Information Retrieval

Jacques Savoy
Université de Neuchâtel
226 PUBLICATIONS **3,436** CITATIONS
SEE PROFILE

Eric Gaussier
Université Grenoble Alpes
270 PUBLICATIONS **6,362** CITATIONS
SEE PROFILE

Information Retrieval

Jacques Savoy[1] and Eric Gaussier[2]

[1] Computer Science Dept., University of Neuchatel
rue Emile Argand 11, 2009 Neuchatel (Switzerland)
Jacques.Savoy@unine.ch
[2] Laboratoire LIG, Université Joseph Fourier
385 rue de la bibliothèque, 38041 Grenoble Cedex 9 (France)
Eric.Gaussier@imag.fr

Abstract. This chapter presents the fundamental concepts of Information Retrieval (IR) and shows how this domain is related to various aspects of NLP. After explaining some of the underlying and often hidden assumptions and problems of IR, we present the notion of indexing. Indexing is the cornerstone of various classical IR paradigms (Boolean, vector-space, and probabilistic) which we introduce together with some insights to advanced search strategies used on the Web, such as PageRank. The IR community relies on a strong empirical tradition and we present the basic notions of IR evaluation methodology and show, with concrete examples, why some topic formulations can be hard even with the most advanced search strategies. Various NLP techniques can be used to, at least partially, improve the retrieval performance of IR models. We devote a section of this chapter to an overview of these techniques.

1 Introduction

The Information Retrieval (IR) [1] domain can be viewed, to a certain extent, as a successful applied domain of NLP. The speed and scale of Web take-up around the world has been made possible by freely available and effective search engines. These tools are used by around 85% of Web surfers when looking for some specific information [2].

But what precisely is IR? We can define it the following way: "IR deals with the representation, storage, organization of, and access to information items. These information items could be references to real documents, documents themselves, or even single paragraphs, as well as Web pages, spoken documents, images, pictures, music, video, etc." [3]

As information items, we will focus mainly on documents written in the English language, but the ideas and concepts we introduce can also be applied, with some adaptations however, to other media (music, image, picture, video) and other languages (e.g., German, Spanish, Russian, Chinese, Japanese, etc.).

One of the first and most important characteristics of IR is the fact that an IR system has to deal with imprecise and incomplete descriptions of both user needs and documents queried: in most cases, it is impossible to compute a precise and unambiguous representation for queries of documents. This contrasts with the situation for databases. In a relational table for example, John has a wage of \$1,198.5 and not "almost \$1,200". If SQL queries cannot be ambiguous, users sending queries to Web search engines tend to write very short and ambiguous descriptions of their information needs ("Canadian recipes," "iPhone" or "Britney Spears"). Clearly, users do not describe their information needs with all the needed details, preferring the system to suggest some answers, and selecting, from the answers, the most appropriate items. The search process should be viewed more as a "trial-and-error" problem solving approach than a direct "query-response" paradigm. Finally, and contrary to SQL-based search, the matching between the query and the information item is not deterministic: the system provides the best possible answers (best matches) by estimating their underlying probability of relevance to the submitted query.

The underlying difficulty in retrieving documents relevant to a query resides in the three aspects of all natural languages, namely polysemy, synonymy and, to a lesser extent, spelling errors and variations. In all natural languages, a given word may have more than one precise meaning (polysemy). Some of these senses could be related, but in other cases, the underlying meaning could vary greatly. For example, the word "bank" could be used to designate a financial institution, its building, a synonym of "rely upon" in the expression "I'm your friend, you can bank on me" or the borders of a river. The last sense is of course not related to the previous three. With the word "Java", the possible meanings are less related (an island, coffee, a dance, a domestic fowl, a computer programming language). If this word were submitted as a query, we can understand why the search system may provide "incorrect" answers. Acronyms are also subject of such ambiguity problems as, for example, BSE (Bovine Spongiform Encephalopathy, Bombay Stock Exchange (or Boston, Beirut, Bahrain), Breast Self-Examination, Bachelor of Science in Engineering, Basic Service Element, etc.).

Polysemy corresponds to one face of a coin. Synonymy, i.e. the fact that different words or expressions can be used to refer to the same object, is the second. For example, to refer to a given car accident, we can use the following terms: "accident," "event," "incident," "situation," "problem," "difficulty," "unfortunate situation," "the subject of your last letter," "what happened last month," ... When fixing a rendez-vous, we can meet in a restaurant, hotel, pizzeria, coffee shop, snack bar, café, tearoom, tea house, public house, cafeteria, inn, tavern, or simply at our favorite Starbucks. In the last example below, the expressions written in italics refer to the same person. It is also interesting to note that the string "Major" does not always refer to the same person.

"*Mr Major* arrived in France today. *The Prime Minister* will meet the President tomorrow. *The Conservative leader* will then travel to Moscow where *he* will meet Mr Gorbachev. Mrs Major will join *her husband* in Russia, where *this son of a circus artist* is a relatively unknown figure." [4]

As a result of these two main problems, and as demonstrated empirically in [5], the probability that two people use the same term to describe the same object is below 0.20. The question that then arises is to understand how a search system may operate at a reasonable performance level with such variability, a question we will address by considering the different components that make up a search system.

All search systems assume that similar words tend to correspond to similar meanings. Thus when a query shares many terms with a document, this document is seen as an appropriate answer to the corresponding query. Search systems are thus based on a simple approach: extract words from documents and queries, compare the two sets of words and rank the document according to this comparison (potentially using additional elements, such as PageRank). As shown by Hawking & Robertson [6], as the collection size increases, the achieved performance of such a process tends to increase.

Of course, different components must be described in more detail; the rest of this chapter is organized as follows: Section 2 outlines the main aspects of the indexing process used to build documents or query representations while Section 3 discusses various IR models. Section 4 describes the evaluation methodology used in the IR field in order to compare objectively two (or more) indexing and search strategies. Failure analysis is also described in this section. Finally, Section 5 presents various aspects of NLP approaches that can be used to answer some of the topic difficulties that each IR system encounters.

2 Indexing

Effective search systems do not work directly with the documents (or the queries). They use different techniques and strategies to represent the main semantic aspects of the documents and queries. This process is called indexing. It is described in the following two sections.

2.1 Indexing Dimensions

We focus here on the representation of the semantic content of the documents. External characteristics, such as the publication date, the author name, the number of pages, the book price, the edition, the publisher, the language, can be managed without real problem by a relational database system. Of course, we can use such features to conduct a search (e.g., "return all documents written by Salton"). They tend however to be used as filters to complement the search criteria (e.g., "best picture Oscar later than 2002" or "airplane hijacking, information in English or German").

The indexing process aims to represent the semantic content of documents (or queries) by a set of indexing features. In the most common case, these indexing units are words for documents, musical notes for music, color values for pictures, etc. If we limit ourselves to written documents, we can consider as indexing units not only single words (e.g., "language," "natural") but compound constructions ("natural language processing") or thesaurus class numbers. In the opposite direction, we can sometimes consider decompounding compound words (e.g., from "handgun", we may extract the terms "hand" and "gun", a useful feature when searching with the German language [7]). Furthermore, sentences or words may be subdivided into continuous segments of n characters (n-gram indexing [8]). Fixing $n = 4$, the phrase "white house" is represented by the following 4-gram sequence: "whit", "hite", "ite h", "te ho", ..., and "ouse". Such a language independent indexing scheme can be useful when faced with a language for which no morphological analyzers (be it traditional analyzers or stemming procedures) are readily available (e.g., the Korean language [9]) or when dealing with texts containing many spelling errors (e.g., OCR-ed documents [10]).

After defining our indexing units, we have to answer different questions to define a complete indexing strategy. When describing a document, do we need to consider all its details or only the main aspects? In defining this exhaustivity level (completeness of the representation), one can also take into account the document importance with respect to some objectives (e.g., an in-house document may have a deeper coverage). In addition, one must fix the specificity level or the degree of accuracy applied to the selected indexing units: choose general indexing terms (e.g., "drink"), or impose the use of more specific terms ("soft drink"), or even very specific terms ("iced tea", "herbal iced tea", "Nestea"). In some cases, the indexing policy relies on a controlled vocabulary (e.g., Library of Congress Subject Headings, MeSH in the biomedical literature, or the hierarchical thesaurus associated with DMOZ[3]). In such cases, indexing units cannot be freely chosen, but must be present in the given authoritative list.

Traditionally, librarians have adopted a manual indexing strategy in the hope of creating a better representation of their searchable objects. Such manual indexing [11] usually relies on the use of controlled vocabularies in order to achieve greater consistency and to improve manual indexing quality. The advantage of these authority lists is that they prescribe a uniform and invariable choice of indexing descriptors and thus help normalize orthographic variations (e.g., "Beijing" or "Peking"), lexical variants (e.g., "analyzing", "analysis") or examine equivalent terms that are synonymous in meaning. The level of generality may be represented by hierarchical relationships (e.g., "Ford" is a "car"), and related-term relationships (e.g., "see also"). However, while controlled vocabularies should increase consistency among indexers, various experiments demonstrate that different indexers tend to use different keywords when classifying the same document [12], [13]. This illustrates, through yet another example, the variety of wordings human beings have at their disposal to describe the same content. It is thus important to have a certain degree of consistency between document and query representations in order to increase the chance that searchers will be able to locate the information they require [14].

2.2 Indexing Process

Most existing IR systems nowadays rely on automatic indexing of documents and queries. Developed from the beginning of the sixties [15], such an approach allows one to process the huge amounts of information available online. A simple automatic indexing algorithm is composed of four steps:

[3] www.dmoz.org

1. Structure analysis and tokenization
2. Stopword removal
3. Morphological normalization
4. Weighting

Structure analysis and tokenization

In this first step, documents are parsed so as to recognize their structure (title, abstract, section, paragraphs). For each relevant logical structure, the system then segments sentences into word tokens (hence the term *tokenization*). This procedure seems relatively easy but (a) the use of abbreviations may prompt the system to detect a sentence boundary where there is none, and (b) decisions must be made regarding numbers, special characters, hyphenation, and capitalization. In the expressions "don't", "I'd", "John's" do we have one, two or three tokens? In tokenizing the expression "Afro-American", do we include the hyphen, or do we consider this expression as one or two tokens? For numbers, no definite rule can be found. We can simply ignore them or include them as indexing units. An alternative is to index such entities by their type, i.e. to use the tags "date", "currency", etc. in lieu of a particular date or amount of money. Finally, uppercase letters are lowercased. Thus, the title "Export of cars from France" is viewed as the word sequence "export", "of", "cars", "from" and "france".

Stopword removal

In a second step, very frequent word forms (such as determiners ("the"), prepositions ("from"), conjunctions ("and"), pronouns ("you") and some verbal forms ("is"), etc.) appearing in a stopword list are usually removed. This removal is usually motivated by two considerations. Firstly because it allows one to base the matching between queries and documents on content bearing words only. Retrieving a document just because it contains the query words "be", "in" and "the" does not constitute an intelligent search strategy. Stopwords, also called *empty words* as they usually do not bear much meaning, represent noise in the retrieval process and actually damage retrieval performance, since they do not discriminate between relevant and non-relevant documents. Secondly because removing stopwords allows one to reduce the storage size of the indexed collection, hopefully within the range of 30% to 50%.

Although the objectives seem clear, there is no clear and complete methodology to develop a stopword list [16]. For example, the SMART system [15] has 571 words in its stopword list, while the DIALOG information services propose using only nine terms (namely "an", "and", "by", "for", "from", "of", "the", "to", and "with"). Furthermore, some expressions, as "The Who", "and-or gates", or "vitamin A", based on words usually found in stopword list, are very useful in specifying more precisely what the user wants.

Similarly, after converting all characters into lowercase letters, some ambiguity can be introduced as, for example, with the expressions "US citizen" viewed as "us citizen" or "IT scientist" as "it scientist", as both *us* and *it* are usually considered stopwords. The strategy regarding the treatment of stopwords may thus be refined by identifying that "US" and "IT" are not pronouns in the above examples, e.g. through a part-of-speech tagging step. Commercial search engines tend to use, if any, a very short stopword list.

