Masters Internship Report
Extraction of information in large graphs; Automatic search for synonyms

Pierre Senellart
Pierre.Senellart@ens.fr

June 5th 2001 - August 3rd 2001

1 Extraction of the graph of the dictionary

I used the Online Plain Text English Dictionary (OPT 2000) 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. It consists in 27 HTML files (one for each letter of the alphabet, and one for several additions). The problem was to parse these files in order to build the graph of the dictionary: there is a vertex for every word and an arc from vertex i to vertex j if j appears in the definition of i. I encountered several problems in this operation:

- Some words defined in the Webster's dictionary were in fact multi-words (e.g. **All Saints'**, **Surinam toad**). I decided not to include them into the graph, since there is no simple way, when you see two words side-by-side, to decide whether 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**), which I also 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, I 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. I 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 I used by a **/** (e.g **proven/al**, **cr/che**). I 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. I didn't take these mistakes into account.

The resulting graph has 112,169 vertices and 1,398,424 arcs, but because of these remarks, it is far from being a precise graph of semantic relationships. For example, 13,396 lexical units are used in the definitions and not defined. These are of course numbers (e.g. **14159265**, **14th**), mathematical and chemical symbols (e.g. **x3**, **fe3o4**). But when this kind of lexems, which are not real words, are excluded, it remains 12,461 words: 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 graph is available for download at `http://www.eleves.ens.fr:8080/home/senellar/stage_maitrise/graphe`

2 Brief analysis of the graph

Several interesting computations may be carried out on the graph. The first natural question is of its connectivity. When dealing with oriented graphs, there are two different notions of connectivity: a graph is *strongly connected* if there is a path from any vertex to any other one and a graph is just said to be *connected* if the underlying undirected graph is connected. We can derive therefrom the notions of *strongly connected components* and *connected components* of a graph which are the equivalence classes of, respectively, the relations R_1 and R_2, where iR_1j means that there is a path from i to j and a path from j to i and iR_2j means that there is a path from i to j in the underlying undirected graph.

2.1 Connectivity

One could expect the graph of a dictionary to be connected. However, there are 185 different connected components: a large one with 111,982 vertices, 3 two-vertex components and 181 one-vertex components (see list in appendix). This is either due to misspelling or to words not defined in the dictionary: for example, **anguineal** is defined with the single word **anguineous** which is in turn defined as **snakelike**, the definition of which does not exist. Another example is **indissolvableness**, defined as **indissolubleness**, defined as **indissolubility**. But **indissolubility** is not defined, because of a misspelling: **indisdolubility** is defined.

Thus, it appears that the graph is not connected because of small, not very important, details. It just could be use to search the lacks and mistakes of the dictionary. The issue of strong connectivity is much more interesting.

2.2 Strong connectivity

There are 79,348 strongly connected components. Table 1 shows their classification according to their size and Figure 1 how they are connected to each other (additional arrows may exist, as long as they go from top to bottom).

The dictionary may thus be split into two parts: the largest connected component, which is a kind of "core" of the English language, and the peripheral words. As for small components with more than one vertex, they form the semantic field of a specific domain. For example, here are the ten-vertex component and one of the seven-vertex component:

1. **bezpopovtsy**, **dukhobors**, **dukhobortsy**, **judaizers**, **molokane**, **molokany**, **popovtsy**, **raskolnik**, **raskolniki**, **skoptsy**: several Russian dissident churches from the Greek faith or their members. The word "raskolnik" is how the Russian government (before the Revolution) called them all. It is interesting to note that all these words appeared in the "addition file" of the Webster's Dictionary.

2. **theosophy**, **theurgy**, **theurgic**, **theurgical**, **neoplatonic**, **neoplatonist**, **neoplatonism**: the Neoplatonists developed the theurgy, which is a kind of theosophy.

I said above that the largest connected component C could be seen as the core of the language. If this is true, you could understand the definition of any word in the dictionary if you only knew the meanings of the words in C, that is, in the graph where the directions of the arrows are inverted, all vertices are accessible from the vertices in C. Of course, it can't be true since the graph is not connected. But when you exclude the other connected components, you notice that all vertices except 12 of them are accessible. Once more, the reason of these 12 words lies in misspellings, undefined words or such (for example, **dyscrasy** is defined as **dycrasia** instead of **dyscrasia**). The denomination of "core" is therefore valid.

