Automatic extraction of synonyms in a dictionary

Vincent D. Blondel
Division of Applied Mathematics
University of Louvain
Avenue Georges Lemaitre, 4
B-1348 Louvain-la-Neuve, Belgium
Email: blondel@inma.ucl.ac.be

Pierre P. Senellart
Computer Science Department
École normale supérieure
45 rue d'Ulm
F-75230 Paris cedex 05, France
Email: Pierre.Senellart@ens.fr

Abstract

We propose a method for automatic synonym extraction in a dictionary. Our method is based on an algorithm that computes similarity measures between vertices in graphs. This algorithm can be thought of as a generalization of Kleinberg's web search algorithm to structure graphs that are more general than the hub-authority graph used by Kleinberg. We use the 1913 Webster's Dictionary and apply our method on four synonym queries. The results obtained are analyzed and compared to those obtained with two other methods.

1 Introduction

In this paper, we propose a method for automatic synonym extraction in a dictionary (our goal is not exactly the same as that of automatic thesaurus generation techniques: for a given word, we only want a list of good synonyms or near-synonyms). Our method uses a graph constructed from the dictionary and is based on the assumption that synonyms have many words in common in their definitions and are used in the definition of many common words. Our method is based on an algorithm that generalizes an algorithm initially proposed by Kleinberg for searching the web [8].

Starting from a dictionary, we first construct the associated *dictionary graph* G; each word of the dictionary is a vertex of the graph and there is an edge from u to v if v appears in the definition of u. Then, associated to a given query word w, we construct a *neighborhood graph* G_w which is the subgraph of G whose vertices are those pointed by w or pointing to w. Finally, we look in the graph G_w for vertices that are similar to the vertex 2 in the structure graph

$$1 \longrightarrow 2 \longrightarrow 3$$

and choose these as synonyms. For this last step we use a similarity measure between vertices in graphs that was introduced in [3, 6] and is described in Section 2.

The problem of searching synonyms is similar to that of searching similar pages on the web; a problem that is dealt with in [8] and [5]. In these references, similar pages are found by searching authoritative pages in a subgraph focused on the original page. Authoritative pages are pages that are similar to the vertex "authority" in the structure graph

$$\text{hub} \longrightarrow \text{authority}.$$

We ran the same method on the dictionary graph and obtained lists of good hubs and good authorities of the neighborhood graph. There were duplicates in these lists but not all good synonyms were duplicated. Neither authorities nor hubs appear to be the right concepts for discovering synonyms.

In the next section, we describe our method in some detail. In Section 3, we briefly survey two other methods that will be used for comparison. We then describe in Section 4 how we have constructed a dictionary graph from the Webster's dictionary. In a last section we compare all methods on the following words chosen for their variety: **disappear**, **parallelogram**, **sugar** and **science**.

2 A generalization of Kleinberg's method

In [8], Jon Kleinberg proposes a method for identifying web pages that are good *hubs* or good *authorities* for a given query. For example, for the query "automobile makers", the home pages of

Ford, Toyota and other car makers are good authorities, whereas web pages that list these home pages are good hubs. In order to identify hubs and authorities, Kleinberg's methods exploits the natural graph structure of the web in which each web page is a vertex and there is an edge from vertex a to vertex b if page a points to page b. Associated to any given query word w, the method first constructs a "focused subgraph" G_w analogous to our neighborhood graph and then computes hub and authority scores for all vertices of G_w. These scores are obtained as the result of a converging iterative process. Initial hub and authority weights are all set to one, $x^1 = 1$ and $x^2 = 1$. These initial weights are then updated simultaneously according to a mutually reinforcing rule: the hub scores of the vertex i, x_i^1, is set equal to the sum of the authority scores of all vertices pointed by i and, similarly, the authority scores of the vertex j, x_j^2, is set equal to the sum of the hub scores of all vertices pointing to j. Let M_w be the adjacency matrix associated to G_w. The updating equations can be written as

$$\begin{pmatrix} x^1 \\ x^2 \end{pmatrix}_{t+1} = \begin{pmatrix} 0 & M_w \\ M_w^T & 0 \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \end{pmatrix}_t$$

$$t = 0, 1, \ldots$$

It can be shown that under weak conditions the normalized vector x^1 (respectively, x^2) converges to the normalized principal eigenvector of $M_w M_w^T$ (respectively, $M_w^T M_w$).

