Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

Seminar 'Text Mining'

Similarity Thesauri and Cross-Language Retrieval

Sebastian Marius Kirsch
skirsch@moebius.inka.de

Sommersemester 2004

Abstract

This paper describes a method for constructing a thesaurus automatically from a corpus of suitable documents, using standard information retrieval methods. The resulting thesauri can be used for user-initiated query expansion, automatic query expansion, as well as cross-language retrieval.

Researchers at the Swiss Federal Institute of Technology in Zürich developed and evaluated this method in the early 90's, after publication of Schäuble and Knaus [1992].

Contents

1 Introduction

The Merriam-Webster Online Dictionary defines a thesaurus as

> **the·sau·rus** [...]
> **2a:** a book of words or of information about a particular field or set of concepts; *especially:* a book of words and their synonyms **b:** a list of subject headings or descriptors usually with a cross-reference system for use in the organization of a collection of documents for reference and retrieval

Grouping words by their meaning is one of the fundamental methods of ordering a dictionary. The first thesauri for western languages date back to the 19th century, for example P. M. Roget's 'Thesaurus of English Words and Phrases' from 1852, or D. Sanders' 'Deutscher Sprachschatz' from 1873.

This paper describes thesauri not from their linguistical aspects, but from their role in information retrieval: as a valuable tool for the user. The use of thesauri in research and retrieval is already alluded to in sense **b** of the Merriam-Webster definition.

Constructing a thesaurus by hand is a time-consuming and arduous task. As noted in Qiu and Frei [1995], a thesaurus is prone to contain errors and is hardly ever consistent; furthermore, it must be kept up-to-date continually if it is to be of any use.

A carefully crafted thesaurus can improve the effectiveness of an information retrieval system considerably and is therefore an important component. It can aid the researcher in formulating his queries more effectively and provide disambiguation of problematic terms.

A thesaurus constructed automatically from a document collection is useful for two reasons: It can include implicit knowledge about the domain contained in the documents, and it does not suffer from the problems of conventional thesauri mentioned above.

This paper starts with a short review of the vector space model of information retrieval and goes on to describe the construction and updating of a similarity thesaurus from a collection of documents. It then explains two applications of similarity thesauri in an IR system:

automatic query expansion improving retrieval effectiveness by automatically adding terms from the thesaurus to the query

cross-language retrieval retrieving documents in languages other than the language of the query

The paper wraps up with an evaluation of experiments conducted at the Swiss federal institute.

2 Background

Similarity thesauri are constructed using the vector space model of information retrieval, which I review briefly in this chapter.

2.1 The vector space model

In the vector space model, documents and queries are represented by vectors $\vec{d}, \vec{q} \in \mathbb{R}^n$ of real numbers. The components of these vectors signify the weight of indexing features contained in the document and the queries. The similarity between a query and a document is determined by a similarity measure, for example the cosine of the angle between the two vectors. (For normalized vectors, the scalar product $\vec{d} \cdot \vec{q}$ is the same as the cosine.)

The vector space model is a powerful representation for information retrieval: since queries and documents are expressed using vectors, the retrieved documents can be used to modify the query, and the similarity measure provides an implicit ranking of retrieved documents. Queries can be amended and modified by changing their components. Clustering methods for multi-dimensional data can be applied to the document vectors.

I will describe one approach to determining the weight of the indexing features in the next section.

2.2 Features, items, and weighting

In the following description, I use the tuple

$$< \Theta, F, I, ff, if > \tag{1}$$

to model the information retrieval task, where $F = f_1, \ldots, f_n$ is a set of indexing features, and $I = i_1, \ldots, i_m$ is the set of items in the collection. Θ denotes the set of tokens. (See also Sheridan et al. [1997].)

The function

$$f : \Theta \rightarrow F, \theta \mapsto f(\theta)$$

maps a token θ to an indexing feature $f(\theta)$. A second function

$$i : \Theta \rightarrow I, \theta \mapsto i(\theta)$$

maps every token to an item. (See figure 1 for the relationship between tokens, items and features.)

The *feature frequency* of a feature f_i in regard to a item i_j is the number of occurrences of the feature in the item, or

$$ff(f_i, i_j) = |\{\theta \in \Theta | f(\theta) = f_i \wedge i(\theta) = i_j\}|$$

Likewise, the *item frequency* is the number of items containing the feature f_i at least once, or

$$if(f_i) = |\{i \in I | \exists\, \theta \in \Theta : f(\theta) = f_i \wedge i(\theta) = i\}|$$

The *inverse item frequency* of a feature f_i is

$$iif(f_i) = \log\left(\frac{|I|}{if(f_i)}\right),$$

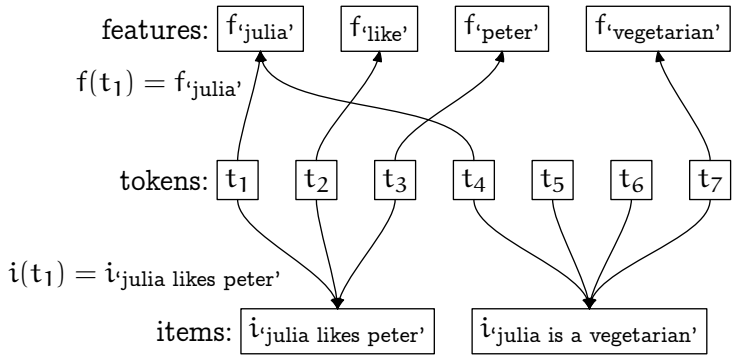


Figure 1: Relation between tokens, items and features

where $|I|$ is the total number of items in the collection. ($\mathrm{iif}(f_i) = \log\left(\frac{1+|I|}{1+\mathrm{if}(f_i)}\right)$ is used in Sheridan et al. [1997], as the formula above fails for features that do not occur in any item.)

The *maximum feature frequency* of an item is

$$\mathrm{maxff}(i_j) = \max_{f \in F} \mathrm{ff}(f, i_j).$$

A weighting scheme is a function

$$\mathrm{weight} : I \times F \to \mathbb{R}, (i_j, f_i) \mapsto \mathrm{weight}(i_j, f_i)$$

This function determines the weight of an item i_j as regards a feature f_i.

