15 *Topics in Information Retrieval*

INFORMATION RETRIEVAL (IR) RESEARCH is concerned with developing algorithms and models for retrieving information from document repositories. IR might be regarded as a natural subfield of NLP because it deals with a particular application of natural language processing. (Traditional IR research deals with text although retrieval of speech, images and video are becoming increasingly common.) But in actuality, interactions between the fields have been limited, partly because the special demands of IR were not seen as interesting problems in NLP, partly because statistical methods, the dominant approach in IR, were out of favor in NLP.

With the resurgence of quantitative methods in NLP, the connections between the fields have increased. We have selected four examples of recent interaction between the fields: probabilistic models of term distribution in documents, a problem that has received attention in both Statistical NLP and IR; discourse segmentation, an NLP technique that has been used for more effective document retrieval; and the Vector Space Model and Latent Semantic Indexing (LSI), two IR techniques that have been used in Statistical NLP. Latent Semantic Indexing will also serve as an example of *dimensionality reduction*, an important statistical technique in itself. Our selection is quite subjective and we refer the reader to the IR literature for coverage of other topics (see section 15.6). In the following section, we give some basic background on IR, and then discuss the four topics in turn.

15.1 Some Background on Information Retrieval

The goal of IR research is to develop models and algorithms for retrieving information from document repositories, in particular, textual information.

AD-HOC RETRIEVAL PROBLEM

The classical problem in IR is the *ad-hoc retrieval problem*. In ad-hoc retrieval, the user enters a query describing the desired information. The system then returns a list of documents. There are two main models.

EXACT MATCH
BOOLEAN QUERIES

Exact match systems return documents that precisely satisfy some structured query expression, of which the best known type is *Boolean queries*, which are still widely used in commercial information systems. But for large and heterogeneous document collections, the result sets of exact match systems usually are either empty or huge and unwieldy, and so most recent work has concentrated on systems which rank documents according to their estimated relevance to the query. It is within such an approach that probabilistic methods are useful, and so we restrict our attention to such systems henceforth.

An example of ad-hoc retrieval is shown in figure 15.1. The query is '"glass pyramid" Pei Louvre,' entered on the internet search engine Alta Vista. The user is looking for web pages about I. M. Pei's glass pyramid over the Louvre entrance in Paris. The search engine returns several relevant pages, but also some non-relevant ones – a result that is typical for ad-hoc searches due to the difficulty of the problem.

Some of the aspects of ad-hoc retrieval that are addressed in IR research are how users can improve the original formulation of a query interactively, by way of *relevance feedback*; how results from several text databases can be merged into one result list (*database merging*); which models are appropriate for partially corrupted data, for example, OCRed documents; and how the special problems that languages other than English pose can be addressed in IR.

RELEVANCE FEEDBACK
DATABASE MERGING

Some subfields of information retrieval rely on a training corpus of documents that have been classified as either relevant or non-relevant to a particular query. In *text categorization*, one attempts to assign documents to two or more pre-defined categories. An example is the subject codes assigned by Reuters to its news stories (Lewis 1992). Codes like CORP-NEWS (corporate news), CRUDE (crude oil) or ACQ (acquisitions) make it easier for subscribers to find stories of interest to them. A financial analyst interested in acquisitions can request a customized newsfeed that only delivers documents tagged with ACQ.

TEXT CATEGORIZATION

FILTERING
ROUTING

Filtering and *routing* are special cases of text categorization with only

[AltaVista] [Advanced Query] [Simple Query] [Private eXtension Products] [Help with Query]

Search the **Web Usenet**
Display results **Compact Detailed**

Tip: When in doubt use lower-case. Check out Help for better matches.

```
Word count: glass pyramid:  about 200; Pei:9453; Louvre:26578
```

Documents 1-10 of about 10000 matching the query, best matches first.

Paris, France
Paris, France. Practical Info.-A Brief Overview. Layout: One of the most densely populated cities in Europe, Paris is also one of the most accessible,...
http://www.catatravel.com/paris.htm - size 8K - 29 Sep 95

Culture
Culture. French culture is an integral part of France's image, as foreign tourists are the first to acknowledge by thronging to the Louvre and the Centre..
http://www.france.diplomatie.fr/france/edu/culture.gb.html - size 48K - 20 Jun 96

Travel World - Science Education Tour of Europe
Science Education Tour of Europe. B E M I D J I S T A T E U N I V E R S I T Y Science Education Tour of EUROPE July 19-August 1, 1995...
http://www.omnitravel.com/007etour.html - size 16K - 21 Jul 95
http://www.omnitravel.com/etour.html - size 16K - 15 May 95

FRANCE REAL ESTATE RENTAL
LOIRE VALLEY RENTAL. ANCIENT STONE HOME FOR RENT. Available to rent is a furnished, french country decorated, two bedroom, small stone home, built in the..
http://frost2.flemingc.on.ca/~pbell/france.htm - size 10K - 21 Jun 96

LINKS
PAUL'S LINKS. Click here to view CNN interactive and WEBNEWSor CNET. Click here to make your own web site. Click here to manage your cash. Interested in...
http://frost2.flemingc.on.ca/~pbell/links.htm - size 9K - 19 Jun 96

Digital Design Media, Chapter 9: Lines in Space
Construction planes... Glass-sheet models... Three-dimensional geometric transformations... Sweeping points... Space curves... Structuring wireframe...
http://www.gsd.harvard.edu/~malcolm/DDM/DDM09.html - size 36K - 22 Jul 95

No Title
Boston Update 94: A VISION FOR BOSTON'S FUTURE. Ian Menzies. Senior Fellow, McCormack Institute. University of Massachusetts Boston. April 1994. Prepared..
http://www.cs.umb.edu/~serl/mcCormack/Menzies.html - size 25K - 31 Jan 96

Paris - Photograph
The Arc de Triomphe du Carrousel neatly frames IM Pei's glass pyramid, Paris 1/6. © 1996 Richard Nebesky.

Figure 15.1 Results of the search '"glass pyramid" Pei Louvre' on an internet search engine.

INFORMATION NEED

two categories: relevant and non-relevant to a particular query (or *information need*). In routing, the desired output is a ranking of documents according to estimated relevance, similar to the ranking shown in figure 15.1 for the ad-hoc problem. The difference between routing and ad-hoc is that training information in the form of relevance labels is available in routing, but not in ad-hoc retrieval. In filtering, an estimation of relevance has to be made for each document, typically in the form of a probability estimate. Filtering is harder than routing because an absolute ('Document d is relevant') rather than a relative assessment of relevance ('Document d_1 is more relevant than d_2') is required. In many practical applications, an absolute assessment of relevance for each individual document is necessary. For example, when a news group is filtered for stories about a particular company, users do not want to wait for a month, and then receive a ranked list of all stories about the company in the past month, with the most relevant shown at the top. Instead, it is desirable to deliver relevant stories as soon as they come in without knowledge about subsequent postings. As special cases of classification, filtering and routing can be accomplished using any of the classification algorithms described in chapter 16 or elsewhere in this book.

15.1.1 Common design features of IR systems

INVERTED INDEX

Most IR systems have as their primary data structure an *inverted index*. An inverted index is a data structure that lists for each word in the collection all documents that contain it (the *postings*) and the frequency of occurrence in each document. An inverted index makes it easy to search for 'hits' of a query word. One just goes to the part of the inverted index that corresponds to the query word and retrieves the documents listed there.

POSTINGS

POSITION INFORMATION

A more sophisticated version of the inverted index also contains *position information*. Instead of just listing the documents that a word occurs in, the positions of all occurrences in the document are also listed. A position of occurrence can be encoded as a byte offset relative to the beginning of the document. An inverted index with position information lets us search for *phrases*. For example, to search for 'car insurance,' we simultaneously work through the entries for *car* and *insurance* in the inverted index. First, we intersect the two sets so that we only have documents in which both words occur. Then we look at the position information and keep only those hits for which the position information

PHRASES

a	also	an	and	as	at	be	but	by
can	could	do	for	from	go			
have	he	her	here	his	how			
i	if	in	into	it	its			
my	of	on	or	our	say	she		
that	the	their	there	therefore	they			
this	these	those	through	to	until			
we	what	when	where	which	while	who	with	would
you	your							

Table 15.1 A small stop list for English. Stop words are function words that can be ignored in keyword-oriented information retrieval without a significant effect on retrieval accuracy.

indicates that *insurance* occurs immediately after *car*. This is much more efficient than having to read in and process all documents of the collection sequentially.

The notion of *phrase* used here is a fairly primitive one. We can only search for fixed phrases. For example, a search for 'car insurance rates' would not find documents talking about *rates for car insurance*. This is an area in which future Statistical NLP research can make important contributions to information retrieval. Most recent research on phrases in IR has taken the approach of designing a separate phrase identification module and then indexing documents for identified phrases as well as words. In such a system, a phrase is treated as no different from an ordinary word. The simplest approach to phrase identification, which is anathema to NLP researchers, but often performs surprisingly well, is to just select the most frequent bigrams as phrases, for example, those that occur at least 25 times.

In cases where phrase identification is a separate module, it is very similar to the problem of discovering collocations. Many of the techniques in chapter 5 for finding collocations can therefore also be applied to identifying good phrases for indexing and searching.

STOP LIST
FUNCTION WORDS

In some IR systems, not all words are represented in the inverted index. A *stop list* of 'grammatical' or *function words* lists those words that are deemed unlikely to be useful for searching. Common stop words are *the*, *from* and *could*. These words have important semantic functions in English, but they rarely contribute information if the search criterion is

a simple word-by-word match. A small stop list for English is shown in table 15.1.

A stop list has the advantage that it reduces the size of the inverted index. According to Zipf's law (see section 1.4.3), a stop list that covers a few dozen words can reduce the size of the inverted index by half. However, it is impossible to search for phrases that contain stop words once the stop list has been applied – note that some occasionally used phrases like *when and where* consist *entirely* of words in the stop list in table 15.1. For this reason, many retrieval engines do not make use of a stop list for indexing.

STEMMING

Another common feature of IR systems is *stemming*, which we briefly discussed in section 4.2.3. In IR, stemming usually refers to a simplified form of morphological analysis consisting simply of truncating a word. For example, *laughing*, *laugh*, *laughs* and *laughed* are all stemmed to *laugh-*. Common stemmers are the *Lovins* and *Porter* stemmers, which differ in the actual algorithms used for determining where to truncate words (Lovins 1968; Porter 1980). Two problems with truncation stemmers are that they conflate semantically different words (for example, *gallery* and *gall* may both be stemmed to *gall-*) and that the truncated stems can be unintelligible to users (for example, if *gallery* is presented as *gall-*). They are also much harder to make work well for morphology-rich languages.

LOVINS STEMMER

PORTER STEMMER

15.1.2 Evaluation measures

Since the quality of many retrieval systems depends on how well they manage to rank relevant documents before non-relevant ones, IR researchers have developed evaluation measures specifically designed to evaluate rankings. Most of these measures combine precision and recall in a way that takes account of the ranking. As we explained in section 8.1, precision is the percentage of relevant items in the returned set and recall is the percentage of all relevant documents in the collection that is in the returned set.