The authority score of a vertex v in a graph G can be seen as a similarity measure between v in G and vertex 2 in the graph

$$1 \longrightarrow 2.$$

Similarly, the hub score of v can be seen as a measure of similarity between v in G and vertex 1 in the same structure graph. As presented in [3, 6], this measure of similarity can be generalized to graphs that are different from the authority-hub structure graph. We describe below an extension of the method to a structure graph with three vertices and illustrate an application of this extension to synonym extraction. We then briefly describe the situation for arbitrary structure graphs.

Let G be a dictionary graph. The neighborhood graph of a word w is constructed with the words that appear in the definition of w and those that use w in their definition. Because of this, the word w in G_w is similar to the vertex 2 in the structure graph (denoted P_3)

$$1 \longrightarrow 2 \longrightarrow 3.$$

For instance, Figure 1 shows a part of the neighborhood graph of **likely**. The words **probable** and **likely** in the neighborhood graph are similar to the vertex 2 in P_3. The words **truthy** and **belief** are similar to, respectively, vertices 1 and 3. We say that a vertex look like the vertex 2 of the preceding graph if it points to vertices looking like the vertex 3 and if it is pointed to by vertices looking like the vertex 1. This mutually reinforcing definition is very similar to Kleinberg's definitions of hubs and authorities.

The similarity between vertices in graphs can be computed as follows. To every vertex i of G_w we associate three scores (as many scores as there are vertices in the structure graph) x_i^1, x_i^2 and x_i^3 and initially set them equal to one. We then iteratively update the scores according to the following mutually reinforcing rule: the scores x_i^1 are set equal to the sum of the scores x_j^2 of all vertices j pointed by i; the scores x_i^2 are set equal to the sum of the scores x_j^3 of vertices pointed by i and the scores x_j^1 of vertices pointing to i; finally, the scores x_i^3 are set equal to the sum of the scores x_j^2 of vertices pointing to i. At each step, the scores are updated simultaneously and are subsequently normalized; $x^k \leftarrow x^k / \|x^k\|$ $(k = 1, 2, 3)$. It can be shown that when this process converges, the normalized vector score x^2 converges to the normalized principal eigenvector of the matrix $M_w M_w^T + M_w^T M_w$. Thus, our list of synonyms can be obtained by ranking in decreasing order the entries of the principal eigenvalue of $M_w M_w^T + M_w^T M_w$.

The algorithm described above can be generalized to arbitrary structure graphs. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two given graphs. We think of G_2 as a "structure graph" and wish to quantify the similarity between the vertex v_1 in G_1 and the vertex v_2 in G_2. It is shown in [3, 6] how similarity scores can be obtained from the principal eigenvector of the matrix

$$M_1 \otimes M_2 + M_1^T \otimes M_2^T$$

where $\otimes$ denotes the Kronecker tensorial product, and M_1 and M_2 are the adjacency matrices associated to G_1 and G_2. When

$$M_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

we obtain Kleinberg's algorithm, when

$$M_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

we obtain the algorithm described above for synonym extraction.

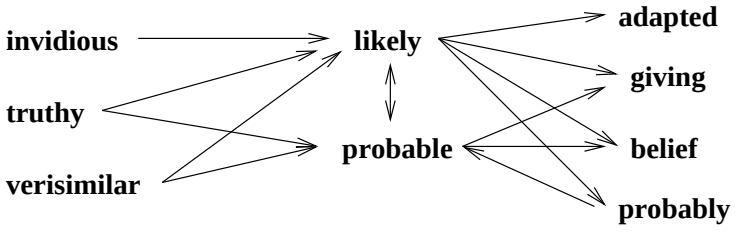

Figure 1: Subgraph of the neighborhood graph of **likely**

3 Other methods

In this section, we briefly describe two synonym extraction methods that will be compared to our method on a selection of 4 words.

