AN APPROACH TO CLASSIFY SYNONYMS IN A DICTIONARY OF VERBS

Awada Ali
Lebanese University
Faculty of sciences – section 1
Hadath - Lebanon

ABSTRACT

Several works in computational linguistics try to study the relationships among dictionary entries. These works consider the dictionary as a graph where words are represented by vertices and relationships between two words by an arrow between the corresponding vertices. Several kinds of relationships exist between two words such as synonymy, antonymy, hyperonymy, hyponymy …

We define the proximity of meaning between two words as the power of synonymy between them. In order to characterize this proximity, we focus our study on path length, circuits, and connected components in the dictionary graph. However, this study encounters frequently the polysemy problem that puts together different word acceptations. The aim of this paper is to solve this problem and proposes an algorithm to measure this proximity.

KEYWORDS

Synonymy, polysemy, meaning order, dictionary, weight, threshold.

1. INTRODUCTION

The synonymy has been widely studied by the scientific community who proposed different approaches based on the dictionaries [Lyons 90]. In general, these approaches represent the dictionary as a graph where vertices are the entries of the dictionary and arrows represent a relationship between two vertices: an arrow from a vertex A to a vertex B exists if B appears in A’s definition as a synonym in general. Therefore, the problem of the synonymy in a dictionary can be considered as a study of the semantic network represented by the graph. In general, these studies consist in detecting components with specific graph properties as the cliques [Ploux, Victorri 1998] and the gangs [Venant 2003]. These components’ detection leads to group synonyms. The content of a component corresponds in general to a “meaning”. However, the existence of polysemous words may create illegal relationships among vertices and thus, contradict the following assumption: “a component corresponds to only one meaning”.

Our concern in this paper is to study the synonymy by examining a graph of a French dictionary of verbs. We propose an approach that builds “meaning components” using synonymic and double antonymic relationships. A meaning component is a group of verbs having the same meaning given by an initial verb. However, the meaning of some verbs in the group may be closer to the initial verb than some others. The next step is to define a measure of closeness between the initial verb and all those in the group. We proposed a measure called the “synonymetry” in [Awada, Chebaro 2004] based on the “semantic proximity” of [Gaume et al.2002]. We defined a grouping criteria based on the “N-connexity” in order to eliminate metaphorical uses of synonyms called “metaphorymy” by [Duvignau et al. 2000].

The study proposed in this document aims to bring concrete answers to the previous problems. It proposes an algorithm to build different synonyms’ groups of an initial verb. We classified these groups using their closeness to initial verb. Each group has a synonymy order starting from one. We assign the smallest value to the closest group to the initial verb. The order is inversely proportional to the closeness to the initial verb.
2. OVERVIEW OF THE PROPOSED APPROACH

As we already specified it in the introduction, our aim is to study the synonymy in a French dictionary of verbs. The dictionary is an XML file that contains several information for each entry. We transform this file to a graph representing the dictionary’s verbs and keep only both synonymic and double antonymic relationships. These two kinds of relationships are represented in the graph by two types of arrows.

Our goal is to find for every verb, a family of synonyms and to classify them in lists of different orders. Thus, we introduce the "synonymy degree" as the resemblance between the synonyms. It corresponds to the quantification of the synonymy strength. Indeed, we apply the following method:

For an initial verb A, we start by building the list of the verbs appearing in its definition. This list is used to build the synonyms of order 1.

We calculate a weight for each verb in the list. The weight of a verb in the list depends on its relationships with the initial verb and with the other verbs of the list. Then we calculate a threshold of acceptance, according to which, every verb of the list will be accepted or rejected. The accepted verbs will be considered as order 1 synonyms of the initial verb.

The list of synonyms of each order is used to build the next order list. This list is initialized by the synonyms of the first order verbs. A calculation of weight is done for every verb of the list as well as a calculation of a second order threshold. The threshold allows selecting some of these verbs as the second order synonyms. This process is repeated until a given order.

3. THE POLYSEMY PROBLEM

A polysemous verb is a verb with several meanings. The existence of such verbs leads to meaning deviation problem. Thus, we can find among the synonyms of an initial verb some verbs with no relationship with the initial one. Indeed, a path may exist between two verbs of different meanings. Such path contains a polysemous verb (vertex) that joins these verbs. The figure 1 illustrates this problem:

We obtain two groups corresponding to direct synonyms of dress and erode. The result of the above association is the intersection of these two groups: the verb “to wear” (see figure 2).

