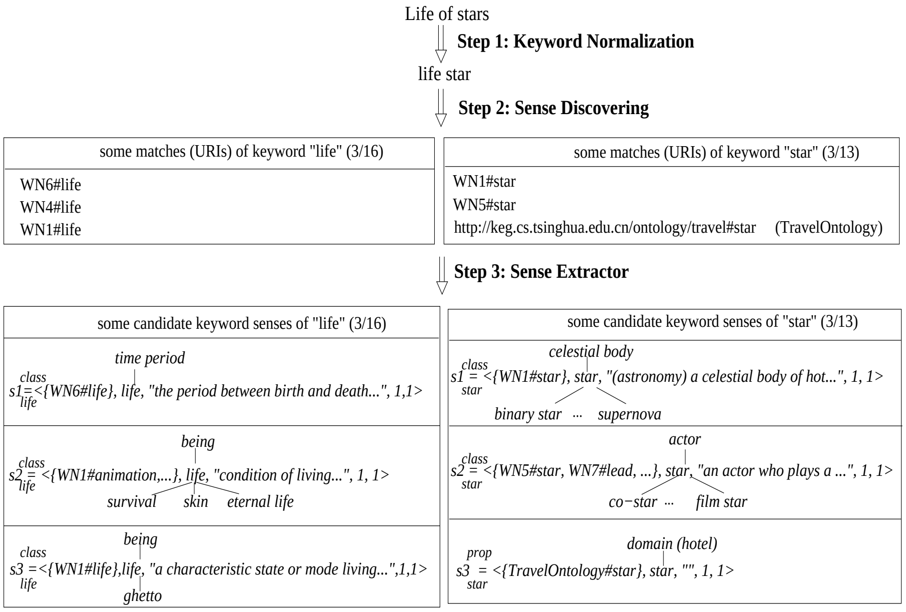

Figure 3: Extraction of senses of the user keywords "life" and "star".

Semantic Enrichment of Keyword Senses with Synonym Senses

We explained before that each user keyword is searched (syntactically) in the ontology pool, and a sense is built for each matching term by considering its ontological context. However, we realized that something was missing with such a simple ontology matching: many relevant ontology terms do not match just because their names differ from the searched keyword; for example, if the user was looking for "lorry" information about "truck" was not obtained. So we decided to make our approach not so dependent on the words chosen (user keywords are just a guide to begin the search of senses) by considering the synonyms found during the process. In other words, the system retrieves similar results whether the user keyword is *lorry* or *truck*, if any ontology classifies them as synonyms.

Therefore our system takes advantage of the shared ontologies available on the Web and semantically enriches the keyword senses with senses extracted from their synonyms. After the senses extractor module obtains the synonym names of a term by consulting the synonym relationships of its ontology; the synonym extractor module performs a process similar to discovering and senses extractor modules, to build new senses from other ontologies which match with the synonym names. Notice that synonym names are stored in the sense structure shown before, which gets upgraded every time a sense is integrated with a (very similar) sense coming from other ontology. In order to evaluate the semantic similarity between the sense of a keyword and their synonyms, the system performs the sense alignment step (detailed in Section 4) which determinates whether the semantics of the keyword sense and each synonym sense found represent the same semantics or not.

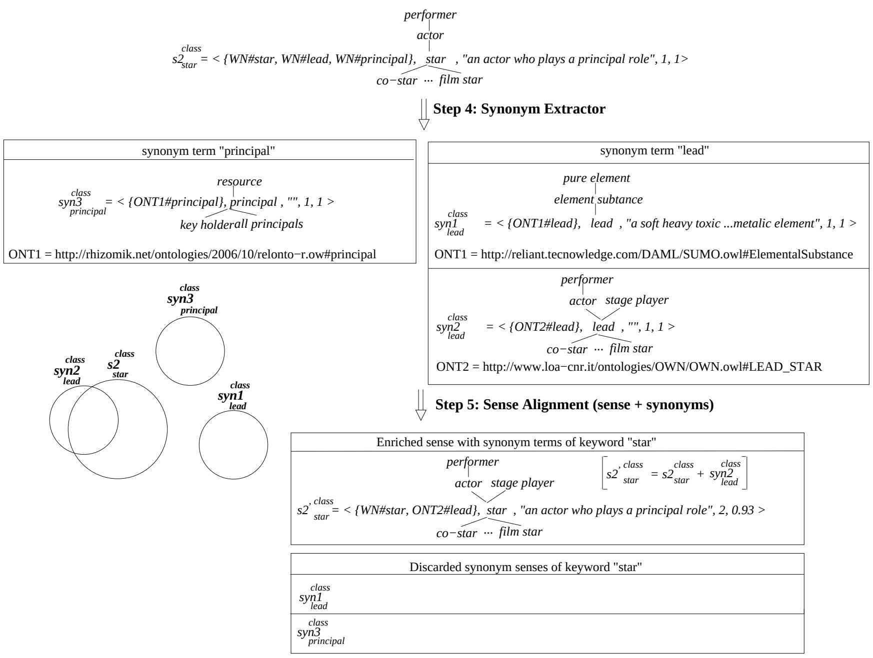

Figure 4: Semantic enrichment of the sense of keyword "star" $s2^{class}_{star}$.

In Figure 4, we show the steps given to semantically enrich the sense of "star" $s2^{class}_{star}$. On the bottom left, circles represent the senses extracted from the ontology pool (the sense of keyword *star* $s2^{class}_{star}$ is represented by the biggest circle). Two senses intersect when they have some degree of synonymy. Our prototype finds terms *lead* and *principal* as synonyms of $s2^{class}_{star}$ in WordNet. However, we see that some "synonyms", for example, $syn1^{class}_{lead}$ from ONT1, do not represent the same semantics as their keyword sense; in the example, $s2^{class}_{star}$ although *lead* was classified as synonym of "star" by some ontology (WordNet in this case): the reason is that the word "lead" has also different semantics in different ontologies. Something similar happens with the synonym named "principal": its *semantic* distance to the sense of *star* $s2^{class}_{star}$is too high to be integrated with it. The system finally discards senses $syn1^{class}_{lead}$ and $syn3^{class}_{principal}$

because they do not correspond to the searched semantics ($s2^{class}_{lead}$). However one matching of *lead* ($syn2^{class}_{star}$) represents a semantics very similar to $s2^{class}_{star}$ and therefore both senses are integrated into only one ($s2'^{class}_{star}$); as suggested by [Ala06], we focus on integrating only the interesting parts (senses) rather than whole ontologies. The values of $s, grph$ and $descr$ of the integrated sense are the union of the equivalent parameters in the original senses; however the *pop* value is the add of the *pop* values of the original senses and *syndgr* is the product of the *syndgr* values of the original senses. As result of this semantic enrichment, now the system *knows* that a "star" is also a "stage player", according to certain possible sense ($s2$). Notice that when consulting different ontologies created by different people, it is very unusual that terms defined as synonyms are expressed exactly in the same way (same ontological graph).

