KASSYS: A DEFINITION ACQUISITION SYSTEM IN NATURAL LANGUAGE

Patrice Hernert
L.I.F.O., R. L. de Vinci, B.P. 6759, 45067 Orléans Cedex 2, France
E-mail: hernert@chambord.univ-orleans.fr

Abstract: This paper is an introduction to KASSYS, a system that has been designed to extract information from defining statements in natural language. Only hyperonymous definitions are dealt with here, for which systematic processing has been devised and implemented in the initial version of the system. The paper describes how KASSYS builds a taxinomic hierarchy by extracting the hyperonyms from these definitions. It also explains the way in which the system can answer closed questions (yes/no), thus enabling the user to check very quickly that a definition has been assimilated correctly. The underlying formalism is that of conceptual graphs, with which the reader is assumed to be familiar.

Keywords: Conceptual graphs, hyperonymous definition, knowledge acquisition, question (yes/no).

1-INTRODUCTION

The aim of KASSYS, the system described here, is to acquire lexicographical definitions expressed in French, to extract from these definitions a carefully chosen conceptual structure, to save this structure in a file, and to then be able to use it, where appropriate, for the semantic analysis of a text or during the search for an answer to a question put by the user. [1] The formalism which has been adopted for the representation of the definitions is that of conceptual graphs, [2] that the reader is assumed to understand. All the examples will be given in the form of statements in natural language, and the operations actually performed by the system on the conceptual structures extracted from these statements will not be described. This paper is limited to hyperonymous definitions. It also shows, very briefly, how KASSYS can answer certain types of question.

2. KNOWLEDGE EXTRACTION

2.1 - Some points concerning hyperonymous definitions

Hyperonymy/hyponymy can be defined as follows: term A is said to be a *hyperonym* of term B, or alternatively that term B is a *hyponym* of term A, if the set of instances of term B is included in the set of instances of term A. This gives the following corollary: the set of semantic features that make up the elements of A is included in the set of semantic features that make up the elements of B; the elements of B are said to *inherit* the semantic features that are common to the elements of A. Here we have the notion of inheritance of semantic features that is fundamental to the theory of semantic networks.

Hyperonymy is thus defined by the general relation:

(1) (A is a hyperonym of B) $\equiv$ ($\forall$ x, B(x) $\supset$ A(x))

The equivalent in natural language of (1) is:

(2) (A is a hyperonym of B) $\equiv$ (All B is A)

For example, the definition of the concept *bee*:

(3) bee: a social insect which produces wax and honey.

From this definition it is possible to extract the following statement, in which *insect* is the hyperonym of *bee*:

(4) All bees are insects.

In formal terms a hyperonymous definition can be written as in (5), in which the defining statement is split into two fundamental components: the hyperonym, followed by a conjunction of semantic features which distinguish the defined from this hyperonym; this conjunction of semantic features is called *specific difference*.

(5) $\forall x, B(x) \supset (A(x) \wedge P1(x) \wedge P2(x) \wedge ... \wedge Pn(x))$

The implication contained in (5) is that KASSYS perceives definitions as statements of conditions that are necessary but not sufficient. It is assumed that concept B as defined in (5) may possess semantic features that have not been specified but the knowledge of which may turn out to be indispensable if it were necessary to differentiate it from an individual belonging to a very similar class.

2.2 - How to extract the hyperonym

It is usually fairly easy to extract the hyperonym from a hyperonymous definition. In nearly all cases the hyperonym of the defined is the first word of the definition. In KASSYS the following heuristics have been implemented:

For the definition of a verb the hyperonym is the first word of the definition; if this word is not a verb the search fails. For nouns, start by checking whether or not the definition begins with a defining prefix, i.e. an expression such as *action of, fact of*, etc.; in some cases the definition may not be hyperonymous; otherwise, if the first word of the definition is a noun it is the hyperonym; if the first N words are adjectives, possibly separated by the conjunction *and* or by a comma, and if the (N+1)th word

[1] For a complete description of the initial version of KASSYS, please refer to (Hernert 93).

