

2

Automatic Discovery of Similar Words

Pierre P. Senellart
Vincent D. Blondel

Overview

We deal with the issue of automatic discovery of similar words (synonyms and near-synonyms) from different kinds of sources: from large corpora of documents, from the Web, and from monolingual dictionaries. We present in detail three algorithms that extract similar words from a large corpus of documents and consider the specific case of the World Wide Web. We then describe a recent method of automatic synonym extraction in a monolingual dictionary. The method is based on an algorithm that computes similarity measures between vertices in graphs. We use the 1913 *Webster's Dictionary* and apply the method on four synonym queries. The results obtained are analyzed and compared with those obtained by two other methods.

2.1 Introduction

The purpose of this chapter is to review some methods used for automatic extraction of similar words from different kinds of sources: large corpora of documents, the Web, and monolingual dictionaries. The underlying goal of these methods is the automatic discovery of synonyms. This goal is, in general, too difficult to achieve since it is often difficult to distinguish in an automatic way synonyms, antonyms, and, more generally, words that are semantically close to each other. Most methods provide words that are “similar” to each other. We mainly describe two kinds of methods: techniques that, upon input of a word, automatically compile a list of good synonyms or near-synonyms, and techniques that generate a thesaurus (from some source, they build a complete lexicon of related words). They differ because in the latter case, a complete thesaurus is generated at the same time and there may not be an entry in the thesaurus for each word in the source. Nevertheless, the

purposes of the techniques are very similar and we therefore do not distinguish much between them.

There are many applications of such methods. For example, in natural language processing and information retrieval they can be used to broaden and modify natural language queries. They can also be used as a support for the compilation of synonym dictionaries, which is a tremendous task. In this chapter we focus on the search for synonyms rather than on applications of these techniques.

Many approaches for the automatic construction of thesauri from large corpora have been proposed. Some of them are presented in Section 2.2. The value of such domain-specific thesauri, as opposed to general handmade synonym dictionaries is stressed. We also look at the particular case of the Web, whose large size and other specific features do not allow their being handled in the same way as more classical corpora. In Section 2.3, we propose an original approach, which is based on monolingual dictionaries and uses an algorithm that generalizes an algorithm initially proposed by Kleinberg for searching the Web. Two other methods working from monolingual dictionaries are also presented.

2.2 Discovery of Similar Words from a Large Corpus

Much research has been carried out on the search for similar words in corpora, mostly for applications in information retrieval tasks. A large number of these approaches are based on the simple assumption that similar words are used in the same contexts. The methods differ in the way the contexts are defined (the document, a textual window, or more or less elaborate syntactical contexts) and the way the similarity is computed.

Depending on the type of corpus, we may obtain different emphasis in the resulting lists of synonyms. The thesaurus built from a corpus is domain-specific to this corpus and is thus more adapted to a particular application in this domain than a general hand-written dictionary. There are several other advantages to the use of computer-written thesauri. In particular, they may be rebuilt easily to mirror a change in the collection of documents (and thus in the corresponding field), and they are not biased by the lexicon writer (but are, of course, biased by the corpus in use). Obviously, however, hand-written synonym dictionaries are bound to be more liable, with fewer gross mistakes.

We describe below three methods that may be used to discover similar words. Of course, we do not pretend to be exhaustive, but rather have chosen to present some of the main approaches. In Section 2.2.1, we present a straightforward method, involving a document vector space model and the cosine similarity measure. This method is used by Chen and Lynch to extract information from a corpus on East-bloc computing [CL92] and we briefly report their results. We then look at an approach proposed by Crouch [Cro90] for the automatic construction of a thesaurus. The method is based on a term vector space model and term discrimination values [SYY75], and is specifically adapted for words that are not too frequent. In

Section 2.2.3, we focus on Grefenstette's SEXTANT system [Gre94], which uses a partial syntactical analysis. Finally, in the last section, we consider the particular case of the Web as a corpus, and discuss the problem of finding synonyms in a very large collection of documents.

