Department of Computer Science
Series of Publications A
Report A-2005-2

Advanced Document Description, a Sequential Approach

Antoine Doucet

Academic Dissertation

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in the Small Hall, University Main Building, on November 19, 2005, at 10 o'clock.

University of Helsinki
Finland

ISSN 1238-8645
ISBN 952-10-2801-7 (paperback)
ISBN 952-10-2802-5 (PDF)
http://ethesis.helsinki.fi/

Computing Reviews (1998) Classification: I.2.7, H.3.1, H.3.3, H.2.8, G.3

Helsinki University Printing House
Helsinki, November 2005 (161 pages)

Advanced Document Description, a Sequential Approach

Antoine Doucet
Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Antoine.Doucet@cs.helsinki.fi
http://www.cs.helsinki.fi/Antoine.Doucet/

Abstract

To be able to perform efficient document processing, information systems need to use simple models of documents that can be treated in a smaller number of operations. This problem of document representation is not trivial. For decades, researchers have tried to combine relevant document representations with efficient processing. Documents are commonly represented by vectors in which each dimension corresponds to a word of the document. This approach is termed "bag of words", as it entirely ignores the relative positions of words. One natural improvement over this representation is the extraction and use of cohesive word sequences.

In this dissertation, we consider the problem of the extraction, selection and exploitation of word sequences, with a particular focus on the applicability of our work to domain-independent document collections written in any language.

After a look at the state of the art of advanced document representations, we present a novel technique to efficiently extract frequent word sequences from document collections of any size.

The second contribution of this dissertation is the definition of a formula and an efficient algorithm to address the problem of computing the probability of occurrence of a discontinued sequence of items. An application of this result is that it permits a direct evaluation of a word sequence through the comparison of its expected and observed frequency.

We finally present a new measure of the phrasal similarity of two documents. We apply this new metric to the task of document retrieval and illustrate the multilingual- and domain-independence of our work by conducting experiments with scientific and general

document collections written in English, Japanese, Korean and Chinese.

Computing Reviews (1998) Categories and Subject Descriptors:

I.2.7	[Artificial Intelligence]: Natural Language Processing - Text analysis
H.3.1	[Information Storage and Retrieval]: Content Analysis and Indexing - Indexing Methods
H.3.3	[Information Storage and Retrieval]: Information Search and Retrieval
H.2.8	[Database Applications]: Data mining
G.3	[Probability and Statistics]: Markov processes

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Lexical Cohesion, Text Data Mining, Information Retrieval, Term Dependence, Phrases, Collocations, N-grams, Multi-Word Units, Sequential Patterns, Maximal Frequent Sequences, Document Retrieval, Automatic Indexing, Information Systems

Acknowledgements

I am most grateful to my supervisor Helena Ahonen-Myka for supporting me whenever I needed it and for leaving me with the freedom to learn from my numerous mistakes the rest of the time. Despite the distance, Bruno Crémilleux has always been present and eager to give advice. His numerous stays in Helsinki, and mine in Caen, have always been the key to major progress for this dissertation.

The conditions of my work could hardly be better than they have been until now in the Department of Computer Science of the University of Helsinki, currently headed by Jukka Paakki, within the From Data to Knowledge (FDK) research unit headed by Esko Ukkonen. My French dialect of English has been improved, all administrative issues have been eased, the IT people have provided me with tens of desktops to pollute whenever needed and I do not regret any of the coffee breaks or Doremi cultural events that I have attended.

The expertise and friendliness of the members of the Document Management, Information Retrieval and Text Mining (Doremi) research group have made an unknown environment easy right away. Our common passion for coffee has probably a lot to do with it as well.

I am grateful for the financial support of the Academy of Finland, the From Data to Knowledge (FDK) research unit and the French-Finnish Association for Scientific and Technical Research.

Proceeding backwards chronologically, I have to thank my friend Nico for inviting me to spend Christmas 2000 in Finland. Without you, who knows if I would ever have gotten to know this country, or even stepped into it? Thank you for that.

I should also get back to my exchange year at the University of Tennessee, where Professors Langston and Thomason gave me the first hint that doing research may be something I would enjoy.

Finally, on the emotional side, I naturally want to thank my family and friends. You know who you are, don't you? Just in case, here comes a list of names in alphabetical order: Arthur D., Cécile D., Chloë D., Fernande W., Gérard D., Guy-Roger L., Hélène D., Henriette D., Jérôme D., Lothaire C., Lucie, C., Ludovic C., Marika H., Monique D., Nathalie B., Nicolas C., Pia K., Renaud P., René W., Romane C., Stéphane S., Susanna K., Tamlin B., and the unavoidable number of important people I forgot to mention here!

Once more, please note that the sequential information in each name list should be ignored and they should rather be processed as a single bag of words (hence challenging the usefulness of the following 161 pages).

Contents

CHAPTER 1

Introduction

Languages, as we speak and write them, are complex. Computers, on the other hand, relying entirely on binary representations, are simple. It is hence natural that, to use computers for document processing, we must develop document models (or representations) to simplify the complexity of human language down to the level of comprehension of a simple computer.

This (voluntarily) exaggerated simplification of the motivation of our work is actually quite accurate. A more advanced explanation is that document processing techniques require a large number of document comparisons. For example, if we enter a query in a search engine, our query is compared to the documents in the search engine's database, and the ranked list of documents that we obtain is a list of the documents that are considered most similar to our query, presented in decreasing order of similarity.

Because we generally wish for *efficient* document processing, this large number of comparisons requires that each of them is computed efficiently. And the best way to compare documents efficiently is to compare simplifications of documents instead. And it is well known that what computers are best at dealing with is numbers. Hence, a common way to simplify a document is to represent it with a vector of values, where each value corresponds to the importance of a word of the document set. The comparison of two documents can then be completed with a set of multiplications and additions.

The major weakness of such a document representation is that it ignores the position of occurrence of the words of a document. This is the reason why this technique is often termed "bag of words".

Evidently, two words are more likely to be related if they occur next to each other than if they are separated by three book chapters of 20 pages each.

Previous studies did get around this weakness by adding new dimensions to the document vector. To supplement the values representing single words, these extra dimensions contain values to represent the importance of multiple words occurring together in a document. The difficulty is then that the number of ways to combine words can be enormous and the representation of each of those associations by a dimension of the vector space can cause efficiency problems. Even if word associations are formed from adjacent pairs only, their number is often too high. At the same time, one may observe that using only adjacent word pairs already means leaving a considerable amount of information out. If the word "and" occurs between two other words, they are certainly related but this is not taken into account by using only adjacent pairs.

The example of the use of adjacent word pairs is very representative of the problem of finding a good phrasal description. We easily end up with too many descriptors that are paradoxically insufficient. This motivates research in the area of *multi-word unit* extraction, where the goal is to extract from text cohesive units of several words, and ignore the majority of joint word occurrences that do not form cohesive units.

A further problem stems from the fact that when a document vector contains both a value for a word and a value for a phrase that contains the same word, its importance in the document is artificially augmented. The question of how to account for this fact is an open problem.

This dissertation is focused on those very problems of extraction, selection, and exploitation of multi-word units. An important particularity of this work is the development of techniques that are entirely language-independent.

1.1 Main Contributions

This thesis presents three main results. They respectively contribute to the extraction, evaluation, and exploitation of the sequential nature of text. Those results are listed below.

1. Maximal frequent sequences (MFSs) are word sequences that are more frequent than a frequency threshold, and they are maximal in the sense that no longer sequence that contains an MFS is frequent. The interest in MFS is due to the fact that they permit a compact document representation. Their extraction is, however, difficult, as it requires counting and comparing numerous word sequences with each other and even $MineMFS$ [AMD05, AM05], the current best-performing technique to extract the set of MFSs from a text collection sometimes fails to produce results in a reasonable amount of time, especially when the collection is large.

   In Chapter 3 we introduce $MFS\_MineSweep$, a partition-rejoin technique that uses $MineMFS$ as a black-box, and permits obtaining an approximation of the set of MFSs of a document collection. This method permits extracting descriptors even from collections with which $MineMFS$ fails. It effectively increases the scope of use of MFSs as document descriptors to document collections of virtually any size. Even for smaller collections, our experiments indicate that $MFS\_MineSweep$ can extract a more exhaustive phrasal description of the document collection and that it does it faster than the standard technique $MineMFS$.

2. The main contribution of our work is the definition of a formula and an efficient algorithm to address the problem of computing the probability of occurrence of a discontinued sequence of items. We formalized the problem to a simple Markov process, and exploited the specificities of the corresponding transition matrix through techniques of linear algebra. This technique goes well beyond the scope of this thesis as it can be applied to any type of sequential data. In text, it is common to estimate the probability of occurrence of word sequences, but the sequences are often defined with fixed relative positions of their word constituents, or sometimes by a maximal distance between the first and last word. To propose probabilities without constraints on the distance between words is new.

   A neat application of this work to textual data is the following. We have extended our technique of computation of the

probability of occurrence of a discontinued sequence towards an efficient algorithm for the calculation of the expected document frequency of such a sequence in a given document collection. The expected document frequency of a word sequence can then be compared to its actual frequency using statistical significance techniques. This provides a general-purpose technique to directly evaluate and rank a set of word phrases. The evaluation of word sequences has always been indirect, heavily relying on the intended application and on the subjective judgment of human assessors. Our technique provides an alternative to evaluate the quality of word phrases from a general point of view, regardless of their intended use.