Figure 15.2 demonstrates why the ranking of documents is important. All three retrieved sets have the same number of relevant and not relevant documents. A simple measure of precision (50% correct) would not distinguish between them. But ranking 1 is clearly better than ranking 2 for a user who scans a returned list of documents from top to bottom

Evaluation	Ranking 1	Ranking 2	Ranking 3
	d1: ✓	d10: ×	d6: ×
	d2: ✓	d9: ×	d1: ✓
	d3: ✓	d8: ×	d2: ✓
	d4: ✓	d7: ×	d10: ×
	d5: ✓	d6: ×	d9: ×
	d6: ×	d1: ✓	d3: ✓
	d7: ×	d2: ✓	d5: ✓
	d8: ×	d3: ✓	d4: ✓
	d9: ×	d4: ✓	d7: ×
	d10: ×	d5: ✓	d8: ×
precision at 5	1.0	0.0	0.4
precision at 10	0.5	0.5	0.5
uninterpolated av. prec.	1.0	0.3544	0.5726
interpolated av. prec. (11-point)	1.0	0.5	0.6440

Table 15.2 An example of the evaluation of rankings. The columns show three different rankings of ten documents, where a ✓ indicates a relevant document and a × indicates a non-relevant document. The rankings are evaluated according to four measures: precision at 5 documents, precision at 10 documents, uninterpolated average precision, and interpolated average precision over 11 points.

(which is what users do in many practical situations, for example, when web searching).

CUTOFF

One measure used is precision at a particular *cutoff*, for example 5 or 10 documents (other typical cutoffs are 20 and 100). By looking at precision for several initial segments of the ranked list, one can gain a good impression of how well a method ranks relevant documents before non-relevant documents.

UNINTERPOLATED AVERAGE PRECISION

Uninterpolated average precision aggregates many precision numbers into one evaluation figure. Precision is computed for each point in the list where we find a relevant document and these precision numbers are then averaged. For example, for ranking 1 precision is 1.0 for d1, d2, d3, d4 and d5 since for each of these documents there are only relevant documents up to that point in the list. The uninterpolated average is therefore also 1.0. For ranking 3, we get the following precision numbers for the relevant documents: 1/2 (d1), 2/3 (d2), 3/6 (d3), 4/7 (d5), 5/8

(d4), which averages to 0.5726.

If there are other relevant documents further down the list then these also have to be taken into account in computing uninterpolated average precision. Precision at relevant documents that are not in the returned set is assumed to be zero. This shows that average precision indirectly measures *recall*, the percentage of relevant documents that were returned in the retrieved set (since omitted documents are entered as zero precision).

INTERPOLATED AVERAGE PRECISION
LEVELS OF RECALL

Interpolated average precision is more directly based on recall. Precision numbers are computed for various *levels of recall*, for example for the levels 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% in the case of an 11-point average (the most widely used measure). At recall level α, precision β is computed at the point of the ranked list where the proportion of retrieved relevant documents reaches α. However, if precision goes up again while we are moving down the list, then we *interpolate* and take the highest value of precision anywhere beyond the point where recall level α was first reached. For example, for ranking 3 in figure 15.2 interpolated precision for recall level 60% is not 4/7, the precision at the point where 60% recall is first reached as shown in the top diagram of figure 15.2. Instead, it is $5/8 > 4/7$ as shown in the bottom diagram of figure 15.2. (We are assuming that the five relevant documents shown are the only relevant documents.) The thinking here is that the user will be willing to look at more documents if the precision goes up. The two graphs in figure 15.2 are so-called *precision-recall curves*, with interpolated and uninterpolated values for 0%, 20%, 40%, 60%, 80%, and 100% recall for ranking 3 in table 15.2.

INTERPOLATE

PRECISION-RECALL CURVES

There is an obvious trade-off between precision and recall. If the whole collection is retrieved, then recall is 100%, but precision is low. On the other hand, if only a few documents are retrieved, then the most relevant-seeming documents will be returned, resulting in high precision, but recall will be low.

Average precision is one way of computing a measure that captures both precision and recall. Another way is the *F measure*, which we introduced in section 8.1:

F MEASURE

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{15.1}$$

where P is the precision, R is the recall and α determines the weighting of precision and recall. The F measure can be used for evaluation at fixed cutoffs if both recall and precision are important.

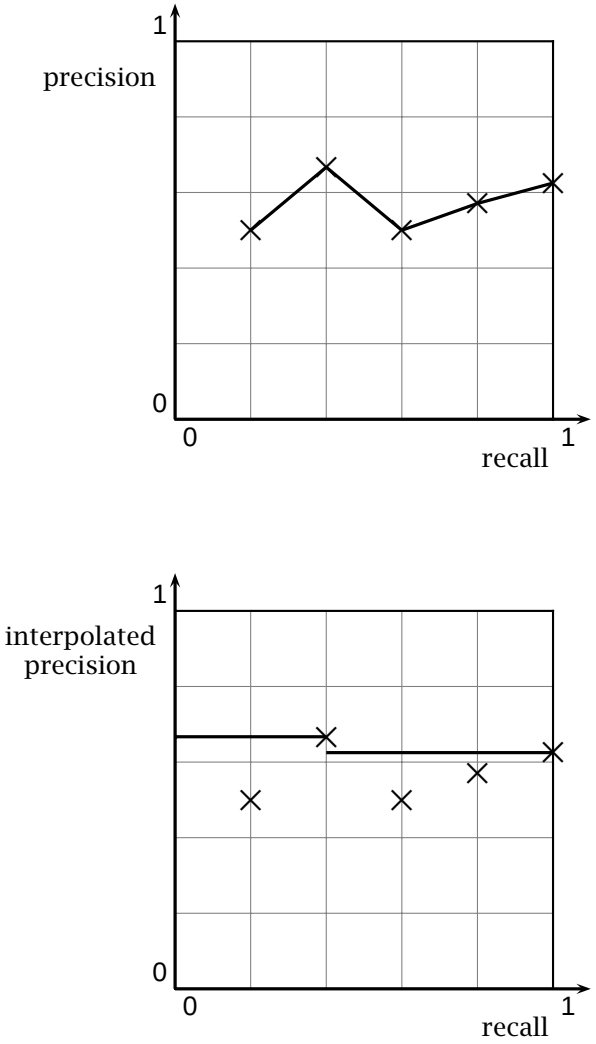

Figure 15.2 Two examples of precision-recall curves. The two curves are for ranking 3 in table 15.2: uninterpolated (above) and interpolated (below).

Any of the measures discussed above can be used to compare the performance of information retrieval systems. One common approach is to run the systems on a corpus and a set of queries and average the performance measure over queries. If the average of system 1 is better than the average of system 2, then that is evidence that system 1 is better than system 2.

Unfortunately, there are several problems with this experimental design. The difference in averages could be due to chance. Or it could be due to one query on which system 1 outperforms system 2 by a large margin with performance on all other queries being about the same. It is therefore advisable to use a statistical test like the t test for system comparison (as shown in section 6.2.3).

15.1.3 The probability ranking principle (PRP)

Ranking documents is intuitively plausible since it gives the user some control over the tradeoff between precision and recall. If recall for the first page of results is low and the desired information is not found, then the user can look at the next page, which in most cases trades higher recall for lower precision.

The following principle is a guideline which is one way to make the assumptions explicit that underlie the design of retrieval by ranking. We present it in a form simplified from (van Rijsbergen 1979: 113):

> **Probability Ranking Principle (PRP).** Ranking documents in order of decreasing probability of relevance is optimal.

The basic idea is that we view retrieval as a greedy search that aims to identify the most valuable document at any given time. The document d that is most likely to be valuable is the one with the highest estimated probability of relevance (where we consider all documents that haven't been retrieved yet), that is, with a maximum value for $P(R|d)$. After making many consecutive decisions like this, we arrive at a list of documents that is ranked in order of decreasing probability of relevance.

Many retrieval systems are based on the PRP, so it is important to be clear about the assumptions that are made when it is accepted.

One assumption of the PRP is that documents are independent. The clearest counterexamples are duplicates. If we have two duplicates d_1 and d_2, then the estimated probability of relevance of d_2 does not change after we have presented d_1 further up in the list. But d_2 does not give

the user any information that is not already contained in d_1. Clearly, a better design is to show only one of the set of identical documents, but that violates the PRP.

Another simplification made by the PRP is to break up a complex information need into a number of queries which are each optimized in isolation. In practice, a document can be highly relevant to the complex information need as a whole even if it is not the optimal one for an intermediate step. An example here is an information need that the user initially expresses using ambiguous words, for example, the query *jaguar* to search for information on the animal (as opposed to the car). The optimal response to this query may be the presentation of documents that make the user aware of the ambiguity and permit disambiguation of the query. In contrast, the PRP would mandate the presentation of documents that are highly relevant to either the car or the animal.

A third important caveat is that the probability of relevance is only estimated. Given the many simplifying assumptions we make in designing probabilistic models for IR, we cannot completely trust the probability estimates. One aspect of this problem is that the *variance* of the estimate of probability of relevance may be an important piece of evidence in some retrieval contexts. For example, a user may prefer a document that we are certain is probably relevant (low variance of probability estimate) to one whose estimated probability of relevance is higher, but that also has a higher variance of the estimate.

VARIANCE

15.2 The Vector Space Model

VECTOR SPACE MODEL

The *vector space model* is one of the most widely used models for ad-hoc retrieval, mainly because of its conceptual simplicity and the appeal of the underlying metaphor of using spatial proximity for semantic proximity. Documents and queries are represented in a high-dimensional space, in which each dimension of the space corresponds to a word in the document collection. The most relevant documents for a query are expected to be those represented by the vectors closest to the query, that is, documents that use similar words to the query. Rather than considering the magnitude of the vectors, closeness is often calculated by just looking at angles and choosing documents that enclose the smallest angle with the query vector.

In figure 15.3, we show a vector space with two dimensions, corre-

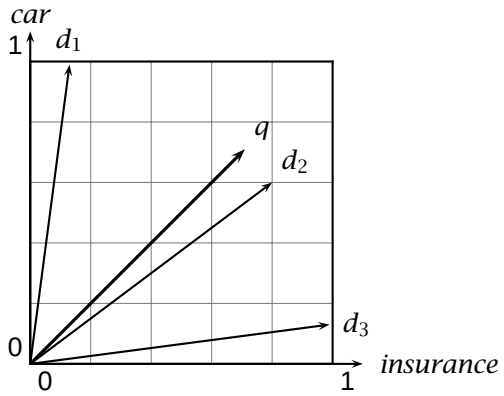

Figure 15.3 A vector space with two dimensions. The two dimensions correspond to the terms *car* and *insurance*. One query and three documents are represented in the space.

sponding to the words *car* and *insurance*. The entities represented in the space are the query q represented by the vector $(0.71, 0.71)$, and three documents d_1, d_2, and d_3 with the following coordinates: $(0.13, 0.99)$, $(0.8, 0.6)$, and $(0.99, 0.13)$. The coordinates or *term weights* are derived from occurrence counts as we will see below. For example, *insurance* may have only a passing reference in d_1 while there are several occurrences of *car* – hence the low weight for *insurance* and the high weight for *car*. (In the context of information retrieval, the word *term* is used for both words and phrases. We say *term weights* rather than *word weights* because dimensions in the vector space model can correspond to phrases as well as words.)

TERM WEIGHTS

TERM

In the figure, document d_2 has the smallest angle with q, so it will be the top-ranked document in response to the query *car insurance*. This is because both 'concepts' (*car* and *insurance*) are salient in d_2 and therefore have high weights. The other two documents also mention both terms, but in each case one of them is not a centrally important term in the document.

15.2.1 Vector similarity

To do retrieval in the vector space model, documents are ranked according to similarity with the query as measured by the *cosine* measure or

COSINE

NORMALIZED CORRELATION COEFFICIENT

normalized correlation coefficient. We introduced the cosine as a measure of vector similarity in section 8.5.1 and repeat its definition here:

$$\cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{15.2}$$

where $\vec{q}$ and $\vec{d}$ are n-dimensional vectors in a real-valued space, the space of all terms in the case of the vector space model. We compute how well the occurrence of term i (measured by q_i and d_i) correlates in query and document and then divide by the Euclidean length of the two vectors to scale for the magnitude of the individual q_i and d_i.