2.2.1 A Document Vector Space Model

The first obvious definition of the context, given a collection of documents, is to say that terms are similar if they tend to occur in the same documents. This can be represented in a multidimensional space, where each document is a dimension and each term is a vector in document space with Boolean entries indicating whether the term appears in the corresponding document. It is common in information retrieval to use this type of vector space model. In the dual model, terms are coordinates and documents are vectors in term space; we show an application of this dual model in the next section.

Thus two terms are similar if their corresponding vectors are close to each other. The similarity between the vector \mathbf{i} and the vector \mathbf{j} is computed using a similarity measure, such as the cosine:

$$\cos(\mathbf{i}, \mathbf{j}) = \frac{\mathbf{i} \cdot \mathbf{j}}{\sqrt{\mathbf{i} \cdot \mathbf{i} \times \mathbf{j} \cdot \mathbf{j}}},$$

where $\mathbf{i} \cdot \mathbf{j}$ is the inner product of \mathbf{i} and \mathbf{j} . With this definition, we have $0 \leq \cos(\mathbf{i}, \mathbf{j}) \leq 1$; θ with $\cos \theta = \cos(\mathbf{i}, \mathbf{j})$ is the angle between \mathbf{i} and \mathbf{j} . Similar terms will tend to occur in the same documents and the angle between them will be small. Thus the cosine similarity measure will be close to one. In contrast, terms with little in common will not occur in the same documents, the angle between them will be close to $\pi/2$, and the cosine similarity measure will be close to zero.

The cosine is a commonly used similarity measure. One must, however, not forget that the justification of its use is based on the assumption that the axes are orthogonal, which is seldom the case in practice since documents in the collection are bound to have something in common and not be completely independent.

In [CL92] Chen and Lynch compare the cosine measure with another measure, referred to as the Cluster measure. The Cluster measure is asymmetrical, thus giving asymmetrical similarity relationships between terms. It is defined by

$$cluster(\mathbf{i}, \mathbf{j}) = \frac{\mathbf{i} \cdot \mathbf{j}}{\|\mathbf{i}\|_1},$$

where $\|\mathbf{i}\|_1$ is the sum of the magnitudes of \mathbf{i} 's coordinates (i.e., the l_1 norm of \mathbf{i}).

For both these similarity measures the algorithm is then straightforward: once a similarity measure has been selected, its value is computed between every pair of terms, and the best similar terms are kept for each term.

The corpus Chen and Lynch worked on was a 200 MB collection of various text documents on computing in the former East-bloc countries. They did not run the algorithms on the raw text. The whole database was manually annotated so that every document was assigned a list of appropriate keywords, countries,

organization names, journal names, person names, and folders. Around 60,000 terms were obtained in this way and the similarity measures were computed on them.

For instance, the best similar keywords (with the cosine measure) for the keyword **technology transfer** were: **export controls**, **trade**, **covert**, **export**, **import**, **micro-electronics**, **software**, **microcomputer**, and **microprocessor**. These are indeed related (in the context of the corpus) and words such as **trade**, **import**, and **export** are likely to be some of the best near-synonyms in this context.

The two similarity measures were compared on randomly chosen terms with lists of words given by human experts in the field. Chen and Lynch report that the Cluster algorithm presents a better Concept Recall ratio (i.e., is, the proportion of relevant terms that were selected) than cosine and human experts. Both similarity measures exhibits similar Concept Precision ratios (i.e., the proportion of selected terms that were relevant), and they are inferior to that of human experts. The asymmetry of Cluster seems to be a real advantage.