3.1 The distance method

One possible way of defining a synonym distance is to declare that two words are close from being synonyms if they appear in the definition of the same words and have the same words in their definition. A way of formalizing this is to define a distance between two words by counting the number of words that appear in one of the definitions but not in both, and add to this the number of words that use one of the words but not both of them in their definition. Let A be the adjacency matrix of the dictionary graph, and i and j be the vertices associated to two words. The distance between i and j can be expressed as

$$d(i,j) = \|(A_{i,\cdot} - A_{j,\cdot})\| + \|(A_{\cdot,i} - A_{\cdot,j})^T\|$$

where $\|\cdot\|$ is the l_1 vector norm (sum of all entries magnitudes). For a given word i we may compute $d(i,j)$ for all j and sort the words according to increasing distance.

Unlike the other methods presented in this paper, we can apply this algorithm directly to the entire dictionary graph rather than on the neighborhood graph. This does however give very bad results: the first two synonyms of **sugar** are **pigwidgeon** and **ivoride**. We will see in Section 5 that much better results are achieved if we use the neighborhood graph.

3.2 ArcRank

ArcRank is a method introduced by Jan Jannink and Gio Wiederhold for building a thesaurus [7]; their intent was not to find synonyms but related words. We did not implement their method but have used instead the online version available at `http://skeptic.stanford.edu/data/` (this online version also uses the 1913 Webster's Dictionary and the comparison with our results is therefore meaningful).

The method is based on the PageRank algorithm, used by the web search engine Google and described in [4]. PageRank assigns a ranking to each vertex of the dictionary graph in the following way. All vertices start with identical initial ranking and then iteratively distribute it to the vertices they point to, while receiving the sum of the ranks from vertices they are pointed by. This process converges to a stationary distribution corresponding to the principal eigenvector of the adjacency matrix of the graph. ArcRank assigns a ranking to each edge according to the ranking of its vertices. If $|a_s|$ is the number of outgoing edges from vertex s and p_t is the page rank of vertex t, then the edge relevance of (s,t) is defined by

$$r_{s,t} = \frac{p_s/|a_s|}{p_t}$$

Edge relevances are then converted into rankings. Those rankings are computed only once. When looking for words related to some word w, one select the edges starting from or arriving to w which have the best rankings and extract the corresponding incident vertices.

4 Dictionary graph

Before proceeding to the description of our experiments, we describe how we constructed the dictionary graph. We used the Online Plain Text English Dictionary [1] which is based on the "Project Gutenberg Etext of Webster's Unabridged Dictionary" which is in turn based on the 1913 US Webster's Unabridged Dictionary. The dictionary consists of 27 HTML files (one for each letter of the alphabet, and one for several additions). These files are available from the web site `http://www.gutenberg.net/`. In order to obtain the dictionary graph several choices had to be made.

- Some words defined in the Webster's dictionary are multi-words (e.g., **All Saints'**, **Surinam toad**). We did not include these words in the graph since there is no simple way to decide, when the words are found side-by-side, whether or not they should be interpreted as single words or as a multi-word.

- Some head words of definitions were prefixes or suffixes (e.g., **un-**, **-ous**), these were excluded from the graph.

- Many words have several meanings and are head words of multiple definitions. For, once more, it is not possible to determine which meaning of a word is employed in a definition, we gathered the definitions of a word into a single one.

- The recognition of derived forms of a word in a definition is also a problem. We dealt with the cases of regular and semi-regular plurals (e.g. **daisies**, **albatrosses**) and regular verbs, assuming that irregular forms of nouns or verbs (e.g., **oxen**, **sought**) had entries in the dictionary.

- All accentuated characters were replaced in the HTML file by a \ (e.g, **proven\al**, **cr\che**). We included these words, keeping the \.

- There are many misspelled words in the dictionary, since it has been built by scanning the paper edition and processing it with an OCR software. We didn't take these mistakes into account.

Because of the above remarks, the graph is far from being a precise graph of semantic relationships. For example, 13,396 lexical units are used in the definitions but are not defined. These include numbers (e.g., **14159265**, **14th**) and mathematical and chemical symbols (e.g., **x3**, **fe3o4**). When this kind of lexemes, which are not real words, are excluded, 12,461 words remain: proper nouns (e.g., **California**, **Aaron**), misspelled words (e.g., **aligator**, **abudance**), existing but undefined words (e.g., **snakelike**, **unwound**) or abbreviations (e.g., **adj**, **etc**).