3. Our third contribution permits us to exploit a phrasal document description in information retrieval. As a result of an exploratory attempt to use MFS-based document descriptors in a document retrieval framework, we developed a novel technique to measure the phrasal similarity of documents. The descriptors can be matched more loosely, and a set of parameters is proposed to loosen or tighten constraints, such as the distance between words, their possible use in inverse order, and so on. A number of retrieval experiments were attempted, using MFS-based descriptors with radically different document collections, news-feed articles written in four languages (English, Japanese, Chinese, and Korean), and computer science journal articles in English.

   This exploratory research could not demonstrate the intrinsic quality of MFSs as descriptors that would be particularly suited for document retrieval applications, but the phrasal similarity measure we developed showed a significant improvement on all three Asian language collections.

1.2 Organization of this Monograph

In the following chapter, we will motivate the need to extract descriptors so as to model documents. We will then describe the *vector space model*, the most common model to represent a document, and expose a few of its limitations. Notably, it does not take word

order into account, and it ignores the inherent logical structure of documents. We will present the state of the art in the extraction of descriptors that take the sequential nature of text into account, and criticize their shortcomings.

Chapter 3 will present techniques of sequence extraction that have been successfully used in data mining. We will discuss the problem of adapting such techniques to textual data, before introducing $MineMFS$ [AMD05, AM05], an algorithm developed by Helena Ahonen-Myka to combine the approaches of data mining and collocation discovery. This method still fails to obtain descriptors in a reasonable amount of time for large collections. We hence introduce our first contribution, $MFS\_MineSweep$, a partitioning approach that permits us to extend the scope of $MineMFS$ to document collections of virtually of any size. In general, *MFS_MineSweep* further enables the extraction of a more exhaustive document description than $MineMFS$ alone. These two observations are verified by a set of experiments followed by a discussion that concludes the chapter. Measures of the quantity, size and density of information of an MFS-based phrasal description are defined and the various descriptions obtained with $MFS\_MineSweep$ are extensively compared to each other and to those provided by $MineMFS$ alone.

Chapter 4 proposes a novel algorithm to fill up a major lack of research in multi-word units extraction: the absence of a direct evaluation technique for non-contiguous word sequences that is domain- and language-independent. We present an algorithm to calculate the probability of occurrence of a given non-contiguous sequence. We then extend this algorithm to permit the calculation of the expected document frequency of such a sequence in a given collection. Standard statistical techniques permit us to compare observed and expected frequency, which gives a general measure of interestingness for non-contiguous sequences. It can be used, for example, to sort a set of word sequences by their value of interestingness, possibly to select the best ranked, as an absolute number, or through a comparison with an interestingness threshold.

After having developed techniques to extract more advanced document descriptors from larger document collections and having proposed an automatic way to evaluate the interestingness of sequential non-contiguous descriptors, we explore into a potential application of this work in Chapter 5. We notably introduce a

new measure to calculate a phrase-based similarity between documents. Finally, we experiment with variations of that measure applied to MFS-based descriptors and document collections in four different languages (English, Korean, Chinese, Japanese) and from two different domains (computer science articles and news-feeds). The first observation is that MFS-based phrasal descriptors did not seem to benefit document retrieval, at least not with the techniques we proposed. A promising result for our newly defined similarity measure is that it benefited significantly the retrieval performance on the Chinese, Japanese and Korean document collections.

The conclusions of this monograph are drawn in Chapter 6.

CHAPTER 2

Basic Document Descriptors

The need to model documents has a long history, originating long before computers even existed. We will below motivate the need for automatic document modeling and present the vector space model, the technique most commonly used to describe and compare text documents. After going through its limitations, we will present more advanced attempts to account for the specifics of text, notably its sequential nature (word order matters in most languages) and the structural organization inherent to documents. We will finally criticize the weaknesses of the current state of the art.

2.1 Why Model Documents?

> "A **model** is a simplified framework to organize how we think about a problem." [1]

The problem of organizing documents originates from library science, i.e., from the need for librarians to organize books and documents in such a way that readers can easily find items of their interest. The system of attaching a category to a document and storing it in the corresponding shelf has a clear limitation: to determine the right category for a given document can be difficult, and some documents can duly belong to several categories, which is impossible to transpose on the library's shelves. One book cannot

[1] From David Begg (et al.): Economics, 7th edition.

stand on more than one shelf at a time, and this physical obstacle led library science to the idea of constructing sets of terms pointing towards documents they describe.

The technique of defining a set of *index terms* to point at and represent documents is called *indexing*. A book can then be assigned to as many categories as it duly belongs to. Consequently, regardless of the organization of library shelves, users could check a list of keywords and find directions to a number of corresponding books, as assigned by human indexers. Although the indexing task may seem straightforward and repetitive, it is in fact very difficult as it truly consists in *anticipating the future uses of the document*. The indexer should indeed notably be aware that users may well look up the same indexing term for different reasons. The fact that different people have different needs and different ways to express the same needs is encompassed by the observation that consistency between indexers and between different indexing sessions of the same indexer is difficult to achieve [Kar00].

The constantly growing amount of large document collections available in electronic format has created the need and given the possibility to automate their organization, or at least to assist in the manual task. The improvement in processing speed gave way to the exploitation of automatic techniques for indexing documents. Initially, automatic approaches of indexing were aiming at emulating human indexation. But soon, they were not only based on the document title or on a fixed set of categories anymore, but on the analysis of the content of the document itself.

This permits representing documents automatically and organizing them in a way that serves the information needs of users. The first and foremost application of such a document organization is to search and retrieve the documents relevant to a user need. Information retrieval research permitted to improve the efficiency of this basic application, but also to open the door to more sophisticated uses. It is, for example, possible to classify documents within a set of predefined categories, or simply to group (to cluster) similar ones together. Documents can even be summarized or translated.

The information era has introduced computers as standard tools into homes and libraries, drastically increasing the number of end-users of automatic searching facilities. The Internet gave access to many online document collections, including the web itself, and

caused users to start searching through library collections, or reserving items from home. Paradoxically, the average computer literacy grew, but searching systems started to be used more and more by regular users, not only by professional librarians any further. This fact has for consequence a need for simplicity and efficiency.

All the applications we just mentioned require a crucial ingredient. To be able to group similar documents together, we must compare them. To retrieve the documents that correspond to a user's information needs, we must compare them to these information needs. Formally, the *comparison of documents* consists in giving a numerical evaluation of their similarity. But this evaluation is not an obvious process. Textual data is complex. There is no straight way to obtain a similarity value of two different instances of natural language, the language we speak and write.

This is why we need to define document *models*, simplifications of the reality. As there is no way for machines to comprehend the subtleties of human language, they should instead deal with simplified representations of textual documents, which they can process efficiently, so as to support the constraints of simplicity and efficacy associated with end-user applications. Document comparisons will then be replaced by comparisons of document representations.

The two most important features that a good document model should be able to combine are the ability to integrate the most important elements of the original document and the capacity to calculate numerous document similarities efficiently. The vector space model follows those principles and, to date, remains the most common choice.

2.2 The Vector Space Model

Widely used for its simplicity, the vector space model was developed in the late 1960's by Salton and his students [SWY75]. This model relies on the idea to represent semantic proximity with spatial proximity. Any piece of text can be transformed into a high-dimensional vector. Subsequently, the similarity between documents can be computed using basic techniques of linear algebra.

2.2.1 Principle

Assuming we have a set of V distinct terms to represent a document d, the vector space model representation of d is the vector $\overrightarrow{d}$ of dimension V:

$$\overrightarrow{d} = (w_{d_1}, w_{d_2}, \ldots, w_{d_V}),$$

where V is the dimensionality of the vector space, each w_{d_i} represents the weight of the term i in the document d.

The model is better understood with a simple example. Assume a two-word world, say those words are "blue" and "cheese". Any text could then be represented by a 2-dimensional vector. Using the number of word occurrences as their coordinates, Figure 2.1 shows the vectors representing the documents "blue cheese blue", "cheese cheese cheese blue", and "blue". For the last document, "blue", the coordinate for "cheese" is nil. Sparse vectors, i.e., vectors with a large majority of nil coordinates, are the norm in the real world, where the number of words is tremendously high. For agglutinative languages, such as Finnish, the number of words is theoretically infinite. Taking the example of English, the Oxford English Dictionary defines 600,000 word forms [Sim89][2], not including all possible inflections, proper nouns, and abbreviations. In the Cambridge Encyclopedia of the English Language [Cry03], Crystal suggests that there must be at least a million words in the English language. According to the same author, including all the scientific nomenclature would make an easy reach for two million. One must note here that there is some controversy on this number, or even on the existence of a number, given the various possible understandings of the terms "word" and "vocabulary".

In practice, the number of distinct terms used for the indexation (i.e., the *index term vocabulary*) of a document collection is much more reasonable. A document collection is usually represented by the set, or a subset, of the words that compose it. That subset is the result of a selection of the most "significant" index terms of the document collection, a process often referred to as *feature (or term) selection*. A common way to do so is to use a *stop list*, a list of function words that are very frequent and very unlikely to tell much about the topic of a document. According to Zipf's law

[2] see also: http://oed.com/about/facts.html

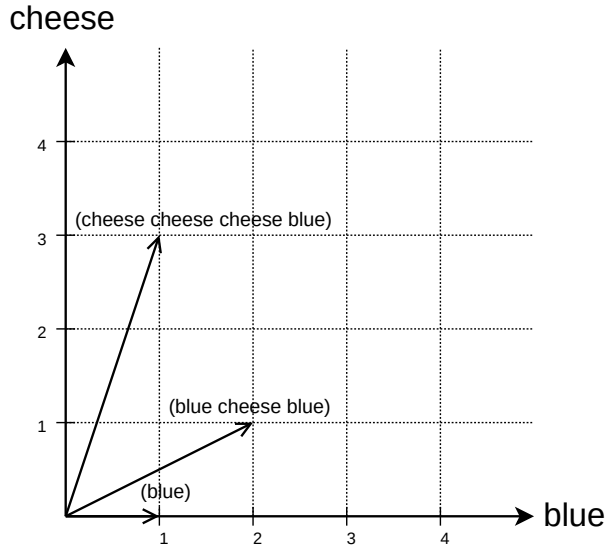

Figure 2.1: A two dimensional vector space. The two dimensions correspond to the words "blue" and "cheese".