An interesting issue would be to determine the minimum number of words which generate the dictionary, that is the minimum number of words you have to know if you want to be able to understand the meaning of all headwords of the dictionary. More formally, this is the minimum number of vertices of a subgraph containing at least one vertex from every directed cycle in G and such that every path in the graph may be prolongated in a path containing a vertex of the subgraph (a subgraph with these two properties will be called a *core subgraph*). Let's call this number the *independence degree* of the graph.

Theorem 1 *Let G be a strongly connected graph and V a subgraph of G. V is a core subgraph if and only if V contains at least one vertex from every directed cycle in G.*

Proof

Let's assume that V contains at least one vertex from every directed cycle in G. Let $k_0 \to k_1 \to \ldots \to k_l$ be a path in G. Since G is strongly connected, there exists a path $k_l \to k_{l+1} \to \ldots \to k_m \to k_0$.

$k_0 \to \ldots \to k_m \to k_0$ is a path prolongating the initial path and contains a vertex of V since it is also a directed cycle, which concludes the proof.

□

Thus, the independence degree of a strongly connected graph is the minimal size of a subgraph containing at least one vertex from every directed cycle. The computation of this value is a NP-complete problem (cf (Garey & Johnson 1983)) so the computation of the independence degree of any graph is a NP-complete problem (the fact that it is NP is clear enough).

There remains to find a good approximation algorithm. I can naturally give the upper bound of $30,595 + 187 + 12 = 30,794$ for the graph of the dictionary.

Another issue I looked at is the diameter of C (i.e. the maximal minimum distance between two vertices). An exact computation would be far too much

time-consuming, but I was able to find it was between 14 and 16: there are two vertices i and j such that $d(i,j) = 14$ ($d(i,j)$ is the minimum length of a path from i to j) and there is a vertex x (**a** or **for** for instance) such that $\forall y, \forall z, d(y,x) + d(x,z) \leq 16$.

2.3 A small world

There is truth in the popular expression: "It's a small world!". In fact, in the graph of human relationships, the minimal distance between any two vertices is always very small: the diameter of the graph is probably not much greater than the number six advanced in John Guare's play *Six Degrees of Separation*. This phenomenon arises in graphs that are called "small worlds".

We will need to know what a random graph is prior to defining small world graphs. There are two classical definitions of random graphs:

1. A random graph $M(n,m)$ is a graph of n vertices and m edges being chosen uniformately out of all possible edges

2. A random graph $M(n,p)$ is a graph of n vertices whose each pair of points is the support of an edge with the probability p

In practice, the properties of the two kinds of random graphs are very similar, when $p = \frac{2m}{n(n-1)}$. We will call a random graph indifferently either of these sorts of graphs.

In (Watts 1999), Duncan J. Watts proposes the following definition of small world graph:

A small world graph is an undirected, unweighted, sparse and connected graph of n vertices and average degree k verifying the two following conditions:

1. The mean minimal length of a path between any two vertices (which is called the *characteristic path length*) L is close to that of a random graph with same n and k.

2. The mean over all vertices of the ratio of the number of edges in the neighborhood graph by the number of possible edges in the same subgraph (which is called the *clustering coefficient*) γ is much greater than that of a random graph of same n and k.

Small world graphs are very frequent in very various fields. The Kevin Bacon graph, for instance is a small world: each vertex is an actor and there is an edge between two actors if they appeared in the same movie. Other examples of small worlds are the Web, power distribution graphs or the nervous system of some worm.

Is our graph of the dictionary a small world? Our graph is directed and not connected. In order to give some sense to this question, we took the underlying undirected graph and we kept only the members of the largest connected component, in order to have an undirected, unweighted, sparse (because $k \approx 24.6$) and connected graph. Then, we computed the values of L and γ and compared them to the values for a random graph, using the approximative equations given by Duncan J. Watts:

1. $L \approx 2,40 \ \sim \ 3.61 \approx L_{\text{random}}$

2. $\gamma \approx 0.45 \gg 2.19\ 10^{-4} \approx \gamma_{\text{random}}$

It seems then that the graph of the dictionnary is a small world. The characteristic path length is even impressive, since it is smaller than that of a random graph. This is due to the simple fact that some words (**of**, **a**...) connect together most words.