4 Alignment and merging of redundant keyword senses

We explain in this section the sense alignment process that can be seen in Figure 2, on the right. Notice that this step is used in two situations by our system: 1) to check which synonym senses from other ontologies represent the same semantics as their candidate keyword senses, and 2) to avoid redundancy in the list of possible senses of each user keyword. However both tasks share a common goal: to find when two given senses represent very similar semantics; in that case they will be considered synonyms and both senses will be integrated[5]. Later, in this section, we detail the synonymy measure that compares the ontology context of two senses and how the merging (integration) of two senses is performed if they are synonyms.

In the following we explain graphically why our proposal for sense alignment is not just a comparison between two senses but an iterative process. Let us suppose that the system has extracted a *new sense*, which must be compared to the previously obtained candidate sense list (<*sense1*, *sense2*>). In Figure 5, the *new sense* is compared to *sense1* (step a) but their synonymy degree is below a certain threshold[6] (they semantically intersect only in terms within the area C). Then, the *new sense* is compared to *sense2* (step b); in this case, as their common knowledge is big enough (area D) they are integrated and the result is *new sense'*. Other approaches finish here their sense alignment step and the new list of different senses would be <*sense1*, *new sense'*>. However, although *sense1* and *sense2* were different enough (they do not represent the same semantics), we can see in step c that it is possible for the new integrated sense to be integrated with *sense1*: their common knowledge (areas B and C) is

[5] The integration process we propose, which is not the main goal of this paper, can be found in [EGTM06].

[6] Graphically, we consider that two senses should integrate if their intersection is above 25% of their areas. In our prototype, the threshold used is 0.65 (65%).

above the threshold, therefore they should be integrated, and then the resulting list of possible senses is just <*new sense"*>. In other words, each new integrated sense must be considered as candidate to integrate with the rest. For the same reason, new senses that do not integrate are stored because they could become the missing semantic gap between two senses that do not integrate. Although this method is costly (we limit its execution time), it performs a much better ontology alignment among senses.

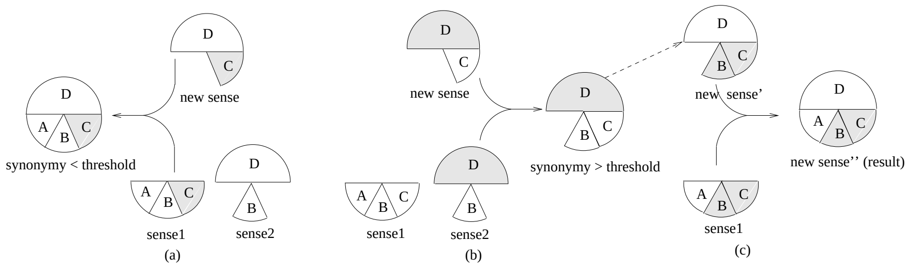

Figure 5: Iterative sense alignment: new integrated senses are considered for integration

The enrichment of senses could be extended by considering other semantic relationships. For example, a query for *hotel facility* could retrieve senses including classes, such as *rooming house restaurant*, *hotel online reservations*, or *casino hotel*, that are related by a kinship relationship.

In the following we detail how the system computes the *synonymy probability* between two candidate senses in order to decide if they must be integrated (as a single sense) or not.

4.1 Synonymy probability between two keyword senses

The aim of estimating the probability of synonymy is to conclude whether the semantics of two ontology terms represent the same sense or not. Thus the system avoid redundancy among the possible senses of a keyword. Our system does not compare senses coming from the same ontology; we suppose that different occurrences of a term in the same knowledge source always describe different senses if that ontology is free of redundancy, as expected in a well-formed ontology. At present, several solutions to determine the matching among ontological terms of different ontologies have been proposed, see [PE05] for recent surveys. Our approach computes coefficients of synonymy degree in the [0-1] range, however other approaches as semantic matching [GSY06] can be used as well.

The synonymy measure used relies on both *linguistic and structural* characteristics of ontologies. As most mapping algorithms we follow the following steps: 1) an initial computation using linguistic similarity, which considers labels as strings, 2) a recursive computation using structural similarity, which exploits the semantics of terms (ontological context) until a certain depth, and 3) the above values are combined to obtain the resultant synonymy measure.

In a variety of approaches, the similarity measure is only calculated among ontological terms that are classes. However, we propose a way to obtain the synonym probability according to the type of senses that we compare. As the ontology term that determines a sense can be a class, a property or an individual, a question can arise: is it possible, for example, for a class to be equivalent to a property? for instance, the property "teacher" of class "student" in some ontology $o1$ could seem similar to the class "teacher" in ontology $o2$, as both terms represent teachers. However, according to [MI01], the answer is no, due to the different semantic nature of classes and properties. What it is possible is that the property "$o1$#teacher" of class "$o1$#student" is equivalent to the property "$o2$#name" of class "$o2$#teacher". Thus, synonymy probability is only meaningful between two classes, two properties, or two individuals.

In the following we explain how the synonymy probability between a previous sense s_{t1} and a new one s_{t2} is computed, when $t1$ and $t2$ are classes, properties or individuals.

4.1.1 Synonymy Probability between Classes

We propose a synonymy probability function sp_{class} which is a linear combination of: 1) $l(t1, t2)$, the string similarity between class names $t1$ and $t2$, 2) $vsm(descr_{s_{t1}}, descr_{s_{t2}})$, the similarity between class descriptions of $t1$ and $t2$, 3) sp_{ctx}, the similarity between the hypernym and hyponym hierarchies (let us call them *ontological contexts*) of $t1$ and $t2$, with a certain depth d, and 4) $sp_{propSet}$, the similarity between the property sets of the classes $t1$ and $t2$:

$$\begin{aligned} sp_{class}(s_{t1}, s_{t2}, d) = {} & w_{name} \cdot l(t1, t2) + w_{descr} \cdot vsm(s_{t1descr}, s_{t2descr}) + \\ & w_{ctx} \cdot sp_{ctx}(s_{t1grph}, s_{t2grph}, d) + \\ & w_{propSet} \cdot sp_{propSet}(s_{t1}, s_{t2}, d) \end{aligned}$$

$$w_{name}, w_{descr}, w_{ctx}, w_{propSet} \geq 0 \wedge w_{name} + w_{descr} + w_{ctx} + w_{propSet} = 1$$

In our implementation, we use the Jaro-Winkler metric [WT91] to calculate the string similarity between two class names (l function). This process also uses the synonyms extracted for each term in the comparison[7]. In order to calculate vsm($s_{t1descr}$,$s_{t2descr}$) we use the Vector Space Model (VSM) algorithm [RW86].

[7] Due to this reason, when the system computes l("*star*", "*lead*") the result is 1.