The simplest approach to weighting uses just the feature frequency: the weight of a feature is the number of times it occurs in an item.

$$\mathrm{weight}_{\mathrm{ff}}(i_j, f_i) = \mathrm{ff}(f_i, i_j)$$

However, this does not take into account the differing item lengths, nor the differing distribution of features in the item collection.

The $\mathrm{tf} \cdot \mathrm{idf}$ scheme by Salton and Buckley provides a better approach: The weight of an item i_j as regards the feature f_i is calculated from the feature frequency and the inverse item frequency as

$$\mathrm{weight}_{\mathrm{tf}\cdot\mathrm{idf}}(i_j, f_i) = \mathrm{ff}(f_i, i_j) \cdot \mathrm{iif}(f_i) \tag{2}$$

This formula incorporates our intuition that a feature is more important if it occurs often in this item, but is rare in the document collection as a whole.

Another desirable property of a weighting scheme is a normalization of weights, so that a vector of feature weights

$$\vec{i} = (\mathrm{weight}(i, f_1), \ldots, \mathrm{weight}(i, f_n))^\top$$

has a length of one. In this case, the scalar product between two vectors is equal to the cosine of the angle between the vectors. This can be achieved by adding a normalization

factor:

$$\text{weight}_{\text{norm.}}(i_j, f_i) = \frac{\text{weight}(i_j, f_i)}{\sqrt{\sum_{f \in F} (\text{weight}(i_j, f))^2}} \tag{3}$$

A modified $\text{tf} \cdot \text{idf}$ scheme is used in Qiu and Frei [1993, 1995]; it can be normalized using formula (3):

$$\text{weight}_{\text{tf} \cdot \text{idf, mod.}}(i_j, f_i) = \left(0.5 + 0.5 \frac{\text{ff}(f_i, i_j)}{\max\text{ff}(i_j)}\right) \cdot \text{iif}(f_i) \tag{4}$$

See [Baeza-Yates and Ribeiro-Neto, 1999, pages 27ff] for more information on weighting methods.

2.3 Weighting in text retrieval

In text retrieval, weighting methods like the one above are used to determine the relevance of a document in respect to a query. Analogous to formula (1), we use a tuple

$$< \Theta, T, D, \text{ff}, \text{if} >$$

that is, a set of documents $D = d_1, \ldots, d_m$ is indexed by a set of indexing terms $T = t_1, \ldots, t_n$. Terms play the roles of indexing features, and the items indexed are documents, hence the name 'tf · idf' for the weighting scheme: 'term frequency times inverse document frequency'.

In the vector space model, a document d is represented by a vector

$$\vec{d} = (\text{weight}(d, t_1), \ldots, \text{weight}(d, t_n))^\top.$$

in the *term vector space* $\text{TVS} = \mathbb{R}^{|T|}$. Queries are represented by a vector

$$\vec{q} = (\text{weight}(q, t_1), \ldots, \text{weight}(q, t_n))^\top,$$

in the term vector space. $\text{weight}(q, t_i)$ is the weight of search term t_i in the query. For a query containing tokens $\{\theta_1, \ldots, \theta_k\} \subset \Theta$, this is a vector with components

$$\vec{q}_i = \begin{cases} 1 & \text{if } t_i \in \{f(\theta_1), \ldots, f(\theta_k)\} \\ 0 & \text{otherwise} \end{cases}$$

The similarity between a document and a query is computed using the scalar product after normalizing both vectors.

2.4 The dual structure

The structure described above, with terms for features and documents for items, is widely used for text retrieval. However, it is unsuitable for determining the similarity between

terms: this weighting scheme determines the weight of a term regarding a document, not the weight of a document regarding a term.

In order to achieve this, we simply exchange terms and documents, obtaining the *dual structure*

$$< \Theta, D, T, \mathrm{ff}, \mathrm{if} >$$

where documents play the role of indexing feature, and terms are indexed by documents.

The similarity thesaurus is constructed from a set $T = t_1, \ldots, t_n$ of terms (playing the role of the items I) and $D = d_1, \ldots, d_m$ of documents (playing the role of the indexing features F). The formulae in section (2.2) are adapted as follows; the experiments in Qiu and Frei [1993, 1995] used the modified weighting scheme (4) in its normalized form:

$$\mathrm{ff}(d_i, t_j) = |\{\theta \in \Theta | d(\theta) = d_i \wedge t(\theta) = t_j\}| \tag{5}$$

$$\mathrm{if}(d_i) = |\{t \in T | \exists\, \theta \in \Theta : d(\theta) = d_i \wedge t(\theta) = t\}| \tag{6}$$

$$\mathrm{iif}(d_i) = \log\left(\frac{|T|}{\mathrm{if}(d_i)}\right) \tag{7}$$

$$\mathrm{maxff}(t_j) = \max_{d \in D} \mathrm{ff}(d, t_j) \tag{8}$$

$$\mathrm{weight}(t_j, d_i) = \frac{\left(0.5 + 0.5\frac{\mathrm{ff}(d_i, t_j)}{\mathrm{maxff}(t_j)}\right) \cdot \mathrm{iif}(d_i)}{\sqrt{\sum_{d \in D}\left(\left(0.5 + 0.5\frac{\mathrm{ff}(d, t_j)}{\mathrm{maxff}(t_j)}\right) \cdot \mathrm{iif}(d)\right)^2}} \tag{9}$$

The reader will note that as regards the feature frequency, the result is the same in the dual structure as in the conventional structure.

Using the dual structure, a term t is represented by a vector

$$\vec{t} = (\mathrm{weight}(t, d_1), \ldots, \mathrm{weight}(t, d_m))^\top,$$

in the *document vector space* $DVS = \mathbb{R}^{|D|}$, where $\mathrm{weight}(t, d_i)$ signifies the weight of the term t as regards the document d_i in the dual structure. A similarity measure such as the scalar product determines the similarity between two terms:

$$\mathrm{SIM}(t_p, t_q) := \vec{t}_p \cdot \vec{t}_q \tag{10}$$

3 Constructing and Updating

Similarity thesauri can be readily integrated into most IR systems that use the vector space model: most of the indexing structures needed for construction will already be available. I will discuss the initial construction as well as updating the thesaurus in this section.