The resulting graph has 112,169 vertices and 1,398,424 edges. It can be downloaded from `http://www.eleves.ens.fr:8080/home/senellar/stage_maitrise/graphe`.
We analyzed several features of the graph: connectivity and strong connectivity, number of connected components, distribution of connected components, degree distributions, graph diameter, etc. Our findings are reported in [9].

We also decided to exclude too frequent words in the construction of neighborhood graphs, that is words who appeared in more than L definitions (best results were obtained for $L \approx 1,000$). (The most often occurring words and the number of occurrences are: **of**: 68187, **a**: 47500, **the**: 43760, **or**: 41496, **to**: 31957, **in**: 23999, **as**: 22529, **and**: 16781, **an**: 14027, **by**: 12468, **one**: 12216, **with**: 10944, **which**: 10446, **is**: 8488, **for**: 8188, **see**: 8067, **from**: 7964, **being**: 6683, **who**: 6163, **that**: 6090).

5 Results

In order to be able to compare the different methods and to judge their relevance, we will examine the first ten results given by each of them for four words, chosen for their variety:

1. **disappear**: a word with various synonyms such as **vanish**.

2. **parallelogram**: a very specific word with no true synonyms but with some similar words: **quadrilateral**, **square**, **rectangle**, **rhomb**...

3. **sugar**: a common word with different meanings (in chemistry, cooking, dietetics...). One can expect **glucose** as a candidate.

4. **science**: a common and vague word. It is hard to say what to expect as synonym. Perhaps **knowledge** is the best option.

Words of the English language belong to different parts of speech: nouns, verbs, adjectives, adverbs, prepositions, etc. It is natural, when looking for a synonym of a word, to get only words of the same type. The Websters's Dictionary provides for each word its part of speech. But this presentation has not been standardized and we counted not less than 305 different categories. We have chosen to select 5 types: nouns, adjectives, adverbs, verbs, others (including articles, conjunctions and interjections) and have transformed the 305 categories into combinations of these types. A word may of course belong to different types. Thus, when looking for synonyms, we have excluded from the list all words that do not have a common part of speech with our word. This technique may be applied with all synonym extraction methods but since we did not implement ArcRank, we did not use it for ArcRank. In fact, the gain is not huge, because many words in English have several grammatical natures. For

instance, **adagio** or **tete-a-tete** are at the same time nouns, adjectives and adverbs.

We have also included lists of synonyms coming from two hand-made sources: WordNet [2] and the dictionary of synonyms of Microsoft Word 97. The order of appearance of the words for these two last sources is arbitrary, whereas it is well defined for the distance method and for our method. The results given by the Web interface implementing ArcRank are two rankings, one for words pointed by and one for words pointed to. We have interleaved them into one ranking. We have not kept the query word in the list of synonyms, since this has not much sense except for our method, where it is interesting to note that in every example we have experimented, the original word appeared as the first word of the list (a point that tends to give credit to the method).

In order to have an objective evaluation of the different methods, we asked a sample of 21 persons to give a mark (from 0 to 10, 10 being the best one) to the lists of synonyms, according to their relevance to synonymy. The lists were of course presented in random order for each word. Tables 1, 2, 3 and 4 give the results.

Concerning **disappear**, the distance method and our method do pretty well: most of the words given by hand-made dictionaries (**vanish**, **cease**, **fade**, **die**, **pass**) appear (one must not forget that verbs necessarily appear without their postposition). Other words such as **dissipate** or **faint** are relevant too. However, some words like **light** or **port** are completely irrelevant, but they appear only in 6th, 7th or 8th position. If we compare these two methods, we observe that our method is better: an important synonym like **pass** takes a good ranking, whereas **port** or **appear** go out of the top ten words. It is hard to explain this phenomenon, but we can say that the mutually reinforcing aspect of our method is apparently a positive point. On the contrary, ArcRank gives rather poor results with out of the point words like **eat**, **instrumental** or **epidemic**.

