Automatic Discovery of Similar Words

Pierre Senellart and Vincent D. Blondel

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 World Wide Web, and monolingual dictionaries. The underlying goal of these methods is in general the automatic discovery of synonyms. This goal, however, is most of the time too difficult to achieve since it is often hard to distinguish in an automatic way between synonyms, antonyms and, more generally, words that are semantically close to each others. Most methods provide words that are "similar" to each other, with some vague notion of semantic similarity. 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 which 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 while there may not be an entry in the thesaurus for each word in the source. Nevertheless, the purposes of both sorts of techniques are very similar and we shall therefore not distinguish much between them.

There are many applications of methods for extracting similar words. For example, in natural language processing and information retrieval, they can be used to broaden and rewrite 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 of similar words 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. The interest of such domain-specific thesauri, as opposed to general-purpose human-written synonym dictionaries, will be stressed. The question of how to combine the result of different techniques will also be broached. We then look at the particular case of the World Wide Web, whose large size and other specific features do not allow to be dealt with in the same way as more classical

corpora. In Section 3, we propose an original approach, which is based on a monolingual dictionary and uses an algorithm that generalizes an algorithm initially proposed by Kleinberg for searching the Web. Two other methods working from a monolingual dictionary are also presented. Finally, in light of this example technique, we discuss the more fundamental relations that exist between text mining and graph mining techniques for the discovery of similar words.

2 Discovery of similar words from a large corpus

Much research has been carried out about the search for similar words in textual corpora, mostly for applications in information retrieval tasks. The basic assumption of most of these approaches is that words are similar if they 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 grammatical contexts) and the way the similarity function 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 human-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, human-written synonym dictionaries are bound to be more liable, with fewer gross mistakes. In terms of the two classical measures of information retrieval, we expect computer-written thesauri to have a better *recall* (or coverage) and a lower *precision* (except for words whose meaning is highly biased by the application domain) than general-purpose human-written synonym dictionaries.

We describe below three methods which may be used to discover similar words. We do not pretend to be exhaustive, but have rather chosen to present some of the main approaches, selected for the variety of techniques used and specific intents. Variants and related methods are briefly discussed where appropriate. In Section 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 [3] and we briefly report their results. We then look at an approach proposed by Crouch [4] for the automatic construction of a thesaurus. The method is based on a term vector space model and term discrimination values [16], and is specifically adapted for words that are not too frequent. In Section 2.3, we focus on Grefenstette's SEXTANT system [9], which uses a partial syntactical analysis. We might need a way to combine the result of various different techniques for building thesauri: this is the object of Section 2.4, which describes the *ensemble* method. Finally, we consider the

particular case of the World Wide Web as a corpus, and discuss the problem of finding synonyms in a very large collection of documents.

2.1 A document vector space model

The first obvious definition of similarity with respect to a context is, given a collection of documents, 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 the document space with boolean entries indicating whether the term appears in the corresponding document. It is common in text mining to use this type of vector space model. In the dual model, terms are coordinates and documents are vectors in term space; we see 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 *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 $|\cos(\mathbf{i}, \mathbf{j})| \leq 1$, defining an angle θ with $\cos\theta = \cos(\mathbf{i}, \mathbf{j})$ as the angle between $\mathbf{i}$ and $\mathbf{j}$. Similar terms tend to occur in the same documents and the angle between them is small (they tend to be *collinear*). Thus, the cosine similarity measure is close to ± 1. On the contrary, terms with little in common do not occur in the same documents, the angle between them is close to $\pi/2$ (they tend to be *orthogonal*) and the cosine similarity measure is close to zero.

Cosine is a commonly used similarity measure. One must however not forget that the mathematical 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.

Chen and Lynch compare in [3] 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:

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

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 like *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 recall (that is, the proportion of relevant terms which are selected) than cosine and human experts. Both similarity measures exhibit similar precisions (that is, the proportion of selected term which are relevant), which are inferior to that of human experts, as expected. The asymmetry of the cluster measure seems to be here a real advantage.