3.1 Constructing from scratch

In most information retrieval and data mining systems, disk access quickly becomes the limiting factor for the performance of the whole system. A naïve algorithm, using formula (10), needs $O(|T|^2)$ disk accesses to compute the similarity between all term-term

pairs, because computation of the weights requires looping over the whole document collection.

However, one can readily split (9) into an unnormalized weight and a normalization coefficient in the denominator, similar to formula (3):

$$\text{weight}'(t_j, d_i) = \left(0.5 + 0.5\frac{\text{ff}(d_i, t_j)}{\text{maxff}(t_j)}\right) \cdot \text{iif}(d_i) \tag{11}$$

$$\begin{aligned} \text{SIM}(t_i, t_j) &= \vec{t_i} \cdot \vec{t_j} \\ &= \sum_{d \in D} \text{weight}(t_i, d) \cdot \text{weight}(t_j, d) \\ &= \sum_{d \in D} \frac{\text{weight}'(t_i, d)}{\sqrt{\sum_{d \in D} (\text{weight}'(t_i, d))^2}} \frac{\text{weight}'(t_j, d)}{\sqrt{\sum_{d \in D} (\text{weight}'(t_j, d))^2}} \\ &= \frac{\sum_{d \in D} \text{weight}'(t_i, d) \cdot \text{weight}'(t_j, d)}{\sqrt{\left(\sum_{d \in D} (\text{weight}'(t_i, d))^2\right) \cdot \left(\sum_{d \in D} (\text{weight}'(t_i, d))^2\right)}} \end{aligned} \tag{12}$$

Obviously, the term-term similarities can be computed by looping over the document collection only once. We call the numerator $\text{sim}(t_i, t_j)$, the *unnormalized similarity*:

$$\text{sim}(t_i, t_j) = \sum_{d \in D} \text{weight}'(t_i, d) \cdot \text{weight}'(t_j, d) \tag{13}$$

and the denominator is composed of the *global normalization coefficients* $c(t_i), c(t_j)$ of terms t_i and t_j:

$$c(t) = \sum_{d \in D} (\text{weight}'(t, d))^2 \tag{14}$$

The normalized similarity is then expressed as

$$\text{SIM}(t_i, t_j) = \text{SIM}(t_j, t_i) = \frac{\text{sim}(t_i, t_j)}{\sqrt{c(t_i) \cdot c(t_j)}}$$

The algorithm following from this expression is found in figure 2; it needs $O(m)$ disk accesses and $O(md^2 + n^2)$ computations, where m is the number of documents, d the average document length and n the number of terms.

3.2 Updating a similarity thesaurus

Constructing a similarity thesaurus is computationally expensive, but it is a one-time expense. After initial construction, the thesaurus can be used with negligible computational cost.

As new documents are added to the collection and old ones are deleted, the domain knowledge contained in the collection changes. Since the thesaurus contains domain knowledge, it will become out of date and will be less effective.

initialization
$c \equiv sim \equiv SIM \equiv 0$

calculation of within-document weights and normalization coefficients
for every document d **do**
    read document d from disk
    for every term t_i in document d **do**
        $c(t_i) := c(t_i) + (weight'(t_i, d))^2$
        for every term t_j in d with $i < j$ **do**
            $sim(t_i, t_j) := sim(t_i, t_j) + weight'(t_i, d) \cdot weight'(t_j, d)$
        end
    end
end

normalization
for every term t_i in the collection **do**
    for every term t_j with $i < j$ and $sim(t_i, t_j) > 0$ **do**
        $SIM(t_i, t_j) := SIM(t_j, t_i) := \frac{sim(t_i, t_j)}{\sqrt{c(t_i) \cdot c(t_j)}}$
    end
end

Figure 2: Algorithm for constructing a similarity thesaurus

By saving the unnormalized similarities (13) and the normalization coefficients (14) in a help file during the initial construction of the thesaurus, we can facilitate updating the thesaurus. We denote the added documents with D_+ and the removed documents with D_- and update the $\mathrm{sim}(t_i, t_j)$ and $c(t_i)$ values:

$$\begin{aligned}\mathrm{sim}(t_i, t_j) :=\ & \mathrm{sim}(t_i, t_j) \\ & + \sum_{d \in D_+} \mathrm{weight}'(t_i, d) \cdot \mathrm{weight}'(t_j, d) \\ & - \sum_{d \in D_-} \mathrm{weight}'(t_i, d) \cdot \mathrm{weight}'(t_j, d) \\ c(t_i) :=\ & c(t_i) + \sum_{d \in D_+} (\mathrm{weight}'(t_i, d))^2 - \sum_{d \in D_-} (\mathrm{weight}'(t_i, d))^2\end{aligned}$$

The similarity values $\mathrm{SIM}(t_i, t_j)$ only need to be recomputed for terms that actually occur in the added or removed documents; the other values remain unchanged. This is obviously much more efficient than reconstructing the entire thesaurus.

Unfortunately this method of incrementally constructing the thesaurus is not equivalent to constructing from scratch: The maximum feature frequency $\mathrm{maxff}(t)$ of a term t may change when documents are added and removed, as may the inverse item frequency $\mathrm{iif}(d)$ of a document d. (Recall that $\mathrm{maxff}(t)$ is the maximum number of times a term occurs in a document, and $\mathrm{iif}(d)$ depends of the number of indexing terms.)

We solve this problem by adopting a simpler weighting scheme that is independent of the number of terms and of the maximum document length. We change (7) to