Because the neighborhood graph of **parallelogram** is rather small (30 vertices), the first two algorithms give similar results, which are not absurd: **square**, **rhomb**, **quadrilateral**, **rectangle**, **figure** are rather interesting. Other words are less relevant but still are in the semantic domain of **parallelogram**. ArcRank which also works on the same subgraph does not give as interesting words, although **gnomon** makes its appearance, since **consequently** or **popular** are irrelevant. It is interesting to note that hand-made dictionaries are less rich, because they focus on a particular aspect (**quadrilateral** for Wordnet, **rhomb** for Microsoft Word).

Once more, the results given by ArcRank for **sugar** are mainly irrelevant (**property**, **grocer**, ...). Our method is again better than the distance method: **starch**, **sucrose**, **sweet**, **dextrose**, **glucose**, **lactose** are highly relevant words, even if the first given near-synonym (**cane**) is not as good. Its given mark is even better than for the two hand-made dictionaries. The dictionary of synonyms of Microsoft Word amusingly focuses on a very specific aspect of the word.

The results for **science** are perhaps the most difficult to analyze. The distance method and ours are comparable. ArcRank gives perhaps better results than for other words but is still poorer than the two other methods. Once again, the dictionary of synonyms of Word gives very few words, though highly relevant ones.

As a conclusion, the first two algorithms give interesting and relevant words, whereas it is clear that ArcRank is not adapted to the search for synonyms. The variation of Kleinberg's algorithm and its mutually reinforcing relationship demonstrates its superiority on the basic distance method, even if the difference is not obvious for all words. Of course, the obtained relevance cannot be reasonably compared with those of hand-made lists. Still, these automatic techniques show their interest, since they present more complete aspects of a word than hand-made dictionaries. They could profitably be used to broaden a topic (see the example of **parallelogram**) and to help with the compilation of synonyms dictionaries.

6 Future perspectives

A first immediate improvement of our method would be to work on a larger subgraph than the neighborhood subgraph. The neighborhood graph we have introduced may be rather small, and may therefore not include important near-synonyms. A good example is **ox** of which **cow** seems to be a good synonym. Unfortunately, **ox** does not appear in the definition of **cow**, neither does the latter appear in the definition of the former. Thus, the methods described above cannot find this word. The first idea would be to extend our neighborhood graph, either as Kleinberg does in [8] for searching similar pages on the Web or as Dean and Henziger do in [5] for the same purpose. However, such subgraphs are not any longer focused on the original word. That implies that our variation of

	Distance	Our method	ArcRank	Wordnet	Microsoft Word
1	**vanish**	**vanish**	**epidemic**	**vanish**	**vanish**
2	**wear**	**pass**	**disappearing**	**go away**	**cease to exist**
3	**die**	**die**	**port**	**end**	**fade away**
4	**sail**	**wear**	**dissipate**	**finish**	**die out**
5	**faint**	**faint**	**cease**	**terminate**	**go**
6	**light**	**fade**	**eat**	**cease**	**evaporate**
7	**port**	**sail**	**gradually**		**wane**
8	**absorb**	**light**	**instrumental**		**expire**
9	**appear**	**dissipate**	**darkness**		**withdraw**
10	**cease**	**cease**	**efface**		**pass away**
Average mark	3.6	6.3	1.2	7.5	8.6
Standard deviation	1.8	1.7	1.2	1.4	1.3

Table 1: Proposed synonyms for **disappear**

	Distance	Our method	ArcRank	Wordnet	Microsoft Word
1	**square**	**square**	**quadrilateral**	**quadrilateral**	**diamond**
2	**parallel**	**rhomb**	**gnomon**	**quadrangle**	**lozenge**
3	**rhomb**	**parallel**	**right-lined**	**tetragon**	**rhomb**
4	**prism**	**figure**	**rectangle**		
5	**figure**	**prism**	**consequently**		
6	**equal**	**equal**	**parallelepiped**		
7	**quadrilateral**	**opposite**	**parallel**		
8	**opposite**	**angles**	**cylinder**		
9	**altitude**	**quadrilateral**	**popular**		
10	**parallelepiped**	**rectangle**	**prism**		
Average mark	4.6	4.8	3.3	6.3	5.3
Standard deviation	i2.7	2.5	2.2	2.5	2.6