[2] Cf. (Sowa 84).

is a noun, then this noun is the hyperonym.

These two heuristics are commonly used by systems to search for hyperonyms in definitions, sometimes with improvements to take into account the special cases for which these heuristics are not suitable. [3]

2.3 - How to build the taxinomic hierarchy

The hyperonym is obviously a fundamental element of a hyperonymous definition. Taken alone, a concept and its hyperonym [4] are sufficient to build an elementary semantic network in which all the nodes are connected by the same link IS-A. The semantic network is limited to a simple taxonomic hierarchy which can be built and maintained far more easily than a complete semantic network.

KASSYS carries out a certain number of checks on the proposed hyperonym. If, for example, it is too general, the user is asked to choose another; if it has already been used as a hyperonym, the system suggests that maybe one of its hyponyms could be a better candidate. Let us now look in more detail at what KASSYS does when the user defines the same concept more than once. Let us take the following definition patterns:

(6) $\forall x, A(x) \supset (B(x) \wedge C(x))$
(7) $\forall x, A(x) \supset (B'(x) \wedge C'(x))$

In (6) and (7), concept A has been defined by the hyperonyms B and B' respectively and the specific differences C and C'. There are four different cases:

1. If B=B' and C=C', the definitions are identical and the second one is therefore redundant.
2. If B=B' and C<>C', the second definition can be considered as additional information which should be merged with the definition that has already been memorised.
3. If B<>B' and C=C', the definitions are identical but for one hyperonym; the system will therefore ask the user to choose between (6) and (7); note that, if B has been defined with B' as its hyperonym (respectively B' with B), the system will suggest keeping (6) (respectively (7)).
4. If B<>B' and C<>C', the user will have to choose one of these two definitions.

Note that, if the second definition (7) does not mention the hyperonym of A, the system will find this hyperonym thanks to the first hyperonymous definition that has been entered, which necessarily contains the structure (6).

2.4 - The circularity of definitions

Whether definitions come from a French dictionary or have been produced by a user who is not a lexicographer, they usually contain characteristics that are considered to make them totally useless. Definitions are too often found to be repetitive or inconsistent; however, once these problems have been identified they can almost always be corrected. But this is not true of circular definitions which, today, are accepted as being inevitable. [5]

As far as KASSYS is concerned, the presence of cycles in definitions would have the unfortunate result of leading the program into infinite loops. In order to avoid this, an algorithm has been implemented which searches each new hyperonymous definition for words that will lead to a circular definition. Let us examine the following example:

(8) swarm: group of bees that leaves an overcrowded hive to settle elsewhere.
(9) bee: social insect of the Hymenoptera group, called *honey fly* that lives in swarms and produces wax and honey.
(10) hive: shelter designed for a swarm of bees.

If the definitions are submitted to the system in this order, the circularity due to the presence of *swarm* in the definition of the concept *bee* is detected as soon as this definition is entered. The user is therefore asked to modify at least one of the two definitions (8) and (9). One possible solution would be to replace (9) by (11):

(11) bee: social insect of the Hymenoptera group, called *honey fly* that lives in colonies and produces wax and honey.

Now (8) and (11) are accepted without any difficulty. But there is still a problem with (10) since the definition of *hive* contains the noun *swarm*. This circularity can be got rid of by removing *hive* from the definition of *swarm* or *swarm* from the definition of *hive*. The first solution leads to seriously truncating the definition of the noun *swarm*:

(12) swarm: group of bees.

The second leads to a somewhat unnatural definition:

(13) hive: shelter designed for a group of bees.

This example shows that it is sometimes almost inevitable to have recourse to a circular definition and it is for this reason that KASSYS can be configured to accept such definitions. However, the danger is that, when the knowledge base is consulted, certain algorithms which are used in this consultation and which, at the present time, are unable to check their own evolution, may lead to infinite loops with a consequent loss of information that has not previously been memorised.

