A SPANISH e-DICTIONARY OF SYNONYMS AS A FUZZY TOOL FOR INFORMATION RETRIEVAL

Santiago Fernández Lanza
Depto. de Lógica y Filosofía Moral
Univ. de Santiago de Compostela
Campus Sur s/n
15782 - Santiago de Compostela
Spain
sflanza@usc.es

Jorge Graña Gil
Depto. de Computación
Univ. de La Coruña
Campus de Elviña s/n
15071 - La Coruña
Spain
grana@udc.es

Alejandro Sobrino Cerdeiriña
Depto. de Lógica y Filosofía Moral
Univ. de Santiago de Compostela
Campus Sur s/n
15782 - Santiago de Compostela
Spain
lflgalex@usc.es

Abstract

We start by analyzing the role of imprecision in information retrieval in the Web, some theoretical contributions for managing this problem and its presence in search engines, with special emphasis on the use of thesaurus in order to increase the relevance of the documents retrieved. We then present a Spanish electronic dictionary of synonyms that compute degrees of synonymy, and an efficient implementation of it by using deterministic acyclic finite-state automata. We conclude by conjecturing that the use of this e-dictionary in a Spanish web searcher will increase precision and recall without diminishing latency.

Keywords: Dictionary of synonyms, finite-state automata, fuzzy information retrieval.

1 IMPRECISION AND INFORMATION RETRIEVAL ON THE WEB

Information retrieval is perhaps as old as the existence of libraries, institutions where information is stored to be consulted. In order to improve the efficiency of these consultations, librarians classify the information using some form of indexation system (alphabetical index of authors, subject index, etc.), which makes it quick and easy to access the documents.

At present, information retrieval is automatic and this is largely due to the success of computer technology. Computers have mode digital libraries possible, where information is stored in electronic devices. In these new libraries, information is not always managed by the normative criteria of librarians. Perhaps the largest digital library that exists at the moment is the Web, which has an enormous amount of documents in every possible style or format. The Dublin Core metadata suggest that every web page should define tags relative to its form and content, but this initiative has not met with universal success. Moreover, in the Web a lot of redundant (the number of repeated pages is estimated at 20% of the whole), false and out-of-date information is stored. Consequently, there are a lot of data, but finding useful and interesting information is quite a complicated task.

In order to help in this task, Web searchers appeared. They belong to two main different classes, *directories* or *search engines*, although today both of them provide mixed services. For example, YAHOO! offers the search engine of GOOGLE and GOOGLE provides access to the OPEN DIRECTORY PROJECT classification.

In the Web, the most commonly used search process is a lexical-grammatical one, based on the possible matching between the terms of a query and some word of the index in the database of the searcher, which is linked to a document. Moreover, there are other models: the logical one, for which retrieval is a synonym of infer; or the cognitive, in which retrieval is the simulation of the behaviour of a human agent searching for information. What follows refers only to lexical model.

There are three main ways of matching in the lexical paradigm: exact, vector space and probabilistic models. Exact matching is the most common and widely implemented in the Web searchers, because it is simple and offers reasonably good results. In exact matching a query is reduced to a set of terms, the document to a set of keywords from the index and the matching is the identity between a query term and an index term. But the relevance of a page rescued as the answer to a query is not always a matter of yes or no. If it were, it would give very poor results. In most cases, it is a question of degree, largely due to the uncertainty present in the query or in the document. In queries, not all the terms may have the same weight when it comes to expressing what we are searching for.

In the index, not all the words represent the document with equal strength. According to this, other kinds of matching functions have been proposed: the vector model and the probabilistic model are the best-known ones. They represent good theoretical improvements, but are criticized: estimations of probabilities about the order of documents in a way that will come close to user judgments about relevance are often based on the absence of any examples of relevant documents.