$$\mathrm{iif}(d_i) = \frac{1}{\log(\mathrm{if}(d_i) + 1)} \tag{15}$$

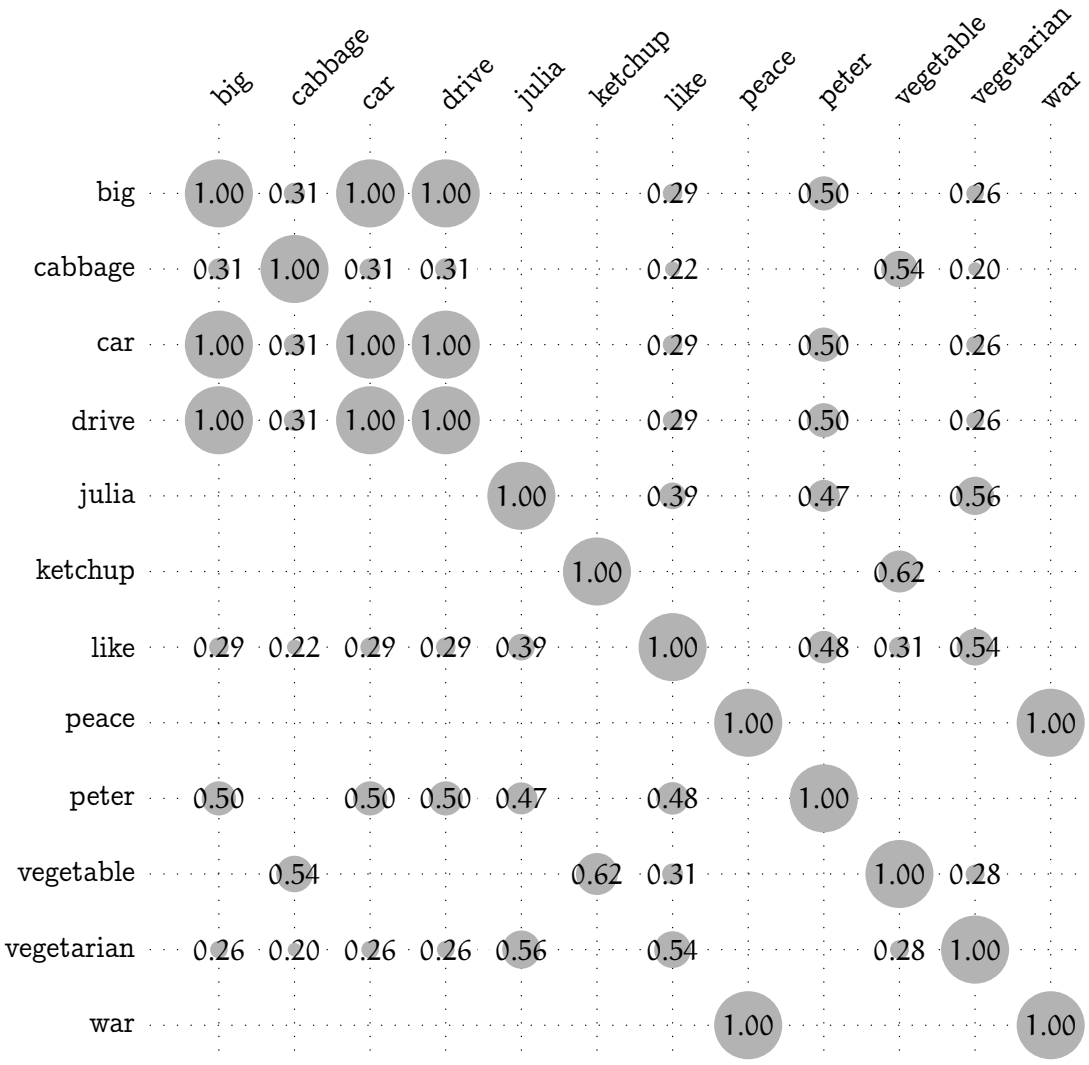
and use the original $\mathrm{tf} \cdot \mathrm{idf}$ weighting from formula (2):

$$\mathrm{weight}'(t_j, d_i) = \mathrm{ff}(d_i, t_j) \cdot \mathrm{iif}(d_i)$$

This new weighting scheme also takes into account the document length, but it is independent of the number of terms in the collection. A similarity thesaurus constructed using this weighting scheme can be constructed incrementally by adding and removing documents without loss of precision. A sample thesaurus is found in figure 3.

4 Query Expansion

An important function of a thesaurus in an IR system is supplying additional search terms for a query. The user can either browse through the thesaurus himself, or the IR system can use the domain knowledge in the thesaurus to choose additional search terms automatically with appropriate weighting. Qiu and Frei [1993, 1995] focus on the second task, because it does not require additional user input and is thus easier to quantify and evaluate.



Items big, cabbage, car, drive, julia, ketchup, like, peace, peter, vegetable, vegetarian, war

Features peter drives a big car, julia likes peter, julia is a vegetarian, vegetarians like vegetables, cabbage is a vegetable, big vegetarians who like cabbage do not drive cars, war is peace, ketchup is a vegetable

Figure 3: A sample similarity thesaurus, constructed using a weighting scheme with (2) and (15)

In our model, similarities between terms are expressed in the *document vector space* DVS; however, a query vector is a member of the term vector space TVS. Therefore, a query vector has to be mapped to the DVS for query expansion. The *concept* of the query is expressed in the DVS as a virtual term vector

$$\vec{q}_c = \sum_{t \in T} \text{weight}(q, t) \cdot \vec{t}$$

The similarity between this virtual term vector and term vectors corresponding to actual terms in the document collection is again expressed using the scalar product:

$$\text{simqt}(q, t_i) := \vec{q}_c \cdot \vec{t}_i = \sum_{t \in T} \text{weight}(q, t) \cdot \vec{t} \cdot \vec{t}_i$$

Note that $\vec{t} \cdot \vec{t}_i$ is the term-term similarity $\text{SIM}(t, t_i)$ contained in the similarity thesaurus, and is already precomputed.

We expand the original query by a query expansion vector

$$\vec{q}_e = \left(\frac{\text{simqt}(q, t_1)}{\sum_{t \in T} \text{weight}(q, t)}, \ldots, \frac{\text{simqt}(q, t_n)}{\sum_{t \in T} \text{weight}(q, t)}\right)^\top;$$

the resulting expanded query is

$$\vec{q}_{\text{exp.}} = \vec{q} + \vec{q}_e$$

This vector needs to be normalized before it can be compared to the document vectors.

Qiu and Frei [1993] call this method 'concept-based query expansion'. They argue that the *concept* of a query is not contained in the search terms alone, but rather in the relation between the search terms and the terms in the collection. A similarity thesaurus captures the relationship between the terms, and is thus suitable for determining the concept of the query.