VSM is a way of representing documents through the words that they contain. Each document is represented as a vector, where components are term weights assigned to that document. The weights of the terms are computed using their tf (term frequency) and *idf* (inverse document frequency) factors in each document. Our prototype builds a document for each description ($s_{t1descr}$ and $s_{t2descr}$) after filtering non-containing-bearing "stopwords". The structural similarity (sp_{ctx}) in our system depends on the factor d (*depth*) chosen:

$$sp_{ctx}(s_{t1}, s_{t2}, d) = \begin{cases} w_h \cdot vsm(h_{s_{t1}}, h_{s_{t2}}) + w_H \cdot vsm(H_{s_{t1}}, H_{s_{t2}}) & if\ d = 1 \\ \\ cd \cdot (w_h \cdot vsm(h_{s_{t1}}, h_{s_{t2}}) + w_H \cdot vsm(H_{s_{t1}}, H_{s_{t2}})) & \\ +(1 - cd) \cdot hH(s_{t1}, s_{t2}, d - 1) & otherwise \end{cases}$$

$$hH(s_{t1}, s_{t2}, d) = w_h \cdot tl(h_{s_{t1}}, h_{s_{t2}}, d) + w_H \cdot tl(H_{s_{t1}}, H_{s_{t2}}, d)$$

$$tl(l1, l2, d) = \frac{\sum_{i \in l1} \max_{j \in l2} sp_{class}(i,j,d) + \frac{\sum_{j \in l2} sp_{class}(i,j,d)}{|l2|}}{2 \cdot |l1|}$$

$$w_h, w_H \geq 0 \wedge w_h + w_H = 1$$

where, $h_{s_{t1}}, h_{s_{t2}}$ are *sampled sets* of hyponyms of the hierarchical graph in senses s_{t1} and s_{t2}, respectively; $H_{s_{t1}}$, $H_{s_{t2}}$ are sampled hypernyms, $|X|$ is the cardinality of the set X, and cd represent the *certainty degree* (see [RMBA05]) for obtaining the sample size for the four sets of terms: $h_{s_{t1}}, h_{s_{t2}}, H_{s_{t1}}$, and $H_{s_{t2}}$. The $vsm(h_{s_{t1}}, h_{s_{t2}})$ and $vsm(H_{s_{t1}}, H_{s_{t2}})$ compute the linguistic similarity between the contexts; while the function $hH(m, n, d)$ calculates synonymy probability of the contexts (hypernyms and hyponyms) of the ontological terms m and n, with depth d by means of the function tl (similarity between two sets of hypernyms/hyponyms).

The structural similarity sp_{ctx} is propagated through neighboring entities (hypernym/hyponym relationships). It follows a statistical approach to incrementally calculate the degree of similarity between two terms by analyzing a *sample* of the hypernym/hyponym relationships between terms, with a certain depth; sampling techniques are used to improve performance in case of ontologies are very populated. The $vsm(h_{s_{t1}}, h_{s_{t2}})$ and $vsm(H_{s_{t1}}, H_{s_{t2}})$ are performed similarly to $vsm(s_{t1descr}, s_{t2descr})$ but considering "bags of ontological terms" (extracted from hyponyms and hypernyms respectively) instead of descriptions. The function $tl(l1, l2, d)$ detects the contribution of each element of the first set ($l1$) with all the elements of the second set ($l2$). In this way we avoid obtaining high values of similarity when only one element in both sets matches.

The synonymy probability $sp_{propSet}$ between the property sets of the two class senses s_{t1}, and s_{t2} is computed in the next way:

$$sp_{propSet}(s_{t1}, s_{t2}, d) = \frac{\sum_{m \in propSet_{s_{t1}}}^{n \in propSet_{s_{t2}}} sp_{prop}(m,n,d)}{|propSet_{s_{t1}}||propSet_{s_{t2}}|}$$

where, $propSet_s$ denotes the property set of ontology terms belonging to sense s, and $sp_{prop}(m, n, d)$ estimates the synonymy probability of two ontology properties, with a certain depth d (explained in the following).

4.1.2 Synonymy Probability between Properties.

In order to calculate the synonymy probability between two properties s_{p1} and s_{p2}, i.e. $sp_{prop}(s_{p1}, s_{p2}, d)$, we consider the information obtained from the semantic context of properties[8] s_{p1} and s_{p2} and the semantic context of their domain and range classes (using the method shown previously for classes).

$$\begin{aligned} sp_{prop}(s_{p1}, s_{p2}, d) = {} & w_{nameP} \cdot l(p1, p2) + w_{ctxP} \cdot sp_{ctx}(s_{p1}, s_{p2}, d) \\ & + w_{domain} \cdot sp_{class}(domain(s_{p1}), domain(s_{p2}), d) \\ & + w_{range} \cdot sp_{class}(range(s_{p1}), range(s_{p2}), d) \end{aligned}$$

$$w_{nameP}, w_{ctxP}, w_{domain}, w_{range} \geq 0 \wedge w_{nameP} + w_{ctxP} + w_{domain} + w_{range} = 1$$

where w_{nameP}, w_{ctx}, w_{domain}, and w_{range} are the weights of the above synonymy measures, and $domain(s_{pi})$ and $range(s_{pi})$ return the domain class and range class of the property s_{pi}, respectively.

4.1.3 Synonymy Probability between Individuals.

To compute the synonymy probability $sp_{indv}(s_{i1}, s_{i2}, d)$ between two individuals of different ontologies, we consider the synonymy of the class they belong to, with a certain depth d.

$$sp_{indv}(s_{i1}, s_{i2}, d) = w_{nameI} \cdot l(i1, i2) + w_{classI} \cdot sp_{class}(class(s_{i1}), class(s_{i2}), d)$$

$$w_{nameI}, w_{classI} \geq 0 \wedge w_{nameI} + w_{classI} = 1$$

where w_{nameI}, and w_{classI} are the weights of the above synonymy measures, and $class(s_i)$ returns the concept from which s_i is an instance.

[8] Properties are organized in hierarchies in some ontologies.

Concerning our example, we provide the details of the synonymy probability functions described above. We consider the sense $s2^{class}_{star}$ and the synonym sense $syn2^{class}_{lead}$ (shown in Figure 4) in order to compute their synonymy probability in the semantic enrichment step. Remember that both senses are *classes*, therefore we use $sp_{class}(s_{t1}, s_{t2}, d)$ in order to decide if that two senses must be integrated (as a single sense) or not. In Figure 6, we show the senses $s2^{class}_{star}$ and $syn2^{class}_{lead}$ (of the keywords "star" and its synonym "lead" respectively) with a full description of their ontological context to depth=2 (in this case there are not hyponyms with depth=2). On the right, the term "lead" have two hypernyms ("actor" and "stage player") and four hyponyms ("co-star", "film star", "idol" and "television star"). Notice that although the terms "actress", "heavy", "ham" and "mime" (enclosed in the shadow ellipse) do not describe the term "lead", they are considered because these terms are used to compute the degree of similarity of their neighbouring entities.