However, the graph of the dictionary does not fit the model proposed by Duncan J. Watts. He constructs an infinite family of graphs, indexed by a parameter ϕ (the percentage of edges which are shortcut, that is which bind edges otherwise at a distance strictly greater than two), which is supposed to model small worlds. But, for the value of ϕ for the graph of the dictionary ($\phi \approx 0.16$), the model predicts $\gamma \approx 0.70$ and $L \approx 4.45$. There remains thus to find another model which may describe the graph of the dictionary as a small world.

One should not forget either that this graph is a directed graph and that we lose much information when we do not take into account "the direction of arrows". It would be perhaps necessarily for a precise description of this graph to construct models of directed small worlds.

2.4 Degree distributions

The indegree and outdegree distributions of the vertices in the Web graph follows a Zipfian distribution ((Kleinberg, Kumar, Raghavan, Rajagopalan & Tomkins 1999)): the probability that a node has indegree or outdegree i is proportional to $\frac{1}{i^\alpha}$ for some α.

Figure 2 and 3 show the indegree and outdegree distribution of the vertices in the dictionary graph on a log-log scale. We notice the same kind of Zipfian distribution. $\alpha \approx 1.6$ for the indegrees and $\alpha \approx 3.1$ for the outdegrees. However, there is a difference between these two graphs. Unlike the indegree distribution, the outdegree distribution show two discrepancies from the Zipfian model. First, the outdegree is bounded by a rather small ammount, which was also true for the Web graph. This is logical: there cannot be too many words in a definition (whereas there is no a priori limit to the number of words that use some fixed word in their definition) as there cannot be too many links in a webpage. Secondly, the plot is not linear in the range of small outdegrees. This inflection is also present for the Web graph but is less visible.

The similitude between the degree distributions of the Web and of our graph which are two small world directed graphs may mean that a small world directed model should take into account these degree distributions.

All these parameters (the number of strongly connected components, their size and diameter, whether the largest one is a core, the small-world parameters, degree distributions, etc) could possibly be interesting tools to compare graphs of different dictionaries, for example in different languages. One could for instance imagine that there are invariants for dictionaries of a given language. Unfortunately, I was unable to find another full-text dictionary in electronic format and cannot thus make use of this remark.

Number of vertices	Number of components
30,595	1
10	1
7	3
6	13
5	21
4	50
3	222
2	1,457
1	77,580

Table 1: Distribution of the size of strongly connected components

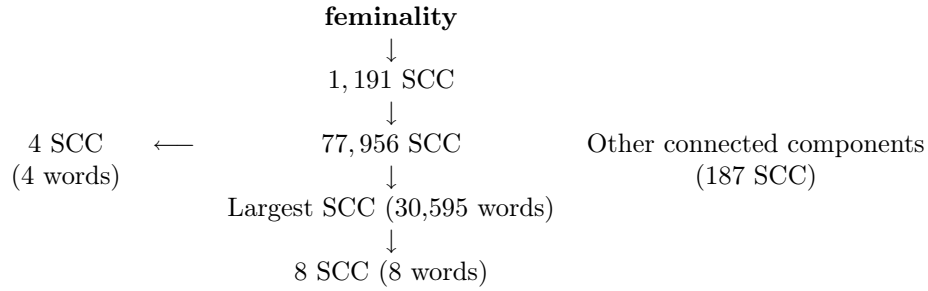

Figure 1: Graph resulting of the contraction of each Strongly Connected Component in one single vertex