4.1 Expanding a query

Using the thesaurus in figure 3, we use the following query vector for a query about 'julia vegetable':

$$\vec{q} = (0, 0, 0, 0, \underset{\text{julia}}{1}, 0, 0, 0, 0, \underset{\text{vegetable}}{1}, 0, 0)$$

The simqt values are as follows:

$$\begin{aligned}
\text{simqt}(\vec{q}, \text{cabbage}) &= \text{SIM}(\text{vegetable}, \text{cabbage}) = 0.54 \\
\text{simqt}(\vec{q}, \text{julia}) &= \text{SIM}(\text{julia}, \text{julia}) = 1 \\
\text{simqt}(\vec{q}, \text{ketchup}) &= \text{SIM}(\text{vegetable}, \text{ketchup}) = 0.62 \\
\text{simqt}(\vec{q}, \text{like}) &= \text{SIM}(\text{julia}, \text{like}) + \text{SIM}(\text{vegetable}, \text{like}) = 0.39 + 0.31 = 0.70 \\
\text{simqt}(\vec{q}, \text{peter}) &= \text{SIM}(\text{julia}, \text{peter}) = 0.47 \\
\text{simqt}(\vec{q}, \text{vegetable}) &= \text{SIM}(\text{vegetable}, \text{vegetable}) = 1 \\
\text{simqt}(\vec{q}, \text{vegetarian}) &= \text{SIM}(\text{julia}, \text{vegetarian}) + \text{SIM}(\text{vegetable}, \text{vegetarian}) \\
&= 0.56 + 0.28 = 0.84
\end{aligned}$$

We arrive at an expanded query of

$$\vec{q}_{\text{exp.}} = (0, \underset{\text{cabbage}}{0.27}, 0, 0, \underset{\text{julia}}{1.5}, \underset{\text{ketchup}}{0.31}, \underset{\text{like}}{0.35}, 0, \underset{\text{peter}}{0.24}, \underset{\text{vegetable}}{1.5}, \underset{\text{vegetarian}}{0.42}, 0)$$

The expanded query still contains 'julia' and 'vegetable' as the terms with the highest weight, but it also searches for documents containing 'vegetarian', 'like', 'ketchup', 'cabbage' and 'peter', because those terms are similar to the query terms.

4.2 Choosing terms for query expansion

As noted in Qiu and Frei [1993], most term-term pairs in the examined collections have a very low similarity value (between $0 - 0.2$), with a select few having a higher similarity. How to choose the terms for query expansion is an important question: adding all the low-similarity terms to the expanded query would lead to a severely reduced performance, as the performance of an IR systems depends crucially on the number of terms in the query. (Information retrieval requires the manipulation of sparse matrices with millions of rows and columns; performance of the system can only be acceptable if the great majority of values in these matrices is zero.)

Setting a weight threshold is similarly problematic, as it does not allow to predict the number of added terms, and thus the performance. The recommended way is to choose the top r similar terms to the query.

5 Cross-Language Retrieval

Cross-language information retrieval is a topic of increasing importance in politics, law, research – and business, in this age of globalization. While English may be the lingua franca of the internet, in most other areas of life day-to-day affairs are conducted in the local language. Legislation is always in the local language, and multilingual countries like Switzerland and Belgium have several official languages.

Switzerland, where the initial research on similarity thesauri was conducted, has four official languages: French, German, Italian and Romansch. With the eastern expansion on May 1st, 2004, the number of official languages in the European Union rose from eleven to twenty.[1] The ISO-639-2 standard lists codes for about 500 of the most common languages.

A host of information is published in these languages every day, and a researcher who is not limited to searching in a single language has a competitive advantage.

Very often, a researcher will have a working knowledge of several languages, or can rely on machine translation services. However, his knowledge may not be good enough to allow the formulation of successful queries in all of those languages. Cross-language retrieval can help in finding relevant documents.

[1] http://www.europa.eu.int/comm/education/policies/lang/languages/lang/europeanlanguages_en.html

How to integrate cross-language retrieval into the formalism from section 2.2 is shown in the next section.

5.1 Feature structures and subsumption

The description of how to construct a multi-lingual thesaurus in Sheridan et al. [1997] is sketchy at best. Here we use techniques from computational linguistics for the theoretical foundation of cross-language IR.

In order to process multilingual documents, we augment the features and items introduced in section 2.2 by feature structures. Feature structures consist of attribute-value pairs, the values being simple types or other feature structures. (For more information about feature structures and their role in computational linguistics, see Padó [2000].)

Feature structures are usually written down using *attribute-value matrices* (AVMs). For example, the feature structure corresponding to the sentence 'julia likes peter' is

$$\begin{bmatrix}\text{lemma:} & \text{julia likes peter}\end{bmatrix}$$

If this sentence is found in an english text, we add the language information to the feature structure:

$$\begin{bmatrix}\text{lemma:} & \text{julia likes peter}\\ \text{language:} & \text{english}\end{bmatrix}$$

The feature structure for the name 'julia' is

$$j = \begin{bmatrix}\text{lemma:} & \text{julia}\end{bmatrix}$$

If this name occurs in an english text, we add the language to the feature structure. Likewise, if our indexing component is able to detect that 'julia' is a proper name, we also add this information:

$$j' = \begin{bmatrix}\text{lemma:} & \text{julia}\\ \text{language:} & \text{english}\\ \text{proper name:} & +\end{bmatrix}$$

A feature structure f is said to *subsume* f' if all the attributes in f also occur in f', and furthermore the values of the features in f subsume the values of the corresponding features in f'. For example, we say that $j \sqsubseteq j'$ ('j subsumes j''), or

$$\begin{bmatrix}\text{lemma:} & \text{julia}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{lemma:} & \text{julia}\\ \text{language:} & \text{english}\\ \text{proper name:} & +\end{bmatrix}$$

because all the attributes in j are present in j' and have the same values in j'. We also say that j' is *more specific* than j, and j is *more general* than j'

Subsumption is a partial order on feature structures, that is, a relation with the following properties:

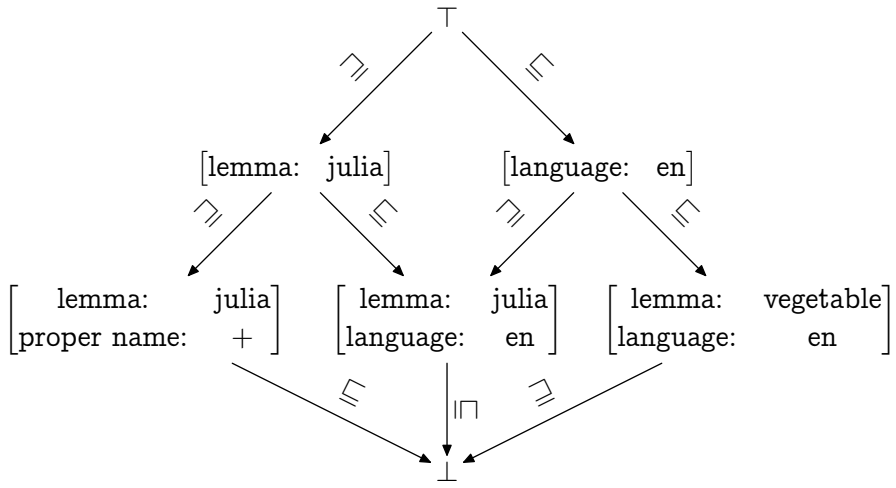


Figure 4: Feature structures form a lattice under the subsumption order.