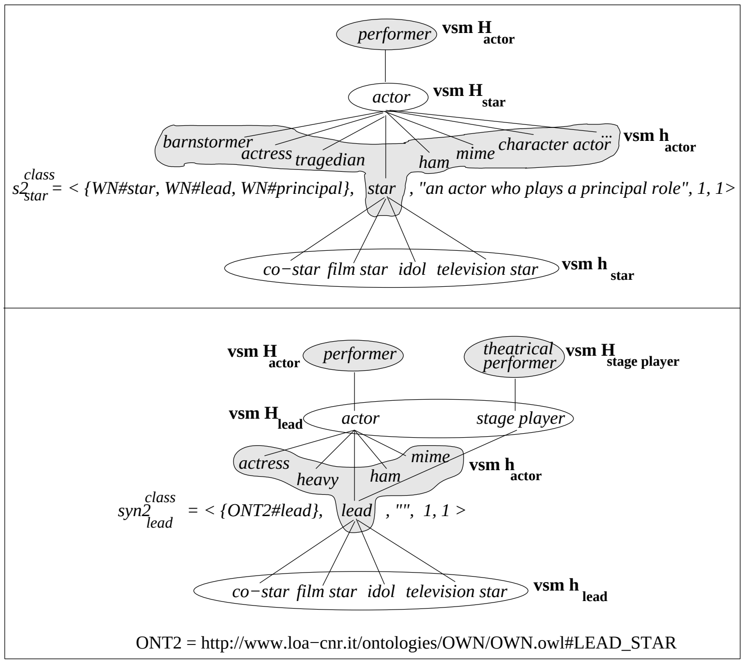

Figure 6: Full description of the context of $s2^{class}_{star}$ of "star" and $syn2^{class}_{lead}$ of "lead".

Firstly, our algorithm computes the string similarity between the terms "star" and "lead" (l("$star$", "$lead$")). This process uses in the comparison, the synonyms extracted of each term, that is *star, lead* and *principal* for "star" and only *lead* for "lead"; therefore the returned value is 1. In order to compare the description of the terms ($vsm(s2^{class}_{star}descr, syn2^{class}_{lead}descr)$) the description of star is represented by the set $\{actor, play, principal, role\}$ and the term "lead" does not have a description, so the obtained value is 0.

Secondly, in order to calculate the neighboring information of both terms, our algorithm computes the similarity between the values $vsm(h_{star}, h_{lead})$ and $vsm(H_{star}, H_{lead})$ with $depth = 1$. In our motivating example H_{star} is the set $\{actor\}$, while that H_{lead} is the set $\{actor, stage\ player\}$. Then, it computes (sp_{ctx}) for depth=2 and detects the contribution of each element of the first set with all the elements of the second set; the algorithm compares the term "actor" of the sense $s2^{class}_{star}$ with the terms "actor" and "stage player" of the sense $syn2^{class}_{lead}$.

Finally, the linguisty matching calculates the synonymy probability between the properties of "star" and "lead". These senses do not have properties so the result of $sp_{propSet}$ is 0.

Notice that sp_{class} and sp_{prop} formulas are recursive and the same calculations can be required several times, so partial results are stored to compute each comparison only once to improve the performance. The system stores pairs of terms already compared. Each pair is represented by a tuple mt = <sourceTerm, targetTerm, typeTerm, synonymy probability>, where *sourceTerm* and *targetTerm* are the names of the terms compared, *typeTerm* describes the type of terms (classes, properties or individuals) and *synonymy probability* is the result of computing the synonymy probability between both terms. Two terms will be merged if the value of *synonymy probability* exceeds a certain threshold (in our prototype 0.65). We show a summary of the performed comparisons in Table 1 and the resultant integrated sense is shown in the bottom of Figure 4.

source Term	*target Term*	*type*	*synonymy*
star	lead	Class	0.93
actor	actor	Class	0.65
co-star	co-star	Class	0.95
film star	film star	Class	0.95
...	...	...	...

Table 1: List of mapping elements between the senses of $s2^{class}_{star}$ and $syn2^{class}_{lead}$.

In Figure 7 we show the results of aligning the candidate keyword senses of user keywords. Due to space limitations we only show a partial list of senses managed by the system. Remember that $s2^{class}_{star}$ of keyword "star" was semantically enriched with $syn2^{class}_{lead}$ and became $s2'^{class}_{star}$. Now the final sense alignment process integrates sense $s3^{class}_{life}$ of keyword "life" with $s10^{class}_{life}$ and $s5^{class}_{life}$ into the single sense $s3'^{class}_{life}$.

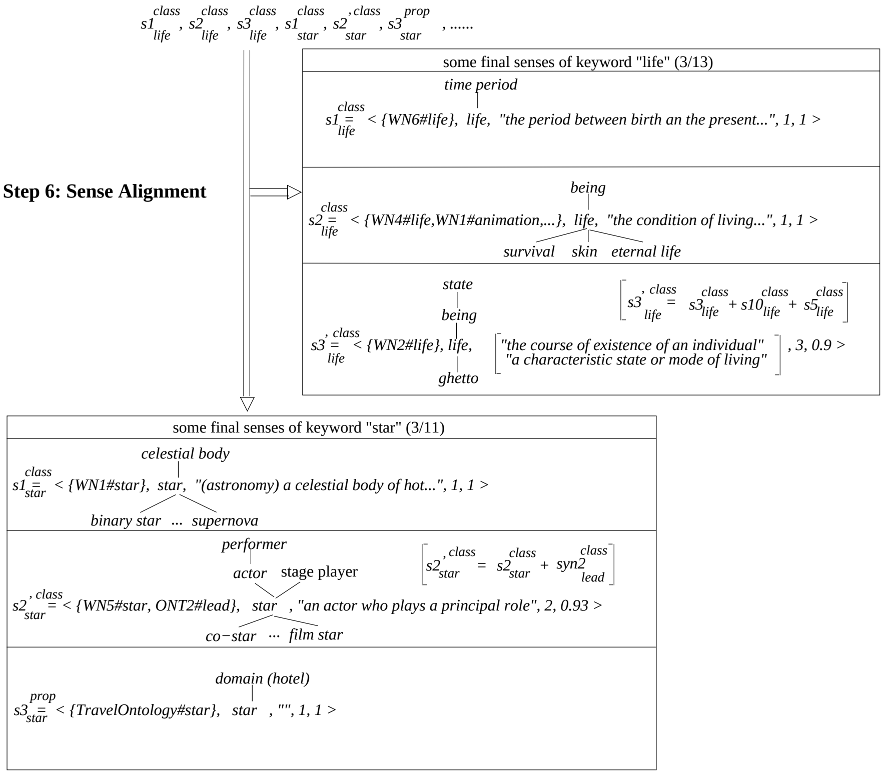

Figure 7: Sense alignment and merging of candidate keyword senses for "life" and "star".

5 Disambiguation of Keyword Senses

The previous steps provide a set of possible (non-redundant) senses for each user keyword, but what is the intended meaning of each keyword? In most cases the user intention is implicitly contained in the set of keywords that he or she entered. For example, if the keywords are "star astronomy planet", the intended meaning of "star" could be "celestial body"; but in "Hollywood film star" the meaning for "star" is probably "famous actor". To identify the right meaning in these contexts is an easy task to a human observer, but it is difficult to be performed automatically. Other examples are still ambiguous even for a human, as our motivating example “life of stars” (where the user wants to retrieve information about the life of famous people).

