Lexical, Syntactic, and Referencial Disambiguation Using a Semantic Network Dictionary

Alexander F. Gelbukh

Natural Language Processing Laboratory,
Centro de Investigación en Computación,
Instituto Politécnico Nacional.
07738 México D.F.

gelbukh@pollux.cic.ipn.mx,
gelbukh@micron.msk.ru

Abstract

A procedure is proposed for determining the measure of "nearness" of two words in a semantic network dictionary. Applications of this procedure are discussed in detail for lexical, syntactical, and referential disambiguation in natural language analysis, as well as in machine translation. A simplified version of the same procedure is used for automatic translation of the semantic network itself into other languages. This simplifies creation and maintenance of semantic network dictionaries for different languages, and thus makes the described methods available for processing of texts in languages other than English.

Keywords

Natural language processing, text processing, syntactic analysis, disambiguation, semantic network.

1. Introduction[1]

This article describes an ongoing research being conducted in the Natural Language Laboratory, CIC, IPN, Mexico D.F.

1.1. The curse of ambiguity

The most unpleasant problem that nearly any algorithm dealing with the natural language faces is the curse of ambiguity. Be it just one word, or a phrase, or a text, there always are several possible interpretations of what it means or what structure it has. We

[1] This article is an improved and significantly enlarged version of our paper [Gelbukh, 1997].

consider ambiguity resolution at all the levels of the language the most important problem of natural language processing. In much larger number of cases than it seems at the first glance, to resolve the ambiguity complicated reasoning or deep knowledge is necessary, often of semantic, pragmatic, or extralinguistic nature.

There is a large number of researches on ambiguity resolution that employ marked up text corpora, dictionaries [Alpha, 1995], thesauri [Yarowsky, 1992], semantic networks [Sussna, 1993; Voorhees, 1993], or a combination of such lexical information sources [Yarowsky, 1995]. However the problem is far from being completely solved up to the date.

Ideally, ambiguity resolution might be just a side effect of some kind of "understanding," by which we would mean construction of some detailed model of the whole situation described in the text and embedding it in the world model based on the hand-coded knowledge, experience, or other texts read. The "true" linguistic knowledge, mostly lexical, ideally should be stored in vast dictionaries, such as combinatory dictionaries developed in frame of the Meaning ⇔ Text theory [Mel'cuk, 1974; Steel, 1990], or programmed in very sophisticated procedures, like in the Word Expert Parser model [Berleant and Daniel, 1995].

However compilation of such resources is extremely labor-consuming and is hardly affordable in the nearest decades. On the other hand, such "true understanding" is too demanding computationally to be considered now; what is more, it seems that such a way is too computationally demanding even for human brain.

A less computationally demanding way is to use some pre-constructed pieces of "typical" situations and first of all to check the ambiguous constructions against them, addressing to a deeper analysis only when a serious contradiction arises in the understanding process. Such pre-constructed pieces of information can be of different nature, such as syntagmatic, semantic, pragmatic, etc.

For example, syntagmatic patterns could be represented by frequently used or "meaningful" word combinations, such as *take a bus*, *take a pen*, as opposed to **take weather* [Bolshakov *et al.*, 1995a]. Such a simplified set of syntagmatic patterns can be used (and probably is used by a human) in syntactic analysis instead of expensive "true understanding."

Similarly, instead of a computationally demanding reasoning, a set of simplified "typical" semantic patterns can be used for disambiguation. Such semantic patters could describe some atomic pieces of typical situations involving the words of the text. One of the form of representation of such knowledge is a semantic network, a set of semantic relationships between words in their specific senses.

1.2. The use of a semantic network for ambiguity resolution

As we show in this article, a semantic network dictionary can be used to measure the degree of “semantic closeness” in a typical context between two given words. This measure of closeness can be used for resolving ambiguities of different types as well as for related tasks such as automatic translation of texts or even dictionaries.

For example, to resolve **syntactic** ambiguity, a variant of parsing can be chosen in which syntactically related words are more closely related semantically. To resolve **lexical** ambiguity between homonyms or word senses in a context, the lexical variant can be chosen that is most closely related to the global or local topic of the document, or to the nearest words in the context. Similarly, to resolve **referential** ambiguity, the closest candidate is chosen to the words in the local context. In **text translation**, if the homonyms are not separated in the bilingual dictionary used for translation, the procedure of lexical disambiguation can be applied in the target language at the stage of text generation. Finally, in **translation of dictionaries** including the semantic network itself, lexical disambiguation can be performed on the reverse translation of the results back to the source language.

2. Distance measurements in a semantic network

Here we discuss the basic notions of measuring the semantic distances between two given words. The algorithms are given in the last section of the article.

2.1. The structure of a semantic network dictionary

In our research we use the FACTOTUM® SemNet semantic network dictionary [FACTOTUM, 1997] developed by Dr. P.Cassidy of MICRA, Inc. It is an English dictionary, though below we describe how to use it for other languages, currently our target language being Spanish[2].

Logically FACTOTUM® SemNet dictionary is a set of so called relationships between pairs of concepts (in rare cases between sets, here we omit this for simplicity).

A **concept** is usually a word, e.g., *book*, or a word combination, e.g., *address book*, referring to a specific thing or idea. In most cases textual words have several meanings, in this case they are marked with different numbers, e.g., $bill_1$ (banknote), versus $bill_2$ (check), $bill_3$ (declaration), $bill_4$ (pike), or $bill_5$ (ax). Very often such word senses, or homonyms, have different translations to other languages: in Spanish $bill_1$ is *billete*, $bill_2$

[2] Other semantic networks have been described, most notably the WordNet [Miller, 1990], which has been widely investigated for natural language processing because of its availability; a Spanish version is to be available soon in frame of European WordNet project. The FACTOTUM® semantic network, however, has a larger number of types and a more flexible representation of semantic relationships, making it more suitable for natural language understanding applications.

is *cuenta*, *bill3* is *declaración*, *bill4* is *pico*, *bill5* is *hacha*. All such senses of any word, even closely related ones, have different numbers in the dictionary, are located in different positions, and have different sets of relationships to other words; if necessary, they can be connected with each other explicitly by some relationships. Thus one textual word can represent many different concepts.

Similarly, one concept can be represented by several words. In this case they are considered synonymous in these particular meanings, and are listed together to represent and disambiguate the concept, e.g., *bill1*, *note, banknote*. Thus generally by a concept we always mean a group of synonymous word senses. However for convenience we name the concepts with just one of the words of the group.

Relationships are used to connect a pair of, or rarely more, concepts. They are labeled with different type names, such as IS_A, USES, CAUSES, etc. A relationship can be also viewed as a simplest statement expressing a "typical" fact, e.g., *computer IS_A equipment*, *explosion* CAUSES *damage*. There are some attributes, or properties, of individual relationships, like MAYBE, USUALLY, RARELY, etc., e.g., *seeing MAYBE* USES *telescope*. For human convenience there are different ways to express the same fact in FACTOTUM ® SemNet dictionary, e.g., *telescope* IS_USED_FOR *seeing*, though all such expressions can be formally converted to a common internal representation. A fragment of such a network is shown on the Fig. 1.