Recall also from section 8.5.1 that cosine and Euclidean distance give rise to the same ranking for normalized vectors:

$$\begin{aligned}
(|\vec{x}-\vec{y}|)^2 &= \sum_{i=1}^{n}(x_i-y_i)^2 \\
&= \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} y_i^2 \\
&= 1 - 2\sum_{i=1}^{n} x_i y_i + 1 \\
&= 2(1 - \sum_{i=1}^{n} x_i y_i) \qquad (15.3)
\end{aligned}$$

So for a particular query $\vec{q}$ and any two documents $\vec{d_1}$ and $\vec{d_2}$ we have:

$$\cos(\vec{q},\vec{d_1}) > \cos(\vec{q},\vec{d_2}) \quad \Leftrightarrow \quad |\vec{q}-\vec{d_1}| < |\vec{q}-\vec{d_2}| \tag{15.4}$$

which implies that the rankings are the same. (We again assume normalized vectors here.)

If the vectors are normalized, we can compute the cosine as a simple dot product. Normalization is generally seen as a good thing – otherwise longer vectors (corresponding to longer documents) would have an unfair advantage and get ranked higher than shorter ones. (We leave it as an exercise to show that the vectors in figure 15.3 are normalized, that is, $\sqrt{\sum_i d_i^2} = 1$.)

15.2.2 Term weighting

We now turn to the question of how to weight words in the vector space model. One could just use the count of a word in a document as its term

Quantity	Symbol	Definition
term frequency	$\mathrm{tf}_{i,j}$	number of occurrences of w_i in d_j
document frequency	df_i	number of documents in the collection that w_i occurs in
collection frequency	cf_i	total number of occurrences of w_i in the collection

Table 15.3 Three quantities that are commonly used in term weighting in information retrieval.

Word	Collection Frequency	Document Frequency
insurance	10440	3997
try	10422	8760

Table 15.4 Term and document frequencies of two words in an example corpus.

TERM FREQUENCY
DOCUMENT FREQUENCY
COLLECTION FREQUENCY

weight, but there are more effective methods of term weighting. The basic information used in term weighting is *term frequency*, *document frequency*, and sometimes *collection frequency* as defined in table 15.3. Note that $\mathrm{df}_i \leq \mathrm{cf}_i$ and that $\sum_j \mathrm{tf}_{i,j} = \mathrm{cf}_i$. It is also important to note that document frequency and collection frequency can only be used if there is a collection. This assumption is not always true, for example if collections are created dynamically by selecting several databases from a large set (as may be the case on one of the large on-line information services), and joining them into a temporary collection.

The information that is captured by term frequency is how salient a word is within a given document. The higher the term frequency (the more often the word occurs) the more likely it is that the word is a good description of the content of the document. Term frequency is usually dampened by a function like $f(\mathrm{tf}) = \sqrt{\mathrm{tf}}$ or $f(\mathrm{tf}) = 1 + \log(\mathrm{tf}), \mathrm{tf} > 0$ because more occurrences of a word indicate higher importance, but not as much importance as the undampened count would suggest. For example, $\sqrt{3}$ or $1 + \log 3$ better reflect the importance of a word with three occurrences than the count 3 itself. The document is somewhat more important than a document with one occurrence, but not three times as important.

The second quantity, document frequency, can be interpreted as an indicator of informativeness. A semantically focussed word will often occur several times in a document if it occurs at all. Semantically unfocussed words are spread out homogeneously over all documents. An example

from a corpus of *New York Times* articles is the words *insurance* and *try* in table 15.4. The two words have about the same collection frequency, the total number of occurrences in the document collection. But *insurance* occurs in only half as many documents as *try*. This is because the word *try* can be used when talking about almost any topic since one can *try* to do something in any context. In contrast, *insurance* refers to a narrowly defined concept that is only relevant to a small set of topics. Another property of semantically focussed words is that, if they come up once in a document, they often occur several times. *Insurance* occurs about three times per document, averaged over documents it occurs in at least once. This is simply due to the fact that most articles about health insurance, car insurance or similar topics will refer multiple times to the concept of insurance.

One way to combine a word's term frequency $\mathrm{tf}_{i,j}$ and document frequency df_i into a single weight is as follows:

$$\text{weight}(i,j) = \begin{cases} (1+\log(\mathrm{tf}_{i,j}))\log\frac{N}{\mathrm{df}_i} & \text{if } \mathrm{tf}_{i,j} \geq 1 \\ 0 & \text{if } \mathrm{tf}_{i,j} = 0 \end{cases} \tag{15.5}$$

where N is the total number of documents. The first clause applies for words occurring in the document, whereas for words that do not appear ($\mathrm{tf}_{i,j} = 0$), we set $\text{weight}(i,j) = 0$.

Document frequency is also scaled logarithmically. The formula $\log\frac{N}{\mathrm{df}_i} = \log N - \log \mathrm{df}_i$ gives full weight to words that occur in 1 document ($\log N - \log \mathrm{df}_i = \log N - \log 1 = \log N$). A word that occurred in all documents would get zero weight ($\log N - \log \mathrm{df}_i = \log N - \log N = 0$).

INVERSE DOCUMENT FREQUENCY
IDF
TF.IDF

This form of document frequency weighting is often called *inverse document frequency* or idf weighting. More generally, the weighting scheme in (15.5) is an example of a larger family of so-called *tf.idf* weighting schemes. Each such scheme can be characterized by its term occurrence weighting, its document frequency weighting and its normalization. In one description scheme, we assign a letter code to each component of the tf.idf scheme. The scheme in (15.5) can then be described as "ltn" for logarithmic occurrence count weighting (l), logarithmic document frequency weighting (t), and no normalization (n). Other weighting possibilities are listed in table 15.5. For example, "ann" is augmented term occurrence weighting, no document frequency weighting and no normalization. We refer to vector length normalization as cosine normalization because the inner product between two length-normalized vectors (the query-document similarity measure used in the vector space model) is

Term occurrence		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (natural)	df_t	n	(no normalization)
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t	$\log \frac{N}{\mathrm{df}_t}$	c	(cosine)
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$				$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}}$

Table 15.5 Components of tf.idf weighting schemes. $\mathrm{tf}_{t,d}$ is the frequency of term t in document d, df_t is the number of documents t occurs in, N is the total number of documents, and w_i is the weight of term i.

their cosine. Different weighting schemes can be applied to queries and documents. In the name "ltc.lnn," the halves refer to document and query weighting, respectively.

The family of weighting schemes shown in table 15.5 is sometimes criticized as 'ad-hoc' because it is not directly derived from a mathematical model of term distributions or relevancy. However, these schemes are effective in practice and work robustly in a broad range of applications. For this reason, they are often used in situations where a rough measure of similarity between vectors of counts is needed.

15.3 Term Distribution Models

An alternative to tf.idf weighting is to develop a model for the distribution of a word and to use this model to characterize its importance for retrieval. That is, we wish to estimate $P_i(k)$, the proportion of times that word w_i appears k times in a document. In the simplest case, the distribution model is used for deriving a probabilistically motivated term weighting scheme for the vector space model. But models of term distribution can also be embedded in other information retrieval frameworks.

ZIPF'S LAW

Apart from its importance for term weighting, a precise characterization of the occurrence patterns of words in text is arguably at least as important a topic in Statistical NLP as *Zipf's law*. Zipf's law describes word behavior in an *entire corpus*. In contrast, term distribution models capture regularities of word occurrence in *subunits of a corpus* (e.g., documents or chapters of a book). In addition to information retrieval, a good understanding of distribution patterns is useful wherever we want to assess the likelihood of a certain number of occurrences of a specific word in a unit of text. For example, it is also important for author identifi-

cation where one compares the likelihood that different writers produced a text of unknown authorship.

Most term distribution models try to characterize how informative a word is, which is also the information that inverse document frequency is getting at. One could cast the problem as one of distinguishing content words from non-content (or function) words, but most models have a graded notion of how informative a word is. In this section, we introduce several models that formalize notions of informativeness. Three are based on the Poisson distribution, one motivates inverse document frequency as a weight optimal for Bayesian classification and the final one, *residual inverse document frequency*, can be interpreted as a combination of idf and the Poisson distribution.

15.3.1 The Poisson distribution

POISSON DISTRIBUTION

The standard probabilistic model for the distribution of a certain type of event over units of a fixed size (such as periods of time or volumes of liquid) is the *Poisson distribution*. Classical examples of Poisson distributions are the number of items that will be returned as defects in a given period of time, the number of typing mistakes on a page, and the number of microbes that occur in a given volume of water.

The definition of the Poisson distribution is as follows.

Poisson Distribution. $\mathrm{p}(k;\ \lambda_i) = e^{-\lambda_i}\frac{{\lambda_i}^k}{k!}$ for some $\lambda_i > 0$

In the most common model of the Poisson distribution in IR, the parameter $\lambda_i > 0$ is the average number of occurrences of w_i per document, that is, $\lambda_i = \frac{\mathrm{cf}_i}{N}$ where cf_i is the collection frequency and N is the total number of documents in the collection. Both the mean and the variance of the Poisson distribution are equal to λ_i:

$$E(\mathrm{p}) = \mathrm{Var}(\mathrm{p}) = \lambda_i$$

Figure 15.4 shows two examples of the Poisson distribution.

In our case, the event we are interested in is the occurrence of a particular word w_i and the fixed unit is the document. We can use the Poisson distribution to estimate an answer to the question: What is the probability that a word occurs a particular number of times in a document. We might say that $P_i(k) = \mathrm{p}(k;\ \lambda_i)$ is the probability of a document having exactly k occurrences of w_i, where λ_i is appropriately estimated for each word.

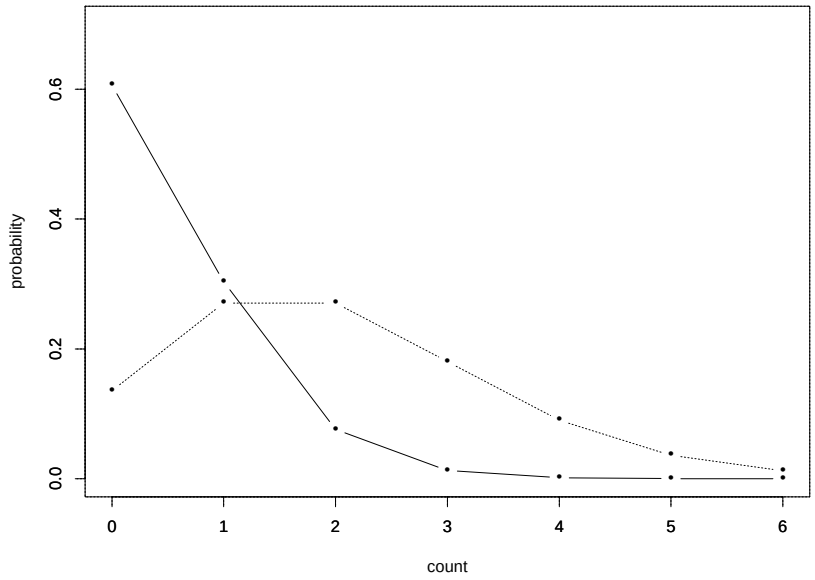

Figure 15.4 The Poisson distribution. The graph shows $\mathrm{p}(k;\ 0.5)$ (solid line) and $\mathrm{p}(k;\ 2.0)$ (dotted line) for $0 \leq k \leq 6$. In the most common use of this distribution in IR, k is the number of occurrences of term i in a document, and $\mathrm{p}(k;\ \lambda_i)$ is the probability of a document with that many occurrences.

The Poisson distribution is a limit of the binomial distribution. For the binomial distribution $\mathrm{b}(k;\ n, p)$, if we let $n \to \infty$ and $p \to 0$ in such a way that np remains fixed at value $\lambda > 0$, then $\mathrm{b}(x;\ n, p) \to \mathrm{p}(k;\ \lambda)$. Assuming a Poisson distribution for a term is appropriate if the following conditions hold.