We understand *disambiguation* as the process of picking up the most suitable sense of a polysemous word according to a given context. In our case the initial context to disambiguate is the set of user keywords, but it can be enriched with the ontological context of their associated senses, as we have extracted them on the previous steps.

In short, the method that we use to perform this disambiguation step processes each user keyword iteratively, taking into account the possible senses of the other keywords to disambiguate it. Its output is a single sense for each keyword.

Such an algorithm compares each keyword sense with the selected ones of previously disambiguated keywords, and with all the possible senses of still not processed keywords. This comparison is performed by computing a relatedness measure based on Google frequencies[9] to measure each possible semantic relation between each sense and the senses of the other keywords.

We define the *Google-based semantic relatedness* between two search terms x and y as:

$$relGoogle(x, y) = e^{-2NGD(x,y)} \tag{1}$$

where NGD(x,y) is the *Normalized Google Distance* [RLC07] between x and y. Initially the function (1) can be applied to *any* pair of search terms indexed by Google. Nevertheless in our disambiguation algorithm we need to discover how much semantically related a pair of senses are, so we must convert keyword senses into convenient search terms (strings) that let us to perform queries on Google. These strings are built using part of the available semantics that characterize the considered keyword senses s_{nk_i} and s_{nk_j}, extracted from the ontology or ontologies where the keyword senses were found.

We consider two levels of characterization: *Level 0* the term itself and its synonyms, and *Level 1* its *father* terms and their synonyms. Consequently two different search terms can be built to characterize a single sense s_{nk_i}: $s_{nk_i}^{lev0}$ and $s_{nk_i}^{lev1}$. We calculate the semantic relatedness between senses computing each level separately and then combining them with certain weights.

$$relGoogle(s_{nk_i}, s_{nk_j}) = w_0 \cdot relGoogle(s_{nk_i}^{lev0}, s_{nk_j}^{lev0}) + w_1 \cdot relGoogle(s_{nk_i}^{lev1}, s_{nk_j}^{lev1})$$

The construction of these two levels of characterization can follow different strategies. The one we consider here is to define $s_{nk_i}^{lev0}$ as a disjunction of synonyms (if they are available):

$$synonym_1 +\ OR\ + synonym_2 +\ OR\ + ... + synonym_n$$

[9] any other search engine can be used

The construction of $s_{nk_i}^{lev1}$ depends on the type of term considered (class, property or individual):

1. $s_{nk_i} \in S_{nk_i}^{class}$: We characterize $s_{nk_i}^{lev1}$ by their direct hypernyms, as the conjunction of the disjunction of synonyms of each direct hypernym in the correspondent ontology:

$$(synonym_{11} + \ OR \ + synonym_{12} + \ OR \ + ...) + \ AND$$
$$+(synonym_{21} + \ OR \ + synonym_{22} + \ OR \ + ...) + \ AND$$
$$+... + (synonym_{N1} + \ OR \ + synonym_{N2} + \ OR \ + ...)$$

   where $synonym_{ij}$ is the $j - th$ synonym of the hypernym $i - th$.

2. $s_{nk_i} \in S_{nk_i}^{prop}$: Although properties have their own $is - a$ hierarchy in ontologies, it is not usual in most of them. Therefore we adopt the *domain* of the property (that is a class) to characterize the *level 1* instead of its fathers in the hierarchy of properties. We built its search term as a disjunction of the synonyms of the domain.

3. $s_{nk_i} \in S_{nk_i}^{indv}$: If the value is an instance of a class, we build the search term for $s_{nk_i}^{lev1}$ as a conjunction of the synonyms of the associated concept. If it is an instance of a property, we use the conjunction of the synonyms of its domain.

For example we can compute the semantic relatedness between $s3'^{class}_{life}$ and $s2'^{class}_{star}$ in this way:

$$relGoogle(s3'^{class}_{life}, s2'^{class}_{star}) = w_0 \cdot relGoogle(\text{“}life\text{”}, \text{“}lead\ OR\ star\text{”}) + w_1 \cdot$$
$$relGoogle(\text{“}being\text{”}, \text{“}actor\ AND\ \text{“}stage\ player\text{””})$$

resulting a value of 0.476 (with $w_0 = w_1 = 0.5$).

The computed relatedness among senses described above is used to calculate a relevance degree for each keyword sense: we choose the maximum ones computed between the sense and the set of senses of each other keyword (as they do in [TP05]). We add them and normalize according to the number of keywords, giving a final relevance degree for each sense. Then the list of initial senses are rearranged according to these values. A filtering process is also applied to omit the less relevant ones, according to a configurable threshold.

The method can operate in two modes: 1) Automatic: The system selects the one with the highest relevance degree, so avoiding user intervention but trusting the decision taken by the computer. 2) Semiautomatic: The system proposes to the user the rearranged and filtered list of senses obtained automatically, and he or she selects the most suitable one for each keyword.

Our disambiguation method does not need large semantically tagged corpora needed by supervised learning approaches, and also overcomes other limitations of many traditional disambiguation techniques [Res95, TP05, LC98]: It is independent on a single lexical resource as source of keyword senses (many traditional methods only deal with WordNet synsets), and it is independent on a particular ontology or corpus to compute the semantic relatedness measure used by the algorithm (the above mentioned relatedness relies on Google, where almost any possible keyword and implicit interpretation could have been indexed, instead of depending on a single ontology where its description could be limited or even non-existent).

As result of applying this disambiguation method to our motivating example our system selects automatically these senses as the most probable ones: for the keyword "life" the selected sense $s3'^{class}_{life}$ is "the course of existence of an individual or a characteristic state of mode of living", and the sense $s2'^{class}_{star}$ ("an actor who plays a principal role") is proposed for the keyword "star" (see Figure 8.).

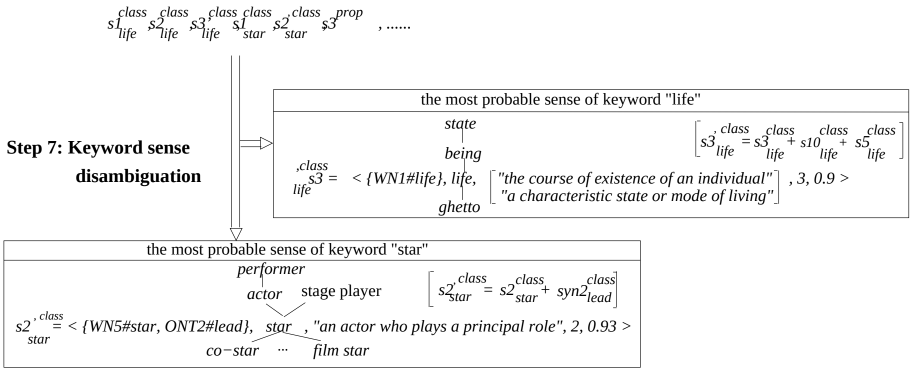

Figure 8: Disambiguation of possible keyword senses for "life" and "star".