(of which principle was actually first suggested by Estoup [Est16]), to discard only a few frequent words permits reducing the size of a document collection drastically. *Morphological conflation* is another common way to reduce the number of word terms. Word forms such as "models", "modeling" and "modeled" certainly relate to a very similar topic. Morphological analysis permits conflating such word forms and raise their combined weight. For English, the well-known Porter algorithm [Por80] uses a list of suffixes as its basis to truncate words (i.e., to *stem* them). Such stemming algorithms are efficient but sometimes faulty, in the sense that word forms with different meanings can be truncated to the same conflation. For instance, the Porter algorithm would stem both the words "generous" and "generally" to "gener". Furthermore, stemming techniques are much harder to conceive for morphology-rich languages.

It is important to underline that even though the use of term selection has been widespread, it is more and more criticized. The use of term selection is in fact strongly dependent on the application. In the field of text classification, a small number of features is often sufficient, whereas for document retrieval, the more terms

are used for document indexation, the more documents can be potentially retrieved. Feature selection was long used to deal with lower-dimensional vector spaces and thus improve efficiency, but this motivation is losing ground with the improvement of computational infrastructures. The gain in computational time that can be obtained by reducing the number of dimensions is getting smaller and hence it is less and less worth the associated loss in performance.

Whether term selection is used or not, we eventually obtain a set of index terms to represent the dimensions of document vectors in the vector space. This issue will be developed later, but it is interesting to note already that words are not the only features that can be included in the set of index terms.

A document is represented by a vector of dimension V, the size of the index term vocabulary. The values of this document vector are weights representing the importance of the corresponding terms. We will now discuss a few of the ways to calculate such weights.

2.2.2 Term Weighting

The effectiveness of IR applications can be significantly affected by the assignment of appropriate term weights [SB88]. Ideal term weights in a document vector should reflect the significance of those terms within the document. The very basic way to weight terms is to simply account for their presence or absence in a document. The binary weight of an index term for a given document is 1 if it occurs therein, 0 if not.

A more advanced indication of the significance of a term within a document is its frequency of occurrence therein. Termed *tf* for *term frequency*, this measure was proposed originally as early as the 1950s by Luhn [Luh57]. This is the measure that we used in Figure 2.1. However, even though it is meaningful, this measure is not totally satisfying. Its weakness is that it does not take the specificity of the terms into account.

A term which is common to many documents is less useful than a term common to only a few documents. This is the motive for introducing a measure of the specificity of a term in a document collection. The *document frequency (df)* of a term in a document collection of size N is the number of documents that term appears

in. First presented by Spärck Jones [SJ72], the *inverted document frequency (idf) factor* is, as the name tells, based on the inversion of the document frequency. To avoid the dramatic effect of low values, the usual formula is based on logarithms rather than straight measures, for each word $w \in d$:

$$idf_w = log\left(\frac{N}{df_w}\right),$$

where N is the total number of documents in the collection, df_w is the document frequency of the word w, and idf_w is its inverted document frequency.

In short, term frequency is a measure of the importance of a term in a document and inverted document frequency is a measure of its specificity within the collection. Best practice has been achieved through the combination of the *tf* and *idf* factors, although the optimal way to combine the factors was shown to vary from collection to collection [SY73]. Generally, *tf* and *idf* factors are simply multiplied:

$$tfidf_w = tf_w \times idf_w.$$

2.2.3 Document Length Normalization

However, *tfidf* weights are not yet quite sufficient. They ignore one important fact: documents vary in size, and this variation favors the similarity of longer documents for the two main reasons pointed out by Singhal et al. [Sin97, SBM96]:

- **Higher term frequencies:** Longer documents deal with a similar topic all along and thus tend to use the same terms repeatedly. Therefore, the term frequency factor may be large for long documents.

- **More terms:** Longer documents also have more different terms. Typically, two random long documents will have more terms in common than two random short documents. It is nonetheless intuitively impossible to admit that, on average, longer documents are more similar to each other than short documents.

This is why we need to account for the size of documents when weighting their terms. Document length normalization is thus used to penalize the term weights of long documents. This normalization is one more thing that is eased by the vector space model representation, since to normalize a vector is a very straightforward process, done by dividing the vector by its norm,

$$\text{Normalized } \overrightarrow{d} = \left(\frac{w_{d_1}}{\| \overrightarrow{d} \|}, \frac{w_{d_2}}{\| \overrightarrow{d} \|}, \ldots, \frac{w_{d_V}}{\| \overrightarrow{d} \|} \right),$$

where a possible way to calculate the norm of $\overrightarrow{d}$ is:

$$\| \overrightarrow{d} \| = \sqrt{w_{d_1}^2 + w_{d_2}^2 + \cdots + w_{d_V}^2}.$$

2.2.4 Similarity Measures

To compare documents, various similarity measures have been proposed. Following the vector space model, Euclidean distance is the most intuitive. An even more frequent measure is the cosine similarity, which measures the cosine of the angle between two vectors, rather than an actual distance. The cosine of the two vectors $\overrightarrow{x}$ and $\overrightarrow{y}$ of size V is defined as:

$$cosine(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} \cdot \overrightarrow{y}}{\| \overrightarrow{x} \| \cdot \| \overrightarrow{y} \|}$$

The main strength of the cosine similarity measure is that it permits very efficient computation for normalized vectors, since in that case the denominator disappears, and the cosine computation simplifies to the inner product:

$$\begin{aligned} cosine(\overrightarrow{x}, \overrightarrow{y}) &= \overrightarrow{x} \cdot \overrightarrow{y} \\ &= \sum_{i=1}^{V} w_{x_i} \cdot w_{y_i} \end{aligned}$$

If the cosine measure is zero, the two document vectors are orthogonal, meaning the two documents have no single term in common. Note that, contradicting a common fallacy found in the literature, a cosine equal to one does not necessarily mean that the

corresponding documents are identical, since they may only contain the same words in different orders. Neither does it mean that the document vectors are identical, it solely indicates that they have equal weights *after* normalization. An example is two documents, where one of the documents consists of a number of consecutive copies of the other. Even when redistributing the words in any order, the cosine of the normalized document vectors will equal unity in most common weighting schemes. We will discuss later why word order deserves more attention.

The length of vectors has obviously no influence on the cosine of the angle they form. The cosine measure is thus length-independent. Unit-length normalization is performed anyway as part of the weighting scheme because this permits more efficient computations (as we just discussed, pre-normalizing vectors permits us to simplify the cosine computation to that of inner product).

Because the results of cosine similarity and Euclidean distances are very similar in nature (see for example [ZK01]), the efficient processing of cosine causes it to be by far the most popular similarity measure. Other measures are e.g., Jaccard and Dice coefficients [MS99].

2.2.5 Is Text just a Bag of Words?

This simple model where the index term vocabulary entirely consists of single words, is often referred to as the "bag of words" model, for it simply ignores the sequential evidence of text and treats documents as sets of words. As we already pointed out, two documents containing the same words in a different order would get the highest mark in terms of similarity, even though they may express different things. The following quotation of "Alice's Adventures in Wonderland" by Lewis Carroll illustrates how we can express different things by placing the same set (bag) of words in a different order.

> 'Do you mean that you think you can find out the answer to it?' said the March Hare.
>
> 'Exactly so,' said Alice.
>
> 'Then you should say what you mean,' the March Hare went on.

> 'I do,' Alice hastily replied; 'at least–at least I mean what I say–that's the same thing, you know.'
>
> 'Not the same thing a bit!' said the Hatter. 'You might just as well say that "I see what I eat" is the same thing as "I eat what I see"!'
>
> 'You might just as well say,' added the March Hare, 'that "I like what I get" is the same thing as "I get what I like"!'

In the vector space model, unfortunately, "I see what I eat" is the same thing as "I eat what I see". This is a problem we can solve by taking word order into account.

2.3 Advanced Descriptors

The vector space model can be improved in numerous ways. In this thesis, we focus our attention on sequential information in text and the different approaches to exploit it. The order in which words are used is an important piece of information, and a vector space model based on single words simply ignores this evidence. Before giving more details on the current trends in techniques for the extraction of word sequences, we will summarize a number of other approaches to ameliorate document description within the vector space model.

2.3.1 Advanced Uses of the Vector Space Model

Documents can nearly always be split into coherent parts and subparts of different depth, with some of these elements possibly being interleaved. This hierarchical structure can be implicit, but thanks to the development and widespread use of markup languages, it is often explicit, simplifying the development and usage of modeling techniques that take document structure into account.

We will later expand on ways to improve the bag-of-words model by exploiting the relative positions of words to extract semantic value. But the semantic meaning of individual words is very valuable already. Techniques have been developed to better exploit the meaning of words, and based upon that, their relations. A special case is geographical and temporal entities, which can be identified and related.

Advanced description based on structural information

Textual data is structured. A straightforward example of this is the delimitation of a book into chapters, sections, and paragraphs. The exponential growth of the amount of textual information in electronic format required explicit ways to express the structure of documents. This is usually done through a markup language that follows strict rules. The use of explicit structures that follow strict grammars facilitates the exploitation of this new data by automatic approaches. We will now present a few of them.