2.2 A thesaurus of infrequent words

Crouch presents in [4] a method for the automatic construction of a thesaurus, consisting of classes of similar words, with only words appearing seldom in the corpus. Her purpose is to use this thesaurus to rewrite queries asked to an information retrieval system. She uses a term vector space model, which is the dual of the space used in 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 is effective and widely used is the "term frequency inverse document frequency" (*tf-idf*), that is, the number of times the word appears in the document multiplied by a (monotonous) function of the inverse of the number of documents the word appears in. Terms that appear often in a document and do not appear in many documents have therefore an important weight.

As we saw earlier, we can use a similarity measure such as cosine to characterize the similarity between two vectors (that is, 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 [16] 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 which tends to raise the distances between documents; a poor discriminator tends to lower the distances between documents; finally,

an indifferent discriminator does not change much the distances between documents. Exact or even approximate computation of all term discrimination values is an expensive task. To avoid this problem, the authors propose to use 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. Neither good discriminators (which tend to be specific to subparts of the original corpus) nor poor discriminators (which tend to be stop words or other universally apparent words) are used here.

Crouch suggests to use low frequency terms to form thesaurus classes (these classes should thus be made of indifferent discriminators). The first idea to build the thesaurus would be to cluster together these low frequency terms with an adequate clustering algorithm. This is not very interesting, however, since, by definition, one has not much information about low frequency terms. But the documents themselves may be clustered in a meaningful way. The *complete link clustering* algorithm, presented next and 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 as the minimum of all similarities (computed by the cosine measure) between pairs of documents in the two clusters—are merged together, 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, forming thus 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 do not generally deal well with such words for which we have little information.

2.3 Syntactical contexts

Perhaps the most successful methods for extracting similar words from text are based on a light syntactical analysis, and the notion of *syntactical context*: For instance, two nouns are similar if they occur as the subject or the direct object of the same verbs. We present here in detail an approach by Grefenstette [9], namely SEXTANT (Semantic EXtraction from Text via Analyzed Networks of Terms); other similar works are discussed next.

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 human-written 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 neither start, end or follow any kind of word in a verb phrase, and so on.

Parsing

Several syntactic relations (or contexts) are then extracted from the bracketed sentences, requiring five successive passes over the text. Table 1, taken from [9], shows the list of extracted relations.

Table 1. Syntactical relations extracted by SEXTANT

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 proposition 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., *he ate an apple*)
IOBJ	a noun in a prepositional phrase modifying a verb	(e.g., *the book was placed on the table*)

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 was used, but it would be more expensive too. Five passes over the text are enough to extract these relations, and since the corpus used 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 dealt with. After the parsing step, a noun has a number of attributes: all

the words which 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 are made. Each attribute is assigned a weight by:

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

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:

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

Results

Table 2. SEXTANT similar words for *case*, from different corpora

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*

Grefenstette used SEXTANT on various corpora and many examples of the results returned are available in [9]. Table 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 interest of working on a domain-specific corpus.

Table 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*.

Table 3. SEXTANT similar words for words with most contexts in Grolier's Encyclopedia animal articles

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*

Other techniques based on a light syntactical analysis

A number of works deal with the extraction of similar words from corpora with the help of a light syntactical analysis. They rely on grammatical contexts, which can be seen as 3-tuples (w, r, w'), where w and w' are two words and r characterizes the relation between w and w'. In particular, [12] and [6] propose systems quite similar to SEXTANT, and apply them to much larger corpora. Another interesting feature of these works is that the authors try and compare numerous similarity measures; [6] especially presents an extensive comparison of the results obtained with different similarity and weight measures.

Another interesting approach is presented in [15]. The *relative entropy* between distributions of grammatical contexts for each word is used as a similarity measure between these two words, and this similarity measure is used in turn for a hierarchical clustering of the set of words. This clustering provides a rich thesaurus of similar words. Only the **DOBJ** relation is considered in [15], but others can be used in the same manner.