Morphological normalization

As a third step, an indexing procedure uses some type of morphological normalization in an attempt to conflate word variants into the same stem or root. Stemming procedures, which aim to identify the *stem* of a word and use it in lieu of the word itself, are by far the most common morphological normalization procedures used in IR. Grouping words having the same root under the same stem (or indexing term) may increase the success rate when matching documents to a query. Such an automatic procedure may therefore be a valuable tool in enhancing retrieval effectiveness, assuming that words with the same stem refer to the same idea or concept, and must be therefore indexed under the same form. We will come back to stemming procedures and morphological analyzers in Section 5.1.

Weighting

As described previously, an IR system automatically segments a given sentence into words, removing the most frequent ones and stripping the suffixes to produce a set of the indexing units. For example, from the sentence "In 1969, the IBM-360 computer was one of the first third generation computers", we can obtain the following indexing units: "IBM-360", "first", "third", "generat", "comput". This result corresponds to a binary indexing scheme within which each document is represented by a set of (stemmed) keywords without any weight assigned. Of course we may consider additional indexing rules as, for example, to consider only the main aspects of each document (low degree of exhaustivity). To achieve this, we can consider as indexing units only terms appearing more often than a given threshold.

Binary logical restrictions may often be too restrictive for a document and query indexing. It is not always clear whether or not a document should be indexed by a given term. Often, a more appropriate answer is neither "yes" nor "no", but rather something in between. Term weighting creates a distinction among terms and increases indexing flexibility. Thus we need to assign higher weight to more "important" features and lower weight to marginal ones. To weight appropriately each indexing unit, we can consider three components, namely the term frequency, the document frequency and the document length [17].

First, one can assume that an indexing unit appearing more often in a document must have a higher importance in describing its semantic content. We can measure this influence by counting its term frequency (i.e. its number of occurrences within a document), a value denoted tf. Thus, if a term occurs three times in a document, its tf will be 3. Of course, one can consider other simple variants, especially when considering that the occurrence of a given term in a document is a rare event. Thus, it may be good practice to give more importance to the first occurrence than to the others. To do so, the tf component is sometimes computed as $log(tf + 1)$ or as $0.5 + 0.5 \cdot [tf/max(tf)]$. In this latter case, the normalization procedure is obtained by dividing tf by the maximum tf value for any term in that document.

As a second weighting component, one may consider that those terms occurring very frequently in the collection do not help us discriminate between relevant and non-relevant documents. For example, the query "computer database" is likely to yield a very large number of articles from a collection about computer science. We meet here the notion of a term's frequency in the collection (i.e. the number of documents in which a term appears), a value denoted *df*, and called "document frequency".

More precisely, we will use the logarithm of the inverse document frequency (denoted by $idf = log(n/df)$, with n indicating the number of documents in the collection), resulting in more weight for rare words and less weight for more frequent ones [18]. With this component, if a term occurs in every document ($df = n$), its weight will be $log(n/n) = 0$, and thus we must ignore it. On the other hand, when a term appears in only one document ($df = 1$), its weight will reach the maximum for the collection, namely $log(n/1) = log(n)$.

To integrate both components (tf and idf), we can multiply the weight corresponding to the importance of the indexing term within the document (tf) by its importance considering the whole collection (idf). We thus obtain the well-known $tf \cdot idf$ formula.

Lastly, one usually considers that the presence of a term in a shorter document provides stronger evidence than it does in a longer document. This phenomenon can be accounted for by taking into account the document length in the weighting of a term, which is usually done by comparing the length of a document to the average document length (denoted *avdl*). Different weighting schemes include the document length within their weighting formula, leading to more complex schemes as described in the next sections.

3 IR Models

To define an IR model, we must explain precisely how information items (documents) and queries are represented and how these representations are compared to produce a set or ranked list of retrieved items. In this section, we will start our presentation with the Boolean model, the oldest paradigm and one that is still used for some specialized applications. Section 3.2 will describe the vector-space paradigm while different probabilistic models will be introduced in Section 3.3, both models corresponding to modern IR approaches, and usually achieving better retrieval effectiveness than Boolean models.

To further improve the retrieval performance as shown in Section 3.4, we may consider an automatic query expansion strategy that takes different term-term relationships into account in order to expand the original query. Finally, in Section 3.5 we will briefly introduce more advanced IR models as well as some search strategies, such as PageRank, based on document relationships.

3.1 Classical Boolean Model

The Boolean model was the first IR model developed and has a long tradition in library science [13]. Documents are represented by a set of keywords, usually obtained by manual indexing [11] based on a controlled vocabulary. Some additional indexing terms can be provided by authors, usually by selecting some terms from an authoritative list and adding free ones. Table 1 shows a very small example with five indexing terms and four documents. To write a query, the user must transform his/her information need into a logical expression using the indexing terms and the Boolean operators AND, OR, and AND NOT. In order to be retrieved, a document must strictly respect the logical constraint imposed by the query. Based on our example, the query "document AND retrieval" will return the documents D_2 and D_3, while the query "search OR retrieval" will extract the documents D_2, D_3 and D_4.

Table 1. Binary Indexing

	Indexing Terms				
Document	linguistic	document	search	compute	retrieval
D_1	1	0	0	0	0
D_2	1	1	0	1	1
D_3	0	1	1	0	1
D_4	1	0	1	0	1

Transforming an information need into a Boolean query is not always a simple task. If you are interested in cats and dogs, you may formulate the query "cat AND dog". But this logical expression imposes the presence of both terms in a document in order to be retrieved. A better formulation is "cat OR dog" allowing documents with only one of these two terms to be retrieved. The conjunctions "and" and "or" in natural languages are thus not equivalent to the corresponding logical operators. This semantic difference must be taken into account when formulating Boolean queries. Within this paradigm, however, users can formulate structured queries to express their information needs with great precision. The query "document AND retrieval" is clearly more specific than writing a broad query such as "document OR retrieval".

In order to return an answer very fast, the indexing information is not internally stored as in a matrix as depicted in Table 1. In fact, as the collection grows, the number of indexing terms also tends to increase rapidly. Therefore, it is usual to have more than tens of millions of terms for the collection containing several million documents. To verify if the documents respect the logical constraint imposed by the query, we only need to have a fast access to the document identifiers indexed under the searched keywords. To achieve this, the system stores in an "inverted file" the document numbers in which indexing terms occurs. For example, Table 2 shows the inverted file corresponding to Table 1. For the sake of clarity, we have denoted in Table 2 the document number with the prefix D which, of course, does not appear in reality. Various techniques have been suggested [19] to reduce storage requirements, to speed up processing, or to allow the use of phrases in queries (e.g., "New York City") as well as the use of proximity operators (e.g. through the use of the adjacency operator as in "New York" ADJ "city").

Table 2. Inverted File

Indexing term	Sequence of document identifiers
linguistic	D_1, D_2, D_4
document	D_2, D_3
search	D_3, D_4
compute	D_2
retrieval	D_2, D_3, D_4

There are however major drawbacks in the Boolean model. First of all, the retrieved documents do not form a ranked list. To rank retrieved documents, a Boolean-based system has to rely on external attributes, such as the publication date, the title, etc. In principle, the ranking of the retrieved documents should reflect their degree of relevance to the submitted query. For example, considering our previous query "search OR retrieval", it seems reasonable to present first the document having both indexing terms (D_3, D_4) before items indexed only under one of them (e.g., D_2). Secondly, binary logical restrictions are often too limiting for document and query indexing. Within this model, it is not possible to specify whether a given term is essential in a user's query or just marginal. Thirdly, a Boolean search system is unable to return documents that partially match the query. As an answer to the query "document AND search", the system cannot extract document D_2, even if it is indexed under the term "document" and "retrieval", a synonym of "search". Fourthly, making the right choice of the search keywords has a real impact on the quality of the returned list. If certain terms appear in the query, the system may return a large number of documents (output overload) from which it is difficult to detect relevant ones.

In order to solve some of these problems, various attempts have been proposed to include the possibility to assign weights during the indexing of documents and/or queries (hybrid Boolean models). Moreover, this information can be used to rank retrieved documents in a sequence most likely to fulfill user intent [20]. However, all these approaches suffer from some logical deficiencies, and their overall performance (see Section 4) is lower than that of more recent IR approaches.

3.2 Vector-Space Models

Within this IR model [15], [17], documents and queries are indexed according to the strategies described in Section 2. The resulting representation is a set of weighted indexing terms. Thus, the user does not need to express his/her information needs using logical operators; a simple expression in natural language, such as “free speech on the Internet”, or “Italian royal family” is enough. This model clearly provides more user-friendly access to the information.

Within the vector-space model, documents and queries are represented by vectors in a high dimensional space in which each indexing term corresponds to one dimension. Elements of the vectors may be binary, indicating the presence or absence of the term, or fractional weights indicating the relative importance of the term in the document or query. The set of indexing terms forms an orthogonal basis (linearly independent basis vectors). We assume therefore that the indexing terms are independent of one another. For example, if the term “computer” appears in a document, this information implies nothing about the presence (or absence) of other terms such as “algorithm” or “horse”. This represents, of course, a simplified assumption.

Based on a geometric intuition, the vector-space model does not have a solid and precise theory that is able to clearly justify some of its aspects. For example, to compute the degree of similarity between the document representations and the query, we can choose different formulas. If we denote by w_{ij} the weight of the indexing term t_j in a document D_i, and by w_{qj} the weight of the same term in the query Q, the similarly between this document and the query could be computed according to the inner product as follows:

$$Sim(D_i, Q) = \sum_{j=1}^{p} w_{ij} \cdot w_{qj} \tag{1}$$

in which p indicates the number of indexing terms included in query Q. Of course, the vector representing D_i is composed of t values with t representing the number of distinct indexing terms. However, when a term is not present in the query, its contribution to the inner product is null, and has no impact on the similarly level. We can thus restrict the computation to the p query terms.

As an alternative similarity measure, we may compute the cosine of the angle between the vectors representing D_i and Q as follows:

$$Sim(D_i, Q) = \frac{\sum_{j=1}^{p} w_{ij} \cdot w_{qj}}{\sqrt{\sum_{k=1}^{t} w_{ik}^2} \cdot \sqrt{\sum_{k=1}^{p} w_{qk}^2}} \tag{2}$$

In order to avoid computing all elements expressed in the previous formula at retrieval time, we may store the weights associated with each element of D_i in the inverted file. If we apply the well-known weighting scheme $tf \cdot idf$ (see previous section), we can compute and store the weight w_{ij} of each indexing term t_j for the document D_i during the indexing as follows:

$$w_{ij} = \frac{tf_{ij} \ \cdot \ idf_j}{\sqrt{\sum_{k=1}^{t} (tf_{ik} \ \cdot \ idf_k)^2}} \tag{3}$$

Advanced weighting formulas have been proposed within the vector-space model leading to different formulas [21], some being more effective than others.

Moreover, various attempts have been suggested to account for term dependencies [22]. Most of these attempts can be seen as transformations aiming at expanding document representations through a linear transformation $\mathbf{T}$: the vector D_i becomes $\mathbf{T} \cdot D_i$. Often, the matrix $\mathbf{T}$ represents a term-term similarity matrix, which can be defined by "compiling" some a priori given thesaurus, or by automatically building a semantic similarity matrix. In particular, the Generalized Vector-Space Model (GVSM) [22] corresponds to setting $\mathbf{T}$ to the term-document matrix (i.e. the transpose of the document-term matrix, an example is given in Table 1). In this case, the transformation projects the document from a term space to a dual document space. This approach can be generalized by using groups of similar documents instead of isolated documents. The Similarity Thesaurus [24] is a variant of this approach which relies on a particular weighting scheme. Another interesting attempt to take into account term dependencies is the approach known as Latent Semantic Analysis (LSA).