- The probability of one occurrence of the term in a (short) piece of text is proportional to the length of the text.
- The probability of more than one occurrence of a term in a short piece of text is negligible compared to the probability of one occurrence.
- Occurrence events in non-overlapping intervals of text are independent.

We will discuss problems with these assumptions for modeling the distribution of terms shortly. Let us first look at some examples.

Word	df_i	cf_i	λ_i	$N(1 - p(0; \lambda_i))$	Overestimation
follows	21744	23533	0.2968	20363	0.94
transformed	807	840	0.0106	835	1.03
soviet	8204	35337	0.4457	28515	3.48
students	4953	15925	0.2008	14425	2.91
james	9191	11175	0.1409	10421	1.13
freshly	395	611	0.0077	609	1.54

Table 15.6 Document frequency (df) and collection frequency (cf) for 6 words in the *New York Times* corpus. Computing $N(1 - p(0; \lambda_i))$ according to the Poisson distribution is a reasonable estimator of df for non-content words (like *follows*), but severely overestimates df for content words (like *soviet*). The parameter λ_i of the Poisson distribution is the average number of occurrences of term i per document. The corpus has $N = 79291$ documents.

Table 15.6 shows for six terms in the *New York Times* newswire how well the Poisson distribution predicts document frequency. For each word, we show document frequency df_i, collection frequency cf_i, the estimate of λ (collection frequency divided by total number of documents (79291)), the predicted df, and the ratio of predicted df and actual df.

Examining document frequency is the easiest way to check whether a term is Poisson distributed. The number of documents predicted to have at least one occurrence of a term can be computed as the complement of the predicted number with no occurrences. Thus, the Poisson predicts that the document frequency is $\widehat{df}_i = N(1 - P_i(0))$ where N is the number of documents in the corpus. A better way to check the fit of the Poisson is to look at the complete distribution: the number of documents with 0, 1, 2, 3, etc. occurrences. We will do this below.

In table 15.6, we can see that the Poisson estimates are good for non-content words like *follows* and *transformed*. We use the term *non-content word* loosely to refer to words that taken in isolation (which is what most IR systems do) do not give much information about the contents of the document. But the estimates for content words are much too high, by a factor of about 3 (3.48 and 2.91).

This result is not surprising since the Poisson distribution assumes independence between term occurrences. This assumption holds approximately for non-content words, but most content words are much more likely to occur again in a text once they have occurred once, a property

BURSTINESS
TERM CLUSTERING

that is sometimes called *burstiness* or *term clustering*. However, there are some subtleties in the behavior of words as we can see for the last two words in the table. The distribution of *james* is surprisingly close to Poisson, probably because in many cases a person's full name is given at first mention in a newspaper article, but following mentions only use the last name or a pronoun. On the other hand, *freshly* is surprisingly non-Poisson. Here we get strong dependence because of the genre of recipes in the *New York Times* in which *freshly* frequently occurs several times. So non-Poisson-ness can also be a sign of clustered term occurrences in a particular genre like recipes.

The tendency of content word occurrences to cluster is the main problem with using the Poisson distribution for words. But there is also the opposite effect. We are taught in school to avoid repetitive writing. In many cases, the probability of reusing a word immediately after its first occurrence in a text is lower than in general. A final problem with the Poisson is that documents in many collections differ widely in size. So documents are not a uniform unit of measurement as the second is for time or the kilogram is for mass. But that is one of the assumptions of the Poisson distribution.

15.3.2 The two-Poisson model

TWO-POISSON MODEL

A better fit to the frequency distribution of content words is provided by the *two-Poisson Model* (Bookstein and Swanson 1975), a mixture of two Poissons. The model assumes that there are two classes of documents associated with a term, one class with a low average number of occurrences (the non-privileged class) and one with a high average number of occurrences (the privileged class):

$$\mathrm{tp}(k;\ \pi, \lambda_1, \lambda_2) = \pi e^{-\lambda_1} \frac{{\lambda_1}^k}{k!} + (1-\pi) e^{-\lambda_2} \frac{{\lambda_2}^k}{k!}$$

where π is the probability of a document being in the privileged class, $(1-\pi)$ is the probability of a document being in the non-privileged class, and λ_1 and λ_2 are the average number of occurrences of word w_i in the privileged and non-privileged classes, respectively.

The two-Poisson model postulates that a content word plays two different roles in documents. In the non-privileged class, its occurrence is accidental and it should therefore not be used as an index term, just as a non-content word. The average number of occurrences of the word in

this class is low. In the privileged class, the word is a central content word. The average number of occurrences of the word in this class is high and it is a good index term.

Empirical tests of the two-Poisson model have found a spurious "dip" at frequency 2. The model incorrectly predicts that documents with 2 occurrences of a term are less likely than documents with 3 or 4 occurrences. In reality, the distribution for most terms is monotonically decreasing. If $P_i(k)$ is the proportion of times that word w_i appears k times in a document, then $P_i(0) > P_i(1) > P_i(2) > P_i(3) > P_i(4) > \ldots$. As a fix, one can use more than two Poisson distributions. The *negative binomial* is one such mixture of an infinite number of Poissons (Mosteller and Wallace 1984), but there are many others (Church and Gale 1995). The negative binomial fits term distributions better than one or two Poissons, but it can be hard to work with in practice because it involves the computation of large binomial coefficients.

NEGATIVE BINOMIAL

15.3.3 The K mixture

A simpler distribution that fits empirical word distributions about as well as the negative binomial is Katz's K mixture:

$$P_i(k) = (1-\alpha)\delta_{k,0} + \frac{\alpha}{\beta+1}\left(\frac{\beta}{\beta+1}\right)^k$$

where $\delta_{k,0} = 1$ iff $k = 0$ and $\delta_{k,0} = 0$ otherwise and α and β are parameters that can be fit using the observed mean λ and the observed inverse document frequency IDF as follows.

$$\begin{aligned}
\lambda &= \frac{\text{cf}}{N} \\
\text{IDF} &= \log_2 \frac{N}{\text{df}} \\
\beta &= \lambda \times 2^{\text{IDF}} - 1 = \frac{\text{cf} - \text{df}}{\text{df}} \\
\alpha &= \frac{\lambda}{\beta}
\end{aligned}$$

The parameter β is the number of "extra terms" per document in which the term occurs (compared to the case where a term has only one occurrence per document). The decay factor $\frac{\beta}{\beta+1} = \frac{\text{cf}-\text{df}}{\text{cf}}$ (extra terms per term occurrence) determines the ratio $\frac{P_i(k)}{P_i(k-1)}$. For example, if there are $\frac{1}{10}$ as

Word		k									
		0	1	2	3	4	5	6	7	8	≥ 9
follows	act.	57552.0	20142.0	1435.0	148.0	18.0	1.0				
	est.	57552.0	20091.0	1527.3	116.1	8.8	0.7	0.1	0.0	0.0	0.0
trans-	act.	78489.0	776.0	29.0	2.0						
formed	est.	78489.0	775.3	30.5	1.2	0.0	0.0	0.0	0.0	0.0	0.0
soviet	act.	71092.0	3038.0	1277.0	784.0	544.0	400.0	356.0	302.0	255.0	1248.0
	est.	71092.0	1904.7	1462.5	1122.9	862.2	662.1	508.3	390.3	299.7	230.1
students	act.	74343.0	2523.0	761.0	413.0	265.0	178.0	143.0	112.0	96.0	462.0
	est.	74343.0	1540.5	1061.4	731.3	503.8	347.1	239.2	164.8	113.5	78.2
james	act.	70105.0	7953.0	922.0	183.0	52.0	24.0	19.0	9.0	7.0	22.0
	est.	70105.0	7559.2	1342.1	238.3	42.3	7.5	1.3	0.2	0.0	0.0
freshly	act.	78901.0	267.0	66.0	47.0	8.0	4.0	2.0	1.0		
	est.	78901.0	255.4	90.3	31.9	11.3	4.0	1.4	0.5	0.2	0.1

Table 15.7 Actual and estimated number of documents with k occurrences for six terms. For example, there were 1435 documents with 2 occurrences of *follows*. The K mixture estimate is 1527.3.

many extra terms as term occurrences, then there will be ten times as many documents with 1 occurrence as with 2 occurrences and ten times as many with 2 occurrences as with 3 occurrences. If there are no extra terms (cf = df $\Rightarrow \frac{\beta}{\beta+1} = 0$), then we predict that there are no documents with more than 1 occurrence.

The parameter α captures the absolute frequency of the term. Two terms with the same β have identical ratios of collection frequency to document frequency, but different values for α if their collection frequencies are different.

Table 15.7 shows the number of documents with k occurrences in the *New York Times* corpus for the six words that we looked at earlier. We observe that the fit is always perfect for $k = 0$. It is easy to show that this is a general property of the K mixture (see exercise 15.3).

The K mixture is a fairly good approximation of term distribution, especially for non-content words. However, it is apparent from the empirical numbers in table 15.7 that the assumption:

$$\frac{P_i(k)}{P_i(k+1)} = c, \quad k \geq 1$$

does not hold perfectly for content words. As in the case of the two-Poisson mixture we are making a distinction between low base rate of occurrence and another class of documents that have clusters of occurrences. The K mixture assumes $\frac{P_i(k)}{P_i(k+1)} = c$ for $k \geq 1$, which concedes

that $k = 0$ is a special case due to a low base rate of occurrence for many words. But the ratio $\frac{P_i(k)}{P_i(k+1)}$ seems to decline for content words even for $k \geq 1$. For example, for *soviet* we have:

$$\frac{P_i(0)}{P_i(1)} = \frac{71092}{3038} \approx 23.4 \qquad \frac{P_i(1)}{P_i(2)} = \frac{3038}{1277} \approx 2.38$$
$$\frac{P_i(2)}{P_i(3)} = \frac{1277}{784} \approx 1.63 \qquad \frac{P_i(3)}{P_i(4)} = \frac{784}{544} \approx 1.44$$
$$\frac{P_i(4)}{P_i(5)} = \frac{544}{400} \approx 1.36$$

In other words, each occurrence of a content word we find in a text decreases the probability of finding an additional term, but the decreases become consecutively smaller. The reason is that occurrences of content words tend to cluster in documents whose core topic is associated with the content word. A large number of occurrences indicates that the content word describes a central concept of the document. Such a central concept is likely to be mentioned more often than a "constant decay" model would predict.

We have introduced Katz's K mixture here as an example of a term distribution model that is more accurate than the Poisson distribution and the two-Poisson model. The interested reader can find more discussion of the characteristics of content words in text and of several probabilistic models with a better fit to empirical distributions in (Katz 1996).

15.3.4 Inverse document frequency

We motivated inverse document frequency (IDF) heuristically in section 15.2.2, but we can also derive it from a term distribution model. In the derivation we present here, we only use *binary* occurrence information and do not take into account term frequency.