It should be noted that the exact matching model does not exclude the treatment of degrees of relevance, only its semantic limits to do so. An extended boolean model makes it possible to use it as an approximate concept, not an exact one. Of all the extensions of the boolean model, the fuzzy model has obtained some credit [6]. A fuzzy model uses a generalized membership function $F(d_k, t_j)$ representing the set of documents described by an index. Given a query with i terms, it is possible to order those documents with respect to the query by combining the membership values of the individual terms. The most popular modality of fuzzy logic that has been used is the max-min logic:

$$\begin{aligned} F(d_k, t_1 \wedge t_2) &= \min(F(d_k, t_1), F(d_k, t_2)) \\ F(d_k, t_1 \vee t_2) &= \max(F(d_k, t_1), F(d_k, t_2)) \\ F(d_k, \neg t_1) &= 1 - F(d_k, t_1) \end{aligned}$$

If the membership function is in $\{0, 1\}$, the fuzzy model is equivalent to a boolean model.

Some objections have been made to fuzzy models: they do not offer a criterion for assigning weights to the query terms, and it is possible to classify documents with the same ranking using either many or only a few query terms. Trillas' studies in [5] on:

- the formalization of *Black's consistence profiles* with fuzzy logic techniques, allowing us to approximately measure the role of each word in a query,
- and the study of *t-norm and t-conorm families* that allow us to aggregate terms of fuzzy meaning, whilst still respecting their semantics,

are solutions to these objections and contribute to improving the fuzzy model.

The evaluation of a retrieval system is based on three parameters:

- Precision: ratio of relevant documents in the set of retrieved documents.
- Recall: ratio of retrieved documents in the set of relevant documents.
- Latency: speed and scalability of retrieval.

The introduction of new kinds of matching modelling leads to an improvement in precision and recall ratios, but we must take care to test these models with realistic collections of documents and to ensure that latency does not decrease, at least if we want to implement it in a real searcher. These are the main problems of fuzzy models. Latency is a critical factor since everything that delays a search for more than one second must be rejected.

Even though real searchers are based on non-extended boolean matching models whose semantics do not admit imprecision, they use predefined resources that introduce certain fuzziness or generality in the queries. Thus, most of them include proximity operators, such as *near*, which retrieve pages in which the proposed terms occur more or less close together (about 10 or 20 words may appear between them). Another form of query expansion is *stemming* (searching for words from their prefix: `impli/implication`, `implicate`, `implicit`, `implicitly`, `implied`) using *wildcards* (character used for substitution of one or several letters: `impli*`).

The *near* operator and *wildcards* increase recall but decrease precision. In order to increase recall and precision, thesaurus have been used in some lexical models:

- In order to increase recall: Some searchers (such as ALTAVISTA) expand the query by using a thesaurus, so asking about `domestic violence` is also asking about `home violence`, `domestic aggression`, etc. The effect of this is the loss of precision in the answer due to the increase in the number of retrieved pages, although some pages retrieved by the use of a synonym could be more relevant than other pages retrieved by the original term.
- In order to improve precision: Suppose that an excessively generic query has been made, thereby retrieving an excessive number of pages. These pages would have been retrieved by the matching of some words in the index. Applying a dictionary of synonyms, pages with similar meaning or subject could be grouped and consequently classified with the same order number. This method has been used by EXCITE.

Up till now the use of dictionaries of synonyms in information retrieval has been limited to its linguistic resources for associating similar meanings. This has provided an improvement in the search process, but it is possible to go one step further by measuring the proximity of meaning between a term and its synonym through similarity measures. This will enable us to offer a calculation of the degree of synonymy between the entry and the synonyms in a dictionary of synonyms. We will now present an implementation of a Spanish dictionary of synonyms, which calculates the degree of synonymy between two words. We have carried out this implementation by using minimal acyclic finite-state automata. The use of this kind of automata turns the dictionary of synonyms into a quick and efficient tool. It is plausible to conjecture that this will decrease latency and increase precision and recall. The next step will be to implement it in an information retrieval system and to evaluate the improvement of the system.