Latent Semantic Analysis [23] allows the automatic derivation of semantic information (in this case a certain form of synonymy and polysemy) from a document collection through co-occurrence analysis. In particular, two terms which are synonyms of each others are likely to have the same profile (i.e. similar rows) in the term-document matrix. Such correlations are unveiled in LSA by a decomposition of the term-document matrix into singular values. More precisely, let $\mathbf{C}$ represent the term-document matrix. Then, the singular value decomposition (SVD) of $\mathbf{C}$ aims at identifying two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ and a diagonal matrix $\mathbf{\Sigma}$ such that:

$$\mathbf{C} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^t \tag{4}$$

where ^{t} denotes the transpose. Assuming the eigenvalues in $\mathbf{\Sigma}$ are organized in decreasing values, one can reduce the dimensionality by retaining only the first k columns of $\mathbf{U}$ ($\mathbf{U_k}$), the first k rows of $\mathbf{V}$ ($\mathbf{V_k}$) and the first k diagonal elements of $\mathbf{\Sigma}$ ($\mathbf{\Sigma_k}$). The matrix $\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^t$ can be shown to be the closest approximation (wrt the Frobenius norm) of $\mathbf{C}$ of rank k. Documents and queries can then be projected on the obtained latent space (via $\mathbf{U_k}^t D$) and directly compared, for example with the cosine formula, in this space. LSA has been showed to provide improvements over the standard vector-space model on several collections, however not on all. The required computing resources represent the main drawback of the LSA approach.

3.3 Probabilistic Models

Within the probabilistic family of models, the retrieval is viewed as a classification process. For each query, the system must form two classes: relevant and non-relevant. Thus, for a given document D_i, one has to estimate the probability that it belongs to the relevant class (class denoted R) or to the non-relevant one (denoted $\bar{R}$). With two classes, the decision rule is rather simple: retrieve D_i if $Prob[R|D_i] > Prob[\bar{R}|D_i]$. The main theoretical foundation of this model is given by the following principle [25]:

"The probability ranking principle (PRP): if a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data."

Of course, this principle does not indicate precisely what data must be used and how to estimate the underlying probabilities. To estimate them, we need to make some assumptions. First, we assume that the relevance of a document is independent of the other documents present in the collection. Second, we assume that the number of relevant documents does not affect the relevance judgment. Both assumptions represent simplification as (a) a particular document may be a good complement to another document, relevant to the query, without being relevant alone, and (b) relevance judgements are affected by the documents already judged.

In addition, we assume for the moment that the document D_i is represented by a set of binary indexing terms. It is important to note that we do not need to compute a precise value for the underlying probabilities $Prob[R|D_i]$. What is required is to produce a ranked list of documents reflecting these values. Using Bayes rule, we can estimate the probability that D_i belongs to the relevant class and the non-relevant one as:

$$Prob[R|D_i] = \frac{Prob[D_i|R] \cdot Prob[R]}{Prob[D_i]} \tag{5}$$

$$Prob[\bar{R}|D_i] = \frac{Prob[D_i|\bar{R}] \cdot Prob[\bar{R}]}{Prob[D_i]} \tag{6}$$

where $Prob[R]$ ($Prob[\bar{R}]$) indicates the prior probability of relevance (and of non-relevance) of a random document (with $Prob[R] + Prob[\bar{R}] = 1$). $Prob[D_i]$ is the probability of selecting D_i. From this, we can see that the ranking order depends only on $Prob[D_i|R]$ and $Prob[D_i|\bar{R}]$ (the other factors being constant).

As described in [26], we can assume conditional independence between terms, and thus write $Prob[D_i|R]$ as the product of the probabilities for its components (binary indexing). For a given document, we denote by p_j the probability that the document is indexed by term t_j given that the document belongs to the relevant set. Similarly, q_j is the probability that the document is indexed by term t_j given that the document belongs to the non-relevant set. Thus we need to estimate, for each term t_j, the following two probabilities:

$$p_j = Prob[d_j = 1 \mid R] \tag{7}$$

$$q_j = Prob[d_j = 1 \mid \bar{R}] \tag{8}$$

from which we can then compute the probability of relevance (and non-relevance) of document D_i as:

$$Prob[D_i \mid R] = \prod_{j=1}^{t} p_j^{d_j} \cdot (1 - p_j)^{1-d_j} \tag{9}$$

$$Prob[D_i \mid \bar{R}] = \prod_{j=1}^{t} q_j^{d_j} \cdot (1 - q_j)^{1-d_j} \tag{10}$$

where d_j is either 1 or 0, depending on the fact that the term t_j appears or not in the representation of document D_i.

One way to estimate the underlying probabilities p_j and q_j is to model the distribution of terms according to a probability distribution such as the 2-Poisson model [27]. In this case, we model the term distribution in the relevance class by a Poisson distribution, while the distribution over the non-relevant class follows another Poisson distribution. These estimates can be refined based on the a set of known relevant and non-relevant items [28] for the current query (relevance feedback). For example, we can estimate the required probabilities as follows:

$$p_j = \frac{r_j}{r} \quad \text{and } q_j = \frac{df_j - r_j}{n - r} \tag{11}$$

where r indicates the number of relevant documents, r_j the number of relevant documents indexed with term t_j, n the number of documents in the collection, and df_j the number of documents in which the term t_j occurs.

Modern probabilistic IR models take into account new variables such as term frequency, document frequency and document length to provide useful insights regarding the probability that a given document is relevant to a query or not (e.g., the Okapi or BM25 model [29]). Among more recent proposals, a very interesting one is DFR (*Divergence from Randomness*), proposed by Amati & van Rijsbergen [30], which represents a general probabilistic framework within which the indexing weights w_{ij} attached to term t_j in document D_i combine two information measures as follows:

$$w_{ij} \;=\; Inf_{ij}^1 \;\cdot\; Inf_{ij}^2 = -\log_2 \left[Prob_{ij}^1(tf)\right] \;\cdot\; (1 - Prob_{ij}^2(tf)) \tag{12}$$

The first component measures the informative content (denoted by Inf_{ij}^1) based on the observation that in the document D_i we found tf occurrences of the term t_j. The second one measures the risk (denoted by $1 - Prob_{ij}^2(tf)$) of accepting the term t_j as a good descriptor, knowing that in document D_i there are tf occurrences of term t_j.

In the first information factor, $Prob_{ij}^1(tf)$ is the probability of observing tf occurrences of the term t_j in document D_i by pure chance. If this probability is high, term t_j may correspond to a non content-bearing word in the context of the entire collection [27]. For the English language, these words generally correspond to determiners, such as "the", prepositions like "with" or verb forms like "is" or "have", considered as being of little or not use in describing a document's semantic content. Various nouns can also appear in numerous documents within a particular corpus (for example "computer" and "algorithm" for a computer science collection). On the other hand, if $Prob^1_{ij}(tf)$ is small (or if $-log_2[Prob^1_{ij}(tf)]$ is high), term t_j would provide important information regarding the content of the document D_i. Several stochastic distributions can be chosen for $Prob^1$ (see [30, 31]), for example the geometric distribution:

$$Prob^1_{ij}(tf) = \left[\frac{1}{(1+\lambda_j)}\right] \cdot \left[\frac{\lambda_j}{(1+\lambda_j)}\right]^{tf} \quad \text{with} \quad \lambda_j = tc_j/n \tag{13}$$

where tc_j indicates the number of occurrences of term t_j in the collection. n is the number of documents in the collection.

The term $Prob^2_{ij}(tf)$ represents the probability of having $tf+1$ occurrences of the term t_j, knowing that tf occurrences of this term have already been found in document D_i. This probability can be evaluated using the Laplace's law of succession as $Prob^2(tf) = (tf+1)/(tf+2) \approx tf/(tf+1)$, which leads, by taking into account the document length to:

$$Prob^2_{ij}(tf) = \frac{tfn_{ij}}{(tfn_{ij}+1)} \quad \text{with} \quad tfn_{ij} = tf_{ij} \cdot log_2\left[\frac{1+(c \cdot avdl)}{l_i}\right] \tag{14}$$

where *avdl* is the mean length of a document, l_i the length of document D_i, and c a constant whose value depends on the collection.

The work presented in [31] relates Laplace's law of succession to a well-known phenomenon in text modeling, namely the one of *burstiness*, which refers to the behavior of words which tend to appear in bursts: once they appear in a document, they are much more likely to appear again. This work also shows that a single distribution can be used as the basis for $Prob^1$ and $Prob^2$.

Another interesting approach is the one known as *non-parametric probabilistic modeling* [4], which is based on a statistical language model (LM) [32]. As such, probability estimates are based on the number of occurrences in document D_i and the collection C. Within this language model paradigm, various implementations and smoothing methods [33] can be considered. We will limit ourselves here to a simple model proposed by Hiemstra [32], and described in Equation 15 (Jelinek-Mercer smoothing), combining an estimate based on the document ($P[t_j|D_i]$) and one based on the collection ($P[t_j|C]$):

$$Prob[D_i|Q] = Prob[D_i] \cdot \prod_{t_j \in Q} [\lambda_j \cdot Prob[t_j|D_i] + (1-\lambda_j) \cdot Prob[t_j|C]] \tag{15}$$

$$Prob[t_j|D_i] = tf_{ij}/l_i \quad \text{and} \quad Prob[t_j|C] = df_j/lc \quad \text{with} \quad lc = \sum_k df_k \tag{16}$$

[4] The term non-parametric is misleading here, as one can view this model as being based on a multinomial distribution. Non-parametric refers here to the fact that the number of parameters grows with the size of the collection, but this is also the case with the previous models we presented.

where λ_j is a smoothing factor (constant for all indexing terms t_j, and usually fixed at 0.35) and lc an estimate of the size of the collection C. Both probability estimates $P[t_j|D_i]$ and $P[t_j|C]$ are based on a ratio, between tf_{ij} and the document size on the one hand, and the number of documents indexed with the term t_j and the size of the whole collection on the other hand.

Lastly, we would like to mention the risk minimization framework developed in [34], which constitutes an attempt to unify several IR models into a single framework.

3.4 Query Expansion and Relevance Feedback

To provide better matching between user information needs and documents, various query expansion techniques have been suggested. The general principle is to expand the query using words or phrases having meanings similar or related to those appearing in the original query, either by using information from a thesaurus (see for example [35, 36] and Section 5.4), or by deriving this information from the collection. To achieve this, query expansion approaches rely on (a) relationships between words, (b) term selection mechanisms, and (c) term weighting schemes. The specific answers to these three questions may vary, leading to a variety of query expansion approaches [37].

In the first attempt to find related search terms, we might ask the user to select additional terms to be included in an expanded query. This can be handled interactively through displaying a ranked list of retrieved items returned by the first query.

As a second strategy, Rocchio [38] proposed taking the relevance or non-relevance of top-ranked documents into account, as indicated manually by the user. In this case, a new query would then be built automatically in the form of a linear combination of the terms included in the previous query and the terms automatically extracted from both relevant (with a positive weight) and non-relevant documents (with a negative weight). More precisely, each new query term was derived by applying the following formula:

$$w'_{qi} = \alpha \cdot w_{qi} + \beta \cdot \sum_{j=1}^{r} w_{ij} - \gamma \cdot \sum_{j=1}^{nr} w_{ij} \tag{17}$$

in which w'_{qi} denotes the weight attached to the ith query term, based on the weight of this term in the previous query (denoted by w_{qi}), and on w_{ij} the indexing term weight attached to this term in both the relevant and non-relevant documents appearing in the top k ranks. The value r (respectively nr) indicates the number of relevant (respectively non-relevant) documents appearing in the first k positions. The positive constants α, β, and γ are fixed empirically, usually with $\alpha \geq \beta$, and $\beta = \gamma$. Empirical studies have demonstrated that such an approach is usually quite effective.

As a third technique, Buckley *et al.* [21] suggested that even without looking at them, one can assume that the top k ranked documents are relevant. Using this approach, we simply set $r = k$ and $\gamma = 0$ in Equation 17. This method, denoted pseudo-relevance feedback or blind-query expansion, is usually effective (at least when handling relatively large text collections).

Relevance feedback can be very effective when the results of the original query are somehow correct. In other cases, the retrieval performance may decrease. Peat & Willett [39] provide one explanation for such poor performance. In their study they show that query terms have a greater occurrence frequency than do other terms. Query expansion approaches based on term co-occurrence data will include additional terms that also have a greater occurrence frequency in the documents. In such cases, these additional search terms will not prove effective in discriminating between relevant and non-relevant documents. In such circumstances, the final effect on retrieval performance could be negative.