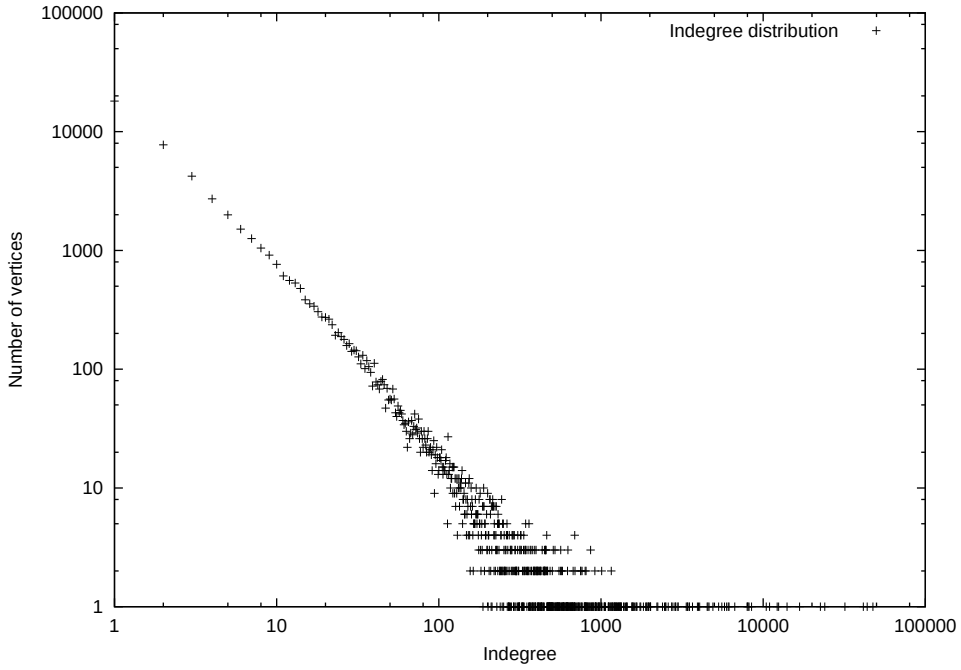

Figure 2: Indegree distribution

3 Three methods for discovering near-synonyms

The problem of searching near-synonyms, that is "similar words", may seem close to that of searching similar pages on the Web, which has been dealt rather satisfactorily for instance in (Kleinberg 1999) or (Dean & Henzinger 1999). However, this is definitely not the same problem: in these two articles, similar pages were in fact authoritative pages in a subgraph focused on the original page, which was implicitly supposed to be authoritative enough. This is completely different for near-synonyms. The words we start from have no reason to be authorities of their neighborhood. Yet these methods are a good starting point for developing dictionary-specific methods.

In this section, I describe three different methods for extracting near-synonyms from the dictionary graph. In section 4, I compare these three methods on several examples.

The near-synonyms algorithms described below will be applied on a subgraph of the graph of the dictionary, depending on a fixed vertex (i.e. word) i, and consisting of i, all parents of i and all children of i. However, I decided to exclude from the selection too frequent words, that is words who appeared in more than L definitions (best results were obtained for $L \approx 1,000$). The 20 most often occurring words are given in appendix.

The graph obtained this way will be called *the neighborhood graph* of vertex i. In the following, we assume that i is fixed and that the neighboring graph of i is represented by its transition matrix $A = (a_{i,j})_{i,j=1...n}$.

3.1 The vectors method: a rather elementary one

A rather natural way to define near-synonyms is to say that two words are semantically close if they appear in the definition of the same words and have the same words in their definition.

The principle is to compute some distance between lines and columns of the transition matrix: if the jth and kth lines are "close" and if the jth and kth columns are "close", j and k point to and are pointed by about the same vertices, and they are thus near-synonyms.

It gives an immediate algorithm for discovering near-synonyms of a word i:

1. For each $1 \leq j \leq n, j \neq i$, compute:

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

   (where $\|\ \|$ is some vector norm, $A_{i,\cdot}$ and $A_{\cdot,i}$ are respectively the ith line and the ith column of A).

   For instance, if we choose the Euclidean norm, we compute:

$$\left(\sum_{k=1}^{n}(A_{i,k} - A_{j,k})^2\right)^{\frac{1}{2}} + \left(\sum_{k=1}^{n}(A_{k,i} - A_{k,j})^2\right)^{\frac{1}{2}}$$

2. Sort the different words according to the computed value and return the k first ones.

Unlike the two other algorithms described below, one could try to apply this algorithm directly on the entire graph. But this gives very bad results: the first two near-synonyms of **sugar** are **pigwidgeon** and **ivoride**. However, we'll see that pretty good results are achieved if we use the neighborhood graph.