As other illustrative example we will mention that the disambiguation of the keyword set "Astronomy star planet" (our prototype provides 1, 10 and 4 possible senses respectively for those keywords) selects a sense for "star" as subclass of "celestial body", giving it a higher score than the other senses (as the one with description "an actor who plays a principal role"), which seems to be in accordance with the intuition of a human observer.

For a more detailed description of the disambiguation process see [GTEM06].

6 Construction and ranking of queries

In this section, we detail our approach to translate the keyword senses selected in the previous step into queries expressed in a knowledge representation language. Although the system knows the correct sense of each keyword, the user could have used such keywords for different information requests (different queries). For example, although the keyword "vehicle" means "a conveyance that transports people or objects" and the keyword "house" means "a dwelling that serves as living quarters for one or more families" for a specific user, the user could be asking for motor homes or houses with a parking lot. In other words, different queries are possible even when each individual keyword has a fixed sense. The possible queries depend on the expressivity of the query language used.

There exists a module called *Parser Builder* which, taking as input a free context grammar describing the syntax of a knowledge representation language, outputs a new grammar. The new grammar generates all possible queries for each permutation of terms, taking into account the expressivity of the knowledge representation language chosen. The resulting grammar has two kinds of rules: 1) the type of constraint of each operator (e.g., *And* is a *Class* constraint) and 2) the ordered list of operands of each operator (e.g., the parameters of *Fills* are a property and a class). Notice that this process filters out the syntax sugar of the input language. In Figure 9 we can see the grammar obtained for a subset of the knowledge representation language BACK [Pel91]. Thus, the parser corresponding with the output grammar will be used to build the possible formal queries for the terms obtained from the user keywords. This parser is built just once for each knowledge representation language.

Subset of BACK language grammar
Class → And(Class, Class)
Class → Fills(Property, Individual)
Class → All(Property, Class)

Grammar obtained from characteristics of BACK
Class → And | All | Fills | className
Property → propertyName
Individual → individualName
And → Class Class
All → Property Class
Fills → Property Class

Figure 9: Subset of BACK language and its equivalent grammar obtained.

The main steps followed to construct queries that combine the keyword senses selected previously are the following (see Figure 10):

– *Permutation of terms.* In this step, the system obtains all the possible permutations of the input terms. For example, if the input is a property p_1, a class c_1 and another class c_2, the permutations obtained are: p_1 c_1 c_2, p_1 c_2

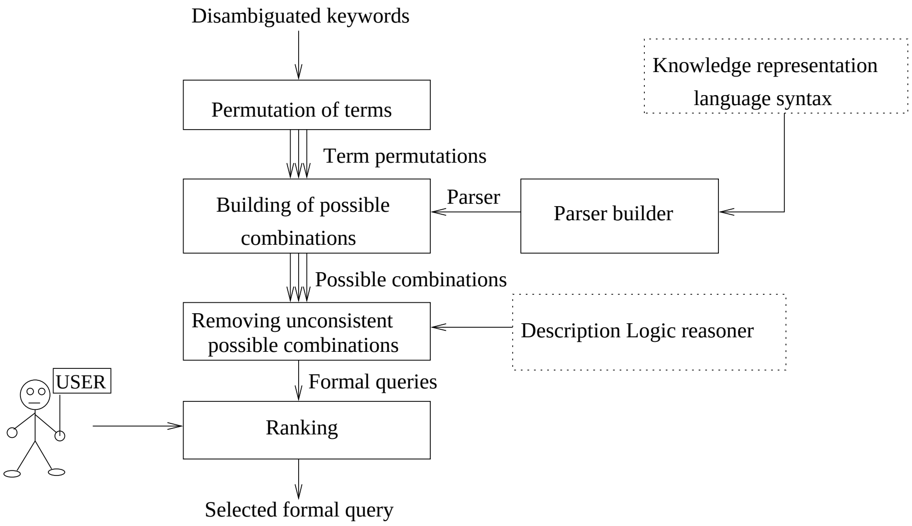

Figure 10: Construction and ranking of queries.

c_1, c_1 p_1 c_2, c_2 p_1 c_1, c_1 c_2 p_1 and c_2 c_1 p_1. In this way, we can build queries where the position of the keywords is different from the original; that is, the order of the keywords is relevant but it does not limit the possible queries.

– *Building of possible combinations.* Using the parser built in the process of configuration of the module, all possible queries are generated for each permutation of terms.

– *Removing inconsistent possible combinations.* Some queries obtained in the previous step could be inconsistent combinations, e.g., *and*(c_1, c_2) is inconsistent if it happens that *disjoint*(c_1, c_2); i.e., although *and*(c_1, c_2) is correct syntactically, it is not correct *semantically* (no object belongs to *and*(c_1, c_2)). The system automatically removes such queries by using a Description Logics reasoner [BCM+03] (compliant with the chosen language) such as RacerPro, FACT, Pellet or BACK [BCM+03, Pel91].

– *Ranking of queries.* Each query in the previous step defines possible classes of objects which the user could be interested in. Probably, the user is interested only in one of the output queries. So, it is needed to compute a *relevance probability* (the probability that a query expresses the intention of the user) for each possible query. The system proposes the most relevant query to the user; then, she/he can change this default selection if it is not right. In order to compute the *relevance probability* the system takes into account the following factors: 1) the order of keyword senses, the more similar to the original order of terms the query is the more relevant it is, 2) the kind of selected senses considered (class, property or instance), and 3) the type of operators used in the construction of queries.

Compared with the well-known system SeamSearch [LUM06], this approach has the following advantages: 1) it does not depend on a specified knowledge language, 2) it is not required that one of the keywords entered by the user is a class, and 3) combinations of operands and operators are not predefined with templates so our prototype considers all the possible combinations.

7 Experimental Evaluation

We have developed a prototype in order to put into practice our ideas. Even though we have to test our system in a more detailed and systematic way to evaluate the benefits of our approach, we have already performed some initial experiments whose results are very promising. Our goal is to evaluate if the system discovers the semantics assigned by the user to the input keywords.

Our initial tests have been oriented to evaluate the quality of the semantics that we give in a particular scenario: when the user query is highly ambiguous (even for a human observer). We had the intuition that, when there exist many possible interpretations, traditional search engines fail in providing some of them among their first results, due to the dominance of some terms highly extended on the Web.

From a larger set of deliberately constructed ambiguous queries, we finally selected ten (the ones with higher ambiguity according to the point of view of several users). After that, we requested 20 users to assign their own meanings to each keyword set. Each keyword set was the input of our system and each user had to identify the first sense provided by our system that described well his assigned meaning. Similarly, users had to identify the first web document returned by Google with the intended meaning after entering each keyword set. We want to remark that we are not comparing directly the output of our system with the output of Google (we return semantic queries while they return links to web pages). We want to compare, the *semantics* contained in the queries that our system returns with the (implicit) *semantics* underlying the pages that Google retrieves.