There are several works focusing on (pseudo-)relevance feedback and the best way to derive related terms from search keywords. For example, we might use large text corpora to derive various term-term relationships and apply statistical or information-based measures. For example, Qiu & Frei [24] suggested that terms extracted from a similarity thesaurus that had been automatically built through calculating co-occurrence frequencies in the search collection could be added to a new query. The underlying effect was to add idiosyncratic terms to those found in underlying document collections, and related to query terms in accordance to the language being used. Kwok *et al.* [40] suggested building an improved request by using the Web to find terms related to search keywords. Additional information about relevance feedback approaches can be found in [1, Chapter 9] and in [37].

3.5 Advanced Models

As we already mentioned, document and query representations are imprecise and uncertain. It has been shown that different document representations or search strategies tend to have similar overall retrieval performances, although based on different retrieved items [41]. This observation prompted investigations of possible enhancements to overall retrieval performance by combining different document representations [7] (e.g., single terms, n-grams, phrases) or different search strategies (different vector-space and/or probabilistic implementations) [42].

Such merging strategies, known as data fusion, tend to improve the overall performance for three reasons. First, there is a skimming process in which only the m top-ranked retrieved items from each ranked list are considered. In this case, one can combine the best answers obtained from various document representations (which retrieve various relevant items). Second, one can count on the chorus effect, by which different retrieval schemes retrieve the same item, and as such provide stronger evidence that the corresponding document is indeed relevant. Third, an opposite or dark horse effect may also play a role. A given retrieval model may provide unusually high (low) and accurate estimates regarding the relevance of a document. In this case, a combined system can return more relevant items by better accounting for those documents having a relatively high (low) score or when a relatively short (or long) result lists occurs. Such data fusion approaches however require more storage space and processing time. Weighing advantages and disadvantages of data fusion, it is unclear if it is worth deploying in a commercial system.

A second interesting research area is to propose a better estimate for prior probabilities. In the probabilistic models, we have denoted by $Prob[D_i]$ the prior probability of document D_i which appears in different implementations (see Equation 5, or in the LM paradigm, see Equation 15). Usually, without any additional information we assume that this value is the same for all documents. On the other hand, we know that all documents are not equally important. The "80-20 rule" may apply in large document collections or IR databases, meaning that around 20% of the documents present in the collection provide the expected answer to 80% of information needs. In such situations, it may be useful to rely on user feedback to dynamically adapt the prior probability for each document.

In the Web, it is known that users have a preference for the home page of a site, a page from which they can directly buy or obtain the needed service or information (e.g., flight ticket, consult a timetable, obtain an address, reserve an hotel). If we look at the corresponding URL describing such pages (e.g., "www.apple.com", "www.easyjet.co.uk"), one immediately sees that it is composed only of the root element (sometimes with a standard file name such as "index.html" or "default.htm"). As described in [43], we can consider this information to assign higher probability to entry pages. Such practice (used by commercial search engines) can significantly improve retrieval performance.

In addition, Web search engines usually take into account the anchor texts, i.e. the sequence of words used to indicate the presence of a hyperlink and describing, in a compact manner, the target Web page. Thus, to refer to Microsoft home page, we can collect all anchor texts written by various authors and pointing to "www.microsoft.com". All these expressions can be viewed as forming a set, manually built, of terminological variants indexing the target page (e.g., "see Microsoft", "MicroSoft", "Micro$oft", "Bill Gates' Empire" or "The $$ Empire").

Lastly, to improve existing IR models on some collections, one can consider various relationships between documents. Bibliographic references are such an example and can viewed as a set of relationships between documents, under the assumption that the main purpose of citing earlier articles is to give credit to works (concepts, facts, models, results, methodology, etc.) the subjects of which are related to the present document (at least in the author's mind). A main advantage of bibliographic references [44] is that they are independent of a particular use of words, and even languages. Thus they do not suffer (a) from the underlying ambiguity of all natural languages, (b) from the fact that the majority of subject indexers are specialized in a given domain, whereas documents may contain information pertaining to more than one specific domain of knowledge, and (c) from the fact that terms used to describe the content of a document may become obsolete in e.g. the scientific and technological literature.

One of the bibliographic measures is *bibliographic coupling* [45], which measures subject similarity on the basis of referenced documents. To define the bibliographic coupling measure between two articles, one simply counts the number of documents cited by the two papers. Another measure is the *co-citation measure*, used by Small [46] to build a network of documents: for each pair of documents, one counts the number of papers which cite both. To be strongly co-cited, two documents must be cited together by a large number of papers. In this case, the underlying hypothesis is that co-citation measures the subject similarity established by an author group.

On the Web, each page typically includes hypertext links to other pages, and such links are clearly not created by chance. Based on the previous bibliographic measures, one can establish and measure an association strength between Web pages. In a similar perspective, Google uses PageRank (PR) [47] as one (among others) source of information to rank retrieved Web pages. In this link-based search model, the importance assigned to each Web page is partly based on its citation pattern. More precisely, a Web page will have a higher score if many Web pages point to it. This value increases if there are documents with high scores pointing to it. The PR value of a given Web page D_i (value noted as $PR(D_i)$), having $D_1, D_2, ..., D_m$ pages pointing to it, is computed according to the following formula:

$$PR^{c+1}(D_i) = (1-d) \cdot \frac{1}{n} + d \cdot \left[\frac{PR^c(D_1)}{C(D_1)} + ... + \frac{PR^c(D_m)}{C(D_m)} \right] \tag{18}$$

where d is a parameter (usually set to 0.85 [47]) and $C(D_j)$ is the number of outgoing links for Web page D_j. The computation of the PR value can be done using an iterative procedure (few iterations are needed before convergence). After each iteration, each PR value is divided by the sum of all PageRank values. As initial values, one can set $PR(D_i)$ to $1/n$, where n indicates the number of documents in the collection.

A third link-based approach consists of HITS [48]. In this scheme, a Web page pointing to many other information sources must be viewed as a “good” hub, while a document with many Web pages pointing to it is a “good” authority. Likewise, a Web page that points to many “good” authorities is an even better hub, while a Web page pointed to by many “good” hubs is an even better authority. For Web page D_i, formulas for hub and authority scores ($H^{c+1}(D_i)$ and $A^{c+1}(D_i)$) are given, at iteration $c+1$, by:

$$A^{c+1}(D_i) = \sum_{D_j \in parent(D_i)} H^c(D_j) \tag{19}$$

$$H^{c+1}(D_i) = \sum_{D_j \in child(D_i)} A^c(D_j) \tag{20}$$

Such scores are computed for the k top-ranked documents (typical values of k are $k = 200$) retrieved by a classical search model, together which their children and parents (as given by the citation network). Hub and authority scores are updated for a few iterations, and a normalization procedure (e.g., dividing each score by the sum of all squared values) is applied after each step.

Based on these two main hyperlink-based algorithms, different variants have been suggested [49]. It is important to note, however, that using such algorithms alone to retrieve documents does not yield very interesting results, as shown in various TREC evaluation campaigns [50]. This being said, their integration in a general search engine provides new, interesting functionalities (as the detection of spam Web pages, or the improvement of the rank of well-known Web pages).

More recently, several researchers have investigated the possibility of using machine learning approaches in IR. Let us assume that we have a large collection of documents, a large number of queries and relevance judgements (see Section 4) for these queries on the collection. We can see such judgements as a set of annotated (query, document) pairs, where the annotation takes e.g. the value 1 if the document is relevant to the query, and -1 if it is not. Next, one can transform a given (query, document) pair into a vector which represents an example. Such a transformation needs to abstract away from the particular terms present in the query, and usually relies on a variety of features based on characteristics [51], such as

1. number of common terms between the query and the document,
2. average inverse document frequency of the query terms present in the document,
3. width of the window in which query terms appear in the document,
4. cosine similarity between the query and the document,
5. probabilistic measure of relevance of the document,
6. and, when available:
   - PageRank value of the document,
   - number of incoming (outgoing) links, ...

Once (query, document) pairs are seen as vectors, we end up with a standard binary classification problem, the aim of which being to build a classifier that discriminates well between positive and negative examples. Once such a classifier is built, documents relevant to a new query are those documents classified in the positive class (each document of the collection is, of course, first paired to the query, the pair being then transformed, as above, into a vector which will be the input of the classifier). Many different classifiers can be used here. Because of their success in different settings, Support Vector Machines (see for example [1]) are quite often chosen for the task.

Nallapati [51] shows that the approach based on machine learning outperforms standard IR approaches when both the number of features used in the vector representation and the number of relevance judgements are sufficiently large. It is also possible to go beyond the framework of binary classification, and try to directly learn a ranking function, through pairwise preferences [52] or listwise preferences [53]. These last two approaches outperform the one based on a direct classification of documents.

As one may have noted, the basic building block of machine learning approaches to IR is a set of relevance judgements. If some collections (as the one found in TREC for example) do have a set of associated relevant judgements, most collections do not. The situation is somewhat different for the Web and Web search engines, as Web search companies have access to clickthrough data (data recording the fact that, for a given query, a user has clicked on particular documents proposed by the search engine) which can partly be used as annotated data to develop machine learning tools (see for example [54] and more recently [55]). Furthermore, it is probable that such companies try to manually develop relevance judgements in order to deploy accurate machine learning techniques. The field of research on machine learning for IR is thus very active at the time of writing this chapter, and will certainly give birth to new, interesting approaches to IR.

4 Evaluation and Failure Analysis

In order to know whether one search model is better than another, an evaluation methodology must be adopted. In the IR domain, this must be applied to the search process as a whole (user-in-the-loop paradigm) which means evaluation by real users with their real information needs. It would be of interest to analyze a range of characteristics such as the answer speed, its quality, the user's effort needed to write a query, the interface of the search system, the coverage of the collection, etc. All these aspects are certainly important but (a) user studies are costly, and (b) some features are hard to measure objectively. Thus traditional IR evaluation approaches are usually limited to system performance, and more particularly to the quality of the answer (retrieval effectiveness).

In this vein, to measure the retrieval performance [56], we first need a test collection containing a set of information units (e.g., documents), and a set of topic (or query) formulations together with their relevance assessments. As described in the following section, these corpora are usually the result of an international cooperative effort. Having such a benchmark, we can evaluate several IR models or search strategies by comparing their relative retrieval performance (measured by precision-recall values).

As shown in Section 4.3, even after decades of research in this field, we still need a better understanding of the reasons explaining why some topics are still hard. The analysis of some difficult queries will give us some insights on this, as well as on the ways existing IR models can be improved or designed more effectively.

4.1 Evaluation Campaigns

Modern IR evaluations are based on rather large test collections built during different evaluation campaigns. The oldest and best known of these campaigns is TREC[5] [57] (Text REtrieval Conference) established in 1992 in order to evaluate large-scale IR, to speed the transfer of technology from research labs into products and to increase the availability of appropriate evaluation methodologies. Held each year, TREC conferences have investigated retrieval techniques in different medium (written documents, Web pages, spoken documents, OCR,

[5] trec.nist.gov

image, video) as well as different search tasks (*ad hoc*, interactive, routing, filtering, categorization, question/answering) or languages (English, Arabic, Chinese, Spanish, etc.).

Around the world, three other evaluation campaign series have been launched. Beginning in 1999, the NTCIR[6] conference is held every 18 months in Japan and is more oriented to problems related to Far-East languages (e.g., Japanese, traditional or simplified Chinese, Korean) used in conjunction with several search tasks (patent retrieval, Web, *ad hoc*, question/answering, summarization).

In Europe, CLEF[7] [58] (Cross-Language Evaluation Forum) was founded to promote, study and evaluate information access technologies using various European languages. Held each year since 2000, the CLEF campaigns have produced test collections in more than twelve languages related to different tasks (*ad hoc*, bilingual, multilingual and cross-lingual retrieval, image search, question/answering, domain-specific retrieval, etc.).