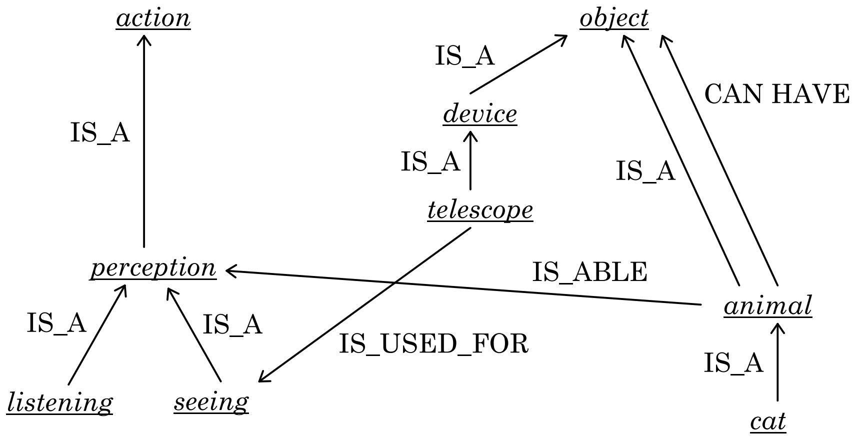

Fig. 1. Semantic network.

From this example, we can see that a telescope is a tool to see, an animal can have an object, etc.

In the human-editable form of the dictionary the most extensive set of relationships, namely most of the IS_A relationships, are represented implicitly by just placing the concepts in hierarchical order in the dictionary. This does not imply that a concept may not be a subtype of several concepts, for example, a *girl* IS_A *child* and IS_A *female*, in this case one of the relationships is indicated in the dictionary explicitly.

Clearly, there are many possible relationships that can be easily inferred by some general rules from other relationships; e.g., transitive relationships like IS_A and IS_PART_OF: if *a* IS_PART_OF *b* and *b* IS_PART_OF *c*, then *a* IS_PART_OF *c*. In such cases only some of the relationships, the most immediate ones, are explicitly included in the dictionary, to keep its size maintainable, e.g., *car* HAS_PART *motor* and *motor* HAS_PART *screw* implies *car* HAS_PART *screw*.

There are other rules of inference involving particular relationships or groups of relationships. E.g., the most obvious one is that if *a IS_A b* and *b R c*, then *a R c*, where R is any relationship. In some cases, such inheritance of characteristics from higher categories is defeasible, i.e., it may be blocked explicitly by a special notation in the definition of a concept, or it may be canceled where contradictory information is inherited from more than one higher node.

2.2. Paths in the semantic network

A semantic network is a graph. A **path** in such a graph is a chain of relationships r_1, ..., r_n such that r_i and r_{i+1} have exactly one common word, $i = 1, ..., n - 1$. If a word A is the beginning of the path and the word B is its end, we say that the path leads from A to B. There are several reasons to use paths for measure the semantic closeness of words in the network.

First, since some of the relationships are present in the network only implicitly and can be inferred by application of the inference rules, a problem arises of generating all the relationships, including the implicit ones, between, say, two given words. The problem can be formulated as enumerating all the network paths with some special condition that lead from one given word to another.

Second, in some cases important commonality between words may not be expressed in terms of any existing named type of relationship. E.g., on Fig. 1, it can be seen that "a *cat* CAN HAVE something that IS_USED_FOR *seeing*". There is no named type of relationship expressing this fact, so we have to represent the commonality between these two words by just a path of the two relationships.

Third, some rules of inference may have fuzzy character, being rather common-sense observations. At least this may imply that applying too many rules can make the result less reliable. We can express this lost of reliability, in some cases, by adding the MAYBE attribute to the resulting inferred relationship, though in general case we should use some kind of lengths, or weights, of relationships and paths, which are described below.

Finally, there may exist some participants of the situation described in the text that are not mentioned explicitly. E.g., in the phrase "*The seller asked the buyer for too high price.*" there may be no explicit relationship in the dictionary between the words *seller* and *buyer*, though the commonality between them can be established through the implied actant, *goods*: *buyer* CONSUMES *goods* HAS_SOURCE *seller*, a path of two relationships.

2.3. Lengths of the relationships and paths

In general we need to assign some weight, or "length," to each relationship and calculate the "length" of a path based not just on the number of links in it, but on their individual lengths. This value gives the quantitative estimation of how closely related are the two given words, while the path itself with the labeled links gives the qualitative estimation of exactly how the two given words are related.

For **explicit** relationships, such a length can reflect the degree of the importance of the relationship, e.g., IS_A relationships indicate that the words are close and probably substitutable for each other in most contexts. On the other hand, CONSUMES relationship reflects much less degree of closeness, i.e., it is "longer."

For **inferred** relationships, their lengths can depend on the kind of relationships involved in the logical inference (i.e., be an attribute of the corresponding rule) as well as on the length of the logical chain. E.g., many applications of the transitivity rule for IS_A relationships can increase the length of the resulting IS_A relationship. On Fig. 1, it is true but "less reliable" that *cat* CAN HAVE *telescope*, due to too many applications of IS_A transitivity rule. The fuzzy character of the inference rules is quite obvious when it comes to such relationships as IS_SIMILAR_TO, which is "to some degree" transitive.

For a **path**, its length should increase with the total length of the constituting links. The more is the length of the path between two words, the less semantically close the words are.

The easy way to assign the lengths to the links is to relate a specific length value to each **type** of the links, e.g., 1 to IS_A, 5 to SIMILAR_TO, and 20 to CAUSED_BY. Such assignment may be context-dependent or may vary according to the type of information being retrieved from the text. E.g., in a text with the principal topic "toys" the relationship SIMILAR_APPEARANCE can be more important than in a text with another principal topic. We leave such considerations for the future.

In some cases it might be desirable to assign a length to **individual** links or groups of links. Ideally, each individual link should have some specific length expressing the degree of commonality between the two specific words. Assignment of these values hardly can be done by hand, instead, some procedure for training on a large corpus might be used in the future. There are cases, though, when additional individual coefficients can be assigned automatically.

For example, special precautions are to be taken in order to prevent the algorithm from abuse of hierarchical links like IS_A. Namely, any concept referring to an object IS_A *object*: *car* IS_A *object*, *book* IS_A *object*. Thus any two objects are connected with a path of two (usually implicit) links: *car* IS_A *object* HAS_SUBTYPE *book*, which normally should not imply a great degree of commonality between them. On the other hand, in some cases a path of two IS_A relationships does imply commonality: *Ford* IS_A *car* HAS_SUBTYPE *BMW*.