1. reflexive: $f \sqsubseteq f$ for all feature structures
2. transitive: $f \sqsubseteq f'$ and $f' \sqsubseteq f''$ implies that $f \sqsubseteq f''$
3. anti-symmetrical: $f \sqsubseteq f'$ and $f' \sqsubseteq f$ implies that $f = f'$

Feature structures form a lattice under the subsumption order.

The most specific element in the lattice is $\bot$ ('bottom'), and the most general element is $\top$ ('top'). For every feature structure holds $\top \sqsubseteq f \sqsubseteq \bot$.

5.2 Processing multi-lingual documents

For processing multi-lingual documents, we add the language information to all index terms. We use either supplied language information, or perform language detection, for example using n-grams. (The 300 most frequent n-grams usually correlate highly to the language; see Cavnar and Trenkle [1994].) Language detection can be performed on document level, paragraph level or sentence level, with decreasing accuracy. Trying to determine the language of single words is usually not feasible.

We redefine feature frequency and item frequency in terms of subsumption:

$$\mathrm{ff}(f_i, i_j) = |\{\theta \in \Theta | f_i \sqsubseteq f(\theta) \wedge i_j \sqsubseteq i(\theta)\}| \tag{16}$$

$$\mathrm{if}(f_i) = |\{i \in I | \exists\, \theta \in \Theta : f_i \sqsubseteq f(\theta) \wedge i_j \sqsubseteq i(\theta)\}| \tag{17}$$

A feature f_i is deemed to occur in an item i_j for every token θ where

1. f_i is equal to or more general than that token's feature $f(\theta)$, and
2. i_j is equal to or more general than that token's item $i(\theta)$.

A sample cross-lingual similarity thesaurus is found in figure 5. It was produced from the sentences in figure 3 and their German translations. The words in the english sentences are found at the top, and the words in the german translation are on the right. As can be seen from the diagonal, the similarity values between the English words and their German translations are very high. The words that are similar to each other in the monolingual thesaurus tend to have a comparable similarity value to the German counterparts in the cross-lingual thesaurus.

5.3 Choosing a suitable collection

The ideal collection for construction of a cross-lingual similarity thesaurus is a collection of multilingual documents, ie. of documents containing passages in more than one language. However, collections like that are scarce.

Parallel or comparable corpora are much easier to find, and can be used to construct a surrogate multilingual collection. *Parallel corpora* are collections which contain translationally equivalent versions of the same document in several languages; they are an important tool in corpus-based linguistics. *Comparable corpora* contain documents about the same topic in several languages.

Documents from these corpora are aligned and then combined into multilingual documents; this is easy for parallel corpora (document-level alignment), but harder for comparable corpora. How to align documents from comparable corpora with a minimum of human intervention is a topic of ongoing research.

6 Evaluation

The performance of information retrieval systems is usually measured by *precision* and *recall*, using standard test collections and queries. The test collections come with a set of standard queries and information about which documents are relevant to the queries. Whether a document is relevant to a query is usually determined by an expert.

For each query, the collection contains a set of relevant documents R; the information retrieval system produces a set of documents A that it deems relevant to the query.

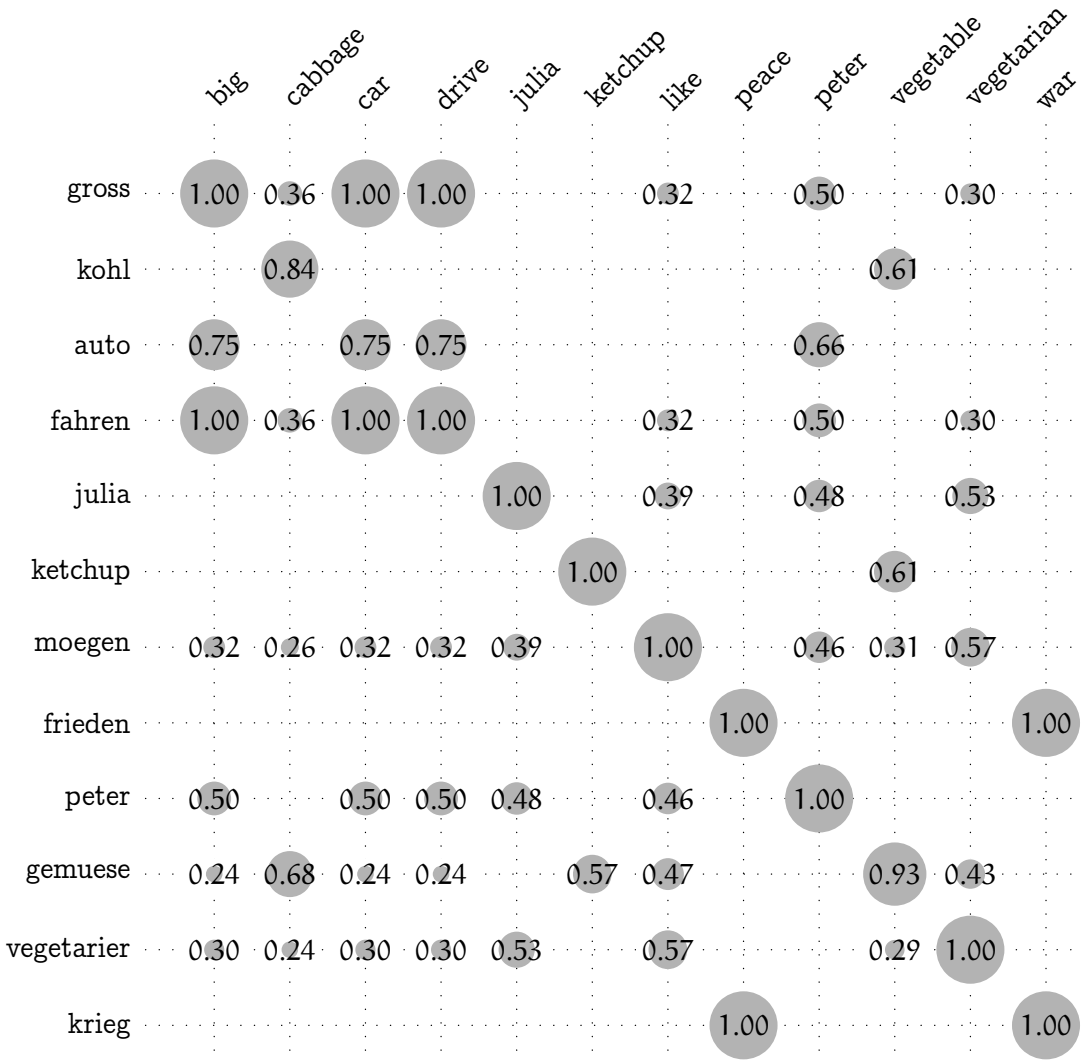
We call the *recall level* the percentage of relevant documents actually retrieved by the system:

$$\text{Recall} = \frac{|R \cap A|}{|R|}$$

This measure is closely tied to the *precision* of the result set, or what percentage of the result set is relevant to the query:

$$\text{Precision} = \frac{|R \cap A|}{|A|}$$

These two measures correlate because of the ranking methods of IR systems: In order to get more relevant documents, ie. increase the recall level, we retrieve more of the lower-



Items big, cabbage, car, drive, julia, ketchup, like, peace, peter, vegetable, vegetarian, war, gross, kohl, auto, fahren, julia, ketchup, moegen, frieden, peter, gemuese, vegetarier, krieg

Features [peter drives a big car, peter faehrt ein grosses auto], [julia likes peter, julia mag peter], [julia is a vegetarian, julia ist vegetarierin], [vegetarians like vegetables, vegetarier moegen gemuese], [cabbage is a vegetable, kohl ist gemuese], [big vegetarians who like cabbage do not drive cars, grosse vegetarier die gemuese moegen fahren keine autos], [war is peace, krieg ist frieden], [ketchup is a vegetable, ketchup ist gemuese]

Figure 5: A cross-lingual similarity thesaurus

ranked documents. However, the precision of the result set will suffer, since there are also more documents among the lower-ranked results that are *not* relevant to the query.

Precision is usually measured at standard recall levels: We retrieve enough documents to get to the required recall level and then measure the precision. This results in a graph as in figure 7: As the recall level increases. the precision of the result set decreases. The precision figure for recall level zero is obtained by interpolation.

The *average precision* is the precision of an IR system averaged over a range of standard recall levels, for example 0.25, 0.5 and 0.75, or 0.0–1.0 in steps of 0.1.

(Note that the recall level is the same as the *true positive rate* in the ROC formalism that has been popular lately for evaluating machine learning algorithms. Unfortunately, the *false positive rate* cannot be determined from precision and recall figures alone, thus a direct conversion between the two measures is not possible.)

For more information on how to compute and interpolate precision and recall, see [Baeza-Yates and Ribeiro-Neto, 1999, pages 73ff].

6.1 Query expansion

Qiu and Frei [1993, 1995] used the following collections for evaluating query expansion by similarity thesauri:

Medline a collection of article excerpts from medical journals. Also known as 'MED'.

CACM a collection of abstracts from 'Communications of the ACM'.

NPL a collection of about 10000 document titles

These and other collections are available for download at the University of Glasgow.[2]

Measuring 1–3 MB in size, these test collections are tiny when compared to current test collections. (The current TREC corpus measures about 8 GB, and the Reuters Corpus vol. 1 about 2.5 GB.)

The main characteristics of the three collections are given in table 1, along with the average improvement of precision using the expanded queries. The average precision was taken at three representative recall levels of 0.25, 0.5 and 0.75.

The figures indicate that the expanded queries yield a considerable improvement of up to 30% for the largest collection. Furthermore, they suggest that the improvement increases with the size of the collection. (The largest collection used, the NPL collection, is still tiny compared to today's standards, at 3 MB.)

Figure 6 shows that for the largest collection, the improvement increases with the number of additional terms, whereas for the two smaller collections, the improvement peaks at a number of 80–100 terms and then decreases. This may be due to the fact that more search terms are needed to distinguish effectively between the documents in the larger collection. If this trend is shown to continue, query expansion is bound to bring even better results for today's collections ranging in the gigabytes of data and millions of documents.

[2] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/

Table 1: Improvement using expanded queries (from [Qiu and Frei, 1993, page 6])

Collection	Medline	CACM	NPL
documents	1033	3240	11429
queries	30	52	93
average precision of original queries	0.5446	0.2718	0.1818
number of additional terms	80	100	800
average precision of expanded queries	0.6443	0.3339	0.2349
improvement	18.31%	22.85%	29.21%

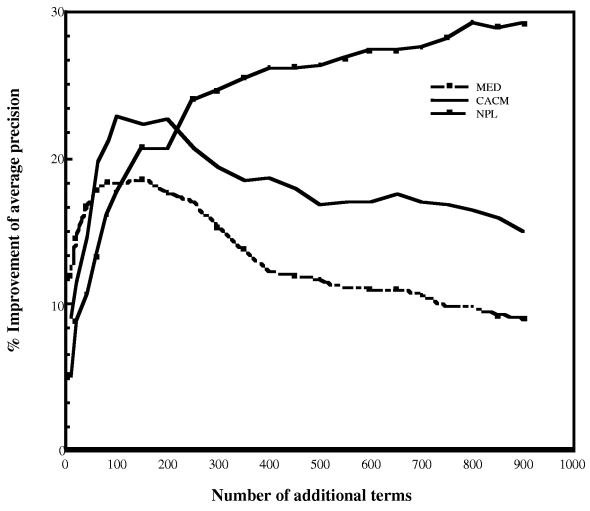


Figure 6: Improvement of query precision by adding similar terms (reproduced from [Qiu and Frei, 1993, page 6])