The resulted ambiguity was treated by [Victorri and Fuchs 1996] who proposed a dynamic construction model of meaning. In this model, they associated a semantic space to each polysemous unit. The meaning of the unit in a given statement results from a dynamic interaction with the other linguistic units in the statement (the cotexte) [François, et al. 2003].
4. WEIGHT AND THRESHOLD

In order to solve the polysemy problem, we elaborate an approach that consists in starting from an initial verb and searching for all its direct and indirect synonyms. An indirect synonym is obtained transitively or using synonymic and/or double antonymic relationships (opposite of an opposite). Afterwards, we associate to each synonym a weight that corresponds to its proximity to the initial verb. The weight is obtained by examining the different paths between the initial verb and the synonym. Then, we use a tolerance threshold to keep synonyms having acceptable weight. The acceptable verbs are put in a group of synonyms with a given order of closeness. Non-acceptable synonyms will be examined later as candidates to enter a weaker (higher) order (farer from the initial verb).

The weight of a verb (or degree of proximity) is proportional to the power of its synonymy with the initial verb. Therefore, a “good” synonym is a synonym with a big weight. Note that a good synonym influences its descendants (direct synonyms) that acquire their weights according to the good synonym weight. Hence, they will have a great chance to be accepted as synonyms of the initial verb too.

The threshold plays the role of filter in the resolution of the polysemy problem. Its value must be chosen in order to eliminate bad synonyms obtained using paths that contain a polysemous verb.

5. THE SEARCH FOR THE DIFFERENT SYNONYMY ORDERS

We elaborated and experienced several strategies to find synonyms based on the graph properties. All these strategies are based on the attribution of a weight to the synonyms and the use of an acceptance/rejection threshold. A verb weight is used to calculate the weights of its direct synonyms. Thus, if B is a synonym of A, B’s weight is a function of A’s weight. In this document, we decided to present the most efficient strategy.

5.1 Description

Consider the following reflection: the antonyms of the antonyms are supposed to be potential synonyms. This kind of relationship can be considered as an enforcement of the synonymy one. Thus, the double antonymy is taken into account in the weight attribution. Note that the double antonymy is not a direct relationship and must then be considered as a weak one. We decided to use it only to strengthen the synonymy. Hence, we consider two kinds of arrows in the graph. An arrow exists from A to B if B is a synonym of A, or if B is a synonym and a double antonym of A. The second type of arrow is stronger than the first one and will allow attributing a higher weight to the related synonym.

In order to build the list, corresponding to the order, of synonymy, we process as follow:

- list, is initialized with the preceding order (i-1) verbs’ synonyms. A weight is then attributed to each verb in list, A verb weight reflects the relationship between this verb and all those in list, and list_{i-1}. It depends on the related verbs in the graph.

This solution suffers from the possible presence of circuits or strongly connected components in the graph. A strongly connected component is a sub-graph in which, for each couple of vertices A and B, there are a path p_1 from A to B and another path p_2 from B to A. It is obvious that the elements of a circuit are equivalent and that two circuits with a common element will constitute one strongly connected component. The propagation of weights in a circuit will lead to infinite loop and the resulting problem must be solved.

Finally, we note that the construction of the first level synonyms differs from the other levels construction. We made this distinction in order to respect the following constraint: the verbs being in the definition of the initial verb must be taken for its first order synonyms according to the structure of the dictionary.
5.2 Algorithm

5.2.1 Determination of the first order

The following function FirstOrder returns the set of all accepted first order synonyms of the verb V in the initial graph IG.

\[
\text{ListVerbs FirstOrder}(\text{graph IG, verb V})
\]

\[
\{
\text{ListVerbs } S_1, AS_1; /* direct synonyms & accepted direct synonyms */
\text{Graph RG; /* the reduced graph corresponding to IG */}
\text{int T; /* Threshold */}
\text{verb V;}
\}
\]

\[S_1 = \text{the set of direct synonyms in IG;}
\]

\[\text{AS}_1 = \emptyset; \]

\[\text{AffectWeightInitial}(S_1, IG); /* Affect weight to verbs in the initial graph */
\]

\[\text{RG} = \text{ReduceGraph}(\text{IG});
\]

\[\text{AffectWeightReduced}(\text{RG}); /* Affect weight to verbs in the reduced graph */
\]

\[\text{UpdateWeight}(\text{RG, IG}); /* Update weights of verbs in the initial graph */
\]

\[T = \text{average(weight(verbs of IG)); /* Threshold calculation */}
\]

\[\text{for each verb } V \text{ in } S_1
\]

\[\text{if (weight}(V) > T)
\]

\[\text{AS}_1 = \text{AS}_1 \cup \{V\};
\]

\[\text{return } \text{AS}_1
\]

The function AffectWeightInitial attributes an initial weight W to each direct synonym as follow:

- \(W = 60\) if \(V\) is a synonym of the initial verb.
- \(W = 70\) if \(V\) is a synonym and a double antonym of the initial verb.