2.4 Combining the output of multiple techniques

The techniques presented above may use different similarity measures, different parsers, or may have different inherent biases. In some contexts, using a combination of various techniques may be useful to increase the overall quality of lists of similar words. A general solution to this problem in the general context of machine learning is the use of *ensemble* methods [8]; these methods may be fairly elaborate, but a simple one (*Bayesian voting*) amounts to perform some renormalization of the similarity scores and averaging them together. Curran uses such a technique in [5] to aggregate the results of different techniques based on a light parsing; each of these uses the same similarity measure, making the renormalization step useless. Another use of the combination of different techniques is to be able to benefit from different kinds of sources: Wu and Zhou [21] extend Curran's approach to derive a thesaurus of similar words from very different sources: a monolingual dictionary (using a method similar to the distance method of Section 3.3), a monolingual corpus (using grammatical contexts) and the combination of a bilingual dictionary and a bilingual corpus with an original algorithm.

2.5 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 (see the discussion in Section 3.7). And of course it is not a domain-specific source, though domain-specific parts of the Web could be extracted by restricting ourselves to pages matching appropriate keyword queries. 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 [18]. 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 validated against 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, and each of these use the advanced search capabilities of the Altavista search engine (`http://www.altavista.com/`).

$$\begin{aligned}
\text{score}_1(j) &= \frac{\text{hits}(i \text{ AND } j)}{\text{hits}(j)} \\
\text{score}_2(j) &= \frac{\text{hits}(i \text{ NEAR } j)}{\text{hits}(j)} \\
\text{score}_3(j) &= \frac{\text{hits}\left((i \text{ NEAR } j) \text{ AND NOT } ((i \text{ OR } j) \text{ NEAR } \textit{not})\right)}{\text{hits}\left(j \text{ AND NOT}(j \text{ NEAR } \textit{not})\right)} \\
\text{score}_4(j) &= \frac{\text{hits}\left((i \text{ NEAR } j) \text{ AND } \textit{context} \text{ AND NOT } ((i \text{ OR } j) \text{ NEAR } \textit{not})\right)}{\text{hits}\left(j \text{ AND } \textit{context} \text{ AND NOT}(j \text{ NEAR } \textit{not})\right)}
\end{aligned}$$

In these expressions, hits$(\cdot)$ 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 are not separated by more than ten 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 score_2 and score_3 was introduced in order not to assign a good score to antonyms.

The four scores are presented in increasing order of quality of the corresponding results. score_3 gives the right synonym for 73.75% of the questions from TOEFL (score_4 was not applicable since no context was given) and score_4 gives the right 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 global synonym dictionary, as 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.

3 Discovery of similar words in a dictionary

3.1 Introduction

We propose now a method for automatic synonym extraction in a monolingual dictionary [17]. 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 the *HITS* algorithm initially proposed by Kleinberg for searching the Web [11].

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 [1].

The problem of searching synonyms is similar to that of searching similar pages on the Web, a problem that is dealt with in [11] and [7]. 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 concept for discovering synonyms.

In the next section, we describe our method in some detail. In Section 3.3, we briefly survey two other methods that are used for comparison. We then describe in Section 3.4 how we have constructed a dictionary graph from 1913 Webster's dictionary. We compare next the three methods on a sample of words chosen for their variety. Finally, we generalize the approach presented here by discussing the relations existing between the fields of text mining and graph mining, in the context of synonym discovery.

3.2 A generalization of Kleinberg's method

In [11], Jon Kleinberg proposes the HITS 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 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 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 score 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

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

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

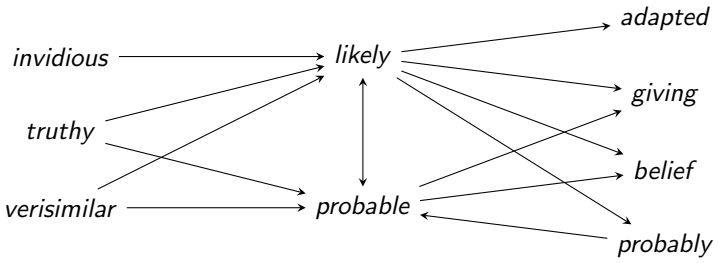

Fig. 1. Subgraph of the neighborhood graph of *likely*

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 score x_i^1 is set equal to the sum of the scores x_j^2 of all vertices j pointed by i; the score x_i^2 is 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 score x_i^3 is 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:

$$x^k \leftarrow \frac{x^k}{\|x^k\|} \qquad (k = 1, 2, 3).$$

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 eigenvector of $M_w M_w^T + M_w^T M_w$.