3. QUERYING THE KNOWLEDGE BASE

3.1 - Simple questions

KASSYS is able to answer yes/no type questions, i.e. it can compute the truth of certain statements. This paper does not deal with elementary queries of the type *Is an A*

[3] For example, (Byrd 87) identifies several hyperonyms in the same definition, separated by a conjunction; in (Véronis 89) there is a heuristic which, in certain cases, allows the hyperonym of a noun defined by the prefix *action of* to be extracted.

[4] Right from the beginning it has been assumed that no concept possesses more than one immediate hyperonym; from this point of view this immediate hyperonym coincides with the *genus* in the Aristotelian sense of the term.

[5] Cf, for example, (Weinreich 70), page 81.

a B?, which the system handles without any problem.

Let us suppose that the following definitions have been submitted to the system, which then analyses and memorises them:

(14) revolver: small-arm with a revolving cylinder that can contain six cartridges.

(15) pistol: small-arm with a removable cartridge clip in which the cartridges are loaded.

(16) small-arm: short, portable firearm.

(17) firearm: arm that fires shots through the detonation of an explosive mixture.

Let us begin with the simplest questions, i.e. those it is possible to answer by consulting just one definition. For example:

(18) Does a revolver have a cylinder?

Using definition (14), this question can be answered in the affirmative. This is exactly what KASSYS does, by simply projecting the conceptual graph associated to the question onto the conceptual graph of the defining statement *a revolver is a small-arm with etc.*. Note that KASSYS knows that if A is said to be with B, then A has B.

It may happen that a query is projected onto the body of a definition but not onto the defining statement that has been obtained from the defined and the definition. For example:

(19) Does a cylinder contain cartridges?

The graph of this query is not projected onto that of *a revolver is a small-arm with etc.* but onto that of the definition properly speaking of *revolver*, which contains the pattern *a cylinder contains cartridges*. The system cleverly deduces that there exist cylinders which contain cartridges and so, in answer to question (19), replies *Sometimes*.

3.2 - An algorithm using type expansion

This section deals with the case of questions that cannot be answered by consulting just one hyperonymous definition. It is assumed that these questions contain neither modal verbs nor negations.

The following algorithm has been implemented so as to be able to answer these questions: [6]

Search in the assertion to be verified for the concepts to which a type definition has been associated; for each concept C that is found:

1. Search for the definition of C.
2. For this definition, perform all possible type expansions (the strategy that has been implemented is a breadth first search); for each definition that is obtained, try to project the graph of the query onto the graph of this definition; if a projection succeeds, the answer is *Yes*; go to 5; if no projection succeeds, continue.
3. For each of the hyponyms of C, return to 1; if a projection succeeds, the answer is *Sometimes*; go to 5; if no projection succeeds, continue.
4. No projection has succeeded; the system is unable to answer.
5. If an answer has been found, display it; otherwise display *I don't know*; stop.

Let us take the query:

(20) Does a pistol fire shots?

Starting from the concept *pistol*, then performing type expansion on its hyperonym *small-arm*, followed by a second type expansion on the concept *firearm*, KASSYS builds the graph of the following definition:

(21) A pistol is a short portable arm which fires shots, etc.

We are back to the case of the previous paragraph, where just one hyperonymous definition is enough to be able to answer the question. It is easy to see that the graph of the query (20) is projected onto that of the definition (21). This is what KASSYS does, and so it replies in the affirmative to question (20).

It should be noted that this algorithm can be very time consuming, if the assertion to be verified contains more than one concept that has been defined in the knowledge base. One possible solution would be to look for the answer starting from a priority concept that we shall call the *focus* of the query and that is defined as being the concept to which the questioning applies. It is a somewhat vague notion and is rather difficult to explain clearly. To begin with, it was necessary to define a naive, focus extraction heuristic. Although far from perfect, this heuristic is never dangerous since the previous algorithm guarantees that all the concepts will be tried. However, where the heuristic computes a focus leading to a successful conclusion, the time saved is proportional to the number of concepts contained in the assertion and on which type expansion can be performed. In the example of question (20), the focus determined by the heuristic is *pistol*, which leads to a successful conclusion. The amount of time is saved here is nil but would be considerable if the definition of the concept *shot*, for example, were to be inserted in the knowledge base.