Lastly, and more recently, a campaign, called INEX[8], was launched in order to evaluate retrieval systems in (semi-)structured collections.

Each year, these evaluation campaigns propose additional challenging tracks such as novelty (retrieval of new and unseen information items), robust (improving the performance of hard topics), spam filtering, IR on blog corpus, question/answering using Wikipedia, cross-language retrieval on audio data, multilingual Web IR, geographic-based IR (search involving spatial aspects), multimodal summarization for trend information, etc.

The user is not absent in all evaluation studies. For example the interactive track at TREC [59] presented an interesting set of studies on various aspects of human-machine interactions. More specific experiments pertaining to cross-lingual information retrieval systems were presented during various CLEF evaluation campaigns [60].

One of the main objectives of these evaluation campaigns is to produce reliable test collections. To achieve this, corpora are extracted from various newspapers, news agencies, on the Web or from other private sources (e.g., libraries, private companies). These data are preprocessed to guarantee some standardization (same encoding, homogenization of the tags, segmentation into information items).

Besides the creation and clean-up of the corpora themselves, the organizers prepare a set of topic formulations (usually 50 per year). Each of them is usually structured into three logical sections comprising a brief title (T), a one-sentence description (D) and a narrative (N) specifying the relevance assessment criteria. For example, we can find the following topic description:
<title> Oil Prices </title>
<desc> What is the current price of oil? </desc>
<narr> Only documents giving the current oil price are relevant, i.e. the price

[6] research.nii.ac.jp/ntcir
[7] www.clef-campaign.org
[8] inex.is.informatik.uni-duisburg.de

when the document was written. References to oil prices in the past or predictions for price changes in the future are not relevant. </narr>

The available topics cover various subjects (e.g., “Pesticides in Baby Food”, “Whale Reserve”, “Renewable Power” or “French Nuclear Tests”) and may include both regional (“Swiss Initiative for the Alps”) and international coverage (“Ship Collisions”). Depending on the campaigns, topics are manually translated into several languages.

After receiving the topic set, the participants have around one month to send back their answers as computed by their own search system. For each task, guidelines specify exactly the task conditions (usually by imposing that runs must be fully automatic and based only on the title and description (TD) parts of the topic formulation). After this step, the organizers may form a pool of retrieved documents for each topic. As each participant is usually allowed to send 1,000 answers for each topic, the organizers take only the top n documents (e.g., $n = 100$) from each run. These documents are then presented to a human assessor who decides whether each item is relevant or not. This process is blind in the sense that the assessor only has access to a query and a set of documents.

This procedure has been applied over many years and we must recognize that the resulting judgments are a subset of true relevant set because not all documents belonging to the underlying collection were judged. However, as demonstrated by various studies, the difference is not large [56]. It is worth to note that having a large number of very different IR systems is necessary (however not sufficient) to ensure the reliability of the evaluation process. If a test collection is built with a few participants having similar search strategies, the results are questionable. Finally, the retrieval performance that can be drawn from any test collection is never absolute but only relative to the test collection in use.

4.2 Evaluation Measures

To measure retrieval performance [56], we may consider that the only important aspect is to retrieve one pertinent answer. In some contexts, the answer could really be unique, or at least the number of correct answers is rather limited as, for example, when searching for a home page on the Web. In this case, the evaluation measure will be based only on the rank of the first correct answer retrieved.

For any given query, if r is the rank of the first relevant document retrieved, the query performance is computed as $1/r$. This value, called the reciprocal rank (RR), varies between 1 (the first retrieved item is relevant) and 0 (no correct response returned). It should be noted here that ranking the first relevant item in second place instead of first would seriously reduce the RR value, making it 0.5 instead of 1.

To measure the retrieval performance resulting from a set of queries, we simply compute the mean over all the queries. This value known as the mean reciprocal rank (MRR), serves as a measure of any given search engine’s ability to extract one correct answer and list it among the top-ranked items. We thus believe that MRR value closely reflects the expectation of those internet surfers who are looking for a single good response to their queries.

In IR, we usually do not want to measure a search system's ability to rank one relevant item, but to extract all relevant information from the collection. In such contexts, we assume that users want both high precision (fraction of retrieved items that are relevant) and high recall (fraction of relevant items that have been retrieved). In other words they want "the truth, the whole truth (recall), and nothing but the truth (precision)". In order to get a synthetic measure from both precision and recall, the harmonic mean between the two (known as F1 or F score [26]) is sometimes used (specially in NLP tasks). Denoting the precision by P and the recall by R, the F score is defined as:

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{21}$$

It is however more common in IR to compute the average precision (AP) for each query by measuring the precision achieved at each relevant item extracted and then computing an overall average. Then for a given set of queries we calculate the mean average precision (MAP), which varies between 0.0 (no relevant items found) and 1.0 (all relevant items always appear at the top of the ranked list).

However, between these two values it is difficult for a user to have a meaningful and direct interpretation of a MAP value. Moreover, from a user's point of view, the value of the difference in AP achieved by two rankings is sometimes difficult to interpret. For example, in Table 3 we have reported the AP for a topic with three relevant items. With System A, the relevant documents appear in rank 2, 3 and 30. If we computed the AP for this query, we have a precision of 1/2 after the second document, 2/3 after the third document, and 3/30 after the 30th retrieved item given an AP of 0.422. Computing the AP for System B, we found 0.676, showing a relative improvement of 60% over ranking produced by System A.

Table 3. Precision-Recall Computation

Rank	System A	System B
1	NR	**R** 1/1
2	**R** 1/2	**R** 2/2
3	**R** 2/3	NR
...	NR	NR
30	**R** 3/30	NR
...	NR	NR
100	NR	**R** 3/100
AP	0.422	0.676

As an alternative, we may consider that the evaluation should focus on the capability of the search system to retrieve many relevant items on the one hand,

and to present them in the top-n position of the returned list. In this case, we do not attach a great importance to extracting *all* relevant items, assuming that there are too many. To evaluate the retrieval performance in such circumstances, we can compute the precision achieved after retrieving n items. On the Web, we may set this threshold to $n = 10$, corresponding to the first screen returned by a commercial search engine and then compute the precision at this point, a value denoted P@10 or Prec@10. In our previous example (see Table 3), both systems achieved a performance of P@10 = 0.2.

More recently, Järvelin and Kekäläinen introduced [61] a new evaluation measure called Normalized Discounted Cumulative Gain (NDCG), which is well adapted to situations where relevance judgements are graded, i.e. when they take more than just two values (relevant or not). The assumption behind NDCG is that highly relevant documents are (a) more useful when appearing earlier in the list of retrieved documents and (b) more useful than marginally relevant documents, which in turn are more useful than irrelevant documents. NDCG computes a relevance score for a list of documents through the gain brought by each document in the list discounted by the position at which the document appears. This score is then normalized relative to the optimal gain that can be achieved, yielding a score between 0 and 1 (provided the lowest value for the relevance is greater than 1). The NDCG score for the list consisting of the first k documents retrieved by an IR system is thus:

$$N(k) = \overbrace{Z_k}^{\text{normalization}} \underbrace{\sum_{j=1}^{k}}_{\text{cumulative}} \overbrace{(2^{p(j)} - 1)}^{\text{gain}} \; / \; \underbrace{\log_2(j+1)}_{\text{position discount}} \tag{22}$$

where $p(j)$ corresponds to the relevance value of the document appearing at position j. With this formula, System B in Table 3 gets a higher score than System A on the first 10 documents as, even though they both retrieve only two documents, the position discounts for System B are lower than the ones for System A since the relevant documents are ranked higher in the list by System B.

Finally, in an effort to statistically determine whether or not a given search strategy would be better than another, we may apply different statistical tests [62] (Sign test, Wilcoxon signed ranks test, t-test or using the bootstrap methodology [63]). Within these tests the null hypothesis H_0 states that the two retrieval schemes produce similar MAP (or MRR) performance. This null hypothesis is accepted if the two retrieval schemes are statistically similar (i.e. yield more or less the same retrieval performance), and is rejected otherwise.

For example, the Sign test does not take the amount of difference into account, but only the fact that a search system performs better than the other. In a set of 50 topics, imagine that System A produced better MAP for 32 queries (or 32 "+"), System B was better for 16 (or 16 "–"), and for the two remaining queries both systems showed the same performance. If the null hypothesis were true, we would expect to obtain roughly the same number of "+" or "–" signs.

In the current case involving 48 experiments (the two ties are ignored), we have 32 "+" and only 16 "–" signs. Accepting the null hypothesis, the probability of observing a "+" would be equal to the probability of observing a "–" (namely 0.5). Thus, for 48 trials, the probability of observing 16 or fewer occurrences of the same sign ("+" or "–", for a two-tailed test) is only 0.0293 (see tables in [62]). With a significance level fixed at $\alpha = 5\%$, we must reject H_0, and accept the alternative hypothesis stating that there is a difference between System A and B. In such a case, the difference is said to be *statistically significant at the level* $\alpha = 0.05$.

4.3 Failure Analysis

Given the sound procedures described previously for both indexing (see Section 2) and search models (see Section 3), it seems *a priori* that search engines should not fail to retrieve documents queried by users, especially when those documents share many words with the query. However, unexpected and incorrect answers happen from time to time.

From a commercial IR perspective, it is important to understand the reasons why a search system fails, why customers encounter problems when searching for information. In the academic world, this aspect has been studied within several robust tracks [64]. During previous evaluation campaigns, numerous topic descriptions have been created by humans and submitted to different search engines. Some of them (around 5% to 10%) have been found to be hard for almost every search paradigm, without being able to detect, *a priori*, when a query would be hard or not. Only a few studies (e.g. [65]) tend to investigate search system failures.

To illustrate our purpose, we have extracted some topics from our participation in past evaluation campaigns. These hard queries are defined as topics having zero precision after 10 retrieved items (P@10). In an effort to explain the IR model's failure to list at least one pertinent item among the top ten, we might classify the causes into two main groups: first, system flaws (Category #1 to #3) where some advanced processing techniques may improve the performance; Second, topic intrinsic difficulties (Category #4 to #6) where a deeper understanding of user's intent and semantic analysis seems to be required.

Category 1: *Stopword list.* The way letters are normalized (or not) as well as the use of a stopword list may prevent one from finding correct answers. From the topic "Who and whom", the query representation was simply empty because the forms "who", "and" and "whom" were included in the stopword list. A similar problem might be encountered with phrases such as "IT engineer", "US citizen" or "language C". In the first two cases, the search system might fail to recognize the acronyms, treating them as the pronouns "it" or "us", which are usually included in a stopword list, along with the letter "c". This example explains why commercial IR systems may have a particular strategy for dealing with empty words.

Category 2: *Stemming.* The stemming procedure cannot always conflate all word variants into the same form or stem, as illustrated by the topic "Prehistorical Art". In this case, a light stemming approach left unchanged the search term "prehistorical". This term however does not occur in the corpus and the submitted query was therefore limited to "art". Using a more aggressive stemmer is not always the most appropriate solution. Of course, using Porter's stemmer [69], the IR system is able to conflate the forms "prehistorical" and "prehistoric" under the same root and to retrieve a relevant item in the top ten. However, a more aggressive stemmer may lead to overstemming with a negative impact on the search results.

Category 3: *Spelling errors.* When building topic descriptions, the organizers of the evaluation campaign usually check the spelling of each topic so that only a few, if any, of them appear in queries. They do however exist as "tartin" (instead of "tartan") or "nativityscenes" (for "nativity scenes"). The presence of proper nouns may also generate this problem (e.g., "Solzhenitsyn"). A related problem is the fact that spelling may vary across countries (e.g. "color" vs. "colour") or that several variants are acceptable ("fetus" and "foetus"). In all these cases, search systems are often unable to retrieve any pertinent items in the top of their ranked list or may ignore numerous pertinent items written using an alternate spelling. In a commercial environment, spell checking and suggestion is an essential feature for all man-machine interfaces.