Section 2 gives the definition of synonymy and specifies how to calculate the degree of synonymy between two entries of the dictionary. Section 3 describes our general model of dictionary and allows us to understand the role of the finite-state automata here. In Sect. 4, we describe Blecua's Spanish dictionary of synonyms [1] and detail all the transformations performed on it with the help of our automata-based architecture for dictionaries. Finally, Sect. 5 presents our conclusions.

2 SYNONYMY

The most frequent definition of synonymy conceives it as a relation between two expressions with identical or similar meaning. The controversy of understanding synonymy as a precise question or as an approximate question, i.e. as a question of identity or as a question of similarity, has existed from the beginning of the study of this semantic relation. In the present work, synonymy is understood as a gradual relation between words. In order to calculate the degree of synonymy, we use measures of similarity applied on the sets of synonyms provided by a dictionary of synonyms for each of its entries. In the examples shown in this work, we will use as our measure of similarity *Jaccard's coefficient*, which is defined as follows. Given two sets X and Y, their *similarity* is measured as:

$$sm(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

On the other hand, let us consider a word w with m_i possible meanings, where $1 \leq i \leq M$, and another word w' with m_j possible meanings, where $1 \leq j \leq M'$. By $dc(w, m_i)$, we will represent the function that gives us the set of synonyms provided by the dictionary for every entry w in the concrete meaning m_i. Then, the degree of synonymy of w and w' in the meaning m_i of w is calculated as follows [2]:

$$dg(w, m_i, w') = \max_{1 \leq j \leq M'} sm[dc(w, m_i), dc(w', m_j)]$$

Furthermore, by calculating

$$k = \arg \max_{1 \leq j \leq M'} sm[dc(w, m_i), dc(w', m_j)]$$

we obtain in m_k the meaning of w' closest to the meaning m_i of w.

The conception of synonymy as a gradual relation implies a distancing from the idea that considers it as an equivalence relation. This is coherent with the behaviour of synonymy in the printed dictionary, since it is possible to find cases in which the reflexive, symmetrical and transitive properties do not hold:

- The reflexive relation is not usually included in dictionaries in order to reduce the size of the corresponding implementations, since it is obvious that any word is a synonym of itself in each one of its individual meanings.
- The lack of symmetry can be due to several factors. In certain cases, the relation between two words can not be considered as one of synonymy. This is the case of the words `granito` (*granite*) and `piedra` (*stone*), where the relation is a hyponymy. This phenomenon also occurs with some expressions: for instance, the expression `ser uña y carne` (*to be inseparable* or, in literal translation, *to be nail and flesh*) and the word `uña` (*nail*). In other cases, symmetry is not present because a word can have a synonym which is not an entry in the dictionary. One reason for this is that the lemmas of the words are not used when these words are provided as synonyms. Another possible reason is an omission by the lexicographer who compiled the dictionary.
- Finally, if synonymy has been understood as similarity of meanings, it is reasonable that transitivity does not always hold.

In the following section, we will describe a general architecture that uses minimal deterministic acyclic finite-state automata in order to implement large dictionaries of synonyms, and how this general architecture has allowed us to modify an initial dictionary with the purpose of letting the relations between the entries and the expressions provided as answers satisfy the reflexive and symmetrical properties, but not the transitive one.

3 GENERAL ARCHITECTURE OF A DICTIONARY OF SYNONYMS

The use of finite-state automata to implement efficient dictionaries is a well-established technique. The main reasons for compressing a very large dictionary of words into a finite-state automaton are that its representation of the set of words is compact, and that the process of looking up a word in the dictionary is proportional to the length of the word, and therefore very fast. Of particular interest for natural language processing applications are minimal acyclic finite-state automata, which recognize finite sets of words, and which can be constructed in various ways [3, 7]. The aim of this section is to build a general architecture to handle a large Spanish dictionary of synonyms [1].