WWW and HTML documents. The World Wide Web (WWW) is an enormous source of structured information. The particularities of its formatting language, the *HyperText Markup Language* (HTML) [RLHJ99], have been an early and soon major center of interest for the web retrieval community. Thanks to HTML, web pages are highly structured. Different levels of headers can be marked-up, and numerous tags permit to identify important pieces of information. Another important strength of HTML is the possibility to include pointers between web pages, the so-called *hyperlinks*.

The markup of a text fragment offers a clear delimitation and extra knowledge about the meaning and importance of the text that it contains. Further, the markup elements in HTML are predefined and their number is fairly low. It is therefore simple to create rules for text encapsulated in any one of those tags. For document representation, the general technique of exploitation of the text markup of HTML webpages is to increase or decrease the weights of occurrences of words, depending on the tags they are encapsulated in. Cutler et al. [CSW97] proposed to use the structure of HTML documents to improve web retrieval. They studied the set of distinct HTML elements and assigned them into one of 6 disjoint classes. Each set of elements was associated to an importance factor that was reflected in the document model by modifying the weight of words, depending on the element class in which they occurred. In other words, term weights were linearly combined depending on the tags within which they occurred. Ever since, a number of techniques have been based on the same principle (see the latest TREC Web track [CH04] for a recent overview).

In addition to the markup encapsulating text, there is another

important kind of information that can be taken into account in modeling a set of HTML-structured documents, that is, their hyperlink structure. Undeniably, if the author of a document X has decided to place a pointer towards the document Y, the content of X gives us information about the content of Y, and reciprocally. Further, in very large sets of documents, we are more inclined to *trust* documents that have many pointers towards them. The rationale behind this is that the authors of the pointers must have found these documents worthwhile, and hence they must be more important than others. This idea permitted major breakthroughs in web retrieval. The techniques implementing this idea relied on the same principle: representing the documents by a standard vector space model, *and* attaching an importance score to each document, based on hyperlinks (the more incoming hyperlinks, the more important the document), see e.g. [CH03, CH04]. A well-known variation of this principle is the PageRank algorithm [PBMW98], whose main particularity is to account for the importance of a document to calculate the importance of the others. In other words, it does not only base the importance of a document on the number of incoming hyperlinks, but also takes the importance of the documents of origin into account. A more recent trend has been to cross the boundary between document representation and hyperlink-based measure of importance. Therefore, researchers have tried to exploit the relations between content and link structures. The first way to do this is to use the document content to improve link analysis, as in [Hav03, Kle99], while another approach is to propagate content information through the link structure to increase the number of document descriptors [QLZ+05, SZ03].

Content-oriented XML documents. The eXtensible Markup Language (XML) [BPSM+04] is a generalized markup language that allows for a more varied structure than HTML. From a technical point of view, an important difference between HTML and XML is that the set of elements in HTML is fixed and predefined, whereas there exists no general set of predefined elements for a given XML document. In fact, the elements and the way they should be used need to be specified in a separate declaration, the *Document Type Declaration* (DTD), that describes what hierarchies of elements are allowed in corresponding documents. XML is called a *meta-language*, because each DTD can actually define a different

language. HTML, on the other hand, is just one language. Another particularity of XML is that it is also used in the database community. As opposed to database-oriented XML documents, the main focus of interest of the information retrieval community is *content-oriented* XML documents, i.e., documents that consist essentially of textual data. An example of such a document, from the INEX[3] collection, is shown in Figure 2.2. This document gives explicit information about the publication details of a journal article and its content is structured with labels to mark the beginning and the end of paragraphs (<ip1>), sections (<sec>), and their titles (<st>). This document can be represented by the tree shown in Figure 2.3. The absolute XML Standard Path (XPath) expression towards an XML element is an incremental string indicating the path to be followed to reach the element, starting from the root of the tree. Hence, the XPath expression of the element containing the word "Abstract" is "/article/fm/abs/p/b".

We will now give some insight in a few of the ways the structure of content-oriented XML documents has been used to improve the quality of their description.

Yi and Sundaresan [YS00] have used the structure of XML documents in a straightforward way. They concatenated to every word term its XPath of occurrence, and thus augmented the vector space model of another dimension for every distinct path of occurrence of a word term. They applied this document representation to the task of document classification and reported successful results. It must be pointed out that they experimented with a well-structured document set, where each element has a clear signification and high discriminative power. This is unfortunately not a typical situation.

Even though XML documents should ideally be provided with a DTD, real-life data often contains XML documents without one. Given a collection of XML documents without their DTDs, Nierman and Jagadish [NJ02] proposed a technique to evaluate the structural similarity between XML documents, with the aim to cluster together documents that originally derive from the same DTD. The measure of the pairwise similarity between two XML documents is

[3]available at http://inex.is.informatik.uni-duisburg.de/2005/

```
<article>
 <fm>
  <hdr>
   <ti>
    IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING
   </ti>
   <volno>Vol. 15</volno> , <issno>No. 4</issno>
   <mo>JULY/AUGUST</mo> <yr>2003</yr>
  </hdr>
  <atl>
    Topic-Sensitive PageRank: A Context-Sensitive...
  </atl>
  <au> <fnm>Taher H.</fnm> <snm>Haveliwala</snm> </au>
  <abs>
   <p><b>Abstract</b>
     The original PageRank algorithm for improving the...
   </p>
  </abs>
 </fm>
 <bdy>
  <sec><st>Introduction</st>
   <ip1>Various link-based ranking strategies have...</ip1>
   <p>The PageRank algorithm, introduced by Page et...</p>
  </sec>
  <sec><st>Topic-Sensitive Page Rank</st>
   <ss1>
     <st>3.1 ODP-Biasing</st>
     <ip1>The first step in our approach is to...</ip1>
   </ss1>
  </sec>
 ...
 </bdy>
</article>
```

Figure 2.2: A sample content-oriented XML document.

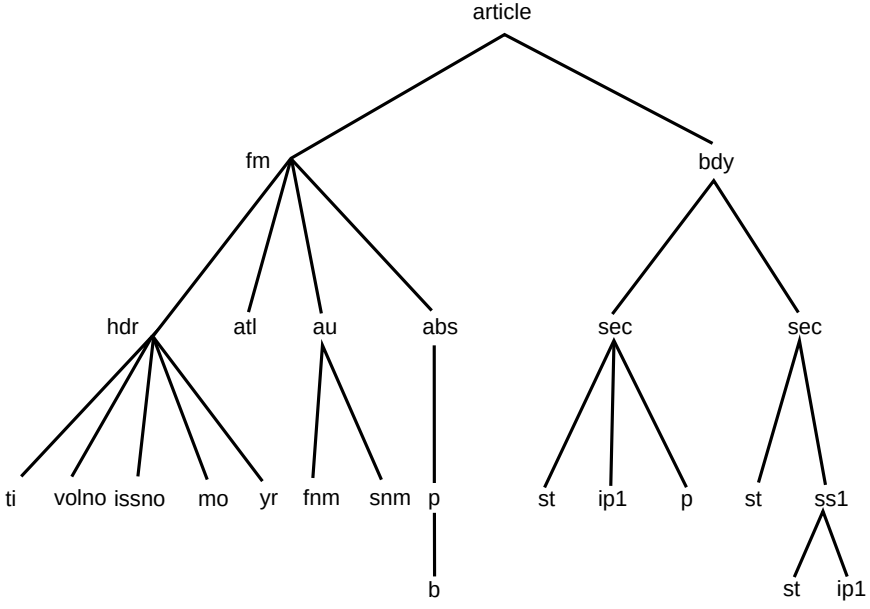

Figure 2.3: An example XML tree corresponding to the document of Figure 2.2.

based on their tree representations, and computed via a tree-edit distance mechanism. This technique performs well with respect to its goal. However, this approach focuses exclusively on the structure. From a content-oriented point of view, we naturally wish to integrate and combine content and structural information.

It is important to observe that XML, as opposed to HTML, was not designed as a language for formatting web pages. It has a much wider use, opening the door to applications beyond this scope. XML is, for example, used in editing, where a document can be very long, for example being an entire book, or a collection of journal articles. This creates new challenges, as it is not always satisfying to treat full documents as independent entities. The clear delimitation inherent to the XML structure form a good background to deal with accurate subparts of the document rather than with entire documents only.

The aim of the INitiative for the Evaluation of XML retrieval (INEX) [FGKL02, FLM03, FLMS05], started in 2002, is to address

the problem of XML retrieval. Its initial case document collection is a set of 12,000 journal articles of the IEEE. The sample XML document given in Figure 2.2 is an article from the INEX collection. This dataset is very relevant to the problem of XML retrieval. Suppose a user wants to find information about "datamining" in IEEE journals. A block of journal volumes certainly contains much relevant information, but is a too large answer to be satisfying. With this type of data, an information retrieval system needs the capacity to return only portions of documents. A simple way to be able to return XML document elements rather than full documents is to use the document structure, and represent each element by an independent vector. In that case, however, it is problematic to use standard weighting procedures, such as $tfidf$. The shortest elements will obtain the highest similarities with the user query, but to return a list of short italicized text fragments containing the word "datamining" will not satisfy the user's information needs either.