2.2.2 A Thesaurus of Infrequent Words

In [Cro90] Crouch presents a method for the automatic construction of thesaurus classes regrouping words that appear seldom in the corpus. Her purpose is to use this thesaurus to modify queries asked of an information retrieval system. She uses a term vector space model, which is the dual of the space used in the previous section: words are dimensions and documents are vectors. The projection of a vector along an axis is the weight of the corresponding word in the document. Different weighting schemes might be used; one that seems effective is the “Term Frequency Inverse Document Frequency” (*TF-IDF*), that is, the number of times the word appears in the document multiplied by a (monotone) function of the inverse of the number of documents in which the word appears. Terms that appear often in a document and do not appear in many documents therefore have an important weight.

As we saw earlier, we can use a similarity measure such as the cosine to characterize the similarity between two vectors (i.e., two documents). The algorithm proposed by Crouch, presented in more detail below, is to cluster the set of documents according to this similarity and then to select *indifferent discriminators* from the resulting clusters to build thesaurus classes.

Salton, Yang, and Yu introduce in [SY75] the notion of *term discrimination value*. It is a measure of the effect of the addition of a term (as a dimension) to the vector space on the similarities between documents. A good discriminator is a term that tends to raise the distances between documents; a poor discriminator tends to lower the distances between documents; finally, an indifferent discriminator does not change the distances between documents much. The exact or approximate computation of all term discrimination values is an expensive task. To avoid this problem, the authors propose using the term document frequency (i.e., the number of documents the term appears in) instead of the discrimination value, since experiments show they are strongly related. Terms appearing in less than about 1% of

the documents are mostly indifferent discriminators; terms appearing in more than 1% and less than 10% of the documents are good discriminators; very frequent terms are poor discriminators.

Crouch therefore suggests using low-frequency terms to form thesaurus classes, which should be made of indifferent discriminators. The first idea to build the thesaurus would be to cluster these low-frequency terms with an adequate clustering algorithm. This is not very interesting, however, since, by definition, one does not have much information about low-frequency terms. But the documents themselves may be clustered in a meaningful way. The complete link clustering algorithm, which produces small and tight clusters, is adapted to the problem. Each document is first considered as a cluster by itself, and iteratively, the two closest clusters (the similarity between clusters is defined to be the minimum of all similarities (computed by the cosine measure) between a pair of documents in the two clusters) are merged, until the distance between clusters becomes higher than a user-supplied threshold.

When this clustering step is performed, low-frequency words are extracted from each cluster. They build corresponding thesaurus classes. Crouch does not describe these classes but has used them directly for broadening information retrieval queries, and has observed substantial improvements in both recall and precision on two classical test corpora. It is therefore legitimate to assume that words in the thesaurus classes are related to each other. This method only works on low-frequency words, but the other methods presented here have problems in dealing with such words for which we have little information.

2.2.3 *The SEXTANT System*

Grefenstette presents in [Gre93, Gre94] an algorithm for the discovery of similar words that uses a partial syntactical analysis. The different steps of the algorithm SEXTANT (Semantic EXtraction from Text via Analyzed Networks of Terms) are detailed below.

Lexical Analysis

Words in the corpus are separated using a simple lexical analysis. A proper name analyzer is also applied. Then each word is looked up in a lexicon and is assigned a part of speech. If a word has several possible parts of speech, a disambiguator is used to choose the most probable one.

Noun and Verb Phrase Bracketing

Noun and verb phrases are then detected in the sentences of the corpus, using starting, ending, and continuation rules: for instance, a determiner can start a noun phrase, a noun can follow a determiner in a noun phrase, an adjective can not start, end, or follow any kind of word in a verb phrase, and so on.

ADJ	:	an adjective modifies a noun	(e.g., civil unrest)
NN	:	a noun modifies a noun	(e.g., animal rights)
NNPREP	:	a noun that is the object of a preposition modifies a preceding noun	(e.g., measurements along the crest)
SUBJ	:	a noun is the subject of a verb	(e.g., the table shook)
DOBJ	:	a noun is the direct object of a verb	(e.g., shook the table)
IOBJ	:	a noun in a prepositional phrase modifying a verb	(e.g., the book was placed on the table)

Figure 2.1. Syntactical relations extracted by SEXTANT.