Category 4: *Synonymy and language use.* The topic "Les risques du téléphone portable" ("Risks with mobile phones") illustrates how vocabulary can change across countries. For this query, the relevant documents used synonyms that are country dependent. In Switzerland, a mobile phone is usually called "natel", in Belgium "téléphone mobile", "cellulaire" in Quebec, and "portable" in France (the same problem occurs in the Chinese language with two different expressions used, one in Taiwan and another in mainland China). All IR systems included in their top ten results certain documents covering the use of mobile phones in the mountains (and the risk of being in the mountains). Other retrieved articles simply presented certain aspects related to mobile phones (new joint ventures, new products, etc.). Other words or English expressions present similar difficulties (e.g, "film" and "movie" in the query "Films set in Scotland", or "car" and "automobile" in the query "European car industry"). In such cases, the query may include one form, and relevant documents another.

Category 5: *Missing specificity.* A fifth failure explanation is found for example in topic "Trade unions in Europe". The specific or desired meaning is not clearly specified or is too broad. This same difficulty occurs with the topics "Edouard Balladur", "Peace-keeping forces in Bosnia", "World soccer championship" or "Computer security". With all these queries, the IR system listed top ranked articles having not one but at least two or three terms in common with the query. Placed at the top of the output list were short articles having all query terms in their title (or 3 out of 4 terms for topic "Peace-keeping"). The unspecified main purpose of the topic was clearly missing; for example, for the topic "World soccer championship" the required information was most probably the "result of the final".

Category 6: *Discrimination ability.* For example, in the topic "Chinese currency devaluation" the pertinent set must contain information about the effects of devaluation. In this case, the three relevant articles had only one or two terms in common with the query. The terms "Chinese" (also appearing in 1,090 other articles) and "currency" (occurring in 2,475 documents) appeared in the first relevant document. In the second, only the term "Chinese" appears to be in common with the topic's title, and in the last only the term "devaluation" (occurring also in other 552 articles). The IR system therefore found it very difficult to discriminate between relevant and non-relevant documents, due to the fact that a lot of the latter had at least two terms in common with the query. The same difficulty arose with the topic "Wonders of ancient world" for which relevant documents describe one wonder without using explicitly the term "wonder", "ancient" or "world".

5 Natural Language Processing and Information Retrieval

The basic building blocks of most natural languages are words. However, the term "words" is ambiguous and we must be more precise in order to distinguish between the surface form (e.g., "horses") corresponding to tokens, and word type or lemma (entry in the dictionary, such as "horse" in our case). Moreover, the specific meaning of the term "horse" is not always an animal with four legs as in the term "horse-fly" or in expressions as "Trojan horse", "light horse", "to work like a horse", "from the horse's mouth", or "horse about".

In the design of effective IR systems, the morphological [66] component plays an important role. For most search systems, and also for most human languages, the words form the basic units to build the phrases, expressions and sentences used to transmit a precise meaning. An appropriate processing of these entities is therefore important for enhancing retrieval performance by promoting pertinent word-sense matching. In addition, the way words combine and the meanings words convey are crucial for understanding the content of a document and for representing this content accurately.

In this section, we will review some of the major relations between IR and NLP. We will do so by examining the traditional layers of Natural Language Text Processing (morphology, syntax, semantics) and see the role they play or can play in IR. Lastly, we will briefly mention various applications which have direct connections with both IR and NLP.

5.1 Morphology

As mentioned previously, the goal of the morphological step in IR is to conflate morphological variants into the same form. In some cases, only an inflectional analysis is performed, so as to get to a lemmatized version of the original text. This step can be followed by a derivational analysis, usually relying on suffixes only, as prefixes tend to radically modify the meaning of a word (indexing the two forms “decompose” and “recompose” as the same token does not make sense from an IR point of view).

In most cases however, these two steps (inflectional and derivational analysis) are not separated, but performed in conjunction. Even though inflectional analyzers, based on electronic lexicons, exist in many languages, this is not the case for derivational analyzers, and the IR community has relied on tools which aim at identifying word stems without necessarily relying on precise morphological processes. Such tools are called *stemmers* (as mentioned in Section 2.2) and are described below. One can find in [67] a comparison of the use of a stemmer and a derivational lexicon for conflating words in the framework of IR.

When defining a stemming algorithm, a first approach will only remove inflectional suffixes. For English, such a procedure conflates singular and plural word forms (“car” and “cars”) as well as removing the past participle ending “-ed” and the gerund or present participle ending “-ing” (“eating” and “eat”).

Stemming schemes that remove only morphological inflections are termed as “light” suffix-stripping algorithms, while more sophisticated approaches have also been proposed to remove derivational suffixes (e.g., “-ment”, “-ably”, “-ship” in the English language). For example, Lovins’s stemmer [68] is based on a list of over 260 suffixes, while Porter’s algorithm [69] looks for about 60 suffixes. In such cases suffix removal is also controlled through the adjunct of quantitative restrictions (e.g., “-ing” would be removed if the resulting stem had more than three letters as in “running”, but not in “king”) or qualitative restrictions (e.g., “-ize” would be removed if the resulting stem did not end with “e” as in “seize”). Moreover, certain *ad hoc* spelling correction rules are used to improve the conflation accuracy (e.g., “running” gives “run” and not “runn”), due to certain irregular grammar rules usually applied to facilitate easier pronunciation. Of course, one should not stem proper nouns such as “Collins” or “Hawking”, at least when the system can recognize them.

Stemming schemes are usually designed to work with general text in any given language. Certain stemming procedures may also be especially designed for a specific domain (e.g., in medicine) or a given document collection, such as that of Xu & Croft [70] who suggest developing stemming procedures using a corpus-based approach which more closely reflects the language used (including the word frequencies and other co-occurrence statistics), instead of using a set of morphological rules in which the frequency of each rule (and therefore its underlying importance) is not precisely known.

As we mentioned above, stemming procedures ignore word meanings, and thus tend to make errors. Such errors may be due to over-stemming (e.g., “general” becomes “gener”, and “organization” is reduced to “organ”) or under-stemming (e.g., with Porter’s stemmer, the words “create” and “creation” do not conflate to the same root). Not surprisingly, the use of an on-line dictionary has been suggested in order to produce better conflations [71], [72].

The development of a morphological analyzer (be it a stemmer or a more refined tool) depends largely on the language considered. The English inflectional morphology is relatively simple. The plural form is usually denoted by adding an '-s' (with some exceptions like "foot" and "feet"). The feminine form is built using some suffixes (e.g., "actor" and "actress") and we do not have to mark the agreement between noun and adjective ("tall man", "tall women"). To build new words (derivational construction), we may add prefixes ("pay", "prepay") and / or suffixes ("bank", "banking", "banker", "bankless", "bankrupt").

For other European languages, the morphological construction can be relatively similar. In the French language, the plural is denoted as in English ("face", "faces"), while the feminine form can simply be denoted by a final '-e' ("employé", "employée") or by a suffix ("acteur", "actrice"). As in English, various suffixes are available to form new words.

For some other languages, the inflectional morphological possibilities are more numerous. In the German language for example, we find four grammatical case endings (e.g., the genitive case by employing an '-s' or '-es' as in "Staates" (of the state), "Mannes" (of the man)). The plural form is denoted using a variety of endings such as '-en' (e.g., "Motor" and "Motoren" (engine)), '-er', '-e' (e.g., "Jahr" and "Jahre" (year)) or '-n' (e.g., "Name" and "Namen" (name)). Plural forms may also use diacritic characters (e.g., "Apfel" (apple) becomes "Äpfel") or in conjunction with a suffix (e.g., "Haus" and "Häuser" (house)). Also frequently used are the suffixes '-en' or '-n' to indicate grammatical cases or for adjectives (e.g., "... einen guten Mann" (a good man) in the accusative singular form)

The Hungarian language makes use of a greater number of grammatical cases (23 in total, although some are limited to a set of nouns or appear only in fixed and predefined forms) than does German. Each case has its own unambiguous suffix however; e.g. the noun "house" ("ház" in nominative) may appear as "házat" (accusative case), "házakat" (accusative plural case), "házamat" ("... my house") or "házamait" ("... my houses"). In this language the general construction used for nouns is as follows: 'stem' 'possessive marker' 'case' as in 'ház' + 'am' + 'at' (in which the letter 'a' is introduced to facilitate better pronunciation because "házmt" could be difficult to pronounce). Similar agglutinative aspects may be found in other languages such as Turkish, where the noun "ev" (house) may take on the form "evler" (the houses), "evlerim" (my houses) and "evlerimde" (in my houses). For these two languages at least, the automatic removing of suffixes does not present a real and complex task [73], [74].

For the Finnish language however, it seems that the design and development of an effective stemming procedure requires a more complex morphological analysis, usually based on a dictionary. The real stemming problem with the Finnish language is that stems are often modified when suffixes are added. For example, "matto" (carpet in the nominative singular form) becomes "maton" (in the genitive singular form, with '-n' as suffix) or "mattoja" (in the partitive plural form, with '-a' as suffix). Once we remove the corresponding suffixes, we are left with three distinct stems, namely "matto", "mato" and "matoj". Of course irregularities such as these also occur in other languages, usually helping to make the spoken language flow better, such as "submit" and "submission" in English.

In Finnish however, these irregularities are more common, and thus they render the conflation of various word forms into the same stem more problematic.

Compound constructions (concatenation of two or more lexemes to form another word, e.g., handgun, worldwide) also appear in other European languages. In Italian, the plural form may alter letters within a word, for example "capoufficio" (chief secretary) becomes "capiufficio" in its plural form. Yet, in other constructions, the stem "capo" is left unchanged (e.g. "capogiro" gives "capogiri" (dizziness) in its plural form).

In German and in most Germanic languages, compound words are widely used and are a source of additional difficulties. For example, a life insurance company employee would be "Lebensversicherungsgesellschaftsangestellter" ("Leben" + 's' + "Versicherung" + 's' + "Gesellschaft" + 's' + "Angestellter" for life + insurance + company + employee). The augment (i.e. the letter "s" in our previous example) is not always present (e.g., "Bankangestelltenlohn" built as "Bank" + "Angestellten" + "Lohn" (salary)). Since compound construction is so widely used and can be written in different forms, it is almost impossible to build a German dictionary providing complete coverage of the language, and an automatic decompounding procedure is required in order to obtain an effective IR system in the German language [7], [75].

Several tools are available for identifying morphological variants in different languages. They are either based on on-line dictionaries and standard morphological analysis (see for example http://www.xrce.xerox.com) or on stemming procedures dedicated to different languages (e.g., http://www.unine.ch/info/clef/ or http://snowball.tartarus.org/).

5.2 Orthographic Variation and Spelling Errors

The standardization of spelling was mainly the fruit of the 19th century. Working with documents written previously, one encounters different spellings for a given term or proper name (e.g. Shakespeare's name appears as "Shakper", "Shakspe", "Shaksper", "Shakspere" or "Shakspeare" in his own works).

The spelling problem can be domain-specific. In the biomedical literature, it is known that several orthographic variants [76] can be found to represent a given name, generally introduced for a variety of reasons. Firstly, there are of course typographic errors and misspellings (performance errors as in "retreival" and "retrieval" or competence errors as in "ecstasy", "extasy", or "ecstacy"). Secondly, punctuation and tokenization may produce variants, mainly due to the lack of a naming convention (e.g. "Nurr77", "Nurr-77" or "Nurr 77"). Thirdly, regional variations also introduce variants (e.g. the difference between British and American English for "colour" and "color" or "grey" or "gray"). Fourthly, the transliteration of foreign names produces some differences (e.g., "Crohn" and "Krohn" or "Creutzfeld-Jakob" and "Creutzfeldt-Jacob").

The standard strategy to reduce the negative impact caused by spelling errors or orthographic variation is to relate similar (however not identical) forms in one way or another. Two main strategies can be used here: (a) compute an edit-distance between different forms (e.g. present in a dynamic dictionary) and normalize the variants with a particular form, (b) adopt an n-gram indexing strategy (n is typically set to 5 across a range of studies). The n-gram method has the advantage of being fast (filters have to be used to avoid comparing unrelated forms in the first approach). Moreover, this method does not require any prior linguistic knowledge and is robust to typographical errors, both in the submitted queries and documents retrieved [8]. For instance, the term "alzheimer" would be decomposed, with an overlapping 5-gram approach, as: "alzhe", "lzhei", "zheim", "heime" and "eimer".