Words in a dictionary of synonyms are manually inserted by linguists. Therefore, our first view of a dictionary is simply a text file, with the following line format:

```
word   meaning   homograph   synonym
```

Words with several meanings, homographs or synonyms use a different line for each possible relation. With no loss of generality, these relations could be alphabetically ordered. Then, in the case of Blecua's dictionary, the point at which the word `concesión` (*concession*) appears could have this aspect:

concesión	1	1	gracia	(*grace*)
concesión	1	1	licencia	(*licence*)
concesión	1	1	permiso	(*permission*)
concesión	1	1	privilegio	(*privilege*)
concesión	2	1	epítrope	(*a figure of diction*)

For a later discussion, we say that the initial version of the dictionary had $M = 27,029$ different words, with $R = 87,762$ possible synonymy relations. This last number is precisely the number of lines in the text file. The first relation of `concesión` appears in the line $25,312$, but the word takes the position $6,419$ in the set of the M different words ordered lexicographically.

Of course, this is not an operative version for a dictionary. Therefore, what is important now is to provide a compiled version to compact this large amount of data, and also to guarantee an efficient access to it with the help of automata. The compiled version is shown in Fig. 1, and its main elements are:

- The `Word_to_Index` function (explained later) changes a word into its relative position in the set of different words (e.g. `concesión` into $6,419$).
- In a *mapping* array of size $M + 1$, this number is changed into the absolute position of the word (e.g. $6,419$ into $25,312$). This new number is used to access the rest of arrays, all of them of size R. The lexicographical ordering guarantees that the relations of a given word are adjacent, but we need to know how many they are. For this, it is enough to subtract the absolute position of the word from the value of the next cell (e.g. $25,317 - 25,312 = 5$ relations).
- The arrays *m1* and *h1* store numbers which represent the meanings and homographs, respectively, of a given word. The arrays *m2* and *h2* have the same purpose for each of its synonyms.
- The array *w2* is devoted to synonyms and also stores numbers. A synonym is a word that also has to appear in the dictionary. The number obtained by the `Word_to_Index` function for this word is the number stored here, since it is more compact than the synonym itself. The original synonym can be recovered by the `Index_to_Word` function (explained later).
- The array *dg* directly stores the degrees of every possible synonymy relation. In this case, no reduction is possible.

Note that the arrays *m2*, *h2* and *dg* store data that are not present in the original version of the dictionary. This new information was easily calculated from the rest of arrays with the formulas explained in Sect. 2, once the dictionary had been compiled into this general model and those initial data could be efficiently accessed. The specific transformations performed on the initial dictionary, including this one, are detailed in Sect. 4.

This is the most compact architecture for storing all the information of the words present in a dictionary, when this information involves specific features of each word, such as the degree of a synonymy relation. Furthermore, this architecture is very flexible: it is easy to incorporate new arrays for other additional data (such as part-of-speech tags), or to remove the unused ones (saving the corresponding space). To complete this model, we only need the implementation of `Word_to_Index` and `Index_to_Word`. Both functions operate over a special type of automata, the numbered minimal acyclic finite-state automata described in [3].

4 IMPROVING THE DICTIONARY

The implementation of the dictionary of synonyms [1] was carried out in several steps, some of which required manual processes whereas others could be made automatically. The initial version of the dictionary had 21,098 entries; however it also included 5,931 expressions that appear as synonyms of others but were not in themselves entries (from now, no-entries). This version had 87,762 pairs of synonyms and our first goal was to fill the information corresponding to the *m2*,