3.3 Other methods

In this section, we briefly describe two synonym extraction methods that will be compared to ours 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 to 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-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 here, 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* in the dictionary graph constructed from Webster's Dictionary are *pigwidgeon* and *ivoride*. We shall see in Section 3.5 that much better results are achieved if we use the neighborhood graph.

ArcRank

ArcRank is a method introduced by Jannink and Wiederhold for building a thesaurus [10]; their intent was not to find synonyms but related words. The method is based on the PageRank algorithm, used by the Web search engine Google and described in [2]. 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. 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, that is words not used in any definition) and sinks (nodes with no outgoing edges, that is 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, one selects the edges starting from or arriving to w which have the best rankings and extract the corresponding incident vertices.

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 [14] 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 website `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 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 (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. 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 inflected 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. Note that a classical stemming would here not be of use, since we do not want to merge the dictionary entries of lexically close words, such as *connect* and *connection*).

The resulting graph has $112,169$ vertices and $1,398,424$ edges, and can be downloaded at `http://pierre.senellart.com/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 [17].

We also decided to exclude stop words in the construction of neighborhood graphs, that is words that appear in more than L definitions (best results were obtained for $L \approx 1,000$).

3.5 Results

In order to be able to compare the different methods presented above (Distance, ArcRank and our method based on graph similarity) and to evaluate their relevance, we examine the first ten results given by each of them for four words, chosen for their variety.

disappear	a word with various synonyms such as *vanish*.
parallelogram	a very specific word with no true synonyms but with some similar words: *quadrilateral*, *square*, *rectangle*, *rhomb*...
sugar	a common word with different meanings (in chemistry, cooking, dietetics...). One can expect *glucose* as a candidate.

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 kind. Websters's Dictionary provides for each word its part of speech. But this presentation has not been standardized and we counted no 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 [20], which is human-written. 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 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 4, 5, 6 and 7 give the results.

Concerning *disappear*, the distance method (restricted to the neighborhood graph) 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 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 words such as *eat*, *instrumental* or *epidemic* that are out of the point.

Table 4. Proposed synonyms for *disappear*

	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

Table 5. Proposed synonyms for *parallelogram*

	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

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 here less rich 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*, *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.

Table 6. Proposed synonyms for *sugar*

	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

Table 7. Proposed synonyms for *science*

	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

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.

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. The quality of the results obtained with these different methods is still quite different to that of human-written dictionaries such as WordNet. Still, these automatic techniques show their interest, since they present more complete aspects of a word than human-written dictionaries. They can profitably be used to broaden a topic (see the example of *parallelogram*) and to help with the compilation of synonyms dictionaries.

3.6 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 [11] for searching similar pages on the Web, or as Dean and Henziger do in [7] 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 may produce irrelevant results. 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 neighborhood graph that depends 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 incoming degree 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 Webster's dictionary or whether the same methods could work on other dictionaries (using domain-specific dictionaries could for instance generate domain-specific thesauri, whose interest was mentioned in Section 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 are undoubtedly differences with other languages. For example, in French, postpositions do not exist and thus verbs do not have 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, the way words are defined vary from language to language. Despite these differences, preliminary studies on a monolingual French dictionary seem to show equally good results.