The second type is obviously stronger than the first one. Note that the values "60" and "70" have been adopted after a certain number of tests and without being based on a precise rule.

ReduceGraph eliminates strongly connected components and replaces each one by a representative vertex. This step is necessary in order to avoid infinite loops during the propagation of weights from each verb to its synonyms. Note that all the vertices in a strongly connected component must have the same weight. The result of the graph reduction is a new graph without circuits nor strongly connected components. Each arrow in the initial graph outgoing from a vertex that belongs to a strongly connected component is represented in the reduced graph by an arrow outgoing from the component representative vertex, and so on for the incoming arrows.

AffectWeightReduced attributes an initial weight to each verb/vertex as follow: if the vertex represents a strongly connected component, its initial weight is the average of the strongly connected component members that are direct synonyms of the initial verb, otherwise, it keeps the same weight as in the initial graph. The final weight in the reduced graph is obtained by iterating the following treatment:

\[\text{for each arrow } A \text{ from a vertex } V_1 \text{ to a vertex } V_2
\]

\[\text{if (A represents a simple synonymy)}
\]

\[\text{weight}(V_2) = \text{weight}(V_1) \times (1 + 60%);
\]

\[\text{else /* A represents a synonymy enforced by a double antonymy */}
\]

\[\text{weight}(V_2) = \text{weight}(V_1) \times (1 + 70%);
\]

UpdateWeight goes back to the initial graph and attributes weights to its vertices as follow:

\[\text{for each verb } V \text{ in IG}
\]

\[\text{if (V belongs to a highly connected component H)}
\]

\[\text{NH} = \text{number of circuits in H};
\]

\[\text{NHV} = \text{number of circuits in H that contain V};
\]

\[\text{weight}(V) = \text{weight(representative of } V \text{ in H}) \times (1 + \text{NHV/NH});
\]

5.2.2 Determination of higher orders

The list \(S_i\) corresponding to the order \(i\) is initialized with the synonyms of the \((i-1)\) order verbs and the verbs rejected in this order. Two cases are taken into account for the calculation of the weight of \(S_i\) verbs:
A verb that has figured in an $S_i$ with $j < i$ without being admitted will be given its last weight if $j = i-1$. Otherwise ($j < i-1$), it will be given its last weight plus additional weight resulting from its relationships with verbs of orders greater than $j$.

The second case concerns verbs that did not appear in a preceding list. Their weight will be the sum of weights resulting from their relationships with the verbs of the preceding order.

As in the first order, the last step is the threshold calculation and the selection of acceptable synonyms $AS_i$.

5.3 A case study

Consider the graph of the figure 3 where the initial verb is $V$, and the bold arrows represent a simple synonymy and the dotted ones a synonymy with a double antonymy:

![Initial Graph and Corresponding Reduced Graph](image)

$S_1 = \{a, b, c\}$

- **AffectWeightInitial**: weight($a$) = 70, weight($b$) = 60, weight($c$) = 60
- **AffectWeightReduced**: In the reduced graph, we have:
  - weight($b$) = 60
  - weight($ade$) = average(weight($a$)) + 70% * weight($b$) = 112
  - weight($cfg$) = average(weight($c$)) + 2 * 60% * weight($b$) = 132
- **UpdateWeight**: Finally, in the initial graph, we have:
  - weight($a$) = weight($d$) = weight($ade$) * (1 + NHV / NH) = 112 * (1 + 2/2) = 224
  - weight($e$) = weight($ade$) * (1 + NHV / NH) = 112 * (1 + ½) = 168
  - weight($b$) = 60
  - weight($c$) = weight($f$) = weight($cfg$) * (1 + NHV / NH) = 132 * (1 + 1/1) = 264

The order1 threshold = average(weight($a$), weight($b$), weight($c$)) = (224+60+264)/3 = 182.67

⇒ $AS_1 = \{a, c\}$ is the list of strong order1 synonyms while $\{b\}$ is considered as a weak order1 synonym.