This clearly poses the problem of *granularity*. We should consider document fragments that are not too long but that are large enough to be able to stand alone as meaningful independent units. An address of this problem has been the definition of Minimal Retrieval Units (MRU) by Doucet et al. [DALP03], where the authors tailor the XML document to a sub-tree, in which the leaves represent the smallest elements that can be returned. The representation of ancestors of the leaf elements is generated on-the-fly, by propagating dimension weights from children to parent elements. A weighting scheme is used to penalize the longest elements, so as to seek for a trade-off between relevance and exhaustivity. Kamps et al. [KRS05] studied the length of the document elements judged as relevant by human readers. They accordingly integrated a length bias into their document model, thus defining specific length normalization techniques for XML elements that permit significant performance improvement.

Another new issue posed by XML retrieval is that of *content overlap* [KLdV04]. As we can observe in Figure 2.2, a paragraph may occur inside a section that may itself occur inside an article. The risk of directly representing XML elements as vectors is then to present the same highly relevant document portion several times, as it belongs to several overlapping elements. Clarke [Cla05] presented a technique to control the overlap. It consists in adjusting the

score of lower-ranked elements, if they contain, or are contained, in higher-ranked elements. The rationale is to penalize elements that contain information that was already seen by the user, assuming she goes through answers in increasing-rank order, as is generally the case.

A full overview of the latest trends in XML retrieval is provided by the INEX workshop proceeding series [FGKL02, FLM03, FLMS05].

Conclusion. There are many alternatives for taking advantage of the hierarchical structure of documents, weighing words depending on their place of occurrence, accounting for context, spreading a document's descriptions through pointers towards other documents, and so on.

We can also improve the basic vector space model, based solely on single words, by looking closer at those single words. A word is a complex and meaningful unit, and we can benefit from acknowledging this fact.

Semantic approaches to advanced document description

Individual words contain already a lot of information by themselves. Each of them carries a specific meaning that can often be determined only through the observation of its context. Many studies have aimed at determining the exact meaning of word occurrences. This supplementary piece of information can be used as such, or to relate, connect or separate words among each other.

Exploiting the meaning of words. In the basic vector space model, every distinct term of the document collection corresponds to a dimension of the vector space. This way, one can apparently not deal with problems of synonymy (when different words have the same meaning, e.g., strong and powerful) and polysemy (when the same word has different meanings, e.g, a mouse can be either a small rodent or a computer device). We can deal with the problem of polysemy by capturing the meaning of words, that is, by trying to figure out whether a given usage of the word mouse is referring to a rodent or to a computer device. A subsequent representation is a vector space with more dimensions, corresponding to distinct word meanings instead of distinct word terms. In the new vector space, there should be one dimension for the word "mouse" mean-

ing a rodent, and another dimension for the word "mouse" meaning a computer device.

A generalization of the problem of finding the meaning of words is Word Sense Disambiguation (WSD) (a good review of the state of the art, although slightly outdated, can be found in [IV98]). Given a word, its context of occurrence, and a set of possible meanings, WSD is the task of determining which is the correct meaning of the word in that context. There exists a number of techniques for this task. The first family of techniques is supervised disambiguation, where a learning algorithm is fed with a training set of documents in which words are manually annotated with a semantic label. Other techniques use dictionaries and the text content of the different definitions of the word as typical indicators of the corresponding meaning. Finally, unsupervised disambiguation consists of using the different contexts of a word to cluster its cases of usage into different categories, without knowledge of the corresponding meanings or associated definitions.

In general, to avoid excessive (or insufficient) dissociation of word senses, it is crucial to have, at least, a practical number of word meanings, arbitrarily given by manual annotations or a dictionary. All the techniques we just covered permit solving the problem of polysemy, but there exist many more semantic relationships between words, which we may wish to take advantage of.

Exploiting the semantic relationships between words. The ability to account for the relationships between words would be a clear improvement over the basic vector space model. The *term independence assumption* consists in considering that the probability of occurrence of a word in a given position is the same regardless of which words occur in other positions. This assumption is common as a matter of mathematical convenience rather than a reality. The vector space model also makes the faulty simplification to ignore correlations between words. The word "telephone" is more related to the word "communication" than to the word "butter", but this is hard to take into account in the vector space model [YBLS83]. A number of techniques have been proposed to account for the semantic relationships between words. They either rely upon handcrafted dictionaries or are automatic.

Started in 1983, *WordNet* [Fel98] is a continued effort that resulted in a thesaurus of English[4] that integrates a vast hierarchy of relationships between words. Each word entry of *WordNet* is associated with a set of synonyms. Depending on its part of speech, a word entry may also be associated, for example, to a set of antonyms (words of opposite meaning), hypernyms and hyponyms. The word Y is a hypernym of the word X if every X is a "kind of" Y. Hyponymy is the opposite relation. Hence, "red" and "blue" are hyponyms of color, and thus color is a hypernym of "red" and "blue".

A trivial sample application of such metrics is numerical evidence to determine sets of words that have similar senses, and that subsequently could be represented by the same dimension of the vector space. Sussna [Sus93] and Resnik [Res95] have used the $WordNet$ word relations to compute advanced semantic word metrics.

Latent Semantic Indexing (LSI) [DDL$^+$90] is a mathematical approach of the problems of synonymy and polysemy that derives from a well-known technique of linear algebra, i.e., Singular Value Decomposition (SVD). LSI is surprisingly effective, given that it is entirely automated, as opposed to $WordNet$ and the enormous human effort its construction required. LSI is a technique of *dimensionality reduction*. Its key idea is to model the documents of the vector space into a smaller space with "latent" dimensions. The high-dimensional document vectors are projected to some lower-dimensional space spanned by significant singular vectors. The expectation is that the dimensions of the smaller space represent the *concepts* of the documents, whereas the initial vector space represents their terms, eventually separating synonyms and uniting polysemes. A number of variations have been proposed, augmenting the basic LSI method by normalization and rescaling [BM05].

Exploiting the spatio-temporal relationships between words. Spatial and temporal pieces of information are apart from other terms. There is a relationship between "Helsinki" and "Finland", but it cannot be described as synonymy, meronymy or hyponymy. Some time indicators can, however, be described by such relations.

[4]There exist related projects for a number of other languages, such as *EuroWordNet* [Vos98] for European languages.

For example, "April" and "May" are two hyponyms of "month". But an important piece of knowledge is still ignored: "April" is intuitively *closer* to "May" than to "November". We can certainly benefit from detecting temporal markers and relating them to a time axis.

Work on spatial and temporal evidence is further motivated by strong information needs. When we look for a restaurant, we usually do not look for a restaurant anywhere on Earth, but rather in a specific place or in its vicinity. To assign one dimension per word term is totally inappropriate in this case. If we search for a restaurant in Helsinki, we also want to get results from Kallio (a Helsinki district), but restaurants from Melbourne will be irrelevant. In a simple vector space model, however, Kallio and Melbourne are equally distinct of Helsinki. There is a similar difficulty in distinguishing Saint-Petersburg, Russia, and Saint-Petersburg, Florida, USA. Woodruff and Plaunt [WP94] summarized a list of this kind of problems, termed as *deficiencies of text-based georeferencing*: lack of uniqueness, spatial boundary change, name changing, naming variation, spelling variation, neologisms. The importance of geographical information is underlined by the appearance of geographically-specialized features in the major search engines. There is even recent work to elicit geographical evidence from user queries, when this bit of information is not explicitly specified [WWX+05].

Geographic information systems are based on two steps. First, geographic references must be detected in text and associated with the appropriate locations. This can be effectively achieved with the help of a dataset associating place names and latitude/longitude coordinate points. An example of such a dataset is the Geographic Names Information System [otI95]. The second step is naturally to make use of the data extracted. Once locations have been georeferenced and associated to their latitude and longitude, it is simple to compute distance metrics among the geographic references found in text. For example, the *GeoVSM* system [Cai02] supports two distinct document descriptions, a word-based vector space model and a spatial description, resulting in spatial and vector space similarities that are gathered to a single similarity measure through a combination function.

Conclusion There are numerous ways to improve the basic vector space model. In the rest of this dissertation, we will, however, focus on the use of the sequential nature of text, giving a particular importance to portability across language borders.

2.3.2 Extraction of Sequential Information

Most document models do not account for word order in a document. However, we can assume that there must exist a way to account for word order, which permits us to improve document description. The sole use of single word terms in the vector space causes some trouble [ZTMFE97]. Some word associations have a totally different meaning of the "sum" of the meanings of the words that compose them (e.g., "hot dog" is most often not used to refer to a dog that is warm). Other lexical units pose similar problems (e.g., "to kick the bucket" is an expression that means "to die"). Given that "to kick" means "to strike out with the foot", and that a "bucket" is a "cylindrical vessel used for holding or carrying liquids or solids", it appears clearly representing the expression "to kick the bucket" by a dimension per word is truly misleading. This problem can be solved by detecting the word sequence "kick the bucket" and treating it as a single entity.

There are two approaches to extracting phrases: (1) using statistical information, i.e., trying to exploit the simple fact that phrases are words that occur together and (2) using linguistic information, i.e., exploiting part-of-speech patterns or syntactical relations between words. Many of today's approaches combine both statistical and linguistic evidence.

Statistical Extraction

Mitra et al. [MBSC97] form a statistical phrase for each pair of two stemmed adjacent words that occur in at least 25 documents of the TREC-1[5] collection. The selected pairs are then sorted in lexicographical order. In this technique, we see two problems. First,

[5]The Text REtrieval Conference (TREC) is a forum that provides data sets and judgments for the evaluation of text retrieval systems. Further information is available at http://trec.nist.gov/

lexicographical sorting means ignoring crucial information about word pairs, that is, their order of occurrence. Furthermore, no gap is allowed, although it is frequent to represent the same concept by adding at least one word between two others. For example, this definition of a phrase does not permit to note any similarity between the two text fragments "XML document retrieval" and "XML retrieval". This model is thus quite far from natural language.