Parsing

Several syntactic relations (or contexts) are then extracted from the bracketed sentences, requiring five successive passes over the text. Figure 2.1, taken from [Gre94], shows the list of extracted relations.

The relations generated are thus not perfect (on a sample of 60 sentences Grefenstette found a correctness ratio of 75%) and could be better if a more elaborate parser were used, but it would be more expensive too. Five passes over the text are enough to extract these relations, and since the corpus dealt with may be very large, backtracking, recursion, or other time-consuming techniques used by elaborate parsers would be inappropriate.

Similarity

Grefenstette focuses on the similarity between nouns; other parts of speech are not discussed. After the parsing step, a noun has a number of attributes: all the words that modify it, along with the kind of syntactical relation (ADJ for an adjective, NN or NNPREP for a noun, and SUBJ, DOBJ, or IOBJ for a verb). For instance, the noun **cause**, which appears 83 times in a corpus of medical abstracts, has 67 unique attributes in this corpus. These attributes constitute the context of the noun, on which similarity computations will be made. Each attribute is assigned a weight by

$$weight(att) = 1 + \sum_{\text{noun } i} \frac{p_{att,i} \log(p_{att,i})}{\log(\text{total number of relations})},$$

where

$$p_{att,i} = \frac{\text{number of times } att \text{ appears with } i}{\text{total number of attributes of } i}.$$

The similarity measure used by Grefenstette is a weighted Jaccard similarity measure defined as follows

1. CRAN (Aeronautics abstract)
case: characteristic, analysis, field, distribution, flaw, number, layer, problem
2. JFK (Articles on JFK assassination conspiracy theories)
case: film, evidence, investigation, photograph, picture, conspiracy, murder
3. MED (Medical abstracts)
case: change, study, patient, result, treatment, child, defect, type, disease, lesion

Figure 2.2. SEXTANT similar words for **case**, from different corpora.

species	bird, fish, family, group, form, animal, insect, range, snake
fish	animal, species, bird, form, snake, insect, group, water
bird	species, fish, animal, snake, insect, form, mammal, duck
water	sea, area, region, coast, forest, ocean, part, fish, form, lake
egg	nest, female, male, larva, insect, day, form, adult

Figure 2.3. SEXTANT similar words for words with most contexts in *Grolier's Encyclopedia* animal articles.

$$jac(\mathbf{i}, \mathbf{j}) = \frac{\sum_{att \text{ attribute of both } \mathbf{i} \text{ and } \mathbf{j}} weight(att)}{\sum_{att \text{ attribute of either } \mathbf{i} \text{ or } \mathbf{j}} weight(att)}.$$

Results

Grefenstette used SEXTANT on various corpora and many examples of the results returned are available in [Gre94]. Figure 2.2 shows the most similar words of **case** in three completely different corpora. It is interesting to note that the corpus has a great impact on the meaning of the word according to which similar words are selected. This is a good illustration of the value of working on a domain-specific corpus.

Figure 2.3 shows other examples, in a corpus on animals. Most words are closely related to the initial word and some of them are indeed very good (**sea, ocean, lake** for **water**; **family, group** for **species**, ...). There remain completely unrelated words though, such as **day** for **egg**.

2.2.4 How to Deal with the Web

The World Wide Web is a very particular corpus: its size can simply not be compared with the largest corpora traditionally used for synonym extraction, its access times are high, and it is also richer and more lively than any other corpus. Moreover, a large part of it is conveniently indexed by search engines. One could imagine that its hyperlinked structure could be of some use too. And of course it is not a domain-specific thesaurus. Is it possible to use the Web for the discovery of similar words? Obviously, because of the size of the Web, none of the above techniques can apply.