Table 2: Proposed synonyms for **parallelogram**

	Distance	Our method	ArcRank	Wordnet	Microsoft Word
1	**juice**	**cane**	**granulation**	**sweetening**	**darling**
2	**starch**	**starch**	**shrub**	**sweetener**	**baby**
2	**cane**	**sucrose**	**sucrose**	**carbohydrate**	**honey**
4	**milk**	**milk**	**preserve**	**saccharide**	**dear**
5	**molasses**	**sweet**	**honeyed**	**organic compound**	**love**
6	**sucrose**	**dextrose**	**property**	**saccarify**	**dearest**
7	**wax**	**molasses**	**sorghum**	**sweeten**	**beloved**
8	**root**	**juice**	**grocer**	**dulcify**	**precious**
9	**crystalline**	**glucose**	**acetate**	**edulcorate**	**pet**
10	**confection**	**lactose**	**saccharine**	**dulcorate**	**babe**
Average mark	3.9	6.3	4.3	6.2	4.7
Standard deviation	2.0	2.4	2.3	2.9	2.7

Table 3: Proposed synonyms for **sugar**

	Distance	Our method	ArcRank	Wordnet	Microsoft Word
1	**art**	**art**	**formulate**	**knowledge domain**	**discipline**
2	**branch**	**branch**	**arithmetic**	**knowledge base**	**knowledge**
3	**nature**	**law**	**systematize**	**discipline**	**skill**
4	**law**	**study**	**scientific**	**subject**	**art**
5	**knowledge**	**practice**	**knowledge**	**subject area**	
6	**principle**	**natural**	**geometry**	**subject field**	
7	**life**	**knowledge**	**philosophical**	**field**	
8	**natural**	**learning**	**learning**	**field of study**	
9	**electricity**	**theory**	**expertness**	**ability**	
10	**biology**	**principle**	**mathematics**	**power**	
Average mark	3.6	4.4	3.2	7.1	6.5
Standard deviation	2.0	2.5	2.9	2.6	2.4

Table 4: Proposed synonyms for **science**

Kleinberg's algorithm forgets the original word and produces irrelevant results. Nevertheless, when we use the vicinity graph of Dean and Henziger, we obtain a few interesting results with specific words: for example, **trapezoid** appears as a near-synonym of **parallelogram** or **cow** as a near-synonym of **ox**. Yet there are also many degradations of performance for more general words. Perhaps a choice of the subgraph depending on the word itself would be appropriate. For instance, the extended vicinity graph may either be used for words whose neighborhood graph has less than a fixed number of vertices, or for words whose indegree is small, or for words who do not belong to the largest connected component of the dictionary graph.

One may wonder whether the results obtained are specific to the Webster's dictionary or whether the same methods could work on other dictionaries, in English or in other languages. Although the latter is most likely since our techniques were not designed for the particular graph we worked on, there will undoubtedly be differences with other languages. For example, in French, postpositions do not exist and thus verbs have not as many different meanings as in English. Besides, it is much rarer in French to have the same word for the noun and for the verb than in English. Furthermore, linguistics teach us that the way words are defined vary from one language to another. This seems to be an interesting research direction.

References

[1] The Online Plain Text English Dictionary, `http://msowww.anu.edu.au/~ralph/OPTED/`, 2000.

[2] Wordnet 1.6, `http://www.cogsci.princeton.edu/~wn/`, 1998.

[3] Vincent D. Blondel, Maureen Heymans, Paul Van Dooren. Feature extraction in large graphs, in preparation.

[4] Sergey Brin and Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems, 30:1–7, pp. 107–117, 1998.

[5] Jeffrey Dean and Monika Rauch Henzinger, Finding Related Pages in the World Wide Web, WWW8 / Computer Networks, 31:11-16, pp. 1467-1479, 1999.

[6] Maureen Heymans, Extraction d'information dans les graphes, et application aux moteurs de recherche sur Internet. Mémoire de fin d'études. Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2001.

[7] Jan Jannink and Gio Wiederhold, Thesaurus Entry Extraction from an On-line Dictionary, Proceedings of Fusion 1999, Sunnyvale CA, 1999.

[8] Jon M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM, 46:5, pp. 604–632, 1999.

[9] Pierre Senellart. Extraction of information in large graphs. Automatic search for synonyms. Masters Internship Report. Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2001.