ODDS OF RELEVANCE

To derive IDF, we view ad-hoc retrieval as the task of ranking documents according to the *odds of relevance*:

$$O(d) = \frac{P(R|d)}{P(\neg R|d)}$$

where $P(R|d)$ is the probability of relevance of d and $P(\neg R|d)$ is the probability of non-relevance. We then take logs to compute the log odds, and apply Bayes' formula:

$$\log O(d) = \log \frac{P(R|d)}{P(\neg R|d)}$$

$$
\begin{aligned}
&= \log \frac{\frac{P(d|R)P(R)}{P(d)}}{\frac{P(d|\neg R)P(\neg R)}{P(d)}} \\
&= \log P(d|R) - \log P(d|\neg R) + \log P(R) - \log P(\neg R)
\end{aligned}
$$

Let us assume that the query Q is the set of words $\{w_i\}$, and let the indicator random variables X_i be 1 or 0, corresponding to occurrence and non-occurrence of word w_i in d. If we then make the conditional independence assumption discussed in section 7.2.1, we can write:

$$\log O(d) = \sum_i [\log P(X_i|R) - \log P(X_i|\neg R)] + \log P(R) - \log P(\neg R)$$

Since we are only interested in ranking, we can create a new ranking function $g(d)$ which drops the constant term $\log P(R) - \log P(\neg R)$. With the abbreviations $p_i = P(X_i = 1|R)$ (word i occurring in a relevant document) and $q_i = P(X_i = 1|\neg R)$ (word i occurring in a non-relevant document), we can write $g(d)$ as follows. (In the second line, we make use of $P(X_i = 1|\_) = y = y^1(1-y)^0 = y^{X_i}(1-y)^{1-X_i}$ and $P(X_i = 0|\_) = 1 - y = y^0(1-y)^1 = y^{X_i}(1-y)^{1-X_i}$ so that we can write the equation more compactly.)

$$
\begin{aligned}
g(d) &= \sum_i [\log P(X_i|R) - \log P(X_i|\neg R)] \\
&= \sum_i [\log(p_i^{X_i}(1-p_i)^{1-X_i}) - \log(q_i^{X_i}(1-q_i)^{1-X_i})] \\
&= \sum_i X_i \log \frac{p_i(1-q_i)}{(1-p_i)q_i} + \sum_i \log \frac{1-p_i}{1-q_i} \\
&= \sum_i X_i \log \frac{p_i}{1-p_i} + \sum_i X_i \log \frac{1-q_i}{q_i} + \sum_i \log \frac{1-p_i}{1-q_i}
\end{aligned}
$$

In the last equation above, $\sum_i \log \frac{1-p_i}{1-q_i}$ is another constant term which does not affect the ranking of documents, and so we can drop it as well giving the final ranking function:

$$g'(d) = \sum_i X_i \log \frac{p_i}{1-p_i} + \sum_i X_i \log \frac{1-q_i}{q_i} \tag{15.6}$$

If we have a set of documents that is categorized according to relevance to the query, we can estimate the p_i and q_i directly. However, in ad-hoc retrieval we do not have such relevance information. That means we

have to make some simplifying assumptions in order to be able to rank documents in a meaningful way.

First, we assume that p_i is small and constant for all terms. The first term of g' then becomes $\sum_i X_i \log \frac{p_i}{1-p_i} = c \sum_i X_i$, a simple count of the number of matches between query and document, weighted by c.

The fraction in the second term can be approximated by assuming that most documents are not relevant so $q_i = P(X_i = 1|\neg R) \approx P(w_i) = \frac{\mathrm{df}_i}{N}$, which is the maximum likelihood estimate of $P(w_i)$, the probability of occurrence of w_i not conditioned on relevance.

$$\frac{1-q_i}{q_i} = \frac{1-\frac{\mathrm{df}_i}{N}}{\frac{\mathrm{df}_i}{N}} = \frac{N-\mathrm{df}_i}{\mathrm{df}_i} \approx \frac{N}{\mathrm{df}_i}$$

The last approximation, $\frac{N-\mathrm{df}_i}{\mathrm{df}_i} \approx \frac{N}{\mathrm{df}_i}$, holds for most words since most words are relatively rare. After applying the logarithm, we have now arrived at the IDF weight we introduced earlier. Substituting it back into the formula for g' we get:

$$g'(d) \approx c \sum_i X_i + \sum_i X_i \mathrm{idf}_i \tag{15.7}$$

This derivation may not satisfy everyone since we weight the term according to the 'opposite' of the probability of non-relevance rather than directly according to the probability of relevance. But the probability of relevance is impossible to estimate in ad-hoc retrieval. As in many other cases in Statistical NLP, we take a somewhat circuitous route to get to a desired quantity from others that can be more easily estimated.

15.3.5 Residual inverse document frequency

RESIDUAL INVERSE DOCUMENT FREQUENCY
RIDF

An alternative to IDF is *residual inverse document frequency* or *RIDF*. Residual IDF is defined as the difference between the logs of actual document frequency and document frequency predicted by Poisson:

$$\mathrm{RIDF} = \mathrm{IDF} - \log_2(1 - \mathrm{p}(0;\ \lambda_i))$$

where $\mathrm{IDF} = \log_2 \frac{N}{\mathrm{df}}$, and p is the Poisson distribution with parameter $\lambda_i = \frac{\mathrm{cf}_i}{N}$, the average number of occurrences of w_i per document. $1 - \mathrm{p}(0;\ \lambda_i)$ is the Poisson probability of a document with at least one occurrence. So, for example, RIDF for *insurance* and *try* in table 15.4 would be 7.3 and 6.2, respectively (with $N = 79291$, verify this!).

	Term 1	Term 2	Term 3	Term 4
Query	user	interface		
Document 1	user	interface	HCI	interaction
Document 2			HCI	interaction

Table 15.8 Example for exploiting co-occurrence in computing content similarity. For the query and the two documents, the terms they contain are listed in their respective rows.

As we saw above, the Poisson distribution only fits the distribution of non-content words well. Therefore, the deviation from Poisson is a good predictor of the degree to which a word is a content word.

15.3.6 Usage of term distribution models

We can exploit term distribution models in information retrieval by using the parameters of the model fit for a particular term as indicators of relevance. For example, we could use RIDF or the β in the K mixture as a replacement for IDF weights (since content words have large β and large RIDF, non-content words have smaller β and smaller RIDF).

Better models of term distribution than IDF have the potential of assessing a term's properties more accurately, leading to a better model of query-document similarity. Although there has been little work on employing term distribution models different from IDF in IR, it is to be hoped that such models will eventually lead to better measures of content similarity.

15.4 Latent Semantic Indexing

In the previous section, we looked at the occurrence patterns of individual words. A different source of information about terms that can be exploited in information retrieval is *co-occurrence*: the fact that two or more terms occur in the same documents more often than chance. Consider the example in table 15.8. Document 1 is likely to be relevant to the query since it contains all the terms in the query. But document 2 is also a good candidate for retrieval. Its terms *HCI* and *interaction* co-occur with *user* and *interface*, which can be evidence for semantic relatedness. *Latent Semantic Indexing* (LSI) is a technique that projects queries and

CO-OCCURRENCE

LATENT SEMANTIC INDEXING

$$A = \begin{array}{l|cccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ \hline \text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{array}$$

Figure 15.5 An example of a term-by-document matrix A.

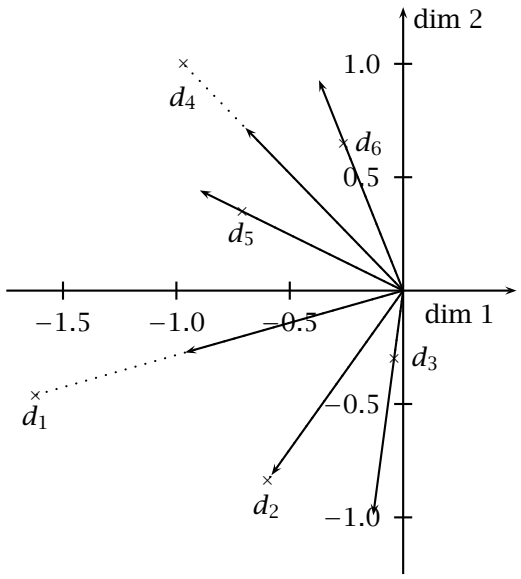

Figure 15.6 Dimensionality reduction. The documents in matrix 15.5 are shown after the five-dimensional term space has been reduced to two dimensions. The reduced document representations are taken from figure 15.11. In addition to the document representations $d_1, \ldots, d_6$, we also show their length-normalized vectors, which show more directly the similarity measure of cosine that is used after LSI is applied.

documents into a space with "latent" semantic dimensions. Co-occurring terms are projected onto the same dimensions, non-co-occurring terms are projected onto different dimensions. In the latent semantic space, a query and a document can have high cosine similarity even if they do not share any terms – as long as their terms are semantically similar according to the co-occurrence analysis. We can look at LSI as a similarity metric that is an alternative to word overlap measures like tf.idf.

DIMENSIONALITY REDUCTION

The latent semantic space that we project into has fewer dimensions than the original space (which has as many dimensions as terms). LSI is thus a method for *dimensionality reduction*. A dimensionality reduction technique takes a set of objects that exist in a high-dimensional space and represents them in a low-dimensional space, often in a two-dimensional or three-dimensional space for the purposes of visualization. The example in figure 15.5 may demonstrate the basic idea. This matrix defines a five-dimensional space (whose dimensions are the five words *astronaut*, *cosmonaut*, *moon*, *car* and *truck*) and six objects in the space, the documents $d_1, \ldots, d_6$. Figure 15.6 shows how the six objects can be displayed in a two-dimensional space after the application of SVD (dimension 1 and dimension 2 are taken from figure 15.11, to be explained later). The visualization shows some of the relations between the documents, in particular the similarity between d_4 and d_5 (*car*/*truck* documents) and d_2 and d_3 (space exploration documents). These relationships are not as clear in figure 15.5. For example, d_2 and d_3 have no terms in common.

There are many different mappings from high-dimensional spaces to low-dimensional spaces. Latent Semantic Indexing chooses the mapping that, for a given dimensionality of the reduced space, is optimal in a sense to be explained presently. This setup has the consequence that the dimensions of the reduced space correspond to the *axes of greatest variation*. Consider the case of reducing dimensionality to 1 dimension. In order to get the best possible representation in 1 dimension, we will look for the axis in the original space that captures as much of the variation in the data as possible. The second dimension corresponds to the axis that best captures the variation remaining after subtracting out what the first axis explains and so on. This reasoning shows that Latent Semantic Indexing is closely related to *Principal Component Analysis* (PCA), another technique for dimensionality reduction. One difference between the two techniques is that PCA can only be applied to a square matrix whereas LSI can be applied to any matrix.

PRINCIPAL COMPONENT ANALYSIS

Latent semantic indexing is the application of a particular mathemat-

ical technique, called Singular Value Decomposition or SVD, to a word-by-document matrix. SVD (and hence LSI) is a least-squares method. The projection into the latent semantic space is chosen such that the representations in the original space are changed as little as possible when measured by the sum of the squares of the differences. We first give a simple example of a least-squares method and then introduce SVD.

15.4.1 Least-squares methods

Before defining the particular least-squares method used in LSI, it is instructive to study the most common least-squares approximation: fitting a line to a set of points in the plane by way of *linear regression*.

LINEAR REGRESSION

Consider the following problem. We have a set of n points: (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). We would like to find the line:

$$f(x) = mx + b$$

with parameters m and b that fits these points best. In a least-squares approximation, the best fit is the one that minimizes the sum of the squares of the differences:

$$SS(m,b) = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \tag{15.8}$$

We compute b by solving $\frac{\partial SS(m,b)}{\partial b} = 0$, the value of b for which $SS(m,b)$ reaches its minimum:

$$\frac{\partial SS(m,b)}{\partial b} = \sum_{i=1}^{n} [2(y_i - mx_i - b)(-1)] = 0$$

$$\Leftrightarrow \quad [\textstyle\sum_{i=1}^{n} y_i] - [m \sum_{i=1}^{n} x_i] - [nb] = 0$$

$$\Leftrightarrow \quad b = \bar{y} - m\bar{x} \tag{15.9}$$

where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ and $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ are the means of the x and y coordinates, respectively.