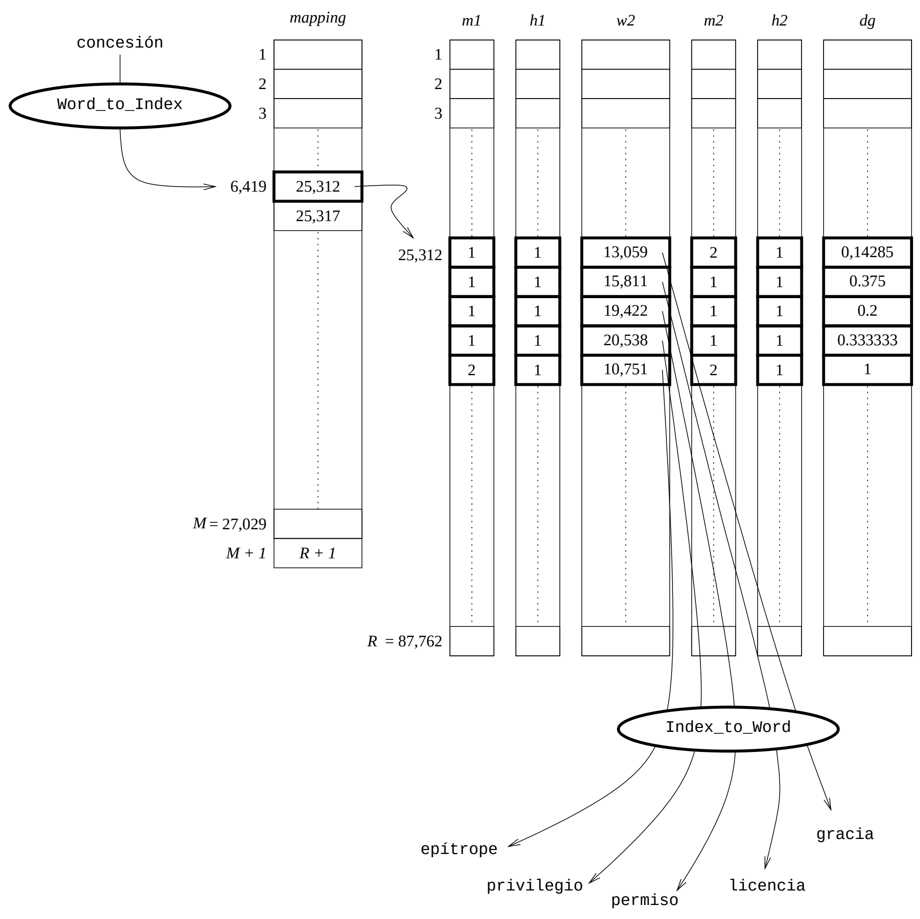

Figure 1: Compact modeling of a dictionary of synonyms

h2 and *dg* arrays described in Sect. 3. With respect to *dg* and *m2*, this could be done mechanically by using the formulas of Sect. 2. The automatic detection of homographs *h2* was carried out by including all the homographs of the second word in the calculation of the degree, but only 300 entries of dictionary proved to have homographs. From these initial data we made further modifications related to the properties that the synonymy relation satisfies in the dictionary:

- Symmetry: One of the reasons why symmetry does not hold is the existence of 11,596 pairs involving no-entries. This means that there exist pairs of the form (entry, no-entry) but not the converse pairs. The next improvement was to add as entries all no-entry expressions. In order to do so, we had to associate a set of synonyms to each no-entry. This set was made up of all entries in which the no-entry expression appears as a synonym. In this way, the number of entries and the number of pairs were increased to 27,029 and 99,358 respectively. At this moment, all the expressions involved in the dictionary appear as entries. Nevertheless, the synonymy relation is still non-symmetric. Since we use a measure of similarity, in this case Jaccard's coefficient, two meanings of two different entries will be non zero synonyms (i.e. will be synonyms) if its associated sets of synonyms have some element in common. Following this criterion, if an entry x has synonyms in common with another y given two respective meanings of them, y will have the same synonyms in common with x for those meanings. We have improved the dictionary again by adding to each set of synonyms X the new entries that had meanings whose associated sets of synonyms had elements in common with X. This step does not modify the number of entries that the dictionary had by then but the number of pairs is modified increasing to 621,265. We obtained a symmetric relation of synonymy and we transformed the initial dictionary into a richer one.