3.3 - Queries that contain a negation

Generally speaking, the handling of negation is a tricky affair for the essential reason that negation in natural language cannot be confused with logical negation. For instance, it is easy to find a statement with a truth value that is identical to that of its negation. [7] However, in a large number of elementary cases, especially where the

[6] This algorithm requires an operation which has not been defined: *type expansion*; this consists in replacing a given word in the graph of a statement by its type definition.

[7] Let us take the example of the statement *My dragon likes baklava*,

negation concerns the main verb of a clause, it is reasonable to accept that the truth value of the clause is the opposite of that of the assertion which is obtained by removing the negation from the clause. This is a heuristic which has proved to be extremely efficient in KASSYS but which would have to be re-examined if certain simplifying hypotheses were to be abandoned.

Let *p* be a statement containing just one verb, and *neg(p)* the negation of this statement, obtained by adding a negation to the verb contained in *p*. The answer given for *neg(p)* is a function of that which has been found for *p*: [8]

p	TRUE	FALSE	SOME	UNDEF
neg(p)	FALSE	TRUE	FALSE	UNDEF

Negation in queries

This table must be read from top to bottom only. Don't forget that it gives the truth value of the statement *neg(p)*, which contains one and only one negation, in function of that of *p*, which contains no negation. Note that the aim of this table is not to define the truth value of *neg(neg(p))*. In the case where *neg(p)* isn't valid, ignoring this restriction leads to attributing two truth values to *neg(neg(p))*.

Let us take as an example the following queried statement, which is the negative answer to (19):

(22) A cylinder does not contain cartridges.

KASSYS answers that statement (22) isn't valid since cylinders exist which contain cartridges, as is consistent with the hypothesis that queried statements are prefixed by a universal quantifier. Statement (22) is interpreted as *Not all cylinders contain cartridges*, and not as *There exist cylinders which do not contain cartridges*. In the latter case it is obviously impossible to answer, for the simple reason that nowhere is it said that there exist cylinders other than those which are parts of revolvers. However, in the hypothesis of a closed world, [9] objects whose existence has not been asserted are assumed not to exist in this world. Given this one proviso, the answer to (22) is negative, whatever its interpretation. On the contrary, a queried statement such as *It is not true that a cylinder does not contain cartridges*, which is the negation of (22), is rejected by the system for the simple reason that it contains two negations. The previous heuristic is not suitable for use in this statement, for which the values TRUE and SOME are possible.

4. A SESSION WITH KASSYS

The following session, given in French, is a faithful illustration of the messages given by the system.