$S_2 = \{\text{synonyms of } AS_1 \text{ verbs}\} = \{d, f\}$

- weight($d$) += 60% * weight($a$) = 358.4
- weight($f$) += 60% * weight($b$) + 60% * weight($c$) = 458.4

The order2 threshold = average(weight($d$), weight($f$)) = 408.4

⇒ $AS_2 = \{f\}$ is the order2 synonyms while $\{d\}$ is rejected.

$S_3 = \{\text{synonyms of } AS_2 \text{ verbs}\} \cup \{S_2 \text{ rejected verbs}\} = \{g\} \cup \{d\}$

- weight($d$) = 358.4, weight($g$) += 60% * weight($f$) = 539.04

The order3 threshold = (358.4+539.04)/2 = 448.72,

⇒ $AS_3 = \{g\}$ is then accepted while $\{d\}$ is rejected.

Finally, $AS_4 = \{d\}$ will be considered as an order4 synonym and $\{e\}$ as an order5 synonym.

5.4 Justification of the algorithm

In the preceding algorithm, we tried to give precedence to verbs having many relationships with other verbs in the dictionary. A verb with several incoming arrows will then cumulate a percentage of the related verbs’
weights. This way allows eliminating synonyms obtained by a path that contains a polysemous verb because
an invalid synonym will naturally have few relationships with the initial verb environment.

Note that the weight of a verb depends on the order where the verb is considered. Thus a verb with a
weight \( w_i \) may be accepted in an order \( i \) and another verb with a higher weight \( w_j \) \( (w_j > w_i) \)
may be rejected in a higher order \( j \) \( (j > i) \). There are no negative effects in such consideration because the threshold is calculated
in each order as the average of the order’s verbs.

Furthermore, the weight attribution in a strongly connected component varies from an element to another.
The weight of a verb does not depend only on its representative weight in the reduced graph. It depends also
on the number of circuits that traverse the corresponding vertex inside the strongly connected component. We
made this choice in order to strengthen the highly connected verbs.

Finally, the inclusion of rejected verbs in higher orders’ initial list gives a second chance to these verbs
and allows them to be synonyms of higher order (weaker synonyms).

6. CONCLUSION

We tried in this paper to study the synonymy in a dictionary and to propose an algorithm to measure the
closeness of synonyms to an initial verb. We wrote a software that implemented the algorithm and allowed
consulting a dictionary and classifying the synonyms of an initial verb into groups of different closeness.

The obtained results are quite satisfactory. However, the interface revealed some problems resulting
mainly from the construction of the dictionary. Indeed, the construction of the dictionary presents some
deficiencies. Several verbs such as make, have ... have meanings that depend on the noun phrase that follows
in the sentence. The dictionary on which we worked takes into account only verbs and do not associate any
noun phrase. These verbs become thus polysemous and pose a lot of problems when developing the different
synonymy orders.

We have taken into consideration only the synonymic and the double antonymic relationships. Other
relationships may be studied and confronted with our approach. On the other hand, the values chosen for both
kinds of relationships were not validated by a theoretical study. We have chosen the weight, the coefficient
and the threshold values by experimentation. We believe that a good choice of these values is crucial because
our approach depends highly on them. A validation study is then necessary and will surely base our approach
on solid foundations.

REFERENCES

Awada, A. and Chebaro, B. 2004, Etude de la synonymie par l’extraction de composantes N-connexes dans les graphes

Généraux du Programme de REcherches en Sciences Cognitives de Toulouse (PRESOCOT), Toulouse, France.

François, J. et al., 2003, La réduction de la polysémie adjectivale en cotexte nominal : une méthode de sémantique
Cahier du Crisco no 14, Université de Caen, France.


Le Blanc, B. et al., 2001, Constitution et visualisation de deux réseaux d'associations verbales. 2nd Colloque sur Agents
Logiciels, Coopération, Apprentissage et Activité humaine (ALCAA), pp. 37-43.

France.


Manguin, J.L. and Victorri, B., 1999, Représentation géométrique d’un paradigme lexical. Conférence TALN 1999,
Cargèse.

Ploux, S. and Victorri, B., 1998, Construction d’espaces sémantiques à l’aide de dictionnaires de synonymes. TAL, 39,
n°1, pp. 161-182.