- Reflexivity: The final improvement was to incorporate the reflexive pairs in the synonymy relation by adding for each entry of the dictionary the entry word itself in all the corresponding sets for each meaning of it. This is useful in order to avoid some problems in the calculation of the degrees of synonymy. For instance, when a word x appears as a unique synonym of another y and y as a unique synonym of x for two specific meanings of them, the corresponding sets of these meanings have no elements in common. In this case, the degree of synonymy is 0; therefore x and y will not be considered as synonyms, which is not very intuitive. By adding the reflexive case in the sets of synonyms we avoid this problem. After this modification the number of pairs is 655,583 and the relation is now reflexive and symmetric.

- Transitivity: Since the criterion followed indicates that two entries are synonyms if their corresponding sets of synonyms have elements in common, it is reasonable to think that the synonymy relation is not necessarily a transitive one. This is because, in general, from the fact that a set of synonyms X has elements in common with Y and Y has elements in common with Z it can not be inferred that X has elements in common with Z. Although there exist some dictionaries of synonyms whose synonymy relation is transitive, the dictionary we have used includes a considerable number of examples showing the non-existence of this property.

With regard to the time figures involved in this final configuration of Blecua's dictionary, the time needed to build the automaton (27,029 words, 27,049 states and 49,239 transitions) is 2.85 seconds, in a Pentium II 300 MHz. under Linux operating system. It is also necessary an extra 16.90 seconds to incorporate the information regarding meanings, homographs, synonyms and degrees, thus giving us a total compilation time of 19.75 seconds. Finally, it should be noted that the recognition speed of our automata is around 80,000 words per second. This figure makes it possible to access the information very rapidly, this proving the correct adaptation of our general architecture for both the processes of improvement and the use of this Spanish dictionary of synonyms.

5 CONCLUSIONS

We have shown an electronic implementation of a Spanish dictionary of synonyms which manages degrees of synonymy using deterministic acyclic finite-state automata. The implementation with automata turns it into a quick and effective tool. This allows us to conjecture that the use of the e-dictionary in a Spanish searcher will lead to an improvement in recall and precision parameters, with no negative effect on latency. It is the first Spanish e-dictionary of synonyms and a pioneer in the task of calculating the degree of synonymy.

The next step will be to integrate this dictionary of synonyms into a searcher, and to measure the improvements produced in the above-mentioned parameters. The use of synonymy relations in the task of automatic query expansion is not a new subject, but the approaches presented until now do not assign a weight to the degree of synonymy that exists between the original terms present in the query and those produced by the process of expansion [4]. In spite of this limitation, the results reported show potential improvements in the coverage, since they retrieve relevant documents which do not contain any of the terms present in the query, particularly in the case of short and incomplete queries. It therefore seems reasonable to think that the use of more sophisticated synonymy relations will allow us to obtain further improvements.

References

[1] Blecua Perdices, J.M. (dir.) (1997) *Diccionario Avanzado de Sinónimos y Antónimos de la Lengua Española.* Vox.

[2] Fernández Lanza, S. (2001). *Una contribución al procesamiento automático de la sinonimia utilizando Prolog.* Publicaciones de la Univ. de Santiago de Compostela (Tesis Doctoral en CD-ROM).

[3] Graña Gil, J.; Barcala Rodríguez, F. M.; Alonso Pardo, M. (2001). Compilation methods of minimal acyclic finite-state automata for large dictionaries. In *Proc. of the Sixth International Conference on Implementation and Application of Automata (CIAA-2001)*, Pretoria, South Africa, pp. 116-129.

[4] Greenberg, J. (2001). Automatic query expansion via lexical-semantic relationships. *Journal of the American Society for Information Science and Technology*, vol. 52(5), pp. 402-415.

[5] Trillas, E.; Alsina, C. (1999). A reflection on what is a membership function. *Mathware & Soft Computing*, vol. 6, pp. 201-215.

[6] Turtle, H.R.; Croft, W.B. (1997). Uncertainty in Information Retrieval Systems. In Motro, A. & Smets, P. (eds.), *Uncertainty Management Information Systems*, Kluwer.

[7] Watson, B.W. (1993). *A Taxonomy of Finite Automata Construction Algorithms.* Computing Science Note 93/43, Eindhoven University of Technology, The Netherlands.