3.7 Text mining and graph mining

All three methods described for synonym extraction from a dictionary use classical techniques from text mining: stemming (in our case, in the form of a simple lemmatization), stop word removal, a vector space model for representing dictionary entries... But a specificity of monolingual dictionaries makes this vector space very peculiar: Both the dimensions of the vector space and the vectors stand for the same kind of objects, words. In other words, rows and columns of the corresponding matrix are indexed by the same set. This peculiarity makes it possible to see the dictionary, and this vector space model,

as a (possibly weighted) directed graph. This allows us to see the whole synonym extraction problem as a problem of information retrieval on graphs, for which a number of different approaches have been elaborated, especially in the case of the World Wide Web [2, 7, 11]. Thus, classical techniques from both text mining (distance between vectors, cosine similarity, tf-idf weighting...) and graph mining (cocitation count, PageRank, HITS, graph similarity measures...) can be used in this context. A study [13] on the Wikipedia online encyclopedia [19], which is similar to a monolingual dictionary, compares some methods from both worlds, along with an original approach for defining similarity in graphs based on *Green measures* of Markov chains.

A further step would be to consider *any* text mining problem as a graph mining problem, by considering any finite set of vectors (in a finite-dimensional vector space) as a directed, weighted, bipartite graph, the two partitions representing respectively the vectors and the dimensions. Benefits of this view are somewhat lower, because of the very particular nature of a bipartite graph, but some notions from graph theory (for instance, matchings, vertex covers, or bipartite random walks), may still be of interest.

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 human-written dictionaries in terms of liability. Computer-written thesauri have however other advantages such as their ease to build and maintain. We also discussed how different methods, with their own pros and cons, might be integrated.

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, etc. 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.

Acknowledgement. We would like to thank Yann Ollivier for his feedback on this work.

References

1. Vincent D. Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: applications to synonym extraction and Web searching. *SIAM Review*, 46(4):647–666, 2004.
2. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
3. Hsinchun Chen and Kevin J. Lynch. Automatic construction of networks of concepts characterizing document databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885–902, 1992.
4. Carolyn J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing and Management*, 26(5):629–640, 1990.
5. James R. Curran. Ensemble methods for automatic thesaurus extraction. In *Proc. Conference on Empirical Methods in Natural Language Processing*, Philadelphia, USA, July 2002.
6. James R. Curran and Marc Moens. Improvements in automatic thesaurus extraction. In *Proc. ACL SIGLEX*, Philadelphia, USA, July 2002.
7. Jeffrey Dean and Monika Rauch Henzinger. Finding related pages in the world wide web. In *Proc. WWW*, Toronto, Canada, May 1999.
8. Thomas G. Dietterich. Ensemble methods in machine learning. In *Proc. MCS*, Cagliari, Italy, June 2000.
9. Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Press, Boston, MA, 1994.
10. Jan Jannink and Gio Wiederhold. Thesaurus entry extraction from an on-line dictionary. In *Proc. FUSION*, Sunnyvale, USA, July 1999.
11. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
12. Dekang Lin. Automatic retrieval and clustering of similar words. In *Proc. COLING*, Montreal, Canada, August 1998.
13. Yann Ollivier and Pierre Senellart. Finding related pages using Green measures: An illustration with Wikipedia. In *Proc. AAAI*, Vancouver, Canada, July 2007.
14. The online plain text english dictionary. `http://msowww.anu.edu.au/~ralph/OPTED/`.
15. Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proc. ACL*, Columbus, USA, June 1993.
16. Gerard Salton, C. S. Yang, and Clement T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975.
17. Pierre Senellart. Extraction of information in large graphs. Automatic search for synonyms. Technical Report 90, Université catholique de Louvain, Louvain-la-neuve, Belgium, 2001.
18. Peter D. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proc. ECML*, Freiburg, Germany, September 2001.
19. Wikipedia. The free encyclopedia. `http://en.wikipedia.org/`.
20. WordNet 1.6. `http://wordnet.princeton.edu/`.
21. Hang Wu and Ming Zhou. Optimizing synonym extraction using monolingual and bilingual resources. In *Proc. International Workshop on Paraphrasing*, Sapporo, Japan, July 2003.