To some degree this problem is already solved by assigning to the implicit relationship a greater length (corresponding to a weaker relationship) than the length of an explicit relationship, when the implicit relationship is computed by application of the inference rules. However the precision of the procedure can be improved by assigning the greater length to the hierarchical links located near the top of the hierarchy, thus, the length of the link *thing* IS_A *object* is more than that of the link *Ford* IS_A *car*. Namely, on the stage of preparation of the relationships database, the *maximum* number of links is determined from each node to the top of the hierarchy, and the links leading to this node are scaled correspondingly.

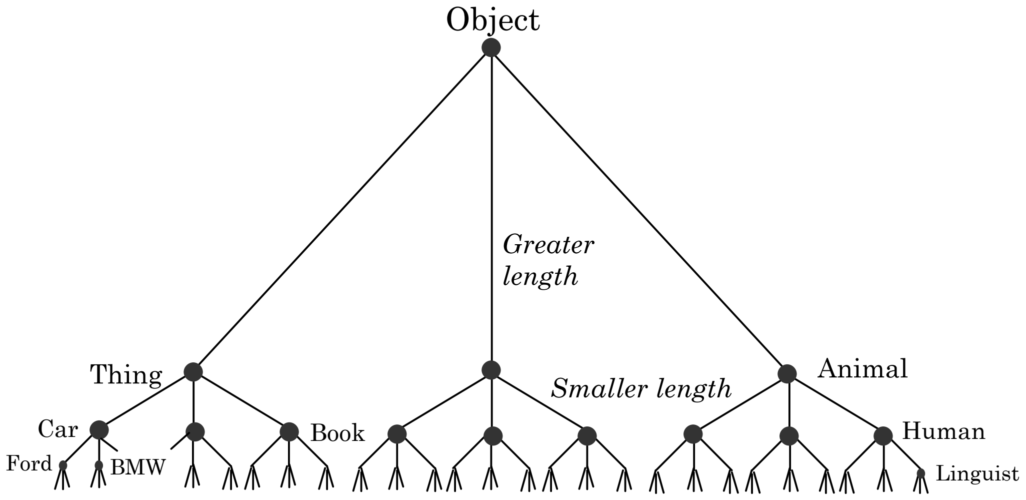

Fig. 2. Different length of the hierarchical links.

E.g., on Fig. 2, the distance between *Ford* and *linguist* is much greater than between *Ford* and *BMW*, though both ones are subtypes of *object*. The distance between *car* and *book* is greater than the distance between *Ford* and *BMW*, though in both pairs there are exactly two explicit links between the nodes.

2.4. Shortest paths problem

To determine the semantic distance between two words, the shortest possible path in the network is to be found; its length can be used as an estimation of the degree of their semantic nearness. Since not all paths can be acceptable in a specific context, in some, and supposedly many, cases the next, then next, etc., shortest path should be used to determine the mesure of the semantic distance between two words in a specific *context*. E.g., if the context suggests that the possible relationship between the two words is USES (expressed by the preposition *with*), then even a shorter relationship of the type IS_A cannot be used as a measure of closeness between these words in this context.

Naturally the curse of ambiguity manifests itself in the full degree in the task of finding such paths. There is virtually infinite number of paths in the network, connecting the two given words. The computational aspects of the problem of finding the shortest path are discussed in the last section of the article; here it is enough to mention that the problem has well-known solutions and the only mathematical issue is computational efficiency. Thus we will first discuss the linguistic applications of such an algorithm, provided that it operates on a large enough semantic network dictionary.

3. Linguistic Applications

An autonatic procedure for finding the shortest paths in the semantic network between the two given words and measuring their "nearness" in the network in a specific context has numerous applications for disambiguation in language processing and automatic translation.

The methods we describe should not be regarded as the ultimate solution of the problem of ambiguity. Instead, we believe that this problem must be solved by "voting" of as many different methods as possible in a particular system, and here we propose one of such "voters."

3.1. Syntactical disambiguation

Consider an English phrase "*John sees a cat with a telescope*." The phrase is syntactically ambiguous: Does it mean 'John uses a telescope to see a cat' or 'John sees a cat that has a telescope,' or 'John sees a cat and a telescope,' or maybe 'John that has a telescope sees a cat,' etc.? This ambiguity cannot be resolved using only lexical or syntactical information, since all the interpretations are quite legal syntactically. On Fig. 3, the first two of above variants are represented.

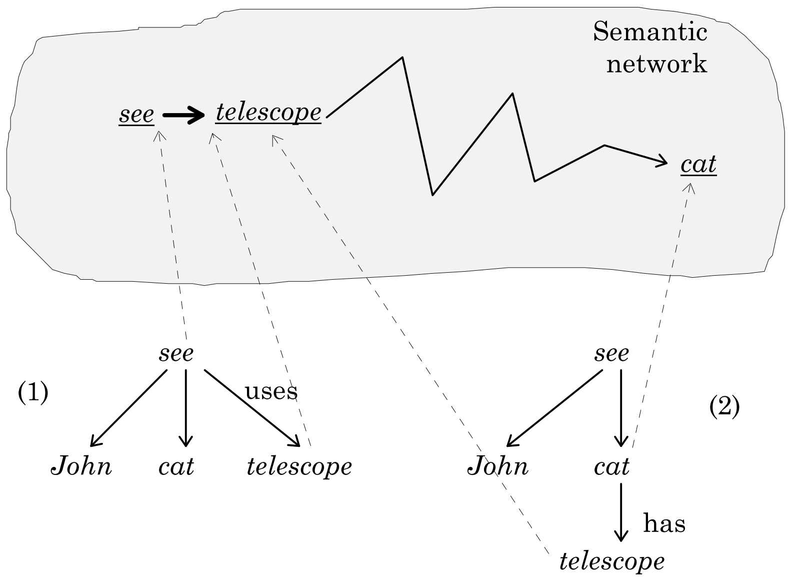

Fig. 3. Resolution of syntactic ambiguity.

The syntactic relationships in question are *see* → *telescope* and *cat* → *telescope*; what are the semantic relationships between these words? There is a short path between *seeing* and *telescope* in the semantic network dictionary. What is more, the type of the relationship(s) constituting this path agrees with the supposed instrumental syntactic relationship between these words in the phrase. On the other hand, the best path between any sense of *cat* and *telescope* that agrees with the type of the supposed syntactic relationship, is much longer. Thus, the variant (1) should be chosen as the probable one.

Note that this of course should not prevent the linguistic processor from being able to revise this decision later if the subsequent sentences disagree with this choice.