Turney partially deals with the issue in [Tur01]. He does not try to obtain a list of synonyms of a word *i* but, given a word *i*, he proposes a way to assign a synonymy score to any word *j*. His method was checked on synonym recognition questions extracted from two English tests: the Test Of English as a Foreign Language (TOEFL) and the English as a Second Language test (ESL). Four different synonymy scores are compared. They use the advanced search functions of the Altavista search engine (<http://www.altavista.com>).

$$\begin{aligned}
 score_1(j) &= \frac{hits(i \text{ AND } j)}{hits(j)} \\
 score_2(j) &= \frac{hits(i \text{ NEAR } j)}{hits(j)} \\
 score_3(j) &= \frac{hits((i \text{ NEAR } j) \text{ AND NOT } ((i \text{ OR } j) \text{ NEAR "not"}))}{hits(j \text{ AND NOT } (j \text{ NEAR "not"}))} \\
 score_4(j) &= \frac{hits((i \text{ NEAR } j) \text{ AND context AND NOT } ((i \text{ OR } j) \text{ NEAR "not"}))}{hits(j \text{ AND context AND NOT } (j \text{ NEAR "not"}))} .
 \end{aligned}$$

In these expressions, *hits* represents the number of pages returned by Altavista for the corresponding query; *AND*, *OR*, and *NOT* are the classical Boolean operators; *NEAR* imposes that the two words not be separated by more than 10 words; and **context** is a context word (a context was given along with the question in ESL; the context word may be automatically derived from it). The difference between *score₂* and *score₃* was introduced in order not to assign good scores to antonyms.

The four scores are presented in increasing order of the quality of the corresponding results. *score₃* gives a good synonym for 73.75% of the questions from TOEFL (*score₄* was not applicable since no context was given) and *score₄* gives a good synonym in 74% of the questions from ESL. These results are arguably good, since, as reported by Turney, the average score of TOEFL by a large sample of students is 64.5%.

This algorithm cannot be used to obtain a list of synonyms, since it is too expensive to run for each candidate word in a dictionary because of network access times, but it may be used, for instance, to refine a list of synonyms given by another method.

2.3 Discovery of Similar Words in a Dictionary

2.3.1 Introduction

We now propose a method for automatic synonym extraction in a monolingual dictionary [Sen01, BS01]. 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 [Kle99].

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 with a given query word w , we construct a *neighborhood graph* G_w which is the subgraph of G whose vertices are those pointed to by w or pointing to w . Finally, we look in the graph G_w for vertices that are similar to the vertex w 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 [BV02, Hey01].

The problem of searching synonyms is similar to that of searching similar pages on the Web; a problem that is dealt with in [Kle99] and [DH99]. 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 2.3.3, we briefly survey two other methods that are used for comparison. We then describe in Section 2.3.4 how we have constructed a dictionary graph from *Webster’s dictionary*. In the last section we compare all methods on the following words chosen for their variety: **disappear**, **parallelogram**, **sugar**, and **science**.

2.3.2 A Generalization of Kleinberg’s Method

In [Kle99], 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 homepages are good hubs. In order to identify hubs and authorities, Kleinberg’s method 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 with 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 to 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 with G_w . The updating equations can be written as

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

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

$$1 \longrightarrow 2.$$

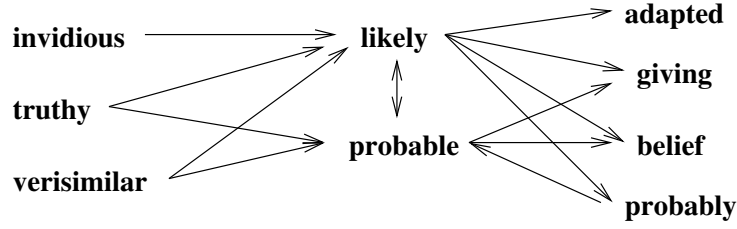
Similarly, the hub score of v can be seen as a measure of similarity between v in G and vertex 1 in the same structure graph. As presented in [BV02, Hey01], this measure of similarity can be generalized to graphs that are different from the authority-hub structure graph. We describe below an extension of the method to a structure graph with three vertices and illustrate an application of this extension to synonym extraction.