1. KASSYS: Bonjour, j'attends vos instructions...
2. USER: Un aéronef est un appareil d'aviation.
3. K: OK, j'ai assimilé le concept <aéronef>.
4. U: Qu'est-ce qu'un aéronef ?
5. K: Un aéronef est un appareil d'aviation.
6. U: Une fusée est un appareil qui sert à envoyer une charge dans l'espace.
7. K: Le concept <appareil> a pour hyponyme <aéronef>; celui-ci est-il un hyperonyme plus proche de <fusée> ?
8. U: Oui.
9. K: Je déduis donc qu'une fusée est un aéronef qui sert à envoyer une charge dans l'espace. Etes-vous d'accord avec cette définition ?
10. U: Oui.
11. K: J'ai mémorisé la définition de <fusée>.
12. U: Une fusée est propulsée par un moteur à réaction.
13. K: S'agit-il d'une question ou d'une affirmation ?
14. U: C'est une affirmation.
15. K: Je sais déjà qu'une fusée est un aéronef qui sert à envoyer une charge dans l'espace. Dois-je intégrer la nouvelle donnée à cette définition ?
16. U: Oui.
17. K: Bien, j'ai compris votre définition de <fusée>.
18. U: Qu'est-ce qu'une fusée ?
19. K: Une fusée est un aéronef propulsé par un moteur à réaction, qui sert à envoyer une charge dans l'espace.
20. U: Un aéronef est un appareil qui peut voler.
21. K: Ok, j'ai assimilé la définition de <aéronef>.
22. U: Un planeur est un appareil semblable à l'avion mais démuni de moteur.
23. K: Le concept <appareil> a pour hyponyme <aéronef>; celui-ci est-il un hyperonyme plus proche de <planeur> ?
24. U: Oui.
25. K: Je déduis donc qu'un planeur est un aéronef semblable à l'avion mais démuni de moteur. Etes-vous d'accord avec cette définition ?
26. U: Oui.
27. K: J'ai mémorisé la définition de <planeur>.
28. U: Un aéronef peut envoyer une charge dans l'espace ?
29. K: Oui.
30. U: Pourquoi ?
31. K: Parce qu'une fusée est un aéronef propulsé par un moteur à réaction, qui sert à envoyer une charge dans l'espace.
32. U: Pourquoi ?
33. K: C'est vous qui me l'avez dit.
34. U: Est-ce que les appareils volent ?
35. K: Quelquefois.
36. U: Un planeur peut voler ?
37. K: Oui.
38. U: Est-ce qu'un planeur possède un moteur ?
39. K: Non.
40. U: On arrête là.
41. K: Ma mémoire n'est pas vide. Dois-je faire une sauvegarde ?
42. U: Non.
71. K: Comme vous voudrez. Merci et à bientôt.

5. CONCLUSION

taken from (Hirst 91); this statement is false because we have no dragon; its negation is equally false, for the same reason.

[8] UNDEF for *undefined*; SOME for *Sometimes*: some occurrences exist for which the queried statement is true.

[9] This hypothesis has not been adopted here.

This paper is a sketchy panorama of KASSYS. In particular, only the handling of hyperonymous definitions has been discussed. You have seen how the taxinomic hierarchy and the knowledge base are built. You have also been told how, broadly speaking, the query/answer module currently running in the system works.

The interest of this work is to show that conceptual graph theory offers an elegant framework in which hyperonymous definitions fit naturally. Careful and judicious use of this framework and the operations defined within it (type expansion, projection) enable information search algorithms to be implemented easily.

6. REFERENCES

(Byrd 87)
R.J. Byrd, N. Galzolari, M.S. Chodorow, J.L. Klavans, M.S. Neff, O.A. Rizky, Tools and methods for computational lexicology, Computational Linguistics, Vol. 13, Nb. 3-4, 1987, pp. 219-240.

(Hernert 93)
P. Hernert, Un système d'acquisition de définitions basé sur le modèle des graphes conceptuels, Thèse de Doctorat, Université Paris XIII, Villetaneuse, juin 1993.

(Hirst 91)
G. Hirst, Existence assumptions in knowledge representation, Artificial Intelligence, Vol. 49, No. 1-3, Elsevier, Amsterdam, 1991, pp. 199-242.

(Sowa 84)
J. Sowa, Conceptual structures - Information processing in mind and machine, Addison Wesley Publishing Company, Reading, Mass., 1984.

(Véronis 89)
J. Véronis, N.M. Ide, N. Wurbel, Extraction d'informations sémantiques dans les dictionnaires courants, Actes du 7ème Congrès Reconnaissances des Formes et Intelligence Artificielle, A.F.C.E.T., Paris, décembre 1989, pp. 1381-1395.

(Weinreich 70)
U. Weinreich, La définition lexicographique dans la sémantique descriptive, Langages, Didier-Larousse, Paris, 1970, pp. 69-86.