We now substitute (15.9) for b in (15.8) and solve $\frac{\partial SS(m,b)}{\partial m} = 0$ for m:

$$\frac{\partial SS(m,b)}{\partial m} = \frac{\partial \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2}{\partial m} = 0$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

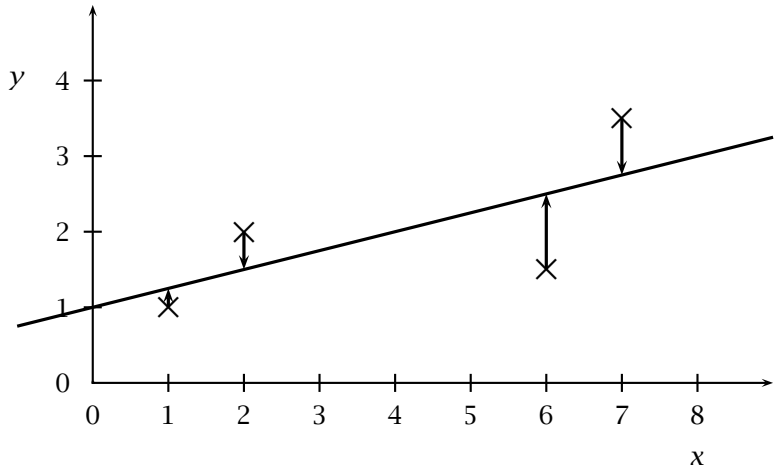

Figure 15.7 An example of linear regression. The line $y = 0.25x + 1$ is the best least-squares fit for the four points (1,1), (2,2), (6,1.5), (7,3.5). Arrows show which points on the line the original points are projected to.

$$\Leftrightarrow \quad 0 = -\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i) + m\sum_{i=1}^{n}(\bar{x} - x_i)^2$$

$$\Leftrightarrow \quad m = \frac{\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i)}{\sum_{i=1}^{n}(\bar{x} - x_i)^2} \tag{15.10}$$

Figure 15.7 shows an example of a least square fit for the four points $(1,1), (2,2), (6,1.5), (7,3.5)$. We have:

$$\bar{x} = 4, \bar{y} = 2, m = \frac{\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i)}{\sum_{i=1}^{n}(\bar{x} - x_i)^2} = \frac{6.5}{26} = 0.25$$

and

$$b = \bar{y} - m\bar{x} = 2 - 0.25 \times 4 = 1$$

15.4.2 Singular Value Decomposition

As we have said, we can view Singular Value Decomposition or SVD as a method of word co-occurrence analysis. Instead of using a simple word overlap measure like the cosine, we instead use a more sophisticated similarity measure that makes better similarity judgements based on word

co-occurrence. Equivalently, we can view SVD as a method for dimensionality reduction. The relation between these two viewpoints is that in the process of dimensionality reduction, co-occurring terms are mapped onto the same dimensions of the reduced space, thus increasing similarity in the representation of semantically similar documents.

Co-occurrence analysis and dimensionality reduction are two 'functional' ways of understanding LSI. We now look at the formal definition of LSI. LSI is the application of Singular Value Decomposition (SVD) to document-by-term matrices in information retrieval. SVD takes a matrix A and represents it as $\hat{A}$ in a lower dimensional space such that the "distance" between the two matrices as measured by the 2-norm is minimized:

$$\Delta = \|A - \hat{A}\|_2 \tag{15.11}$$

The 2-norm for matrices is the equivalent of Euclidean distance for vectors. SVD is in fact very similar to fitting a line, a one-dimensional object, to a set of points, which exists in the two-dimensional plane. Figure 15.7 shows which point on the one-dimensional line each of the original points corresponds to (see arrows).

Just as the linear regression in figure 15.7 can be interpreted as projecting a two-dimensional space onto a one-dimensional line, so does SVD project an n-dimensional space onto a k-dimensional space where $n \gg k$. In our application (word-document matrices), n is the number of word types in the collection. Values of k that are frequently chosen are 100 and 150. The projection transforms a document's vector in n-dimensional word space into a vector in the k-dimensional reduced space.

One possible source of confusion is that equation (15.11) compares the original matrix and a lower-dimensional approximation. Shouldn't the second matrix have fewer rows and columns, which would make equation (15.11) ill-defined? The analogy with line fitting is again helpful here. The fitted line exists in two dimensions, but it is a one-dimensional object. The same is true for $\hat{A}$: it is a matrix of lower rank, that is, it could be represented in a lower-dimensional space by transforming the axes of the space. But for the particular axes chosen it has the same number of rows and columns as A.

The SVD projection is computed by decomposing the document-by-term matrix $A_{t\times d}$ into the product of three matrices, $T_{t\times n}$, $S_{n\times n}$, and $D_{d\times n}$:

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^{\mathrm{T}} \tag{15.12}$$

$$T = \begin{pmatrix} & \text{cosm.} & \text{astr.} & \text{moon} & \text{car} & \text{truck} \\ \hline \text{Dimension 1} & -0.44 & -0.13 & -0.48 & -0.70 & -0.26 \\ \text{Dimension 2} & -0.30 & -0.33 & -0.51 & 0.35 & 0.65 \\ \text{Dimension 3} & 0.57 & -0.59 & -0.37 & 0.15 & -0.41 \\ \text{Dimension 4} & 0.58 & 0.00 & 0.00 & -0.58 & 0.58 \\ \text{Dimension 5} & 0.25 & 0.73 & -0.61 & 0.16 & -0.09 \end{pmatrix}$$

Figure 15.8 The matrix T of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

$$S = \begin{pmatrix} 2.16 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.59 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.28 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.39 \end{pmatrix}$$

Figure 15.9 The matrix of singular values of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

where $n = \min(t, d)$. We indicate dimensionality by subscripts: A has t rows and d columns, T has t rows and n columns and so on. D^{T} is the transpose of D, the matrix D rotated around its diagonal: $D_{ij} = (D^{\mathrm{T}})_{ji}$.

Examples of A, T, S, and D are given in figure 15.5 and figures 15.8 through 15.10. Figure 15.5 shows an example of A. A contains the document vectors with each column corresponding to one document. In other words, element a_{ij} of the matrix records how often term i occurs in document j. The counts should be appropriately weighted (as discussed in section 15.2). For simplicity of exposition, we have not applied weighting and assumed term frequencies of 1.

ORTHONORMAL

Figures 15.8 and 15.10 show T and D, respectively. These matrices have *orthonormal* columns. This means that the column vectors have unit length and are all orthogonal to each other. (If a matrix C has orthonormal columns, then $C^{\mathrm{T}}C = I$, where I is the diagonal matrix with a diagonal of 1's, and zeroes elsewhere. So we have $T^{\mathrm{T}}T = D^{\mathrm{T}}D = I$.)

We can view SVD as a method for rotating the axes of the n-dimensional space such that the first axis runs along the direction of largest variation

$D =$

	d_1	d_2	d_3	d_4	d_5	d_6
Dimension 1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
Dimension 2	−0.29	−0.53	−0.19	0.63	0.22	0.41
Dimension 3	0.28	−0.75	0.45	−0.20	0.12	−0.33
Dimension 4	0.00	0.00	0.58	0.00	−0.58	0.58
Dimension 5	−0.53	0.29	0.63	0.19	0.41	−0.22

Figure 15.10 The matrix D of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

among the documents, the second dimension runs along the direction with the second largest variation and so forth. The matrices T and D represent terms and documents in this new space. For example, the first column of T corresponds to the first row of A, and the first column of D corresponds to the first column of A.

The diagonal matrix S contains the singular values of A in descending order (as in figure 15.9). The i^{th} singular value indicates the amount of variation along the i^{th} axis. By restricting the matrices T, S, and D to their first $k < n$ rows one obtains the matrices $T_{t\times k}$, $S_{k\times k}$, and $(D_{d\times k})^{\text{T}}$. Their product $\hat{A}$ is the best least square approximation of A by a matrix of rank k in the sense defined in equation (15.11). One can also prove that SVD is unique, that is, there is only one possible decomposition of a given matrix.[1] See Golub and van Loan (1989) for an extensive treatment of SVD including a proof of the optimality property.

That SVD finds the optimal projection to a low-dimensional space is the key property for exploiting word co-occurrence patterns. SVD represents terms and documents in the lower dimensional space as well as possible. In the process, some words that have similar co-occurrence patterns are projected (or collapsed) onto the same dimension. As a consequence, the similarity metric will make topically similar documents and queries come out as similar even if different words are used for describing the topic. If we restrict the matrix in figure 15.8 to the first two dimensions, we end up with two groups of terms: space exploration terms (*cosmonaut*, *astronaut*, and *moon*) which have negative values on the second dimension

1. SVD is unique up to sign flips. If we flip all signs in the matrices D and T, we get a second solution.

	d_1	d_2	d_3	d_4	d_5	d_6
Dimension 1	−1.62	−0.60	−0.04	−0.97	−0.71	−0.26
Dimension 2	−0.46	−0.84	−0.30	1.00	0.35	0.65

Figure 15.11 The matrix $B = S_{2\times2}D_{2\times n}$ of documents after rescaling with singular values and reduction to two dimensions. Values are rounded.

	d_1	d_2	d_3	d_4	d_5	d_6
d_1	1.00					
d_2	0.78	1.00				
d_3	0.40	0.88	1.00			
d_4	0.47	−0.18	−0.62	1.00		
d_5	0.74	0.16	−0.32	0.94	1.00	
d_6	0.10	−0.54	−0.87	0.93	0.74	1.00

Table 15.9 The matrix of document correlations $B^{\mathrm{T}}B$. For example, the normalized correlation coefficient of documents d_3 and d_2 (when represented as in figure 15.11) is 0.88. Values are rounded.

and automobile terms (*car* and *truck*) which have positive values on the second dimension. The second dimension directly reflects the different co-occurrence patterns of these two groups: space exploration terms only co-occur with other space exploration terms, automobile terms only co-occur with other automobile terms (with one exception: the occurrence of *car* in d_1). In some cases, we will be misled by such co-occurrences patterns and wrongly infer semantic similarity. However, in most cases co-occurrence is a valid indicator of topical relatedness.

These term similarities have a direct impact on document similarity. Let us assume a reduction to two dimensions. After rescaling with the singular values, we get the matrix $B = S_{2\times2}D_{2\times n}$ shown in figure 15.11 where $S_{2\times2}$ is S restricted to two dimensions (with the diagonal elements 2.16, 1.59). Matrix B is a dimensionality reduction of the original matrix A and is what was shown in figure 15.6.

Table 15.9 shows the similarities between documents when they are represented in this new space. Not surprisingly, there is high similarity between d_1 and d_2 (0.78) and d_4, d_5, and d_6 (0.94, 0.93, 0.74). These document similarities are about the same in the original space (i.e. when we compute correlations for the original document vectors in figure 15.5). The key change is that d_2 and d_3, whose similarity is 0.00 in the original

space, are now highly similar (0.88). Although d_2 and d_3 have no common terms, they are now recognized as being topically similar because of the co-occurrence patterns in the corpus.

Notice that we get the same similarity as in the original space (that is, zero similarity) if we compute similarity in the transformed space without any dimensionality reduction. Using the full vectors from figure 15.10 and rescaling them with the appropriate singular values we get:

$$-0.28 \times -0.20 \times 2.16^2 + -0.53 \times -0.19 \times 1.59^2 +$$
$$-0.75 \times 0.45 \times 1.28^2 + 0.00 \times 0.58 \times 1.00^2 + 0.29 \times 0.63 \times 0.39^2 \approx 0.00$$

(If you actually compute this expression, you will find that the answer is not quite zero, but this is only because of rounding errors. But this is as good a point as any to observe that many matrix computations are quite sensitive to rounding errors.)