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 2.4 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 is similar to vertex 2 of the preceding graph if it points to vertices that are similar to vertex 3 and if it is pointed to by vertices that are similar to vertex 1. This mutually reinforcing definition is analogous to Kleinberg’s definitions of hubs and authorities.

The similarity between vertices in graphs can be computed as follows. With 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 to by

Figure 2.4. Subgraph of the neighborhood graph of **likely**.

i ; the scores x_i^2 are set equal to the sum of the scores x_j^3 of vertices pointed to 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 $M_w M_w^T + M_w^T M_w$.

2.3.3 Other Methods

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

The Distance Method

One possible way of defining a synonym distance is to declare that two words are close to being synonyms if they appear in the definition of many common words and have many common 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 in their definition. Let A be the adjacency matrix of the dictionary graph, and i and j be the vertices associated with two words. The distance between i and j can be expressed as

$$d(i, j) = \|A_{i,\cdot} - A_{j,\cdot}\|_1 + \|(A_{\cdot,i} - A_{\cdot,j})^T\|_1$$

where $\|\cdot\|_1$ is the l_1 vector norm. 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 chapter, we can apply this algorithm directly to the entire dictionary graph rather than to the neighborhood graph. This does, however, give very bad results: the first two synonyms of **sugar** in the dictionary graph constructed from *Webster's Dictionary* are **pigwidgeon** and **ivoride**. We show in Section 2.3.5 that much better results are achieved if we use the neighborhood graph.

ArcRank

ArcRank is a method introduced by Jan Jannink and Gio Wiederhold for building a thesaurus [JW99]; their intent was not to find synonyms but related words. An online version of their algorithm can be run from <http://skeptical.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 [BP98]. 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 that are pointed to them. Under conditions that are often satisfied in practice, the normalized ranking converges to a stationary distribution corresponding to the principal eigenvector of the adjacency matrix of the graph. This algorithm is actually slightly modified so that sources (nodes with no incoming edges, i.e., words not used in any definition) and sinks (nodes with no outgoing edges, i.e., words not defined) are not assigned extreme rankings.

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 , first select the edges starting from or arriving at w which have the best rankings and extract the corresponding incident vertices.

2.3.4 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 [OPT00] 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 *Webster’s Dictionary* are multiwords (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 they should be interpreted as single words or as a multiword (for instance, **at one** is defined but the two words **at** and **one** appear several times side by side in the dictionary in their usual meanings).

- 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. Because, 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 semiregular 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 did not 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, is excluded, 12,461 words remain: proper names (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, and so on. Our findings are reported in [Sen01].

We also decided to exclude too-frequent words in the construction of neighborhood graphs, that is, words that appear in more than L definitions (best results were obtained for $L \approx 1,000$). (The most commonly occurring words and their number of occurrences are: **of**: 68,187, **a**: 47,500, **the**: 43,760, **or**: 41,496, **to**: 31,957, **in**: 23,999, **as**: 22,529, **and**: 16,781, **an**: 14,027, **by**: 12,468, **one**: 12,216, **with**: 10,944, **which**: 10,446, **is**: 8,488, **for**: 8,188, **see**: 8,067, **from**: 7,964, **being**: 6,683, **who**: 6,163, **that**: 6,090).

2.3.5 Results

In order to be able to compare the different methods and to evaluate their relevance, we examine the first 10 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; and
4. **science**: a common and vague word. It is hard to say what to expect as a synonym. Perhaps **knowledge** is the best option.

Words in the English language belong to different parts of speech: nouns, verbs, adjectives, adverbs, prepositions, and so on. It is natural, when looking for a synonym of a word, to get only words of the same type. *Websters's Dictionary* provides the part of speech for each word. 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 WordNet [Wor98], which is handmade. The order of appearance of the words for this last source 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 to 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, in as much as this is only useful for our method, where it is interesting to note that in every example with which 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. Figures 2.5 through 2.8 give the results.