3.2 A variation of Kleinberg's algorithm

In (Kleinberg 1999), Jon Kleinberg proposes an algorithm for identifying among Web pages *hubs* and *authorities* concerning a given query in a search engine. For example, for the query "automobile makers", the home page of *Ford*, *Toyota* or other car makers are good authorities, whereas sites which give a list of these home pages (such as `http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Automotive/Makers/Vehicles/`) are good hubs. The fundamental idea is that of a *mutually reinforcing relationships*: good hubs are pages that point to good authorities and good authorities are pages pointed by good hubs.

He starts on building "a focused subgraph" which is the analogous of our neighborhood graph, and then he tries to determine which vertices of this subgraph "look like" vertex 1 and vertex 2 of the following graph:

$$1 \longrightarrow 2$$

For this purpose, he compute the principal eigenvector (which is assumed to exist) of $A^T A$ and AA^T which give respectively the *authority weights* and *hub weights* of the vertices of the graph. Vertices with highest authority weights are best authorities, that is they look like vertex 2 in the above graph; vertices with highest hub weights are best hubs, that is they look like vertex 1 in the above graph.

Because of the way we built the neighborhood graph, vertex i look very much like the second vertex of the graph:

$$1 \longrightarrow 2 \longrightarrow 3$$

. The idea of this method is to look for other vertices with the same resemblance. We use for this purpose the generalization of Kleinberg's algorithm proposed by Vincent Blondel and Maureen Heymans in (Heymans 2001):

Let $M(m, m)$ and $N(n, n)$ be the transition matrices of two oriented graphs. Let $C = M \otimes N + M^T \otimes N^T$ where $\otimes$ is the Kronecker tensorial product. We assume that the greatest eigenvalue of C is strictly greater than the absolute value of all other eigenvalues. Then, the normalized principal eigenvector X of C gives the "similarity" between a vertex of M and a vertex of N: $X_{i \times n+j}$ characterizes the similarity between vertex i of M and vertex j of N. In particular, if $M = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, the result is that of Kleinberg's algorithm.

We may thus apply this extension of Kleinberg's algorithm with $M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, which is the transition matrix of the graph mentioned above. This is precisely our second method for discovering near-synonyms.

3.3 ArcRank

In (Jannink & Wiederhold 1999), Jan Jannink and Gio Wiederhold propose an algorithm for building a thesaurus starting from a word. They also worked on the 1913 Webster's Dictionary and their algorithm can be tested at `http://skeptic.stanford.edu/data/`. For these reason, I did not implement their algorithm but directly used their results. Moreover, their intent was not to find near-synonyms but related words.

ArcRank is based on the PageRank algorithm, used by Google and described in (Brin & Page 1998). PageRank assigns a ranking to each vertex of the graph in the following way: all vertices begin with a constant initial ranking and they iteratively distribute it to the vertices they point to, while receiving rank from vertices they are pointed by. This process converges to a stationary distribution corresponding to the principal eigenvector of the adjacency matrix. ArcRank assigns a ranking to each arc according to the ranking of vertices. If $|a_s|$ is the number of outgoing arcs from vertex s and p_t is the page rank of vertex t, then the arc relevance of arc (s,t) is defined by:

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

Arc relevances are then converted into rankings.

Those rankings are computed only once. When you are looking for words related to some word x, you select those of the arc starting from or arriving to x which have the best rankings and you get the corresponding vertices.

3.4 Selection by the part of speech

Words of the English language belong to different parts of speech: nouns, verbs, adjectives, adverbs, prepositions... It is natural, when looking for a near-synonym of a word, to get only words of the same type. The Websters's Dictionary provide for each headword its part of speech. But it has not been standardized and we have counted not less than 305 different categories. I chose to select 5 types: nouns, adjectives, adverbs, verbs, others (including articles, conjunctions and interjections) and tried to transform the 305 categories into combinations of these types. A word may of course belong to different types.

Thus, when looking for near-synonyms, we can exclude from the list all words that do not have a common part of speech whith out word. This technique may be applied with all near-synonym algorithms, but as I did not implement ArcRank, I only used it on the first two described above.

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.

4 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 4 words, chosen for their variety:

1. **disappear**: a word with various synonyms or near-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 meaning (in chemistry, cooking, dietetics...). One can expect **glucose** for instance to be a good quasi-synonym.

4. **science**: a very common and vague word. It is hard to say what to expect as for a near-synonym of this word. Perhaps **knowledge** is the best one.