Syntactical Extraction

The technique presented by Mitra et al. [MBSC97] for extracting syntactical phrases is based on a part-of-speech analysis (POS) of the document collection. A set of tag sequence patterns are predefined to be recognized as useful phrases. All maximal sequences of words accepted by this grammar form the set of syntactical phrases. For example, a sequence of words tagged as "verb, cardinal number, adjective, adjective, noun" will constitute a syntactical phrase of size 5. Every sub-phrase occurring in this same order is also generated, with an unlimited gap (e.g., the pair "verb, noun" is also generated). This technique offers a sensible representation of natural language. Unfortunately, to obtain the POS of a whole document collection is very costly. The index size is another issue, given that all phrases are stored, regardless of their frequency. In the experiments, the authors indeed admit to creating no index *a priori*, but instead that the phrases were generated according to each query. This makes the process tractable, but implies very slow answers from the retrieval system, and hence a long wait for the end user.

On top of computational problems, we see a few further issues. The first one is the lack of a minimal frequency threshold to reduce the number of phrases in the index. This means that infrequent phrases are taking up most of the space and having a big influence on the results, whereas their low frequency may simply illustrate an inadequate use or a typographical error. To allow an unlimited gap so as to generate subpairs is dangerous as well: the phrase "I like to eat hot dogs" will generate the subpair "hot dogs", but it will also generate the subpair "like dogs", whose semantical meaning is very far from that of the original sentence.

Collocations

Numerous phrase extraction techniques are combinations of statistical and syntactical methods. A collocation is defined by Smadja as "a recurrent combination of words that co-occur more often than chance and that correspond to arbitrary word usages" [Sma93]. The notion of arbitrariness underlines the fact that if one word of a collocation is substituted by a synonym, the resulting phrase may become peculiar or even incorrect. "To give an elevator" is notably not understood the same way as "to give a lift".

Choueka et al. The initial work on extracting collocations is that of Choueka, Klein and Neuwitz [CKN83]. Comparing to that of Smadja, they had a slightly different definition of a collocation, rather related to that of a statistical phrase: "a collocation is a sequence of adjacent words that frequently appear together". The sequences were theoretically of any length, but were limited to size 6 in practice, due to computational problems. They experimented on an 11 million word corpus from the *New York Times* archive and found thousands of common expressions such as *"home run", "fried chicken"*, and *"Magic Johnson"*. After pointing the limited size of the sequences, one can also regret the impossibility to extract any discontinuous sequence such as *"knock ... door"*, due to the adjacency principle of the definition. Finally, the criteria used to qualify or reject a sequence as a collocation is based on an absolute frequency threshold, which makes the results dependent on the size of the corpus.

Church and Hanks. Church and Hanks described a collocation as a pair of correlated words [CH90]. That is, as a pair of words that occur together more often than chance. The technique is based on the notion of *pointwise mutual information*, as defined in Information Theory [Sha48, Fan61]:

$$I(x,y) = log_2 \frac{P(x,y)}{P(x)P(y)},$$

where $P(x)$ and $P(y)$ are the probabilities of occurrence of the words x and y, and $P(x,y)$ is the probability that both occur simultaneously. There is a degree of freedom in the definition of the

"simultaneous occurrence" of two words. One can consider that two words occur together when they occur in the same document, and thus ignore word order, or consider that two words occur together when they are adjacent. More generally, a window of simultaneity needs to be defined to determine whether or not two words co-occur. This set of techniques permits to retrieve interrupted sequences of words as well as continuous ones. Unfortunately, a consequence of the definition of pointwise mutual information is that the set of candidate sequences is restricted to word pairs. This means that we can only acquire collocations of size 2, when Choueka's technique permitted to reach size 6, even though the drawback was then to require adjacency. Another limitation in the scope of their application of collocations to lexicography is that in addition to true lexical collocations, the technique finds pairs of words that occur simultaneously because their semantic-relatedness causes them to occur in the same contexts, and not because they form collocations. Typical such pairs found by Church and Hanks are *"doctor-nurse" or "doctors-hospitals"*.

Smadja's Xtract. Built on the work of Choueka et al., Smadja proposed a more complex technique. Xtract [Sma93] is a tool that uses a frequency-based metric to find candidate phrases, and a syntactical analysis to extract the collocations.

The first phase of Xtract is a statistical one. For each word, it consists in computing the average-frequency of the other words occurring within a 5-word radius thereof (either forward or backward). Each word pair is then associated to its *z-score*, i.e., its number of standard deviations above the average frequency. Pairs with a z-score below a relative threshold parameter are pruned away. Linguistic filters are then applied to get rid of those pairs, which are not true lexical collocates. For example, for a pair "noun-verb", the technique differentiates the case where the noun is the subject or the object of the verb. Semantically related pairs (such as *doctors-hospitals*) are also removed, by filtering the syntactic pairs of type "noun-noun".

Following the identification of word pairs, the collocation set is recursively extended to longer phrases, by searching for the words that co-occur significantly often together with a collocated pair identified earlier. The final step relies on the part-of-speech analysis

of the co-occurrences, which permits to filter out more candidates.

To evaluate the quality of the final set of collocations, a lexicographer was asked to judge which answers were good collocations. After the full processing, including both the statistical stages and linguistic filtering, 80% of the phrases were evaluated as good collocations. The score was only 40% before the last step of syntactic filtering, illustrating the primary importance of combining both linguistic and syntactic information, in order to find *accurate* lexical collocates.

It is worth underlining that the total amount of good collocations was the same before and after the last step of syntactic filtering. The precision improves from 40% to 80%, but the recall remains identical at 94%. The syntactic analysis is therefore a means to reduce the number of false positives, although a very efficient one indeed. This proves useful in the context of Xtract where the main targeted application is to assist lexicographers identifying new or interesting expressions. In this task, manual filtering through a set of candidate collocations is admittedly much faster than manual scanning through a document collection.However, as discussed earlier in section 2.1, the extraction of document descriptors often needs to be efficient. In this respect, syntactic analysis is a very serious threat.

Further, multilingualism is another major problem to address because of the fact that every single language requires a different syntactic parser. We will get back to this in more detail in section 2.4 when discussing the main weaknesses of existing extraction techniques of advanced document descriptors.

C-value and NC-value. The problem of nested collocations is addressed by Frantzi et al. [FA96]. They introduce the *C-value*, a measure to account for the fact that subsequences of a longer word sequence may actually be more interesting. The first step consists in extracting candidate *adjacent* word sequences with the further restrictions of a user-selected minimal frequency threshold and a maximal length (5 in the experiments). Part-of-speech analysis is then used to select only the candidates that match linguistic patterns such as "$NounNoun^+$" or "Adj^*Noun^+". The *C-value* is a variation of pure frequency. For each candidate sequence, the *C-value* is calculated by subtracting from its frequency the average

frequency of all the frequent sequences that contain it [6]:

$$\text{C-value}(a) = \log_2 |a| \left(freq(a) - \sum_{b \in \text{superseq}(a)} \frac{freq(b)}{|\text{superseq}(a)|} \right),$$

where a is the candidate sequence, $|a|$ is the length of a, $freq(a)$ is its frequency, and superseq(a) is the set of all longer sequences that contain a.

In later work, the same authors extend their work to take into account the context of candidate sequences [FAT98]. This is done in two steps. First, the nouns, adjectives and verbs that occur frequently next to interesting candidate sequences are identified. The candidate sequences that are occurring next to these frequent *context words* are then promoted, through a *context factor*. The assumption made is that *context words*, i.e., words occurring right before or right after the most interesting sequences are typical of them, and their presence in the context of a sequence indicates that it is more likely to be significant. The evidence of those context words is thus used as a pseudo-relevance feedback technique, to re-rank the candidate sequences using a context factor:

$$\text{Ctxt\_factor}(a) = \sum_{b \in C_a} \text{Ctxt\_freq}_a(b) \text{weight}(b),$$

where a is the candidate sequence, C_a is the set of all the context words of a, $\text{Ctxt\_freq}_a(b)$ is how often the word b is adjacent to the sequence a, and weight(b) is the proportion of all candidate sequences of which b is a context word, i.e., given a random candidate sequence, the probability that b is a context word thereof.

Finally, the NC-value is a linear combination of C-value and context factor:

$$\text{NC-value}(a) = 0.8 \text{ C-value}(a) + 0.2 \text{ Ctxt\_factor}(a),$$

where the weights of the linear combination are justified by experimental evidence: "Regarding the weights 0.8 and 0.2 (...), these were chosen among others after a series of experiments. The combination $0.8 - 0.2$ gave the best distribution in the precision of extracted terms" (from [FAT98], p.601).

[6] The original formula (from [FAT98]) is reformulated here.

The data used in the experiments is a small collection (810,719 words) of eye-pathology medical records in English. Unfortunately, no other experiments have been reported. The use of linguistic information combined to the lack of a mathematical justification for the C and NC measures cause some concern with respect to the chances of success of an application of this work to other domains and languages. The C and NC values indeed rely on a number of arbitrary parameter values that have been tuned for Automatic Term Recognition (ATR) in a collection of eye-pathology records in English. ATR is a special case of collocation discovery that aims at technical, domain-specific collections (sometimes also called *sublanguages*).

Defining specific measures and tuning their parameters for a small monolingual and domain-specific collection of well-structured documents, the performance improvement reported is moderate versus the baseline, ranking terms in descending frequency of occurrence. It is not granted that this technique can be applied to general-purpose document collections.