Concerning **disappear**, the distance method and our method do pretty well: **vanish**, **cease**, **fade**, **die**, **pass**, **dissipate**, **faint** are very relevant (one must not forget that verbs necessarily appear without their postposition); **dissipate** or **faint** are relevant too. However, some words such as **light** or **port** are completely irrelevant, but they appear only in the sixth, seventh, or eighth position. If we compare these two methods, we observe that our method is better: an important synonym such as **pass** has a good ranking, whereas **port** or **appear** fall from the top 10 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

	Distance	Our method	ArcRank	Wordnet
1	vanish	vanish	epidemic	vanish
2	wear	pass	disappearing	go away
3	die	die	port	end
4	sail	wear	dissipate	finish
5	faint	faint	cease	terminate
6	light	fade	eat	cease
7	port	sail	gradually	
8	absorb	light	instrumental	
9	appear	dissipate	darkness	
10	cease	cease	efface	
Mark	3.6	6.3	1.2	7.5
Std dev.	1.8	1.7	1.2	1.4

Figure 2.5. Proposed synonyms for **disappear**.

	Distance	Our method	ArcRank	Wordnet
1	square	square	quadrilateral	quadrilateral
2	parallel	rhomb	gnomon	quadrangle
3	rhomb	parallel	right-lined	tetragon
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	
Mark	4.6	4.8	3.3	6.3
Std dev.	2.7	2.5	2.2	2.5

Figure 2.6. Proposed synonyms for **parallelogram**.

gives rather poor results with words such as **eat**, **instrumental**, or **epidemic** that are imprecise.

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 Wordnet is less rich here because it focuses on a particular aspect (**quadrilateral**).

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**, and **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 Wordnet.

	Distance	Our method	ArcRank	Wordnet
1	juice	cane	granulation	sweetening
2	starch	starch	shrub	sweetener
2	cane	sucrose	sucrose	carbohydrate
4	milk	milk	preserve	saccharide
5	molasses	sweet	honeyed	organic compound
6	sucrose	dextrose	property	saccarify
7	wax	molasses	sorghum	sweeten
8	root	juice	grocer	dulcify
9	crystalline	glucose	acetate	edulcorate
10	confection	lactose	saccharine	dulcorate
Mark	3.9	6.3	4.3	6.2
Std dev.	2.0	2.4	2.3	2.9

Figure 2.7. Proposed synonyms for **sugar**.

	Distance	Our method	ArcRank	Wordnet
1	art	art	formulate	knowledge domain
2	branch	branch	arithmetic	knowledge base
3	nature	law	systematize	discipline
4	law	study	scientific	subject
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
Mark	3.6	4.4	3.2	7.1
Std dev.	2.0	2.5	2.9	2.6

Figure 2.8. Proposed synonyms for **science**.

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.

To conclude, 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 demonstrate its superiority on the basic distance method, even if the difference is not obvious for all words. The quality of the results obtained with these different methods is still quite different from that of handmade dictionaries such as Wordnet. Still, these automatic techniques show their interest, since they present more complete aspects of a word than handmade dictionaries. They can profitably be used to broaden a topic (see the example of **parallelogram**) and to help with the compilation of synonym dictionaries.

2.3.6 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. Larger neighborhood graphs could be obtained either as Kleinberg does in [Kle99] for searching similar pages on the Web, or as Dean and Henzinger do in [DH99] for the same purpose. However, such subgraphs are no longer focused on the original word. That implies that our variation of Kleinberg's algorithm "forgets" the original word and may produce irrelevant results. When we use the vicinity graph of Dean and Henzinger, 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 neighborhood graph that depends on the word itself would be appropriate. For instance, the extended vicinity graph may be used for words whose neighborhood graph has less than a fixed number of vertices, or for words whose incoming degree is small, or for words that do not belong to the largest connected component of the dictionary graph.