Tables 2, 3, 4 and 5 give the corresponding results. I also included lists of synonyms coming from two hand-made sources: WordNet (Wor 1998) and the dictionary of synonyms of Microsoft Word 97. The order of appearance of the words for these two last sources is rather arbitrary, whereas it is well defined for the Vectors and Kleinberg algorithms. As for the ArcRank Algorithm, the results given by the Web interface are two rankings, one for words pointed by and one for words pointed to. I interleaved them into one ranking.

I chose not to keep the query word in the list of near-synonyms, since this has not much sense for all methods but the variation of Kleinberg's algorithm, where it is really interesting to note that in every example I experimented, the original word appeared as the first word of the list (a point that tends to give credit to the method).

	Vectors	Kleinberg	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**

Table 2: Near-synonyms for **disappear**

Concerning **disappear**, the Vectors method and the variation of Kleinberg's algorithm 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 like **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 Kleinberg's seems better: an important near-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 Kleinberg's method is apparently a positive point: Because **pass** is pointed by good hubs and point to good authorities, this word is chosen

by the method. On the contrary, ArcRank gives rather poor results with out of the point words like **eat**, **instrumental** or **epidemic**.

	Vectors	Kleinberg	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**	**parallelopiped**		
7	**quadrilateral**	**opposite**	**parallel**		
8	**opposite**	**angles**	**cylinder**		
9	**altitude**	**quadrilateral**	**popular**		
10	**parallelopiped**	**rectangle**	**prism**		

Table 3: Near-synonyms for **parallelogram**

Because the niegborhood 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 a **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)

	Vectors	Kleinberg	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**

Table 4: Near-synonyms for **sugar**

Once more, the results given by ArcRank for **sugar** are mainly irrelevant (**property**, **grocer**...). Kleinbers's algorithm is again better than the Vectors one: **starch**, **sucrose**, **sweet**, **dextrose**, **glucose**, **lactose** are highly relevant words, even if the first given near-synonym (**cane**) is not as good. 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 analyse. The Vectors and Kleinberg's algorithms are comparable. ArcRank gives perhaps better

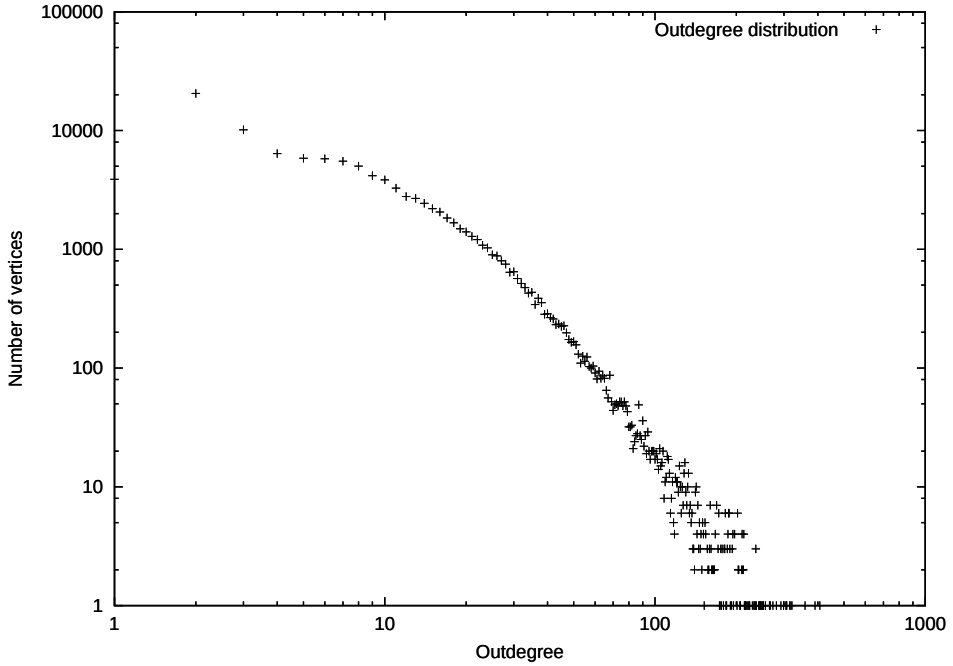

Figure 3: Outdegree distribution

	Vectors	Kleinberg	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**	

Table 5: Near-synonyms for **science**

results than for other words but still seems poorer than the two other methods. Once again, the dictionary of synonyms of Word gives very few words.