In the simplest case, just the quantitative measure of the nearness (the weighted length of the path) can be used for comparison. However for better quality of analysis the whole path should be checked against the expected syntactical type of the relationship. E.g., in a phrase "*John sees a cat with a boy"* there is a short path between *seeing* and *boy*: *boy* IS_ABLE *see*, but the type of the relationship contradicts with the hypothesis that *boy* here is a tool to *see* with. This is why the procedure for finding the paths should be able to enumerate the paths until an acceptable one is found.

3.2. Lexical disambiguation

Similar problems arise with the selection of a particular meaning of a word in the phrase. Sometimes they can be resolved at syntactic level, usually when the choice is made between different parts of speech, e.g., in the phrase "*John tables the dishes"* the word *tables* is clearly a verb. However in many cases, especially when a word has different meanings within the same part of speech, semantic information has to be employed.

Compare, for example, the phrases "*There were fruits and drinks on the table*" and "*The numbers were arranged in a table*." By addressing a semantic network, it can be determined that in the first phrase the shortest path exists between other words and the sense 'table as a furniture,' while in the second phrase, the shortest path leads from *numbers* to 'table as a picture.'

It is not as clear as it is with syntactic ambiguity, with what words in the phrase the given word is to be compared. Good candidates can be words close in the syntactical structure to the given one. Other good candidates are the words describing the main global or local topics of the document. For example, if the document in general is on mathematics, the word *table* will likely be used in it as 'table as a picture,' even if the nearest words don't suggest this directly.

Local and global topics of the document can be determined with the approach developed by Dr. A.Guzmán of CIC in his program CLASITEX [Guzmán-Arenas, 1997]. In this approach, all the textual words of the document without any preliminary disambiguation are clustered with the help of a semantic network dictionary. The centers of the largest clusters represent the main topics of the document. The noise thus is filtered out, since the wrong homonyms of the words form smaller clusters. This approach can be applied to the parts of the document, revealing the local topics. These topics can also be used in the disambiguation process: the distance is to be measured between them and the current word being disambiguated.

Since in the process of disambiguation many words or word senses (global and local topics, surrounding words, etc.) possibly have to be tried and the results have to be accumulated, the procedure may be computationally demanding. However, in comparison with, say, Word Expert Parser model [Small and Rieger, 1982], our procedure requires easier available data and can be used in frame of the traditional text processing algorithms.

3.3. Referential disambiguation

The problem of referential disambiguation arises each time a pronoun, ellipsis, or zero subject (very common in such languages as Spanish) is used in the text. In general, at the stage of text analysis such a reference must be replaced with another word probably used somewhere in the text. Though there are linguistic considerations on selecting the candidates to fill the valence, they usually give ambiguous results when only lexical and syntactic information is considered.

However, it is possible to resolve this task into the task of lexical disambiguation. Namely, when several candidates are to be tried to fill the valence, they can be just treated as different "senses" of the pronoun in this particular context. Then the procedure described in the previous section can be applied with nearly no modifications. The only difference is that neither global nor local topics are not considered in the comparisons.

3.4. Machine translation

In general text translation is quite another task than text understanding. Ideally translation should include the steps of text understanding in the source language and then text generating in the target language. If the ambiguity is resolved at the stage of analysis, and if the bilingual dictionary is good enough, there should be no problems with ambiguity during text generation. However, in real life it is not the case, for both practical and theoretical reasons [Narin'yani, 1997].

In practice currently some less sophisticated methods are used, working mostly at the syntactic level. To our knowledge, most commercially available translation programs, such as Globalink's Power Translator [Power Translator, 1997][3], distinguish the senses of the words only by a limited number of semantic classes or by literal recognition of some number of idioms. E.g., this phrase was translated from Spanish by Power Translator

[3] We used the version of 1994 for our experiments.

Professional: "*El artista realiza bien el papel*" ⇒ "*The artist accomplishes well the paper*" (instead of *role*[4]).

For the ideal scheme of translation, there must be available (1) a good disambiguation procedure in the source language, (2) a good bilingual dictionary that translates one-to-one senses to senses, not textual words to sets of words. Both conditions are very difficult to satisfy. For example, there might not be available a Spanish dictionary to disambiguate the two senses of the word *papel*.

What is more, the most elaborated up to date dictionaries, including academic dictionaries, usually provide translations of a word to several possible words in the target language, e.g.: "*papel*: paper; document; role; <...>" [Spanish-English, 1963]. In this case even if the senses have been disambiguated in the source language, the dictionary anyway does not contain the information necessary to translate them one-to-one into words of the target language.

However with the procedure suggested in this article it becomes possible to disambiguate the words after translation in the target language. As in the previous section, we can treat the ambiguous position as a word with several "senses" and then apply the procedure of lexical disambiguation to the generated phrase in the target language.

E.g., in the example above we can notice that there is a shorter path in the English semantic network between *artist* and *role* than between *artist* and *paper* or *document*. This will allow to use a semantic network as a "black box" to improve the results of translation made with help of existing bilingual dictionaries, rather than to develop new sense-to-sense dictionaries, which are expensive to create and difficult to share between different systems due to their tight integration with the other modules of a linguistic processor.

3.5. Automatic translation of the semantic network

The disambiguation procedure can be applied to automatic or semi-automatic translation of the semantic network itself into other languages. Since we have taken part in such a translation project (though the work was mainly done by hand), we are aware of all the deficiencies of the very idea of translation of a semantic network, and of low quality of the resulting dictionary [Bolshakov *et al.*, 1995b]. However, we see at least three reasons to translate semantic networks.

First, creating a semantic network from zero is a very difficult and expensive work. If the the way the results are used is tolerant to the incompleteness and minor inaccuracies, it

[4] Though this program does distinguish these senses in some contexts, it seems to make choice based on nearly literal recognition of the idiom "*jugar un papel*," e.g., "*El diputado juega un papel importante*" ⇒ "*The deputy plays a role important*," but: "*El diputado juega el papel más importante*" ⇒ "*The deputy plays the most important paper*."

may be more efficient to use a lower quality dictionary translated from an existing resource than to wait for a better dictionary to be created in the far future. Actually we believe that due to the nature of the functioning of natural language, any language processing software must be tolerant enough to incomplete and inaccurate information.

However, since the semantic network contains mostly the facts about real-world objects and ideas, and in part due to commonality between the languages, most of the relationships tend to be translated correctly (though this may depend on the languages and subject area).

Second, the linguistic resources for such languages as English, French, Japanese, etc., are maintained by many people and groups all over the world, with much money spent on their development, enlargement, and refinement. It would be a waste of effort to repeat all this work in full size for each language.

Thus for groups that work on, say, Spanish language, to take advantage of the efforts spent in the world on development of English semantic networks, it is necessary not only to translate the first draft of the dictionary from English, but to be able to repeat such translation automatically as new versions of the English dictionaries become available. There is no need to mention that the existing machine translation programs designed for translation of phrases in a discourse are not appropriate to translate structured resources such as dictionaries; thus the necessity to create specialized dictionary translation software.