We have computed document similarity in the reduced space using the product of D and S. The correctness of this procedure can be seen by looking at $A^{\mathrm{T}}A$, which is the matrix of all document correlations for the original space:

$$A^{\mathrm{T}}A = (TSD^{\mathrm{T}})^{\mathrm{T}}TSD^{\mathrm{T}} = DS^{\mathrm{T}}T^{\mathrm{T}}TSD^{\mathrm{T}} = (DS)(DS)^{\mathrm{T}} \tag{15.13}$$

Because T has orthonormal columns, we have $T^{\mathrm{T}}T = I$. Furthermore, since S is diagonal, $S = S^{\mathrm{T}}$. Term similarities are computed analogously since one observes that the term correlations are given by:

$$AA^{\mathrm{T}} = TSD^{\mathrm{T}}(TSD^{\mathrm{T}})^{\mathrm{T}} = TSD^{\mathrm{T}}DS^{\mathrm{T}}T^{\mathrm{T}} = (TS)(TS)^{\mathrm{T}} \tag{15.14}$$

One remaining problem for a practical application is how to fold queries and new documents into the reduced space. The SVD computation only gives us reduced representations for the document vectors in matrix A. We do not want to do a completely new SVD every time a new query is launched. In addition, in order to handle large corpora efficiently we may want to do SVD for only a sample of the documents (for example a third or a fourth). The remaining documents would then be folded in.

The equation for folding documents into the space can again be derived from the basic SVD equation:

$$\begin{aligned} & A = TSD^{\mathrm{T}} && (15.15)\\ \Leftrightarrow\quad & T^{\mathrm{T}}A = T^{\mathrm{T}}TSD^{\mathrm{T}} \\ \Leftrightarrow\quad & T^{\mathrm{T}}A = SD^{\mathrm{T}} \end{aligned}$$

So we just multiply the query or document vector with the transpose of the term matrix T (after it has been truncated to the desired dimensionality). For example, for a query vector $\vec{q}$ and a reduction to dimensionality k, the query representation in the reduced space is ${T_{t\times k}}^{\mathrm{T}}\vec{q}$.

15.4.3 Latent Semantic Indexing in IR

LATENT SEMANTIC INDEXING

The application of SVD to information retrieval was originally proposed by a group of researchers at Bellcore (Deerwester et al. 1990) and called *Latent Semantic Indexing* (LSI) in this context. LSI has been compared to standard vector space search on several document collections. It was found that LSI performs better than vector space search in many cases, especially for high-recall searches (Deerwester et al. 1990; Dumais 1995). LSI's strength in high-recall searches is not surprising since a method that takes co-occurrence into account is expected to achieve higher recall. On the other hand, due to the noise added by spurious co-occurrence data one sometimes finds a decrease in precision.

The appropriateness of LSI also depends on the document collection. Recall the example of the vocabulary problem in figure 15.8. In a heterogeneous collection, documents may use different words to refer to the same topic like *HCI* and *user interface* in the figure. Here, LSI can help identify the underlying semantic similarity between seemingly dissimilar documents. However, in a collection with homogeneous vocabulary, LSI is less likely to be useful.

The application of SVD to information retrieval is called *Latent Semantic Indexing* because the document representations in the original term space are transformed to representations in a new reduced space. The dimensions in the reduced space are linear combinations of the original dimensions (this is so since matrix multiplications as in equation (15.16) are linear operations). The assumption here (and similarly for other forms of dimensionality reduction like principal component analysis) is that these new dimensions are a better representation of documents and queries. The metaphor underlying the term "latent" is that these new dimensions are the true representation. This true representation was then obscured by a generation process that expressed a particular dimension with one set of words in some documents and a different set of words in another document. LSI analysis recovers the original semantic structure of the space and its original dimensions. The process of assigning different words to the same underlying dimension is sometimes interpreted

as a form of soft term clustering since it groups terms according to the dimensions that they are represented on in the reduced space.

One could also argue that the SVD representation is not only better (since it is based on the 'true' dimensions), but also more compact. Many documents have more than 150 unique terms. So the sparse vector representation will take up more storage space than the compact SVD representation if we reduce to 150 dimensions. However, the efficiency gain due to more compact representations is often outweighed by the additional cost of having to go through a high-dimensional matrix multiplication whenever we map a query or a new document to the reduced space. Another problem is that an inverted index cannot be constructed for SVD representations. If we have to compute the similarity between the query and every single document, then an SVD-based system can be slower than a term-based system that searches an inverted index.

The actual computation of SVD is quadratic in the rank of the document by term matrix (the rank is (bounded by) the smaller of the number of documents and the number of terms) and cubic in the number of singular values that are computed (Deerwester et al. 1990: 395).[2] For very large collections subsampling of documents and selection of terms according to frequency is often employed in order to reduce the cost of computing the Singular Value Decomposition.

NORMALITY ASSUMPTION

One objection to SVD is that, along with all other least-squares methods, it is really designed for normally-distributed data. But, as can be seen from the discussion earlier in this chapter, such a distribution is inappropriate for count data, and count data is, after all, what a term-by-document matrix consists of. The link between least squares and normal distribution can be easily seen by looking at the definition of the normal distribution (section 2.1.9):

$$\mathrm{n}(x;\,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

where μ is the *mean* and σ the *covariance*. The smaller the squared deviation from the mean $(x-\mu)^2$, the higher the probability $n(x;\,\mu,\sigma)$. So the least squares solution is the maximum likelihood solution. But this is only true if the underlying data distribution is normal. Other distri-

2. However, others have suggested that given the particular characteristics of the matrices that SVD is applied to in information retrieval the complexity is linear in the number of documents and (approximately) quadratic in the number of singular values. See (Oard and DeClaris 1996), (Berry et al. 1995), and (Berry and Young 1995) for discussion.

butions like Poisson or negative binomial are more appropriate for term counts. One problematic feature of SVD is that, since the reconstruction $\hat{A}$ of the term-by-document matrix A is based on a normal distribution, it can have negative entries, clearly an inappropriate approximation for counts. A dimensionality reduction based on Poisson would not predict such impossible negative counts.

In defense of LSI (and the vector space model in general which can also be argued to assume a normal distribution), one can say that the matrix entries are not counts, but weights. Although this is not an issue that has been investigated systematically, the normal distribution could be appropriate for the weighted vectors even if it is not for count vectors.

From a practical point of view, LSI has been criticized for being computationally more expensive than other word co-occurrence methods while not being more effective. Another method that also uses co-occurrence is *pseudo-feedback* (also called *pseudo relevance feedback* and *two-stage retrieval*, Buckley et al. 1996; Kwok and Chan 1998). In pseudo-feedback, the top n documents (typically the top 10 or 20) returned by an ad-hoc query are assumed to be relevant and added to the query. Some of these top n documents will not actually be relevant, but a large enough proportion usually is to improve the quality of the query. Words that occur frequently with query words will be among the most frequent in the top n. So pseudo-feedback can be viewed as a cheap query-specific way of doing co-occurrence analysis and co-occurrence-based query modification.

PSEUDO-FEEDBACK

Still, in contrast to many heuristic methods that incorporate term co-occurrence into information retrieval, LSI has a clean formal framework and a clearly defined optimization criterion (least squares) with one global optimum that can be efficiently computed. This conceptual simplicity and clarity make LSI one of the most interesting IR approaches that go beyond query-document term matching.

15.5 Discourse Segmentation

Text collections are increasingly heterogeneous. An important aspect of heterogeneity is length. On the world wide web, document sizes range from home pages with just one sentence to server logs of half a megabyte.

The weighting schemes discussed in section 15.2.2 take account of different lengths by applying cosine normalization. However, cosine normalization and other forms of normalization that discount term weights

according to document length ignore the distribution of terms within a document. Suppose that you are looking for a short description of angioplasty. You would probably prefer a document in which the occurrences of angioplasty are concentrated in one or two paragraphs since such a concentration is most likely to contain a definition of what angioplasty is. On the other hand, a document of the same length in which the occurrences of angioplasty are scattered uniformly is less likely to be helpful.

We can exploit the structure of documents and search over structurally defined units like sections and paragraphs instead of full documents. However, the best subpart of a document to be returned to the user often encompasses several paragraphs. For example, in response to a query on angioplasty we may want to return the first two paragraphs of a subsection on angioplasty, which introduce the term and its definition, but not the rest of the subsection that goes into technical detail.

Some documents are not structured into paragraphs and sections. Or, in the case of documents structured by means of a markup language like HTML, it is not obvious how to break them apart into units that would be suitable for retrieval.

These considerations motivate an approach that breaks documents into topically coherent multi-paragraph subparts. In the rest of this subsection we will describe one approach to multiparagraph segmentation,

TEXTTILING

the *TextTiling* algorithm (Hearst and Plaunt 1993; Hearst 1994, 1997).

15.5.1 TextTiling

The basic idea of this algorithm is to search for parts of a text where the

SUBTOPIC

vocabulary shifts from one subtopic to another. These points are then interpreted as the boundaries of multi-paragraph units.

Sentence length can vary considerably. Therefore, the text is first di-

TOKEN SEQUENCES

vided into small fixed size units, the *token sequences*. Hearst suggests a size of 20 words for token sequences. We refer to the points between

GAPS

token sequences as *gaps*. The TextTiling algorithm has three main components: the *cohesion scorer*, the *depth scorer* and the *boundary selector*.

COHESION SCORER

The *cohesion scorer* measures the amount of 'topic continuity' or cohesion at each gap, that is, the amount of evidence that the same subtopic is prevalent on both sides of the gap. Intuitively, we want to consider gaps with low cohesion as possible segmentation points.

DEPTH SCORER

The *depth scorer* assigns a depth score to each gap depending on how low its cohesion score is compared to the surrounding gaps. If cohesion

at the gap is lower than at surrounding gaps, then the depth score is high. Conversely, if cohesion is about the same at surrounding gaps, then the depth score is low. The intuition here is that cohesion is relative. One part of the text (say, the introduction) may have many successive shifts in vocabulary. Here we want to be cautious in selecting subtopic boundaries and only choose those points with the lowest cohesion scores compared to their neighbors. Another part of the text may have only slight shifts for several pages. Here it is reasonable to be more sensitive to topic changes and change points that have relatively high cohesion scores, but scores that are low compared to their neighbors.

BOUNDARY SELECTOR

The *boundary selector* is the module that looks at the depth scores and selects the gaps that are the best segmentation points.

Several methods of cohesion scoring have been proposed.

- *Vector Space Scoring.* We can form one artificial document out of the token sequences to the left of the gap (the *left block*) and another artificial document to the right of the gap (the *right block*). (Hearst suggests a length of two token sequences for each block.) These two blocks are then compared by computing the correlation coefficient of their term vectors, using the weighting schemes that were described earlier in this chapter for the vector space model. The idea is that the more terms two blocks share the higher their cohesion score and the less likely they will be classified as a segment boundary. Vector Space Scoring was used by Hearst and Plaunt (1993) and Salton and Allen (1993).
- *Block comparison.* The block comparison algorithm also computes the correlation coefficient of the gap's left block and right block, but it only uses within-block term frequency without taking into account (inverse) document frequency.
- *Vocabulary introduction.* A gap's cohesion score in this algorithm is the negative of the number of new terms that occur in left and right block, that is terms that have not occurred up to this point in the text. The idea is that subtopic changes are often signaled by the use of new vocabulary (Youmans 1991). (In order to make the score a cohesion score we multiply the count of new terms by -1 so that larger scores (fewer new terms) correspond to higher cohesion and smaller scores (more new terms) correspond to lower cohesion.)

The experimental evidence in (Hearst 1997) suggests that Block Comparison is the best performing of these three algorithms.