One may wonder whether the results obtained are specific to *Webster's Dictionary* or whether the same methods could work on other dictionaries (using domain-specific dictionaries could, for instance, generate domain-specific thesauri, the value of which was mentioned in Section 2.2), 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 fewer different meanings than in English. Besides, it is much rarer in French to have the same word for a noun and a verb than it is in English. Furthermore, the way words are defined varies from language to language. This seems to be an interesting research direction.

2.4 Conclusion

A number of different methods exist for the automatic discovery of similar words. Most of these methods are based on various text corpora and three of these are described in this chapter. Each of them may be more or less adapted to a specific problem (for instance, Crouch's techniques are more adapted to infrequent words than SEXTANT). We have also described the use of a more structured source - a monolingual dictionary - for the discovery of similar words. None of these methods is perfect and in fact none of them favorably competes with handmade dictionaries in terms of liability. Computer-written thesauri have, however, other

advantages such as their ease of being built and rebuilt. The integration of different methods, with their own pros and cons, should be an interesting research direction to look at for designing successful methods. For it is most unlikely that a single straightforward technique may solve the issue of the discovery of similar words.

Another problem of the methods presented is the vagueness of the notion of “similar word” they use. Depending on the context, this notion may or may not include the notion of synonyms, near-synonyms, antonyms, hyponyms, and so on. The distinction between these very different notions by automatic means is a challenging problem that should be addressed to make it possible to build thesauri in a completely automatic way.

References

- [BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [BS01] V.D. Blondel and P.P. Senellart. Automatic extraction of synonyms in a dictionary. Technical Report 89, Université catholique de Louvain, Louvain-la-neuve, Belgium, 2001. Presented at the Text Mining Workshop 2002 in Arlington, VA.
- [BV02] V.D. Blondel and P. Van Dooren. A measure of graph similarity between graph vertices. Technical Report, Université catholique de Louvain, Louvain-la-neuve, Belgium, 2002.
- [CL92] H. Chen and K.J. Lynch. Automatic construction of networks of concepts characterizing document databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885–902, 1992.
- [Cro90] C.J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing and Management*, 26:629–640, 1990.
- [DH99] J. Dean and M.R. Henzinger. Finding related pages in the World Wide Web. *WWW8 / Computer Networks*, 31(11–16):1467–1479, 1999.
- [Gre93] G. Grefenstette. Automatic thesaurus generation from raw text using knowledge-poor techniques. In *Making Sense of Words. Ninth Annual Conference of the UW Centre for the New OED and Text Research*. 9, 1993.
- [Gre94] G. Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic, Boston, 1994.
- [Hey01] M. Heymans. Extraction d’information dans les graphes, et application aux moteurs de recherche sur internet, Jun 2001. Université Catholique de Louvain, Faculté des Sciences Appliquées, Département d’Ingénierie Mathématique.
- [JW99] J. Jannink and G. Wiederhold. Thesaurus entry extraction from an on-line dictionary. In *Proceedings of Fusion ’99*, Sunnyvale, CA, Jul 1999.
- [Kle99] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [OPT00] The online plain text english dictionary, 2000. <http://msowww.anu.edu.au/~ralph/OPTED/>.

- [Sen01] P. P. Senellart. Extraction of information in large graphs. Automatic search for synonyms. Technical Report 90, Université catholique de Louvain, Louvain-la-neuve, Belgium, 2001.
- [SYY75] G. Salton, C.S. Yang, and C.T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975.
- [Tur01] P. D. Turney. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the European Conference on Machine Learning*, pages 491–502, 2001.
- [Wor98] Wordnet 1.6, 1998. <http://www.cogsci.princeton.edu/~wn/>.

Survey of Text Mining

Clustering, Classification, and Retrieval

Berry, M.W. (Ed.)

2004, XVII, 244 p. 46 illus., Hardcover

ISBN: 978-0-387-95563-6