Third, currently the ANSI is developing a standard ontology, so-called T2 Reference Ontology. It is developed in English. Automatic or semi-automatic procedure for translation of this resource can be very useful in maintaining compatibility between semantic networks in different languages and the ANSI standard.

A detailed description of the translation procedure is beyond the scope of this article. Here we only discuss the application of the procedure for enumerating the paths in the network between two given points to the task of translation of the semantic network itself.

The main problem of automatic translation of a semantic network is the same: ambiguity. Each word in each its occurrence in the text of the dictionary, presumably in different senses, is translated by an ordinary bilingual dictionary to several different words of the target language.

We propose the following procedure to choose the correct variant of the translation, using the same (English in our case) semantic network. Each variant of translation of a word is translated back to the source language. Then the distance in the source semantic network is measured between the source word and each variant of such a reverse translation. The variant(s) of translation are chosen, at least one of whose reverse translations is located near the source word sense in the network, i.e., there is a “short” enough path from this variant to the source word sense, see Fig. 4.

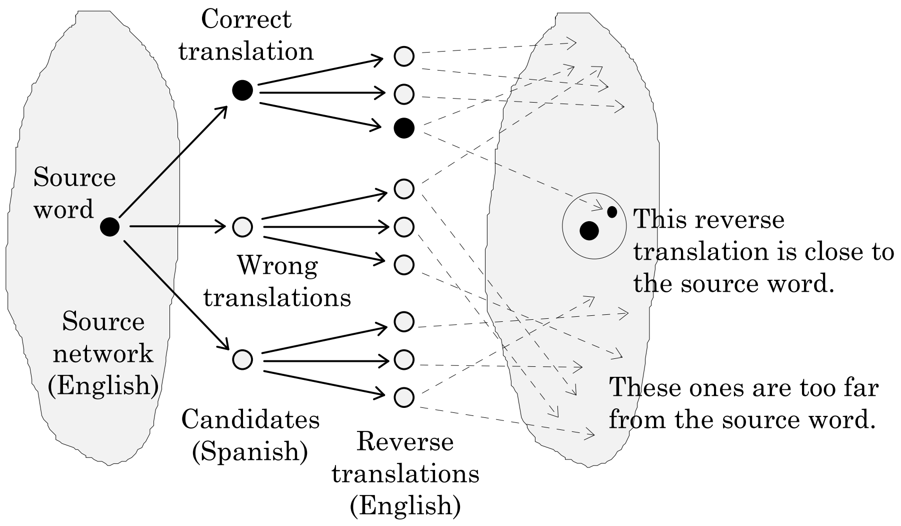
Correct translation
Source word
Source network (English)
Wrong translations
Candidates (Spanish)
Reverse translations (English)
This reverse translation is close to the source word.
These ones are too far from the source word.

Fig. 4. Translation of a semantic network[5].

Naturally, the copy of the source word is removed from the set of the reverse translations. Words having only one reverse translation, namely the same source word, are treated as special cases. They are inserted in the resulting dictionary, and if the source word has different senses, such words are marked when automatically inserted in the dictionary, and then checked by hand.

Ideally, only the words with a reverse translation within the same concept, i.e., at the zero distance from the source word, should be accepted. But in real life, a bilingual dictionary in most cases does not gives such accurate results, therefore the paths of nonzero length should be taken into account.

When searching for the paths in the network, for each textual word all its senses should be tried unless any disambiguating information is available in the bilingual dictionary; usually it is not.

The choice of the candidate is made in two steps. First the "weights," i.e., the lengths of the corresponding paths, of the reverse translations of each candidate are combined to calculate the "weight" of the candidate itself. Second, the candidate(s) are chosen with the best such "weight."

Different procedures can be used for both calculations. To combine the weights of the reverse translations for one candidate, in the simplest case a maximum (but not average) can be taken. In a more sophisticated procedure, all the reverse translations worse than some threshold should be ignored, and the values for the other ones should be accumulated.

[5] Two copies of the same network are shown to simplify the picture.

Similarly, to choose the acceptable candidates of translation, in the simplest case only the best one is taken for each word, or all the candidates are accepted that are better than some threshold value. More sophisticated procedures can also be tried. For example, all candidates better than some threshold value should be accepted, all candidates worse than some threshold value should be rejected, or the best one should be chosen from those candidates whose weight falls between these two thresholds.

The resulting semantic network dictionary may be then post-edited by hand. To be able to repeat the translation as new versions of the source dictionary become available, the changes made by hand should be saved in a special protocol.

In contrast with the procedure of enumerating the paths used for text processing, the procedure used for translation of the dictionary itself can be simplified by ignoring completely the inference rules, since in this case the meanings should be preserved much more precisely. The length of the path can be calculated as just the number of links in it. This makes the implementation of the procedure for translation much more straightforward than for text understanding.

For better results, though, the inference rules may be used, but application of each rule should substantially increment the length of the path. E.g., a chain of transitive relationships like IS_A should be considered long enough, whereas the procedure used for text understanding would use the length near to 1 for such a path. The choice here, as well as the selection of the thresholds mentioned above, is made on the basis of desired compromise between the accuracy of the translation (less usage of inference, higher thresholds) and the number of words that will get any translation at all (more usage of inference, lower thresholds).

In any case only basic relationships, such as IS_A or possibly IS_PART_OF and few others, should be allowed to appear in the paths, while such relationships as USES, etc., should be prohibited.

4. Computational aspects

This section describes the mathematical problem statement and possible algorithms for its solution. The reader who is not interested in computational details can skip this section completely. Generally a simple modification of Dijkstra's shortest path finiding algorithm [Dijkstra, 1959] should be suffictient, though we describe here a more sophisticated modification of Dijkstra's algorithm, adapted to large sparse networks.

4.1. Problem statement

As it has been shown above, finding paths in the network is useful for linguistic applications, primarily to measure the distance in the network between the two given words in a specific context. Usually to measure such a distance in a specific context, the shortest paths between the two points are to be found; however, it is not always true that necessarily the very short, optimal, path should be found first. There are several reasons for this.

First, there are many **restrictions** on suitability of a particular types of paths in a specific context. If, for example, the syntactic relationship between the two given words suggests, say, instrumental relationship, then the semantic relationships of any other types, even very short, are useless in this context. Such restrictions can be applied after the path is found in the network. Thus, it is probable that the very short path will prove unusable in a specific context; in this case the next path should be retrieved and examined. Thus, there is no point to apply a computationally demanding procedure to completely optimize the search process.

Second, the rules for calculating the length of a path in a specific context can be **context-dependent**, complicated, and are better applied after the path has been found. This problem is discussed in more detail below at the end of this section.

Third, the **low precision** of all language data, including in the first place the text itself being analyzed, but also the dictionaries, grammars, etc., makes very precise procedures not so necessary. Everything in language is vague, any text is full with hints, omissions, implied information, metaphors, rather than being a collection of clear and simple logical statements. This makes too precise procedures of text processing in many cases useless.