We have devised a normalized measure to evaluate how closed the user interpretations and an "ideal" distribution are. For example if a user thinks of three possible interpretations for "java" (as "an island", "coffee", or "a programming language"), the ideal distribution for that user is when the system provides these senses in the first three positions of the returned answer without regarding the order. In this case the measure value must be 1. The worst situation (no senses according to the user interpretations were returned) gives a value of 0. Otherwise the measure ranges between 0 and 1. For each keyword set we compute the measure as follows:

1) For each user j we compute the "distance" between the ideal distribution and the real distribution of senses: $D_j = \frac{1}{n} \cdot \sum(pos_i) - \frac{n}{2}$; where pos_i is the position of the i^{th} interpretation returned by the system, and n is the number of user interpretations[10].

2) We average the previous value among all the users, and normalize its value between 0 and 1 proportionally to the proximity to the ideal case. $p = \frac{-1}{30 \cdot m} \cdot \sum(D_j) + 1$; where m is the number of users interviewed and D_j is the value calculated in the previous step.

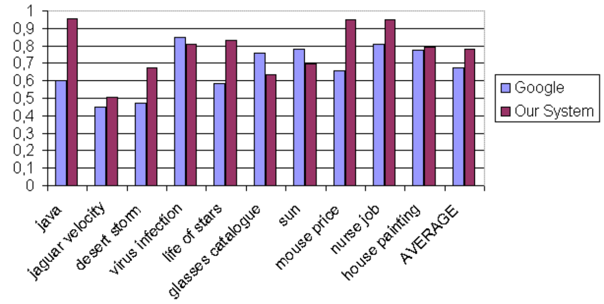

Figure 11: Suitable senses provided by each system

The results are shown in Figure 11. As we can see, our system provides more suitable interpretations to some keyword sets than Google, according to users' opinion. For example, many users interpreted "java" not only as "programming language" but also as "island" or "coffee". Our system provides these senses which are very difficult to find in links returned by Google, as the score shows. Nevertheless, in three of the ten studied cases our system behaved worse than Google due to various reasons: for "glasses catalogue" our system did not find in the ontology pool the senses that "glasses" have in singular ("glass"); for "sun", no accessed ontologies described it as "newspaper" (a popular interpretation); and for "virus infection" some users were not satisfied with the descriptions proposed by our system for "infection" (they were extracted from the ontology pool). Anyway, those detected anomalies can be fixed by improving our morphological analysis (to consider singular and plural forms, for example) and consulting new

[10] If the user interpretation is not found, a position of 30 is applied: we assume that the wanted senses should appear in the first 30 results as the user's patience is finite when using our system or Google.

ontologies, which is very easy to do in our system. The interesting result is that our experiments indicate that *our system behaves better than web search engines when the semantics assigned to keywords by the user is not the most popular on the Web*; that is, our system can provide more interpretations for ambiguous keyword sets than a traditional search engine. Even we do not consider this result conclusive, it shows us a promising result and motivates us to continue in our line.

Regarding the performance, although thousands of ontologies were considered, our system spent six seconds in average[11] to return the possible senses of each keyword set.

8 Related Work

Semantic knowledge extracted from ontologies is used for different purposes: enhanced design of class diagram for the business domain [RS06], development of sophisticated question answering [LMU06], or query enrichment [TGS06]. Keyword senses obtained with our proposal are closely related to the last two applications cited. In [RS06] they use an ontology search engine to retrieve ontology files which contain candidate classes for the purchase order domain; besides their system deals with synonyms to refine the search. Unfortunately they assume that domain experts define a dictionary or ontology that specifies synonyms. On the contrary our system takes advantage of the shared ontologies available on the Web and semantically enriches the keyword senses with senses extracted from their synonyms. In [LMU06] they propose a system which exploits the availability of distributed, ontology-based semantic markup on the Web to answer questions posed in natural language. However they do not precise the mechanism used to index and locate an ontology. Also, unlike this work, we use a measure of synonymy among senses that does not depend on a single lexical resource as WordNet. In [TGS06] they propose a method that uses ontologies to improve the retrieval quality by query enrichment. However, the architecture proposed in this work only accesses their own (single) ontology. Our work has inherent advantages as queries third-party ontology pools; so we advocate a shared knowledge approach.

Concerning our subtask of integrating different ontology terms, from the works that also exploit linguistic and structural information, [QHC06] is the most similar to ours. They propose the idea of virtual documents, which contain the local descriptions and the neighboring information to reflect the intended meaning. Besides they use traditional vector space techniques to compute document similarity. In [Hes06] they propose an iterative algorithm for ontology mapping which is based on established string distance metrics and on a structural

[11] The total time spent depends on the number of candidate senses found in ontologies.

similarity measure based on a vector representation of the relations between entities. In [LDKG04] a linear combination of name similarity, label similarity and description similarity is used in order to discover the mapping between classes. However, these works do not use statistical sampling to optimize the hierarchy comparison in depth, they do not consider the kind of term (class, property, or individual) when comparing and integrating, and our search of synonyms to calculate the name similarity is extended to an ontology pool instead of just a single resource as WordNet.

Regarding the disambiguation step, we have already mentioned some advantages with respect to other traditional approaches [Res95, TP05, LC98]. The most remarkable one is its independence of using a particular lexical resource (both to compute the semantic relatedness measure and as source of keyword senses). Other approaches, as the Structural Semantic Interconnection technique described in [NV05], differ from us due to a similar reason: even its good behavior, they rely on the graph representation that they use in several pre-selected lexical resources. Partial manual pre-processing is also needed. In contrast we want to use a method where the semantic context can be discovered in no pre-established knowledge sources.

9 Conclusions

In this paper we have presented a semantics-guided approach to discover the possible senses of a set of user keywords; the output is a query expressed in a formal language, which represents the user intended semantics. The main features of our proposal are the following:

1. It uses an iterative approach to retrieve from different knowledge repositories the possible senses of each user keyword, in a parallel manner. A sense is represented basically as the (multi)ontological context of a term, and the system is able to deal with senses corresponding to different kind of ontology terms (classes, properties, and individuals).

2. It considers not only the senses corresponding to ontology terms that syntactically match the user keywords, but also the senses of ontology terms matching the synonyms of the user keywords, recursively, in order to semantically enrich the keyword senses retrieved within a certain synonymy threshold.

3. It measures the synonym degree between two senses by considering their linguistic and structural similarity. Statistical techniques like sampling and parallel processing are used to improve the performance of this process. Senses whose synonym degree is above a certain threshold are merged automatically.

4. It applies a disambiguation method to select (automatically or with user intervention) the most suitable sense for each keyword. This method does not rely on a particular ontology or lexical resource and uses a semantic relatedness measure based on Google frequencies.

5. It combines the selected keyword senses into a query (in a knowledge representation language) which expresses without ambiguity the user's information need. This method uses parsing and techniques based on Description Logics and it is valid for most knowledge representation languages.

Our initial experiments show that our system is able to discover keyword senses not easily found by traditional web-based search engines. We believe that this technique to find out the semantics of user keywords can be applied to many fields, such as the retrieval of relevant data underlying ontologies on the Semantic Web, and to improve traditional web search engines.