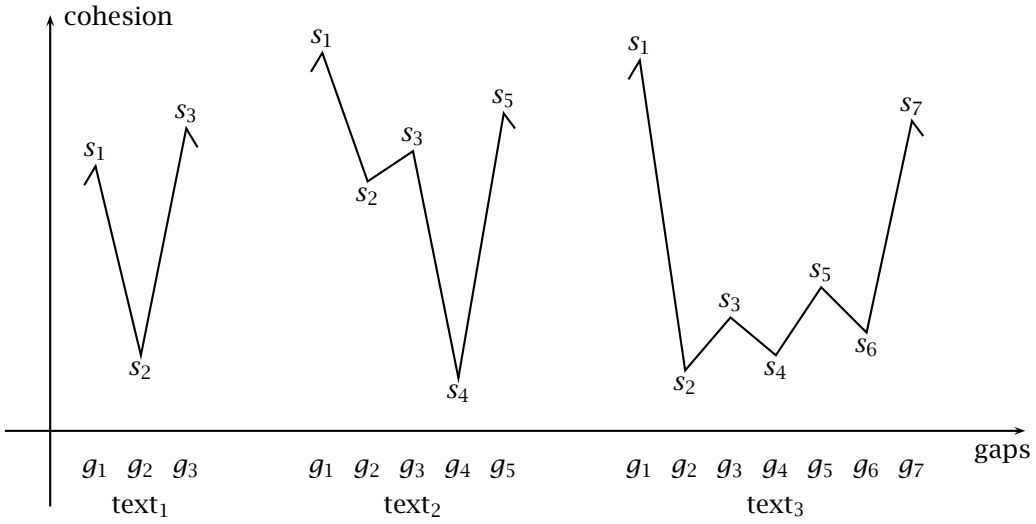

Figure 15.12 Three constellations of cohesion scores in topic boundary identification.

The second step in TextTiling is the transformation of cohesion scores into *depth scores*. We compute the depth score for a gap by summing the heights of the two sides of the valley it is located in, for example $(s_1 - s_2) + (s_3 - s_2)$ for g_2 in text 1 in figure 15.12. Note that high absolute values of the cohesion scores by themselves will not result in the creation of a segment boundary. TextTiling views subtopic changes and segmentation as relative. In a text with rapid fluctuations of topic or vocabulary from paragraph to paragraph only the most radical changes will be accorded the status of segment boundaries. In a text with only subtle subtopic changes the algorithm will be more discriminating.

For a practical implementation, several enhancements of the basic algorithm are needed. First, we need to smooth cohesion scores to address situations like the one in text 2 in figure 15.12. Intuitively, the difference $s_1 - s_2$ should contribute to the depth score of gap g_4. This is achieved by *smoothing* scores using a low pass filter. For example, the depth score s_i for g_i is replaced by $(s_{i-1} + s_i + s_{i+1})/3$. This procedure effectively takes into consideration the cohesion scores of gaps at a distance of two from the central gap. If they are as high as or higher than the two immediately surrounding gaps, they will increase the score of the central gap.

We also need to add heuristics to avoid a sequence of many small segments (this type of segmentation is rarely chosen by human judges when they segment text into coherent units). Finally, the parameters of the methods for computing cohesion and depth scores (size of token sequence, size of block, smoothing method) may have to be adjusted depending on the text sort we are working with. For example, a corpus with long sentences will require longer token sequences.

The third component of TextTiling is the boundary selector. It estimates average μ and standard deviation σ of the depth scores and selects all gaps as boundaries that have a depth score higher than $\mu - c\sigma$ for some constant c (for example, $c = 0.5$ or $c = 1.0$). We again try to avoid using absolute scores. This method selects gaps that have 'significantly' low depth scores, where significant is defined with respect to the average and the variance of scores.

In an evaluation, Hearst (1997) found good agreement between segments found by TextTiling and segments demarcated by human judges. It remains an open question to what degree segment retrieval leads to better information retrieval performance than document retrieval when evaluated on precision and recall. However, many users prefer to see a hit in the context of a natural segment which makes it easier to quickly understand the context of the hit (Egan et al. 1989).

Text segmentation could also have important applications in other areas of Natural Language Processing. For example, in word sense disambiguation segmentation could be used to find the natural units that are most informative for determining the correct sense of a usage. Given the increasing diversity of document collections, discourse segmentation is guaranteed to remain an important topic of research in Statistical NLP and IR.

15.6 Further Reading

Two major venues for publication of current research in IR are the TREC proceedings (Harman 1996, see also the links on the website), which report results of competitions sponsored by the US government, and the ACM SIGIR proceedings series. Prominent journals are *Information Processing & Management*, the *Journal of the American Society for Information Science*, and *Information Retrieval*.

The best known textbooks on information retrieval are books by van

Rijsbergen (1979), Salton and McGill (1983) and Frakes and Baeza-Yates (1992). See also (Losee 1998) and (Korfhage 1997). A collection of seminal papers was recently edited by Sparck Jones and Willett (1998). Smeaton (1992) and Lewis and Jones (1996) discuss the role of NLP in information retrieval. Evaluation of IR systems is discussed in (Cleverdon and Mills 1963), (Tague-Sutcliffe 1992), and (Hull 1996). Inverse document frequency as a term weighting method was proposed by Sparck Jones (1972). Different forms of tf.idf weighting were extensively investigated within the SMART project at Cornell University, led by Gerard Salton (Salton 1971b; Salton and McGill 1983). Two recent studies are (Singhal et al. 1996) and (Moffat and Zobel 1998).

The Poisson distribution is further discussed in most introductions to probability theory, e.g., (Mood et al. 1974: 95). See (Harter 1975) for a way of estimating the parameters π, λ_1, and λ_2 of the two-Poisson model without having to assume a set of documents labeled as to their class membership. Our derivation of IDF is based on (Croft and Harper 1979)). RIDF was introduced by Church (1995).

Apart from work on better phrase extraction, the impact of NLP on IR in recent decades has been surprisingly small, with most IR researchers focusing on shallow analysis techniques. Some exceptions are (Fagan 1987; Bonzi and Liddy 1988; Sheridan and Smeaton 1992; Strzalkowski 1995; Klavans and Kan 1998). However, recently there has been much more interest in tasks such as automatically summarizing documents rather than just returning them as is (Salton et al. 1994; Kupiec et al. 1995), and such trends may tend to increase the usefulness of NLP in IR applications.

CROSS-LANGUAGE INFORMATION RETRIEVAL
CLIR

One task that has benefited from the application of NLP techniques is *cross-language information retrieval* or *CLIR* (Hull and Grefenstette 1998; Grefenstette 1998). The idea is to help a user who has enough knowledge of a foreign language to understand texts, but not enough fluency to formulate a query. In CLIR, such a user can type in a query in her native language, the system then translates the query into the target language and retrieves documents in the target language. Recent work includes (Sheridan et al. 1997; Nie et al. 1998) and the Notes of the AAAI symposium on cross-language text and speech retrieval (Hull and Oard 1997). Littman et al. (1998b) and Littman et al. (1998a) use Latent Semantic Indexing for CLIR.

We have only presented a small selection of work on modeling term distributions in IR. See (van Rijsbergen 1979: ch. 6) for a more systematic introduction. (Robertson and Sparck Jones 1976) and (Bookstein and

Swanson 1975) are other important papers (the latter is a decision theoretic approach). Information theory has also been used to motivate IDF (Wong and Yao 1992). An application of residual inverse document frequency to the characterization of index terms is described by Yamamoto and Church (1998).

The example in table 15.8 is adapted from (Deerwester et al. 1990). The term-by-document matrix we used as an example for SVD is small. It can easily be decomposed using one of the standard statistical packages (we used S-plus). For a large corpus, we have to deal with several hundred thousand terms and documents. Special algorithms have been developed for this purpose. See (Berry 1992) and NetLib on the world wide web for a description and implementation of several such algorithms.

Apart from term-by-document matrices, SVD has been applied to word-by-word matrices by Schütze and Pedersen (1997) and to discourse segmentation (Kaufmann 1998). Dolin (1998) uses LSI for query categorization and distributed search, using automated classification for collection summarization.

Latent Semantic Indexing has also been proposed as a cognitive model for human memory. Landauer and Dumais (1997) argue that it can explain the rapid vocabulary growth found in school-age children.

TEXT SEGMENTATION

Text segmentation is an active area of research. Other work on the problem includes (Salton and Buckley 1991), (Beeferman et al. 1997) and (Berber Sardinha 1997). Kan et al. (1998) make an implementation of their segmentation algorithm publicly available (see website). An information source that is different from the word overlap measure used in TextTiling is so-called *lexical chains*: chains of usages of one or more semantically related words throughout a text. By observing starts, interruptions, and terminations of such chains, one can derive a different type of description of the subtopic structure of text (Morris and Hirst 1991).

LEXICAL CHAINS

Text segmentation is a rather crude treatment of the complexity of written texts and spoken dialogues which often have a hierarchical and non-linear structure. Trying to do justice to this complex structure is a much harder task than merely detecting topic changes. Finding the best approach to this problem is an active area of research in Statistical NLP. The special issue of Computational Linguistics on empirical *discourse analysis*, edited by Walker and Moore (1997), is a good starting point for interested readers.

DISCOURSE ANALYSIS

When Statistical NLP methods became popular again in the early 1990s, discourse modeling was initially an area with a low proportion of statisti-

cal work, but there has been a recent surge in the application of quantitative methods. For some examples, see (Stolcke et al. 1998), (Walker et al. 1998) and (Samuel et al. 1998) for probabilistic approaches to *dialog modeling* and (Kehler 1997) and (Ge et al. 1998) for probabilistic approaches to *anaphora resolution*.

DIALOG MODELING

ANAPHORA RESOLUTION

15.7 Exercises

Exercise 15.1 [⋆]

Try to find out the characteristics of various internet search engines. Do they use a stop list? Try to search for stop words. Can you search for the phrase *the the*? Do the engines use stemming? Do they normalize words to lowercase? For example, does a search for *iNfOrMaTiOn* return anything?

Exercise 15.2 [⋆]

The simplest way to process phrases in the vector space model is to add them as separate terms. For example, the query *car insurance rates* might be translated into an internal representation that contains the terms *car*, *insurance*, *rates*, *car insurance*, *insurance rates*. This means that phrases and their constituent words are treated as independent sources of evidence. Discuss why this is problematic.

Exercise 15.3 [⋆]

Show that Katz's K mixture satisfies:

$$P_i(0) = 1 - \frac{\mathrm{df}_i}{N}.$$

That is, the fit of the estimate to the actual count is always perfect for the number of documents with zero occurrences.

Exercise 15.4 [⋆]

Compute Residual IDF for the words in table 15.7. Are content words and non-content words well separated?

Exercise 15.5 [⋆]

Select a non-content word, a content word and a word which you are not sure how to classify and compute the following quantities for them: (a) document frequency and collection frequency, (b) IDF, (c) RIDF, and (d) α and β of the K mixture. (You can use any reasonable size corpus of your choice.)

Exercise 15.6 [⋆]

Depending on λ, the Poisson distribution is either monotonically falling or has the shape of a curve that first rises, and then falls. Find examples of each. What is the property of λ that determines the shape of the graph?

Exercise 15.7 [⋆]

Compute the SVD decomposition of the term-by-document matrix in figure 15.5 using S-Plus or another software package.

Exercise 15.8 [⋆⋆]

In this exercise, we look at two assumptions about subtopic structure that are made in TextTiling.

First, TextTiling performs a *linear* segmentation, that is, a text is divided into a sequence of segments. No attempt is made to impose further structure. One example where the assumption of linearity is not justified is noted by Hearst: a sequence of three paragraphs that is then summarized in a fourth paragraph. Since the summary paragraph has vocabulary from paragraphs 1 and 2 that does not occur in paragraph 3 a segment boundary between 3 and 4 is inferred. Propose a modification of TextTiling that would recognize paragraphs 1–4 as a unit.

Another assumption that TextTiling relies on is that most segment boundaries are characterized by pronounced valleys like the one in text 1 in figure 15.12. But sometimes there is a longer flat region between two segments. Why is this a problem for the formulation of the algorithm described above? How could one fix it?