On the other hand, since automatic procedures are applied to huge amounts of texts, **performance** is important, as long as the final result of analysis fits in the same "confidence interval." Performance is especially important since the procedure for semantic distance measurement is invoked very many times in typical applications, such as referential disambiguation.

Thus we can consider the following problem:

To enumerate the paths in a network between two given nodes, as a tendency starting from shorter ones, under the following conditions:

- *Various timeouts apply, e.g., a threshold on the length of the paths: only the paths of the length less than a threshold value are considered.*
- *Computational effectiveness is a priority.*
- *Accuracy is the second priority. The better paths should go first, but only as a tendency.*
- *The network is large and stored in a database, so that retrieval of the links leading from a given point is the most time-consuming operation.*

Also at each step estimate (again, with some probability) the lower limit of the length of the paths the procedure can find yet.

The significance of the latter requirement will be discussed below. It is supposed that the calling routine will at some moment stop the enumerating process, or some kind of time-out will be applied to prevent the algorithm from infinite work, such as a restriction on the number of paths, or on their lengths, so that if the procedure cannot find any paths shorter than some threshold, it should stop. Also some qualitative restrictions may be imposed on the desirable paths, e.g., not to contain a particular relationship.

The problem is very similar to the well-known problem of finding the shortest paths in sparse graphs, e.g., [Shier, 1976; Johnson, 1977; Minieka, 1978], but there are some differences in goals and conditions with the classical problems of optimization, shown in Table 1.

Classical Optimization	**Our problem**
There are no restrictions on the length of the path.	Only the paths shorter than some threshold value (i.e., short enough ones) are considered.
Only one path is searched for (in some variants K paths).	Paths must be enumerated until the caller "accepts" one.
The very best path must be found.	The better paths should generally go first, but not necessarily the very best is to be the first.
The length of a path is a mere sum of the lengths, or weights, of the individual links.	The length is calculated according to the fuzzy rules of combination of the relationships.
No previously prepared data is usually used.	Some data can be prepared in the database in advance.
The graph is small enough to be kept in memory.	The graph is very big and is stored on the hard disk.

Table 1. Comparison of the proposed problem and the classical one.

In the Table 1, by better paths we mean the ones with less length, that usually means ones that contain fewer links or links with less lengths. This measure is computed as a combination of the lengths of individual links, with application of, or taking into account, the rules of inference. E.g., a chain of five IS_A relationships may be considered "shorter" than a chain of two IS_SIMILAR_TO relationships. In general such an estimation is a complex problem by itself, and we will not describe it here in more detail.

Note again that the measure of length used by the algorithm can differ from the measure that is used by the calling procedure, the latter being probably calculated or refined by the caller itself with the application of some specific, possibly very complex, rules, for example, making use of the logical structure of the situation described in the text itself.

This difference arises from our intention to separate the information internal to the semantic network from the information used in various applications, and to provide a general procedure (probably implemented as a separate module) that permits the caller to treat the semantic network as a black box. Though, some minimal adjustments of the procedure will anyway be necessary for some applications; they are discussed in the sections on the corresponding applications.

Thus we presume that the algorithm should find the paths just good in some general meaning, and the caller will check if the path is really good for it, though the "generally good" paths should be usually good enough for the caller.

This implies that the algorithm should not even try to absolutely optimize the enumerating process, since anyway chances are little that the very best in general sense path will be the very best for the caller, and we expect usually it will not. This changes the approach to the algorithm as compared with the classical optimization problems.

4.2. Possible algorithms

In spite of the large number of researches devoted to the shortest paths problem, we are not aware of any known algorithm for solving exactly our task. It is not the main goal of this article to propose a mathematically refined algorithm, since at the current stage of the research we are mostly interested in the linguistic applications of the idea itself. However, we describe here the variants of the algorithms we currently use.

4.2.1. *The case of equal lengths of the links*

In this section, a non-weighted graph is considered. A simple algorithm of enumerating all the paths, a modification of Dijksatra's one, is as follows. Let's define a sphere S_r (A) around the point A as the set, actually a tree, of all the paths[6] of the length *r* leading from the point A. Since we consider here the length of each link in the network to be just 1, the radii of the spheres are natural numbers; we can also consider S_0 (A) being the empty path, i.e., the point A itself. Let's call the set of ends of all the paths of the sphere S (A) its surface.

The sphere S_{i+1} (A) can be formed by adding to S_i (A) each link leading from each its *surface* point, see Fig. 5. If necessary, obvious precautions can be taken to prevent the paths being formed from cycles, at least from the cycles formed by two copies of the same link passed in the opposite directions; this can be done by comparision of the link being added to a path with the immediately previous link in the path. Other types of cycles in a sparse network usually do not present much problems for our task.

[6] Each path in the tree is compactly represented by the ending point, additional characteristics such as the length, and a pointer to the previous path in the tree. When a new path is formed by adding one link to the previous one, only such a data structure is to be created in memory.

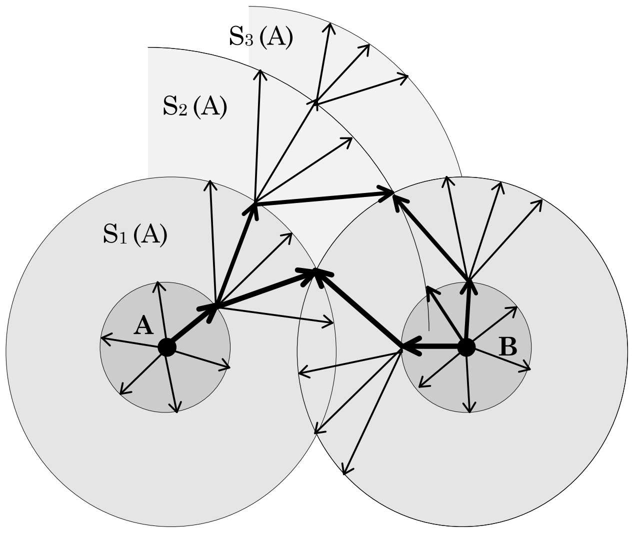

Fig. 5. The spheres in the network.

The algorithm increases the spheres S (A) and S (B) in turn, starting from, say, S (A). At each step, the intersection of the *surfaces* of the spheres S_i (A) and S_j (B), $j = i$ or $j = i - 1$, gives the paths of the length $i + j$. This algorithm enumerates all the paths between A and B, starting from the shortest ones.

For an oriented graph, when only the oriented paths from A to B are to be found, a simple modification of this algorithm can be used. The sphere S^+_{i+1} (A) should be formed only with the links leading *from* the points of S^+_i (A), while the sphere S^-_{i+1} (B) should be formed only with the links leading *to* the points of S^-_i (B). If the paths both from A to B and from B to A are to be found, two spheres S^+ and S^- are maintained for A and B, consisting of the links leading to and from the points, correspondingly. If there are other restrictions on the types of the paths, they also can be taken into consideration at the step of increasing the spheres.