As a conclusion, our first two algorithms give interesting and relevant words, whereas it is clear that ArcRank is not adapted to the search for near-synonyms. The variation of Kleinberg's algorithm and its mutually reinforcing relationship demonstrates its superiority on the basic Vectors 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 list of near-synonyms. 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.

5 Future perspectives

A first immediate improvement of our algorithms would be to work on a larger subgraph than the neighborhood subgraph which may be rather small, and thus not include important near-synonyms. A good example is **ox**. **cow** seems to be a rather good near-synonym of it. Unfortunately, **ox** does not appear in the definition of **cow**, neither does the latter appear in the definition of the former. Thus, the algorithms described above, and in particular the variation of Kleinberg's algorithm, cannot find this word. The first idea would be to extend our neigborhood graph, either as Kleinberg does in (Kleinberg 1999) for searching similar pages on the Web or as Dean and Henziger do in (Dean & Henzinger 1999) for the same purpose. However, such subgraphs are not any longer focused on the original word. That implies that our variation of 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. I did not investigate this idea.

One may wonder whether the results we obtained were 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 chosen in function of the 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. Anyway, and even if it would certainly be a most interesting topic, we are limited to speculations, since it has not been possible to get another full-text dictionary in electronic format.

Another interesting question is to know what other kind of applications the variation of Kleinberg's agorithm described above may have. The fact to be

able to compare vertices of a large graph to vertices of any small reference graph seems promising. The most immediate other application would be to the recognition of similar pages on the Web by keeping our near-synonymy algorithm. However, the results are very disappointing, simply because pages which are pointed to by many websites (that is mostly home pages of a site) seldom point to another site: outgoing links of a website are seldom in its home page. The level which should be considered here would be the entire websites pointing to and pointed by other sites. But this raises a problem: how to identify which pages belong to some site? Moreover, the recursive browsing of websites in order to identify all outgoing links is a very resource-consuming task.

Other questions I already evoked are the problems of finding a model of directed small worlds graphs (or even, more specifically, models of dictionary graphs) and of discovering invariants for languages in the graph of the dictionary.

References

Brin, S. & Page, L. (1998), 'The anatomy of a large-scale hypertextual Web search engine', *Computer Networks and ISDN Systems* **30**(1–7), 107–117.

Dean, J. & Henzinger, M. R. (1999), 'Finding related pages in the world wide web', *WWW8 / Computer Networks* **31**(11-16), 1467–1479.
*http://citeseer.nj.nec.com/dean99finding.html

Garey, M. R. & Johnson, D. S. (1983), *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, New York, NY.

Heymans, M. (2001), 'Extraction d'information dans les graphes, et application aux moteurs de recherche sur internet'. Travail de fin d'étude, Université Catholique de Louvain, Faculté des Sciences Appliquées, Département d'Ingénierie Mathématique.

Jannink, J. & Wiederhold, G. (1999), Thesaurus entry extraction from an online dictionary, *in* 'Proceedings of Fusion '99, Sunnyvale CA'.
*http://citeseer.nj.nec.com/315100.html

Kleinberg, J. M. (1999), 'Authoritative sources in a hyperlinked environment', *Journal of the ACM* **46**(5), 604–632.
*http://citeseer.nj.nec.com/kleinberg97authoritative.html

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S. & Tomkins, A. S. (1999), The Web as a graph: Measurements, models, and methods, *in* T. Asano, H. Imai, D. T. Lee, S. Nakano & T. Tokuyama, eds, 'Proc. 5th Annual Int. Conf. Computing and Combinatorics, COCOON', Springer-Verlag.
*citeseer.nj.nec.com/kleinberg99web.html

OPT (2000), 'The online plain text english dictionary'.
*http://msowww.anu.edu.au/ ralph/OPTED/

Watts, D. J. (1999), *Small Worlds*, Princeton University Press.