As future work, we plan to process the semantic query obtained in the process described in this paper to retrieve relevant data from the Semantic Web.

Acknowledgements

This work is supported by the CICYT project TIN2004-07999-C02-02. Mauricio Espinoza's work is supported by a Santander Central Hispano-University of Zaragoza grant.

References

[Ala06] Alani, H.: "Position paper: Ontology construction from online ontologies"; Proc. 15th International World Wide Web Conference (WWW2006), ACM, Edinburgh, UK, (May 2006), 491-495.

[BCM+03] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. Patel-Schneider, P. F.(eds.): "The Description Logic Handbook: Theory, Implementation, and Applications"; Cambridge University Press, 2003.

[CFV07] Castells, P., Fernandez, M., Vallet, D.: "An adaptation of the vector-space model for ontology-based information retrieval"; IEEE Transactions on Knowledge and Data Engineering, 19, 2 (2007), 261-272.

[DS04] Dean, M., Schreiber, G.: "OWL web ontology language reference"; W3C recommendation, W3C, February 2004.

[EGTM06] Espinoza, M., Gracia, J., Trillo ,R., Mena, E.: "Discovering the semantics of keywords: An ontology-based approach"; Proc. The 2006 International Conference on Semantic Web and Web Services (SWWS'06), CSREA Press, Las Vegas, Nevada, USA (Jun 2006), 193-201.

[FB03] Freitas, F.,Bittencourt, G.: "An ontology-based architecture for cooperative information agents"; Proc. 18th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, Acapulco, Mexico (Aug 2003), 37-42.

[FDP+05] Finin, T., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A., Peng, Y.: "Swoogle: Searching for knowledge on the Semantic Web"; Proc. 20th National Conference on Artificial Intelligence and 17th Innovative Applications of Artificial Intelligence Conference, AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA (Jul 2005), 1682-1683.

[GMV99] Guarino, N., Masolo, C., Vetere, G.: "Ontoseek: Content-based access to the web"; IEEE Intelligent Systems, 14, 3 (1999), 70-80.

[Gru93] Gruber, T. R.: "Towards principles for the design of ontologies used for knowledge sharing"; In N. Guarino and R. Poli, editors, Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, 1993.

[GSY06] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: "Discovering missing background knowledge in ontology matching"; Tech. Rep. DIT-06-005, Informatica e Telecomunicazioni, University of Trento (Feb 2006).

[GTEM06] Gracia, J., Trillo, R., Espinoza, M., Mena, E.: "Querying the web: A multiontology disambiguation method"; Proc. 6th International Conference on Web Engineering (ICWE'06), ACM, Palo Alto, California, USA (Jul 2006), 241-248.

[Hes06] Hess, A.: "An iterative algorithm for ontology mapping capable of using training data"; Proc. 3rd European Semantic Web Conference (ESWC'06), ACM, Budva Montenegro (Jun 2006), 19-33.

[LC98] Leacock, C., Chodorow, M.: "Combining local context and wordnet similarity for word sense identification"; WordNet: An Electronic Lexical Database, MA:MIT Press, Cambridge (1998), 265-283.

[LDKG04] Le, B. T., Dieng-Kuntz, R., Gandon, F.: "On ontology matching problems - for building a corporate semantic web in a multi-communities organization"; Proc. 6th International Conference on Enterprise Information Systems, Porto, Portugal (Apr 2004), 236-243.

[LMU06] Lopez, V., Motta, E., Uren, V.: "Poweraqua: Fishing the semantic web"; Proc. 3rd European Sematic Web Conference, Lect. Notes in Comp. Sci., Springer, Budva, Montenegro (June 2006), 393-410.

[LUM06] Lei, Y., Uren, V. S., Motta, E.: "Semsearch: A search engine for the semantic web"; Proc. 5th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes in Comp. Sci., Springer, Podebrady, Czech Republic (Oct 2006), 238-245.

[MI01] Mena, E., Illarramendi, A.: "Ontology-Based Query Processing for Global Information Systems"; Kluwer Academic Publishers, ISBN 0-7923-7375-8, June 2001.

[Mil95] Miller, G.: "WordNet: A Lexical Database for English"; Communications of the ACM, 38, 11 (Nov 1995), 39-41.

[NV05] Navigli, R., Velardi, P.: "Structural semantic interconnections: A knowledge-based approach to word sense disambiguation";IEEE Trans. Pattern Anal. Mach. Intell., 27, 7 (2005), 1075-1086.

[PE05] Shvaiko, P., Euzenat, J.: "A survey of schema-based matching approaches"; Journal on Data Semantics, 4 (2005), 146-171.

[Pel91] Peltason, C.: "The BACK system – an overview"; ACM SIGART Bulletin, 2, 3 (Jun 1991), 114-119.

[PTBW05] Paz-Trillo, C., Braga, P. P., Wassermann, R.: "An information retrieval application using ontologies"; Journal of the Brazilian Computer Society, 11, 2 (2005), 17-31.

[QHC06] Qu, Y., Hu, W., Cheng, G.: "Constructing virtual documents for ontology matching"; Proc. 15th International World Wide Web Conference, ACM, Edinburgh, Scotland (May 2006), 23-31.

[Res95] Resnik, P.: "Disambiguating noun groupings with respect to Wordnet senses"; Proc. 3rd Workshop on Very Large Corpora, Association for Computational Linguistics, Somerset, New Jersey (1995), 54-68.

[RLC07] Vitnyi, P. M. B., Cilibrasi, R. L.: "The google similarity distance"; IEEE Transactions on Knowledge and Data Engineering, 19, 3 (Mar 2007) 370-383.

[RMBA05] Royo, J.A., Mena, E., Bernad, J., Illarramendi, A.: "Searching the web: From keywords to semantic queries"; Proc. 3rd International Conference on Information Technology and Applications (ICITA'05), IEEE Computer Society, Sydney, Australia (Jul 2005), 244-249.

[RS06] Rungworawut, W., Senivongse, T.: "Using ontology search in the design of class diagram from business process model"; Proc. International Conference on Computer Science (ICCS 2006), Vienna, Austria (Mar 2006), 165-170.

[RW86] Raghavan, V., Wong, S.: "A critical analysis of vector space model for information retrieval"; Journal of the American Society for Information Science, 37 (1986) 279-287.

[TGS06] Tomassen, S. L., Gulla, J. A., Strasunskas, D.: "Document space adapted ontology: Application in query enrichment"; Proc. 11th International Conference on Applications of Natural Language to Information Systems, Springer, Klagenfurt, Austria (May 2006),46-57.

[TP05] Patwardhan, S., Pedersen, T., Banerjee, S.: "Maximizing semantic relatedness to perform word sense disambiguation"; Tech. Rep. UMSI 2005/25, Supercomputing Institute, University of Minnesota (2005).

[WT91] Winkler, W. E., Thibaudeau, Y.: "An application of the fellegi-sunter model of record linkage to the 1990 u.s. decennial census"; Tech. Rep. RR91/09, U.S. Bureau of the Census, Statistical Research Report Series (1991).