4.2.2. *Different lengths and inference rules*

The previous algorithm can be generalized to the case of weighted graphs. We consider here a modification that not always gives the shortest paths first, but does so as a tendency. This algorithm can be easily modified to enumerate the paths in the proper order, but with slightly lower efficiency.

In this algorithm the sets of paths which we will still call spheres actually are not spheres, i.e., the paths in such "spheres" do not have the same lengths. We define these spheres just recursively, the sphere S_{i+1} (A) being formed by adding to S_i (A) all the links leading from some of its surface points[7]. In this case not all the surface points of S_i (A) are

[7] We chose to add *all* the links here since the operation of retrieval of the links is the most time-consuming.

expanded, instead, expanded are only the paths, usually one path, with the minimal length among all the paths of S_i (A). The surface of S_{i+1} (A) is defined by replacing the expanded points of the surface of S_i (A) with the ends of the newly added links.

Again, the spheres S (A) and S (B) are increased in turn, and the intersection of their surfaces gives the different paths between A and B. It is easy to prove that this algorithm enumerates all the paths. Namely, let's call the minimal length of a path in the sphere its *minimal radius*. Each step of the algorithm increases the minimal radius of one sphere, and if the current minimal radii of S (A) and S (B) are r_1 and r_2, then all the paths with the lengths of $r_1 + r_2$ have been already enumerated by this moment.

This algorithm does not enumerate the paths in the order of their length, see Fig. 6, though generally it tends to enumerate the shorter paths before the longer ones. It is possible to store the found paths temporarily without reporting them to the output, until the value $r_1 + r_2$ reaches the length of a temporarily stored path. With this modification the algorithm will enumerate the paths in the proper order, but for our goals we chose to use the path as soon as the algorithm finds it.

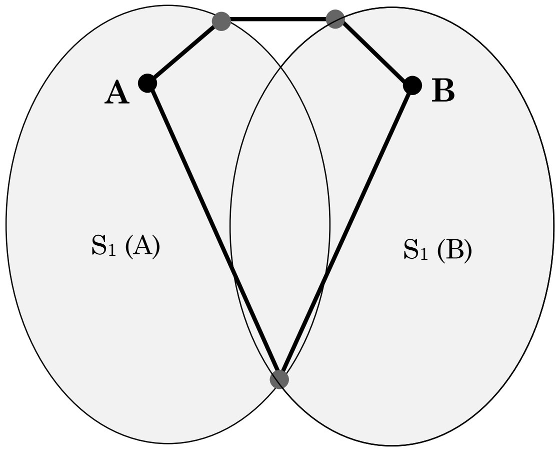

Fig. 6. A counter example: a longer path is found first.

The inference rules and the rules for determining the lengths of different combinations of the relationships can be taken into account at the step of increasing the spheres and at the step of determining the intersection of the spheres. Namely, when a link is added to the path, the length of the path being formed is determined accordingly to the rules of combination of links. Similarly, when a complete path between A and B is formed by connecting together two paths, one of S (A) and another of S (B), the length of the combination may differ from the sum of the length of the two paths. This contributes in the lack of order in enumeration process. However, the shorter paths still tend to be enumerated before the longer ones.

4.2.3. *Use of pre-calculated data*

Due to the high number of "dimensions" of the network, the methods described above are good only to find short enough paths, since spheres of big radii are too large. In practice it may not be a problem if only short paths are searched for.

However if longer paths are required, an additional network of "control nodes" with pre-calculated information about their connections with each other may be used. This is similar to the idea of a cellular telephone system, see Fig. 7.

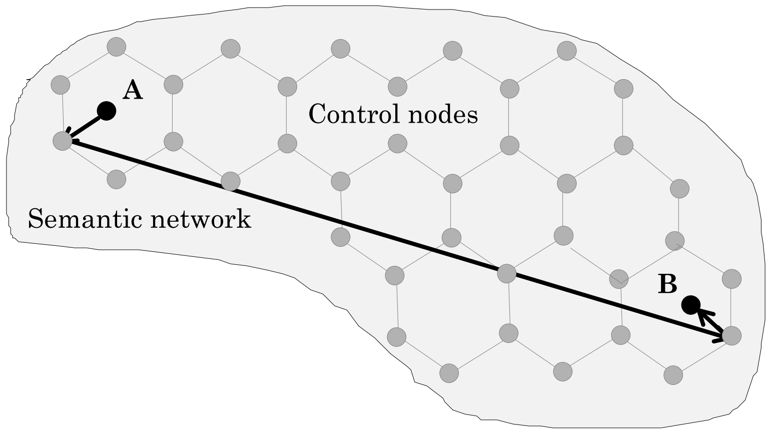

Fig. 7. The network of control nodes.

At the stage of preparation of the database, nodes are added, or existing nodes are used, at nearly equal distances from each other and not further than some threshold distance from any node in the network. The number of such control nodes should be much less than the total number of nodes in the network. The information is stored with these nodes that helps to find the paths leading from each of them to each another. To find a path from an arbitrary point A in the network to another point B, first the paths from these points to the nearest control node are determined using the method of increasing spheres described above, then the path between the control nodes is retrieved from the database, and then the whole resulting path can be varied or optimized locally.

4.2.4. *Multiple comparisons*

In the applications, often not only the distance between two given nodes is to be determined. Instead, the questions to be answered are: What point in some set of points is the nearest to a given one, or what are two nearest points in a two given sets. These problems arise in disambiguation of the binding of a prepositional phrase and in referential disambiguation, correspondingly.

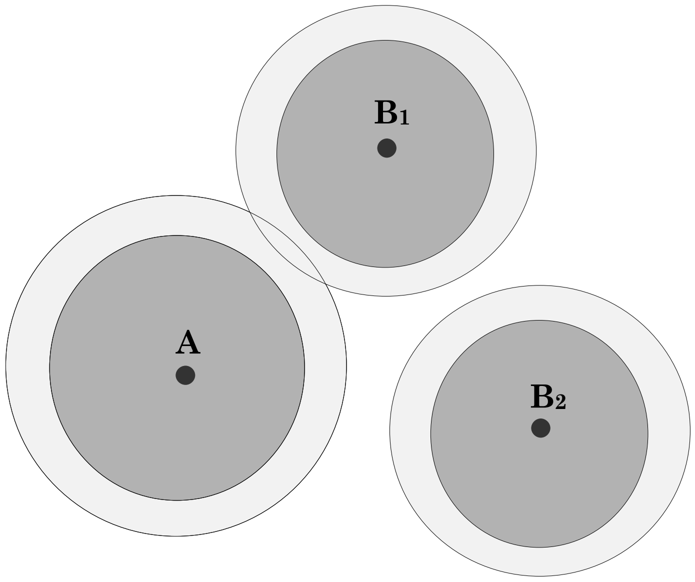

Fig. 8. Determining the distance to a set of nodes.