5.3 Syntax

Most IR systems index documents on the basis of the simple words they contain. However, there have been many attempts to make use of more complex index terms, comprising several terms, in order to get a more precise description of the content of documents. One of the first attempts, referred to as *adjacent pairs* (e.g. described in [77]), considered complex terms made up of two adjacent content words. Even though simple, this approach is appropriate e.g. for the English language, as it allows one to capture terminological elements consisting of *Adjective Noun* or *Noun Noun* sequences (and their combination). For other languages, such as the ones based on a composition of Romance type, i.e. in which compounds are formed through prepositions, simple regular expressions can be used over part-of-speech tags to identify terminological elements and add them to the index terms.

More generally, a syntactic analyzer can be used to identify long-distance dependencies between words, so as to improve the indexing of documents. The first works in this direction were based on "traditional" grammars, in which a complete analysis/representation of a sentence or a phrase was searched for. Fagan [78] investigates for example, the impact of a syntactic parser for noun phrases on IR performance. However, the lack of large-coverage grammars, and the difficulty of obtaining unambiguous and correct parses for many sentences effectively put a stop to this type of research at that time.

Advances in shallow (or light) parsing in the mid-nineties led to a resurgence of this type of work, as such parsers were partly defined to be general and robust. Hull [79] proposes, for example, to use a shallow parser to identify relations within and between complex noun and verb phrases. Pairs of terms thus extracted (as *Subject Verb* pairs or *Adjective Noun* pairs) are then added to simple words to provide a richer index of documents. Experiments conducted on the vector-space model showed a slight improvement on the TREC-5 collection. Similar experiments conducted on French can be found in [80]. More recently, Gao *et al.* [81] proposed an extension of the language modeling approach to IR that can take into account the syntactic dependencies provided by a probabilistic parser deployed on the queried collection.

One of the major problems with the use of syntactic information is that the improvements in IR performance have been limited, and highly dependent on the queried collection. Complex terms acquired through syntactic analysis can be seen as additional information that can help refine query results and thus lead to an increase in the precision at the top of the list of retrieved documents. However, in many cases, having constituents of a compound taken independently of each other or as a whole does not change the retrieval results substantially. For example, a document indexed with the two terms *information* and *retrieval* is very likely to deal with *information retrieval*, so that the addition of this last compound does not really add any new information: all documents containing both *information* and *retrieval* will almost certainly also be indexed by *information retrieval* after a syntactic analysis phase, and the ranking will not change.

In essence, syntactic analysis provides additional indexing dimensions, but does not address the problem of the vocabulary mismatch between queries and documents. A direct way to address this problem is to resort to a semantic analysis in order to replace index terms with concepts and perform the matching between queries and documents at a more abstract level.

5.4 Semantics

In the absence of robust systems providing a complete semantic analysis of sentences, most work in IR and semantics have focused on lexical semantics, and the possibility to replace standard index terms with concepts or word senses. While some studies have shown that word sense disambiguation procedures can be beneficial in some cases [82, 83], the majority have tried to rely on existing semantic resources in order to index both documents and queries at the concept level. Such works have used existing semantic resources, either generic ones, as Wordnet [35], or specialized ones for specific collections (e.g. UMLS - see below).

The usefulness of concept indexing in specific domains has been shown in several studies, mainly in the medical domain for which large-coverage thesauri and semantic networks exist. For example, the best performing systems on text in the ImageCLEFmed task of CLEF [84, 36] use conceptual indexing methods based on vector-space models or language models. In the TREC genomics track, Zhou *et al.* [85] used the MeSH (Medical Subject Headings) thesaurus and *Entrez* databases to select terms from medical publications. Terms in documents and queries were expanded with their variants found in these resources to achieve better indexing. They found that this strategy led to a significant improvement over standard indexing methods.

Other researchers have tried to go beyond the use of concepts by exploiting relations between them. Vintar *et al.* [86] evaluates the usefulness of UMLS concepts and semantic relations in medical IR. They first extract concepts and relations from documents and queries. To select relations in a sentence, they rely on two assumptions: (1) interesting relations occur between interesting concepts; (2) relations are expressed by typical lexical markers such as verbs. However, their experiments with a vector-space model show that using both concepts and relations lower the performance obtained with concepts alone. A similar line of development has been taken on the basis of the language modeling approach to IR. The language modeling approach to IR was first proposed in [87], and extended in [32] (see Section 3.3). Even though smoothed unigram models have yielded good performance in IR, several studies have investigated the use of more advanced representations. Studies like (e.g., [88] or [89]) proposed the combination of unigram models with bigram models. Others studies, e.g. [90] or [81], have extended the model to deal with syntactic dependencies. More recently, Maisonnasse *et al.* [91, 36] have proposed a generalization of these works that can deal with semantic networks as well. In fact, any directed acyclic graph can be used to represent documents and queries.

This latter approach was applied to the medical domain, where documents are indexed at the concept level with UMLS, and where relations between concepts are added by checking whether two concepts co-occurring in the same sentence are present in the *Semantic Network* associated with UMLS. The results show that concept indexing yields a significant improvement over standard indexing in IR performance. The use of semantic relations further improves the precision of the system, namely at the top of the list of retrieved documents, even though slightly.

What these results show is that, provided one has at her disposal semantic resources adapted to the collection (as is UMLS for the medical domain), then significant gains can be achieved with semantic indexing.

5.5 Related Applications

There are a number of applications which directly borrow models and methods from both IR and NLP. A detailed presentation of these applications is beyond the scope of this chapter, and we will just mention what we believe are the more important here.

1. Text categorization is a good example of an application where research has been conducted in the two communities, IR and NLP (to which we should add the Machine Learning and Data Mining ones). Text categorization aims at automatically assigning new documents to existing categories. Most approaches are currently based on machine learning, where already classified documents are used to automatically learn a decision function. The way documents are represented directly derives from the vector-space model (sometimes with additional processing steps, such as named entity and term extraction) and the different weighting schemes of Section 3.2.
2. A second application where methods from both IR and NLP are used is document summarization, which aims at providing a summary, in a few sentences, of a document or a document collection. Current approaches focus on extracting key sentences (sometimes parts of sentences) from the document or document collection and on displaying them in an appropriate way.
3. A third application is BioNLP, which focuses on the processing of text documents in the biological domain. As for the medical domain, there exist several knowledge bases in the biological domain which can be used to get a more accurate representation of documents. However, the kind of information searched for by biologists is complex, and one needs to deploy a whole range of technologies to be able to match the needs of biologists.
   For example, when trying to find articles relevant to an interaction between two proteins (*BRCA1* and *p53* on PubMed), simply searching for the two

terms results in 733 abstracts of which a high proportion do not describe any relationship between the proteins. More precise queries, which include verbs describing interactions, such as ’interact’ and ’regulate’, are often used to significantly reduce the search space. Unfortunately the information loss is unknown and the retrieved abstracts still document other relationships, for example, between *E6* and *p53*[9]. In this case, a tight coupling between the indexing engine, the search engine, and the natural language processing engine is needed. Interested readers are referred to the BioNLP chapter of the current book for a detailed presentation of the models and methods deployed in this domain.

4. The last application we would like to mention is Question-Answering, which aims at providing precise answers (as opposed to whole documents or paragraphs as is traditionally the case in IR) to questions. Most Question-Answering systems rely on a tightly-coupled combination of IR and NLP techniques, leading to systems which integrate many of the existing technologies of these two domains. Here again, we refer interested readers to the Question Answering chapter of the current book for a detailed presentation of such systems.

6 Conclusion

As described in this chapter, the IR field is an extensive applied NLP domain that is able to cope successfully with search and retrieval in the huge volume of information stored on the Web — users are, on average, able to find what they are looking for. This success may however hide other challenging problems. Commercial search engines want to know more about the users and their needs to have a better knowledge of the real question behind the submitted query (what the user is really looking for). For example, a user living in Canberra who submits the query “movie tonight” is not likely to be interested in films displayed in New York City or in Washington D.C. Information about user preferences, e.g. for music, can also be exploited to re-rank retrieved items, e.g. by moving musical comedies to the top of the list. This information can play an important role in commercial advertising. From this perspective, the search system will try to complement its answer page with advertising banners that are appropriate to both the user query and preferences.

Proposing new IR models, suggesting new implementations or adding new functionalities to existing IR models is an important and active domain in IR. There are many collections which are different from the Web. To a certain extent, commercial Web search engines provide only access to the surface Web: the deep Web is largely ignored (private collections, court decisions, patents, etc.). Within a given Web site, or a site inside an enterprise, the search function must provide an effective access to the information required (e.g., in order to allow customers to find what they want or have easy access to past e-mails).

Campaigns involving design, implementation and evaluation of IR systems for different languages, including both European (CLEF), and popular Asian

[9] We thank D. Hawking and T. E. McIntosh who provided us with this example.

(NTCIR) languages have already been running for a few years. Recently, the FIRE evaluation forum has been launched to study IR specialties related to the languages belonging to the Indian sub-continent. Some of these languages seem to be more difficult from an IR perspective, having, for example, a less strict spelling convention. The automatic segmentation of Chinese or Japanese sentences, the automatic decompounding of German words, or the automatic query translation represent examples of NLP problems encountered by modern IR systems.

Finally the relationship between the IR and NLP domains tends to be strong on one hand, and multifaceted on the other. As presented in this chapter, morphology, spelling error correction, syntax, and semantics are important aspects for general or domain-specific IR systems. Moreover, IR can also be one of the steps in a complex processing chain of textual corpora. In this vein, we can mention the design and implementation of Question/Answering systems or the elaboration of opinionated IR within which the retrieved items are not related to a given topic but to personal opinions about the target. In such cases, the retrieval of textual items must be then post-processed by complex NLP system to extract short and precise answers to a question, or to define whether or not the retrieved item contains an opinion on the subject. As indicated in this chapter, the use of machine learning techniques is currently an active research area in developing IR systems with a significant NLP component.

Acknowledgments. The authors would like to thank the two reviewers of this chapter, namely Ellen Voorhees (NIST, Washington, D.C.) and David Hawking (Funnelback, Canberra), for their helpful suggestions and remarks.

References

1. Manning, C. D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (UK), 2008.
2. Wolfram, D., Spink, A., Jansen, B.J., Saracevic, T.: Vox populi: The public searching of the web. Journal of the American Society for Information Science and Technology, 52 (2001) 1073–1074.
3. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (MA), 1999.
4. Gal, A., Lapalme, G., Saint-Dizier, P., Somers, H. : Prolog for Natural Language Processing. Addsion-Wesley, Reading (MA), 1991.
5. Furnas, G., Landauer, T.K., Gomez, L.M., Dumais, S.T. : The vocabulary problem in human-system communication. Communications of the ACM, 30 (1987) 964–971.
6. Hawking, D., Robertson, S.: On collection size and retreival effectiveness. IR Journal, 6 (2003) 99–150.
7. Savoy, J.: Combining multiple strategies for effective monolingual and cross-lingual retrieval. IR Journal, 7 (2004) 121–148.
8. McNamee, P., Mayfield, J.: Character n-gram tokenization for European language text retrieval. IR Journal, 7 (2004) 73–97.
9. Savoy, J.: Comparative study of monolingual and multilingual search models for use with Asian languages. ACM - Transactions on Asian Languages Information Processing, 4 (2005) 163–189.