Dias et al. introduce a sound generalization of conditional probabilities to n-gram extraction [DGBPL00a, DGBPL00b]. In this work, an n-gram is defined as a vector:

$$[w_1 p_{12} w_2 p_{13} \ldots p_{1i} w_i \ldots p_{1n} w_n],$$

where the w_i's are words, and the p_{1j}'s denote the distance between w_1 and w_j. It is important to observe the rigidity of this definition of an n-gram. No similarity is taken into account between two sequences containing the same n words in the same order, if only one more (or one less) word occurs within one of the sequences. For example, the two text sequences "former United States President John Kennedy" and "former United States President John Fitzgerald Kennedy" permit to form two 6-grams:
$[former +1 United +2 States +3 President +4 John +5 Kennedy]$ and $[former +1 United +2 States +3 President +4 John +6 Kennedy]$. The obvious similarity between the two 6-grams is entirely ignored, they are only seen as different.

The *normalized expectation* of occurrence of n words in fixed relative positions is defined based on the average expectations to see each of the words occur in a position, given the occurrence and

position of occurrence of all the others. The cohesion of a lexical multi-word unit is thus calculated as an average of the cohesion of all its subparts.

$$NE([w_1 \dots p_{1i}w_i \dots p_{1n}w_n]) = \frac{p([w_1 p_{12} w_2 p_{13} \dots p_{1i}w_i \dots p_{1n}w_n])}{\frac{1}{n}\left(p([w_2 \dots p_{1i}w_i \dots p_{1n}w_n]) + \sum_{i=2}^{n} p([w_1 \dots \widehat{p_{1i}}\widehat{w_i} \dots p_{1n}w_n])\right)},$$

where the hat sign "ˆ" is written on top of the omitted term of a succession indexed from 1 to n. The actual main metric introduced by Dias et al., by which the list of n-grams is sorted, is a variation of normalized expectation that rewards n-grams occurring more frequently, the *mutual expectation*:

$$ME([w_1 \dots p_{1i}w_i \dots p_{1n}w_n]) = f([w_1 \dots p_{1i}w_i \dots p_{1n}w_n]) \times NE([w_1 \dots p_{1i}w_i \dots p_{1n}w_n]),$$

where $f([w_1 \dots p_{1i}w_i \dots p_{1n}w_n])$ is the absolute frequency of the n-gram.

The calculation of mutual expectation and the corresponding selection and ranking of n-grams is produced as follows. The first step of the extraction technique introduced by Dias is to rank every n-gram ($n \geq 2$) according to its mutual expectation. To avoid the extraction of an n-gram together with all its subparts (in theory, an n-gram has $(2^n - n - 2)$ subparts of size 2 or more), the LocalMaxs algorithm [SDGPL99] is applied to select the most cohesive subparts of an n-gram candidate. An n-gram N is selected as a multi-word lexical unit if and only if its mutual expectation is higher or equal to that of all the $(n-1)$-grams that it contains (unless $n < 2$), and if it is higher than that of all the $(n+1)$-grams that contain N. This technique efficiently exploits the fact that measures for n-grams of different sizes can be directly compared to each other. The LocalMaxs algorithm further permits to avoid the definition of a global *ad hoc* association measure threshold that places the borderline between cohesive and non-cohesive multi-word units, as is the case in related work (among others [CH90, Sma93]).

The approach is fully multilingual and domain-independent. It permits to obtain multi-word units of any size directly, without requiring a prior extraction of all bigrams. It does not require word adjacency either, permitting to find a wider range of multi-word

units. However, the fixed distance required between words appears as excessively rigid. This causes an explosion in the number of distinct candidates and corresponding low frequencies associated to them. The expectation measures are still computed for every candidate, through the transformation of the raw text into a considerable number of n-gram tables. The number of candidates to be considered is certainly a threat regarding the space-complexity of the process.

Finally, a small example can show a weakness of the LocalMaxs algorithm. Assume we have a 2-, a 3- and a 4-gram such that the 2-gram is a subgram of the 3-gram and the 3-gram is a subgram of the 4-gram. The LocalMaxs technique will in no situation qualify the three n-grams as multi-word lexical units. However, there are cases when it should. For example, the following 2-, 3- and 4-grams are true multi-word lexical units, but they cannot all be recognized as such with the LocalMaxs algorithm: "President Kennedy", "President John Kennedy" and "President John Fitzgerald Kennedy".

N-grams. It is worthwhile observing that the general definition of an n-gram is more comprehensive than that of Dias. An n-gram is notably defined by Banerjee and Pedersen [BP03] as a sequence of n tokens in online text, where a token can be defined according to different granularities, e.g., typically as a word or a character. They provided the *Ngram Statistics Package* (NSP), a flexible software for the extraction of n-grams following this definition. Given a maximal length for n and a window size k (the maximal distance allowed between the first and last words of an n-gram), the software outputs a list of all the n-grams together with their frequencies and those of their subgrams. Hence, given a maximal value for n, the output contains all the k-grams and their frequency, for $1 \leq k \leq n$. The generality of those definitions poses a number of computational threats. First, the impossibility to set a minimal frequency threshold means that a very large number of terms need to be stored in the main memory, so as to have their frequency counted. Following this, the practical window size needs to be set very low, hence forcing adjacency between most components of the n-gram.

NSP uses a midway approach on the use of stopwords. In the work we presented earlier, either no stopword list was used, or a

list was used and all stopwords were consequently removed. Banerjee and Pedersen proceed slightly differently, as they use a list of stopwords, but instead of removing all the stopwords, they only remove the n-grams consisting *exclusively* of stopwords. Hence, if an n-gram contains at least one word that is not a stopword, it will be kept.

To remove every n-gram containing a stopword, or, equivalently, to remove all stopwords before extracting the n-grams is a radical option of which limitation has notably been shown in the field of document retrieval through query log analysis. Williams et al. [WZB04] found that 8.4% of the explicit phrase queries of the Excite log contained at least one of the 3 most common words of the dataset, and that 14.4% contained one of the 20 most common words. They further exposed that in many queries, stopwords play an important role. For example, "flights to London" are not "flights from London".

The other extreme is to make no stopword pruning at all, and thus allow for phrases containing only stopwords. A common practice has been to remove the phrases entirely made of stopwords, because of the subsequent considerable increase in the total number of descriptors, with phrases that generally do not carry highly discriminative content. However, still in the domain of document retrieval, Paynter et al. [PWCB00] pointed out the consequence that in this case, a small number of phrase queries cannot be evaluated at all (those consisting of stopwords only), while many more are evaluated incorrectly. Typical examples in favor of keeping stopwords in phrases are, for example, "*to be or not to be*" or the band name "*the who*".

The current trend is nowadays to allow for stopwords in phrases, including stopwords-only phrases. This approach is further supported by industrial evidence, as the current leading search engine Google has been dealing with stopwords in phrase queries since 2002.

2.4 Weaknesses of the Current Utilization of Word Order

We exposed why word order matters, and we presented a number of techniques to exploit this fact. In this section, we will enlighten the weaknesses of the existing techniques, and what should better be taken into account, in an ideal world.

In short, the current state of the art presents multi-word unit extraction techniques that are too specific (domain- or language-dependent), or that extract units that are either too short or too rigid (no variance allowed in the number of words occurring between two words of a unit). The result is that, despite the obvious supplementary information brought by lexical cohesive units, their use in information retrieval applications brings very weak improvement. Sometimes, the results are even worsened.

2.4.1 Multilingualism

A number of techniques rely on language-dependent steps, such as using a stopword list, word stemming, or performing syntactical analysis. Syntactical analysis represents a threat to the need to extract document descriptors efficiently. Further, it is the most critical step, as it cannot be substituted by a language-independent approach. In the contrary, stopword lists can for example be generated automatically using frequencies of occurrence (the least and most frequent words being usually incorporated to the list). Obviously there exists no similar twist for part-of-speech analysis, and any system using parts of speech will require a distinct analyzer for every language it will encounter. We will discuss below why this is not realistic,and argue that a multilingual technique cannot make use of syntactical information.

Various estimates of the number of world languages have been proposed, but David Crystal proposes a plausible approximate figure of 6,000 [Cry00]. One country alone may possibly have hundreds of them (namely, Nigeria is known to have over 400 languages). Certainly in this domain, any precise number is highly controversial, given at least the variable definitions of language and dialect, and the fact that no world languages survey is complete. One cannot seriously consider building a multilingual method that would include

language recognition techniques for 6,000 distinct languages and as many syntactic analyzers and stoplists.

It can be argued to this last point that for a large majority of languages, the available quantity of documents in electronic form is small or null. According to the website of UNESCO's[7] Initiative B@bel, 90% of Internet content is available in approximately 12 languages. The reason is simply that "the creation of content development and information exchange tools and systems has largely been driven by commercial interest", thus excluding languages spoken by small communities. However, the initiative B@bel, among other projects, aims at creating such tools to permit the development of content in smaller languages.

It is further estimated that about 50% of the world languages will get extinct in the coming 100 years, that is on average, the death of one language every two weeks. Responding to the situation, UNESCO adopted the "Endangered Languages Project" in 1993, with the aim to promote and sponsor programs for the description of hitherto unstudied or inadequately documented endangered and dying languages. For the numerous already condemned, the last resort is to archive as many language samples as possible. Such languages draw and will keep drawing attention from scholars who will hardly find any tools for assisting them in their work. It seems, for example, a safe bet to assume that there will not be much research aimed at the development of automatic syntactic analyzers for those languages.

The utilized approaches for dead and endangered languages are therefore bound to be multilingual ones, general in nature, based on data mining or statistical techniques.