A simple modification of the algorithm, shown on the Fig. 8, allows to take advantage of increasing the spheres in turn and of using the same sphere to determine the distances from the given point A to each of the points B_1, ..., B_n. With light color the “surface” of the sphere is shown, since the paths constituting the “sphere” in a weighted graph can have different length. With the darker color, the minimal radius is shown.

All the spheres are increased in turn. If there was found at least one path between A and, say, B_1 such that its length is less than the sum of the minimal radii of the spheres S (A) and S (B_2), and no shorter paths were yet found between A and B_2, then the distance between A and B_1 is smaller than the distance between A and B_2. Note that since the paths are found not in a completely precise order, the check against the minimal radii is very desirable.

With the complicated rules of link combination, the latter criterion is not precise, since the length of a path can be different from the sum of the lengths of its parts. Currently we ignore this complication, since we consider the lost of precision less than normal fuzziness of all data related to natural language. In case of serious problems arising in understanding of a particular phrase, backtracking can be used later to calculate the distances in question more precisely.

Conclusions

In recent years semantic network dictionaries have become available, mostly for English language, such as MICRA’s FACTOTUM® SemNet dictionary.

As it was shown in this work, a simple procedure, namely the search for the shortest paths in a sparse network, can be used for determining the measure of semantic “nearness” of two given word senses in a very large semantic network. This measure is useful for disambiguation in a variety of important tasks of natural language processing, namely for lexical, syntactical, referential disambiguation, as well as in text generation and machine translation.

Also this procedure can be used to automatically translate the semantic network dictionary itself into other languages. This makes the methods described in the work available for processing of languages other than English, and also simplifies creation and maintenance of semantic network dictionaries for these languages. What is more, such automatic translation of a semantic network will be useful in development and maintanance of semantic networks in the languages other than English, which would conform to the Standard Ontology (T2) being developed currently by ANSI.

Acknowledgments

Dr. P.Cassidy of MICRA, Inc. and M.Sc. S.N.Galicia-Haro of CIC have read the manuscript and provided me with very useful advice, discussions, and references. Lic. R.Reyes-González of CIC took part in discussions of the algorithms and implemented the preliminary version of the program.

References

[Alpha, 1995] Alpha K. Luk, *Statistical Sense Disambiguation with Relatively Small Corpora Using Dictionary Definitions*. In Proceedings of the 33rd Annual Meeting of the Amer. Soc. for Comp. Ling., 1995, pp. 181-188.

[Berleant and Daniel, 1995] Berleant, Daniel. *Engineering "word experts" for word sense disambiguation*. In Natural Language Engineering 1, 1995: pp. 339-362.

[Bolshakov *et al.*, 1995a] Bolshakov, I.A., P.J.Cassidy, A.F.Gelbukh. *CrossLexica -- a dictionary of collocations and thesaurus of the general Russian lexicon* (in Russian, abstract in English). In Proceedings of International Workshop Dialogue’95: Computational Linguistics and its Applications, Khazan, 1995.

[Bolshakov *et al.*, 1995b] Bolshakov, I.A., P.J. Cassidy, A.F. Gelbukh. *Parallel English and Russian hierarchical thesauri with semantic links, based on an enriched Roget’s thesaurus* (in Russian, abstract in English). In Proceedings of International Workshop Dialogue’95: Computational Linguistics and its Applications, Khazan, 1995.

[Dijkstra, 1959] Dijkstra, E.W. *A note of two problems in connection with graphs*. Numerische Matematik, 1959, V. 1, pp. 269-271.

[Gelbukh, 1997] Gelbukh, A.F. *Using a Semantic Network for Lexical and Syntactic Disambiguation*. In Proc. of the International Symposium “CIC-97: Nuevas Aplicaciones e Innovaciones Tecnológicas en Computación,” November 12-14, 1997, Mexico D.F.

[Guzmán-Arenas, 1997] Guzmán-Arenas, A. *Determining principal themes in a Spanish article* (in Spanish). In Proc. of the International Symposium "CIC-97: Nuevas Aplicaciones e Innovaciones Tecnológicas en Computación," November 12-14, 1997, Mexico D.F.

[FACTOTUM, 1997] *FACTOTUM® SemNet*, MICRA, Inc., ftp://ftp.cs.cmu.edu/user/ai/-new/fsn*.*, relation.asc; email: cassidy@micra.com.

[Johnson, 1977] Johnson, D.B. *Efficient algorithms for shortest paths in sparse networks*. J. ACM, 1997. Vol. 24, N 1, pp. 1-13.

[Mel'cuk, 1974] Mel'cuk, Igor A. *Experience in theories of Meaning ⇔ Text linguistic models* (in Russian). Moscow: Nauka, 1974.

[Miller, 1990] Miller, George A., ed. *WordNet: An on-line lexical database*. International Journal of Lexicography, 3, 1990: pp. 235-312.

[Narin'yani, 1997] Narin'yani, A.S. *Automatic text understanding – new perspective* (in Russian, abstract in English). In Proceedings of International Workshop Dialogue'97: Computational Linguistics and its Applications, Moscow, 1997.

[Power Translator, 1997] *Power Translator Professional software*, Globalink Inc., http://-www.globalink.com/products.html.

[Shier, 1976] Shier D.R. *Iterative methods for determining the* K *shortest paths in a network*. In Networks, 1976. Vol. 6, N 3, pp. 205-229.

[Small and Rieger, 1982] Small, S.L., and C.J. Rieger. *Parsing and comprehending with word experts (a theory and its realization)*. In Lehnert and Ringle (eds.), Strategies for Natural Language Processing, 1982, pp. 89-147.

[Spanish-English, 1963] *Spanish-English, English-Spanish dictionary*, Pocket books, Inc. NY, 1963.

[Steel, 1990] Steel, James, ed. *Meaning – Text Theory. Linguistics, lexicography, and implications*. University of Ottawa press, 1990.

[Sussna, 1993] Sussna, M. *Word Sense disambiguation for free text indexing using a massive semantic network*. In Proceedings of CIKM, 1993.

[Voorhees, 1993] Voorhees, E.M. *Using WordNet to disambiguate word sense for text retrieval*. In Proceedings of ACM SIGIR Conference, 1993, pp. 171-180.

[Yarowsky, 1995] Yarowsky, David. *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods*. In Proceedings of the 33rd Annual Meeting of the Amer. Soc. for Comp. Ling., 1995, pp. 189-196.

[Yarowsky, 1992] Yarowsky, David. *Word Sense Disambiguation Using Statistical Models of Roget's Categories Training on Large Corpora*. In Proceedings of COLING-92, 1992, pp. 454-460.