Wor (1998), 'Wordnet 1.6'.
*http://www.cogsci.princeton.edu/ wn/

A Connected components

A.1 One-vertex components

1. **abandonedly**
2. **acronyctous**
3. **advantageousness**
4. **affrontedly**
5. **amability**
6. **amorphy**
7. **anchoretish**
8. **aslug**
9. **attently**
10. **barrenly**
11. **bawdily**
12. **brillance**
13. **cancellarean**
14. **canicule**
15. **cankeredly**
16. **chincherie**
17. **consumedly**
18. **cullibility**
19. **culpe**
20. **customably**
21. **daintrel**
22. **dejectly**
23. **deploredly**
24. **depper**
25. **derre**
26. **desertlessly**
27. **deservedness**
28. **despiteously**
29. **disasterly**
30. **disdainishly**
31. **disdainously**
32. **disfavorably**
33. **disingenuity**
34. **disinhabited**
35. **disinteresting**
36. **disordinately**
37. **disperseness**
38. **disprofitable**
39. **domableness**
40. **doubtlessly**
41. **electorial**
42. **epicurely**
43. **erke**
44. **evanescently**
45. **fantasticness**
46. **fastidiosity**
47. **festally**
48. **feverously**
49. **fightingly**
50. **fresh-new**
51. **frication**
52. **funambulation**
53. **gurts**
54. **habitability**
55. **harddihead**
56. **hemicrany**
57. **highmost**
58. **horsly**
59. **hypocritely**
60. **immarcescibly**
61. **immerited**
62. **immixed**
63. **immundicity**
64. **immutate**
65. **impalatable**
66. **impeccancy**
67. **improfitable**
68. **impune**
69. **incedingly**
70. **inchangeability**
71. **incicurable**
72. **incidently**
73. **incito-motory**
74. **incivilly**
75. **inconsumptible**
76. **incontracted**
77. **incultivated**
78. **indepravate**
79. **indesirable**
80. **indetermined**
81. **indignly**
82. **indisputability**
83. **indisputed**

84. **ineptly**
85. **inequable**
86. **inexpectedly**
87. **inexpectedness**
88. **infashionable**
89. **influxious**
90. **ingrately**
91. **innocuity**
92. **inquietness**
93. **insociably**
94. **insomnolence**
95. **insuitable**
96. **intaminated**
97. **intempestively**
98. **intranquillity**
99. **intumulated**
100. **invious**
101. **irefulness**
102. **irrelavance**
103. **jemminess**
104. **joyancy**
105. **loathliness**
106. **manageless**
107. **mistakenness**
108. **mistakingly**
109. **molliently**
110. **mononomial**
111. **narre**
112. **neer**
113. **nemertian**
114. **nemertid**
115. **ner**
116. **nerre**
117. **nocently**
118. **nourishingly**
119. **operosity**
120. **parcity**
121. **perplexly**
122. **pertinately**
123. **physnomy**
124. **plumbage**
125. **polylogy**
126. **populosity**
127. **prisonment**
128. **prosodiacally**
129. **proteinous**
130. **proventricle**
131. **quartzous**
132. **quibblingly**
133. **redempture**
134. **repentless**
135. **restiffness**
136. **revengeless**
137. **rh/tian**
138. **romanticly**
139. **roomily**
140. **salutiferously**
141. **sappare**
142. **scathless**
143. **sleightly**
144. **sortment**
145. **spaky**
146. **spastically**
147. **sperage**
148. **spitously**
149. **sportability**
150. **submissness**
151. **supermaxillary**
152. **superspinous**
153. **supinity**
154. **syndactylic**
155. **synonymally**
156. **tediosity**
157. **temporaneous**
158. **tentifly**
159. **thankly**
160. **thereology**
161. **tickleness**
162. **tireless**
163. **unafiled**
164. **unaquit**
165. **unartistic**
166. **uncautiously**
167. **uncharity**
168. **uncontrovertibly**
169. **undampned**
170. **undwellable**
171. **unkindliness**
172. **unperishably**
173. **unshaked**
174. **unsisting**
175. **untangibly**
176. **unusuality**
177. **urceolar**
178. **vinolency**
179. **volupty**
180. **wistly**
181. **wonderly**

A.2 Two-vertex components

1. **anguineal anguineous**
2. **indissolubleness indissolvableness**
3. **intempestivity untimeliness**

B Most frequent words

Word	Indegree
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