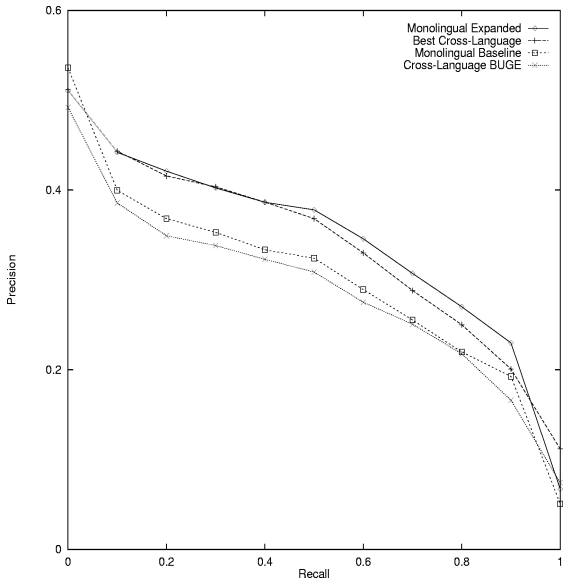


Figure 7: Experimental results of cross-lingual thesauri (reproduced from [Sheridan et al., 1997, page 13].)

6.2 Cross-language retrieval

In order to evaluate the cross-lingual thesaurus, the researchers at the Swiss Federal Institute of Technology found a perfect source for a parallel corpus in the Swiss legal system: As Switzerland has four official languages, all the federal laws are available in all the languages. They can be trivially aligned by their number. Each collection contains about 60000 documents.

The system was tested on the decisions of the Swiss federal court of justice since 1975. The full decisions of the federal court are always given in the local language, with a one-paragraph summary available in French, German and Italian.

Searching the decisions of the federal court is an ideal application for cross-language retrieval: A lawyer may not have sufficient knowledge of all official languages for finding all the relevant articles. However, missing a relevant article in a different language is not an option, as it may well provide a crucial argument.

Interestingly, the cross-language retrieval sometimes performed better than the monolingual baseline in the experiments; this can be attributed to the fact that cross-language IR is a form of query expansion. As can be seen from figure 7, cross-language retrieval performed on par with the expanded monolingual queries, and outperformed the unexpanded queries at all recall levels.

This figure also illustrates the importance of choosing a good collection: The thesaurus constructed from the collection of Swiss federal law produced the best restults. The 'BUGE' cross-lingual thesaurus was constructed from a much smaller collection – only 8000 documents – from the same domain; it produced markedly worse results.

Today, the cross-language retrieval system for court decisions is part of the website of

the Swiss federal court[3] and has proven itself as a valuable tool in Swiss jurisprudence.

In practice however, similarity thesauri are only feasible as the sole means of cross-language IR for collections of one million documents and upwards, according to Schäuble [2004]. For most other systems, they will have to be augmented with machine translation services. The results from Braschler and Schäuble [2001] indicate that the combination of machine translation and similarity thesauri can improve the average precision by another 5–10% when compared to machine translation alone.

7 Conclusion

In the course of this paper, we have seen a number of reasons why similarity thesauri are a valuable approach to the problem of automatic thesaurus construction:

Theoretical foundations Similarity thesauri have a very elegant formulation in the vector space model; all techniques available for working with documents and queries in the vector space model can easily be applied to thesaurus construction.

Performance Similarity thesauri can be constructed with $O(n)$ disk accesses; furthermore, they can be updated incrementally as documents are added to and removed from the collection.

Improved precision Automatic query expansion shows a consistent improvement of query performance, where earlier approaches using co-occurence data failed.

Ease of integration Similarity thesauri are easy to integrate into an existing IR system using the vector space model, as most of the indexing structures are already available.

Multiple use A similarity thesaurus can be used for both automatic query expansion and manual expansion.

Cross-language retrieval Similarity thesauri constructed from a multilingual document collection can be used to facilitate cross-language retrieval.

> *Sample code for the construction of mono- and crosslingual similarity thesauri is available from* `http://sites.inka.de/moebius/comp/simthes/`

[3] `http://www.bger.ch/index/juridiction/jurisdiction-inherit-template/jurisdiction-recht/jurisdiction-recht-leitentscheide1954-direct.htm`

References

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999. http://www.sims.berkeley.edu/~hearst/irbook/.

Martin Braschler and Peter Schäuble. Experiments with the Eurospider retrieval system for CLEF 2000. In Carol Peters, editor, *Cross language information retrieval and evaluation: Workshop of the Cross-Language Evaluation Forum, CLEF 2000, Lisbon, Portugal, September 21–22, 2000*, number 2069 in Lecture Notes in Computer Science, pages 140–148. Springer, 2001. ISBN 3-540-42446-6.

William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994. http://citeseer.ist.psu.edu/68861.html.

Sebastian Padó. Grammatikformalismen. http://www.coli.uni-sb.de/~pado/personal/ps/GF.ps.gz, 2000. Lecture notes.

Yonggang Qiu and Hans-Peter Frei. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, US, 1993. http://citeseer.ist.psu.edu/qiu93concept.html.

Yonggang Qiu and Hans-Peter Frei. Improving the retrieval effectiveness by a similarity thesaurus. Technical Report 225, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1995. http://citeseer.ist.psu.edu/qiu94improving.html.

Peter Schäuble. Personal communications, 2004.

Peter Schäuble and Daniel Knaus. The various roles of information structures. In Otto Opitz, editor, *Information and Classification: Concepts, Methods and Applications. Proceedings of the 16th Annual Conference of the Gesellschaft für Klassifikation*, pages 282–290, Berlin, 1992. Springer.

Páraic Sheridan, Martin Braschler, and Peter Schäuble. Cross-language information retrieval in a multi-lingual legal domain. In Carol Peters and Costantino Thanos, editors, *Proceedings of ECDL-97, 1st European Conference on Research and Advanced Technology for Digital Libraries*, pages 253–268, Pisa, IT, 1997. http://citeseer.ist.psu.edu/sheridan97crosslanguage.html.