10. Voorhees, E.M., Garofolo, J.S.: Retrieving noisy text. In E.M. Voorhees, D.K. Harman (Eds): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (MA) (2005) 183–198.
11. Anderson, J.D., Pérez-Carballo, J.: The nature of indexing: how humans and machines analyze messages and texts for retrieval. Information Processing & Management, 37 (2001) 231–254.
12. Zunde, P., Dexter, M.E.: Indexing consistency and quality. American Documentation, 20 (1969) 259–267.
13. Cleverdon, C.W.: Optimizing convenient on-line access to bibliographic databases. Information Service & Use, 4 (1984) 37–47.
14. Cooper, W.S.: Is interindexer consistency a hobgoblin? American Documentation, 20 (1969) 268–278.
15. Salton, G. (Ed): The SMART Retrieval System. Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs (NJ), 1971.
16. Fox, C.: A stop list for general text. ACM - SIGIR Forum, 24 (1989) 19–21.
17. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing & Management, 24 (1988) 513–523.
18. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 (1972) 11–21.
19. Zobel, J., Moffat, A.: Inverted file for text search engines. ACM Computing Surveys, 38 (2006) 1–56.
20. Savoy, J.: Ranking schemes in hybrid Boolean systems: A new approach. Journal of the American Society for Information Science, 48 (1997) 235–253.
21. Buckley, C., Singhal, A., Mitra, M., Salton, G.: New retrieval approaches using SMART. In Proceedings TREC-4, NIST publication #500-236, Gaithersburg (MD) (1996) 25–48.
22. Wong, S.K.M., Ziarko, W., Raghavan, V.V.: On modelling of information retrieval concepts in vector spaces. ACM - Transactions on Database Systems, 12 (1987) 723–730.
23. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41 (1990) 391–407.
24. Qiu, Y., Frei, H.P.: Concept based query expansion. In Proceedings ACM-SIGIR'93, Pittsburgh (PA) (1993) 160–169.
25. Robertson, S.E.: The probability ranking principle in IR. Journal of Documentation, 38 (1977) 294–304.
26. van Rijsbergen, C.J.: Information Retrieval. 2nd ed., Butterworths, London (UK), 1979.
27. Harter, S.P.: A probabilistic approach to automatic keyword indexing. Journal of the American Society for Information Science, 26 (1975) 197–206.
28. Robertson, S.E., Sparck Jones, K.: Relevance weighting of search terms. Journal of the American Society for Information Science, 27 (1976) 129–146.
29. Robertson, S.E., Walker, S., Beaulieu, M.: Experimentation as a way of life: Okapi at TREC. Information Processing & Management, 36 (2002) 95–108.
30. Amati, G., van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM - Transactions on Information Systems, 20 (2002) 357–389.
31. Clinchant, S., Gaussier, E.: The BNB distribution for text modeling. In Proceedings of ECIR'2008, Glasgow (UK) (2008) 150–161.
32. Hiemstra, D.: Using Language Models for Information Retrieval. PhD. Thesis (2000).

33. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM - Transactions on Information Systems, 22 (2004) 179–214.
34. Zhai, C., Lafferty, J.: A risk minimization framework for information retrieval. Information Processing & Management, 42 (2006) 31–55.
35. Voorhees, E.M.: Using WordNet$^{\text{TM}}$to disambiguate word senses for text retrieval. In Proceedings ACM-SIGIR'93, Pittsburgh (PA) (1993) 171–180.
36. Maisonnasse, L., Gaussier, E., Chevallet, J.P.: Multiplying concept sources for graph modeling. In C. Peters, V. Jijkoun, T. Mandl, H. Müller, D.W. Oard, A. Peñas, V. Petras, D. Santos, (Eds.): Advances in Multilingual and Multimodal Information Retrieval. LNCS #5152. Springer-Verlag, Berlin (2008) 585–592.
37. Efthimiadis, E.N.: Query expansion. Annual Review of Information Science and Technology, 31 (1996) 121–187.
38. Rocchio, J.J.Jr.: Relevance feedback in information retrieval. In G. Salton (Ed.): The SMART Retrieval System. Prentice-Hall Inc., Englewood Cliffs (NJ) (1971) 313–323.
39. Peat, H.J., Willett, P.: The limitations of term co-occurrence data for query expansion in document retrieval systems. Journal of the American Society for Information Science, 42 (1991) 378–383.
40. Kwok K.L., Grunfield, L, Sun, H.L., Deng, P.: TREC2004 robust track experiments using PIRCS. In Proceedings TREC 2004, NIST publication #500-261, Gaithersburg (MD) (2005).
41. Turtle, H., Croft, W.B.: Evaluation of an inference network-based retrieval model. ACM - Transactions on Information Systems, 9 (1991) 187–222.
42. Vogt, C.C., Cottrell, G.W.: Fusion via a linear combination of scores. IR Journal, 1 (1999) 151–173.
43. Kraaij, W., Westerveld, T., Hiemstra, D.: The importance of prior probabilities for entry page search. In Proceedings ACM-SIGIR'2002, Tempere (2002) 27–34.
44. Garfield, E.: Citation Indexing: Its Theory and Application in Science, Technology and Humanities. The ISI Press, Philadelphia (PA), 1983.
45. Kessler, M.M.: Bibliographic coupling between scientific papers. American Documentation, 14 (1963) 10–25.
46. Small, H.: Co-Citation in the scientific literature: A new measure of the relationship between two documents. Journal of the American Society for Information Science, 24 (1973) 265–269.
47. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In Proceedings of the WWW'7, Amsterdam (1998) 107–117.
48. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of the ACM, 46 (1999) 604–632.
49. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: Algorithms, theory, and experiments. ACM - Transactions on Internet Technology, 5 (2005) 231–297.
50. Hawking, D.: Overview of the TREC-9 web track. In Proceedings of TREC-9, NIST Publication #500-249, Gaithersburg (MD) (2001) 87–102.
51. Nallapati, R.: Discriminative models for information retrieval. In Proceedings ACM-SIGIR'2004, Sheffield (UK) (2004) 64–71.
52. Cao, Y., Xu, J., Liu, T;-Y., Li, H., Huang, Y., Hon, H.-W.: Adapting ranking SVM to document retrieval. In Proceedings of ACM-SIGIR'2006, Seattle (WA) (2006) 186–193.

53. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H. Learning to rank: from pariwise approach to listwise approach. In Proceedings of ICML'2007, Corvalis (OR) (2007) 129–136.
54. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM - Transactions on Information Systems, 25 (2007) 1–26.
55. Scholler, F., Shokouni, M., Billerbeck, B., Turpin, A.: Using clicks as implicit judgements: Expectations versus observations. In Proceedings of ECIR'2008, Glasgow (UK) (2008) 28–39.
56. Buckley, C., Voorhees, E.M.: Retrieval system evaluation. In E.M. Voorhees, D.K. Harman (Eds): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (MA), (2005) 53–75.
57. Voorhees, E.M., Harman, D.K. (Eds): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (MA), 2005.
58. Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (Eds.): Advances in Multilingual and Multimodal Information Retrieval. LNCS #5152. Springer-Verlag, Berlin, 2008.
59. Dumais, S.T., Belkin, N.J.: The TREC interactive tracks: Putting the user into search. In E.M. Voorhees, D.K. Harman (Eds): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (MA) (2005) 123–152.
60. Gonzalo, J., Oard, D.W.: The CLEF 2002 interactive track. In C. Peters, M. Braschler, J. Gonzalo, M. Kluck, (Eds.): Advances in Cross-Language Information Retrieval. LNCS #2785. Springer-Verlag, Berlin (2003) 372–382.
61. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. Transaction on Information Systems, 20 (2002) 422–446.
62. Conover, W.J.: Practical Nonparametric Statistics. 3rd edn. John Wiley & Sons, New York (NY) 1999.
63. Savoy, J.: Statistical inference in retrieval effectiveness evaluation. Information Processing & Management, 33 (1997) 495–512.
64. Voorhees, E.M.: The TREC 2005 robust track. ACM SIGIR Forum, 40 (2006) 41–48.
65. Buckley, C.: Why current IR engines fail. In Proceedings ACM-SIGIR'2004, Sheffield (UK) (2004) 584–585.
66. Sproat, R.: Morphology and Computation. The MIT Press, Cambridge (MA), 1992.
67. Hull, D.: Stemming algorithms - A case study for detailed evaluation. Journal of the American Society for Information Science, 47 (1996) 70–84.
68. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics, 11 (1968) 22–31.
69. Porter, M.F.: An algorithm for suffix stripping. Program, 14 (1980) 130–137.
70. Xu, J., Croft, B.W.: Corpus-based stemming using cooccurrence of word variants. ACM - Transactions on Information Systems, 16 (1998) 61–81.
71. Krovetz, R., Croft, B.W.: Lexical ambiguity and information retrieval. ACM - Transactions on Information Systems, 10 (1992) 115–141.
72. Savoy, J.: Stemming of French words based on grammatical category. Journal of the American Society for Information Science, 44 (1993) 1–9.
73. Savoy, J.: Searching strategies for the Hungarian language. Information Processing & Management, 44 (2008) 310–324.
74. Can, F., Kocberber, S., Balcik, E., Kaynak, C., Ocalan, H. C.: Information retrieval on Turkish texts. Journal of the American Society for Information Science and Technology, 59 (2008) 407–421.

75. Braschler, M., Ripplinger, B.: How effective is stemming and decompounding for German text retrieval? IR Journal, 7 (2004) 291–316.
76. Yu, H., Agichtein, E.: Extracting synonymous gene and protein terms from biological literature. Bioinformatics, 19 (2003) i340–i349.
77. Salton, G., Yang, C.S., Yu, C.T.: A theory of term importance in automatic text analysis. Journal of the American Society for Information Science, 26 (1975) 33–44.
78. Fagan, J.L.: The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. Journal of the American Society for Information Science, 40 (1989) 115–132.
79. Hull, D.A., Grefenstette, G., Schültze, B.M., Gaussier, E., Schütze, H., Pedersen, J.O.: Xerox TREC-5 site report: Routing, filtering, NLP, and Spanish track. In Proceedings TREC-5, NIST publication #500-238, Gaithersburg (MD) (1997) 167–180.
80. Gaussier, E., Grefenstette, G., Hull, D., Roux, C.: Recherche d'information en français et traitement automatique des langues. Traitement Automatique des Langues (TAL), 41 (2000) 473–493.
81. Gao, J., Nie, J.-Y., Wu, G., Cao, G.: Dependence language model for information retrieval. In Proceedings ACM-SIGIR'2004, Sheffield (UK) (2004) 170–177.
82. Sanderson, M.: Word sense disambiguation and information retrieval. In Proceedings ACM-SIGIR'94, Dublin (1994) 142–151.
83. Schütze, H., Pedersen, J.O.: Information retrieval based on word senses. In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas (NV) (1995) 161–175.
84. Lacoste, C., Chevallet, J.P., Lim, J.-H., Wei, X., Raccoceanu, D., Le, T.H.D., Teodorescu, R., Vuillenemot, N.: Inter-media concept-based medical image indexing and retrieval with UMLS at IPAL. In C. Peters, P. Clough, F.C. Gey, J. Karlgren, B. Magnini, D.W. Oard, M. de Rijke, M. Stempfhuber (Eds): Evaluation of Multilingual and Multi-modal Information Retrieval. LNCS #4730. Springer-Verlag, Berlin (2007) 694–701.
85. Zhou, W., Yu, C., Smalheiser, N., Torvik, V., Hong, J.: Knowledge-intensive conceptual retrieval and passage extraction of biomedical literature. In Proceedings ACM-SIGIR'2007, Amsterdam (2007) 655–662.
86. Vintar, S., Buitelaar, P., Volk, M.: Semantic relations in concept-based cross-language medical information retrieval. In Proceedings of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining (ATEM), Catvat–Dubrovnik (2003).
87. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In Proceedings ACM-SIGIR'98, Melbourne (1998) 275–281.
88. Srikanth, M., Srikanth, R.: Biterm language models for document retrieval. In Proceedings ACM-SIGIR'2002, Tempere (2002) 425–426.
89. Song, F., Croft, W.B.: A general language model for information retrieval. In Proceedings ACM-CIKM'99, Kensas City (MI) (1999) 316–321.
90. Lee, C., Lee, G.G., Jang, M.G.: Dependency structure applied to language modeling for information retrieval. ETRI Journal, 28 (2006) 337–346.
91. Maisonnasse, L., Gaussier, E., Chevallet, J.P.: Revisiting the dependence language model for information retrieval. In Proceedings ACM-SIGIR'2007, Amsterdam (2007) 695–696.