Should we anyway decide to focus on the most spoken languages, there are still 347 languages with at least one million speakers. Together they account for 94% of the world's population [GJ05]. In any case, 347 is an unrealistic number of part-of-speech analyzers.

At a time when multilingual information retrieval is an increasingly important research domain, we think it is of crucial importance to propose language-independent techniques for the extraction of multi-word units. The difficulty of this task is illustrated by

[7]United Nations Educational, Scientific and Cultural Organization http://www.unesco.org

rare research in this direction, as was suggested by a recent workshop on multi-word expressions [TVBK04] where most of the 11 accepted papers presented monolingual techniques for a total of 6 distinct languages.

2.4.2 Contiguous and Non-Contiguous Rigidity

Due to computational complexity constraints, most extraction techniques require adjacency between words as a prerequisite for them to be able to form candidate multi-word units. This limitation is sometimes artificially lifted by the use of stoplists to priorly remove function words. Even so, requiring word adjacency remains a limitation. For example, the lexical unit "XML retrieval" will be missed in the text fragment "XML information and its retrieval". And only by using a stoplist that removes the function words "and" and "its", can we duly find the pair "information retrieval" (but still miss the pair "XML retrieval").

As we mentioned above, Dias et al. [DGBPL00a, DGBPL00b] introduced a technique that allows for other words to occur between two words of a multi-word unit. This number of other words is unfortunately fixed, meaning, e.g., that the obvious similarity between the two text fragments "XML information retrieval" and "XML retrieval" is ignored.

The main reason for the rigidity of those models, be it due to adjacency constraints (what we define as *contiguous rigidity*) or to fixed distances (*non-contiguous rigidity*), is computational complexity. Such constraints do evidently not provide realistic representations of the complexity of human languages.

This weakness is attenuated by work in the domain of lexicography, where it was shown that most lexical word associations in English involve words that are separated by 5 other words at most[JS74]. It is thus reasonable to think that the damage caused by such rigid models is not as bad in English as it could be in other languages.

While this maximal 5 words distance is agreed upon for English, it is certainly arguable for many other world languages. Clearly, one can imagine that this maximal distance is greater for (1) *isolating languages* (also known as *synthetic*, languages where all the words are invariable, and grammatical relationships are shown through

the use of word order), such as Chinese, Vietnamese or Samoan and for (2) languages where word order is more flexible (think, e.g., of German, where the verb sometimes must occur at the very end of the sentence).

We believe that the rigidity constraints in use in most extraction techniques are harmful. This fact is admitted, but the damage is probably much worse for isolating languages and for those in which word order is more fluctuating. There is need for a technique that accounts for discontiguous association and allows for variation in the relative positions of the words of a lexical unit.

2.4.3 Limitation in the Size of Phrases

Another consequence of computational complexity is that techniques that are defined for sequences of any length often require a maximal size in practice. The issue is a problematic one as very long frequent sequences may occur in text. If a maximal length is set to, say 8, and there exists a frequent multi-word association of size 20, the discovery process has to output all the 8-sequences that are subsequences of the longer one. If we disregard word positions, their number is equal to the number of ways to choose 8 objects from a set with 20 elements, that is $\binom{20}{8} = \frac{20!}{8!12!} = 125\ 970$. Given that this one sequence of size 20 and the $125,970$ sequences of size 8 represent the same quantity of information, it is obvious that a technique that only outputs the longer sequence is preferable.

This fact is striking for English, but in isolating languages, once more, where each word corresponds to one morpheme, the size limitation is even more critical as such languages naturally produce longer frequent sequences.

The ability to extract sequences of any length is very desirable, as it permits to provide accurate text descriptors in a very compact form.

2.4.4 Weak Results

As a matter of fact, using multi-word units in information retrieval has so far produced weak improvement. In some cases, the results have even been worsened. We will next cover results and comments from the domains of text classification and document retrieval.

In his Ph.D. dissertation, David Lewis [Lew92] extracted syntactic phrases using a simple linguistic pattern: in a sentence, any pair of non-function words that are heads of syntactic structures connected by a grammatical relation. To compensate for the weak frequency of phrases, he proposed to add *phrase clustering* in the loop (to gather together phrases with a similar behavior). He evaluated his phrase clusters indirectly through text retrieval and text classification experiments. The initial experiments in text retrieval produced very weak improvement. The author pointed out the fact that text retrieval collections had a relatively small number of queries available, and that only a portion of the phrasal terms is actually used. Therefore, any variation in effectiveness is hardly statistically significant. Lewis further finds it difficult to extract phrases from user queries, whereas this task in the document collection is eased by the possibility to rely on statistics and part-of-speech analysis. Consequently, he decided to use text classification to pursue his goal to evaluate phrasal text representation. The results were even worse. A purely phrasal representation was found to produce much lower effectiveness than a word-based representation. And the use of phrase clusters only worsened the results.

Work on the use of phrases in information retrieval has consistently exposed marginal improvements. Promising early results in 1975 [SYY75] were likely due to a combination of the lack of statistical significance of a too small document collection, and to the fact that the quality of "basic" information retrieval systems gradually improved, which made it harder and harder with time to perform major efficiency improvement by the use of phrases. Reiterations of these early experiments have all produced weaker amelioration [Fag89, MBSC97, TM99]. In 1999, Karen Spärck-Jones [SJ99] discouraged linguistics-based efforts, reaching the conclusion that "linguistically-motivated indexing is not needed for effective retrieval". About multi-word units, she adds "where compound terms are concerned, the statistical facts about term occurrences help as much to make joined terms linguistically legitimate as natural language processing can". Latest experiments still did not show more than marginal improvement [Vec05].

A few reasons have been suggested for these disappointing results. Mitra et al. [MBSC97] suggests that the use of phrases tends to favor documents that deal with only one aspect of multi-faceted

queries. Such an example from TREC-4 is a query that deals with "problems associated with pension plans, such as fraud, skimming, tapping or raiding". Reportedly, most promoted documents were dealing with pension plans, but not with associated problems. This problem is termed as one of *inadequate query coverage*. Another issue is the fact that the result of mixing phrases and words into similar weighting schemes has unclear consequences. Obviously, in a naive scheme, the words occurring in phrases will have their weight indirectly augmented. Precautions must be taken with respect to this fact and the way we should deal with it [SSW+98, Lew92].

Despite discouraging experimental evidence, there is still a strong consensus towards the assumption that the use of phrases must permit to improve the performance of information retrieval applications. The principal ground for this widespread agreement is the fact that phrases have higher information content and specificity than words. Despite more than 30 years of research, this self-evident fact remains surprisingly difficult to exploit.

Experiments have been attempted to put aside the problem of the automatic extraction of multi-word units, by involving humans in the extraction process. Lewis ([Lew92] page 68) improved text retrieval results by manually selecting the phrases to be used. More recent work in the domain of interactive information retrieval by Vechtomova [Vec05] used standard techniques of phrase extraction (e.g., C-value) to *propose* ranked lists of candidate phrases to be manually judged by the end-user. Only the user-selected phrases were used in the document representations, resulting in slightly better performance.

We can see these attempts of involving experts in the extraction process as an illustration of the difficulty of automatically extracting good phrasal descriptors. The modest improvements resulting from the use of manually selected phrases is further support for the idea that the exploitation of phrasal descriptors is not a solved problem either.

Hence, not only is there room for more research towards the extraction of better phrases, but there remains a lot of work to be done towards finding better techniques to exploit them in real information retrieval applications.

2.4.5 Conclusion

Now that we have developed on the current techniques and their limitations, we shall look at the results following a radically different approach of the discovery of interesting word sequences. The data mining field has proposed a number of algorithms for *sequential pattern mining*, an approach that, as its name indicates, is general enough to treat words as *any* other type of sequential data. An obvious consequence is the absence of linguistic information in the basic definitions. Linguistic information can in fact often be taken into account, but it is never necessary.

In the next chapter, we will take a glance at the data mining approach to the discovery of interesting word sequences. We will then present *MineMFS*, a promising technique to extract Maximal Frequent Sequences (MFS) developed at the University of Helsinki by Helena Ahonen-Myka. For relatively large document collections, where $MineMFS$ fails to provide good content descriptors, we will present $MFS\_MineSweep$, a contribution of this dissertation that permits to extract interesting word sequences from document collection of virtually any size. It relies on the idea of partitioning a large collection into cohesive subcollections, before joining the interesting word sequences of each subcollection.

CHAPTER 3

Advanced Document Descriptors - A proposal

After covering the state of the art in collocation discovery, we came to the conclusion that the current multi-word extraction techniques are lacking a number of desirable characteristics. First of all, the multi-word units extraction techniques are often too specific, relying on linguistics or domain-dependent steps. Second, they are nearly always limited in size, which may imply a drastic increase in the amount of space required to store the corresponding set of content descriptors. Finally, rigid descriptors, i.e., descriptors that account for no variance in the number of words between two words of a unit, are too strict a model of the variety of natural language.

In the next section, we will define the concept of a Maximal Frequent Sequence (MFS) [AMD05, AM05]. In short, a sequence is said to be *frequent* if it occurs more than a given frequency threshold. It is said to be *maximal*, if it is not a subsequence of another frequent sequence. The property of maximality guarantees to obtain the longest frequent sequences possible, and none of their subsequences individually. This ensures that the set of MFSs of a document collection is a very compact representation. We believe maximal frequent sequences can be an efficient way to account for the sequential aspect of textual data. Aiming at the extraction of all the MFSs of a document collection, we will present various methods of sequential pattern mining. All of them avoid the problem of sequence rigidity, by allowing for an unlimited gap between any two items (words) of a sequence. None of them uses linguistics

or domain-dependent techniques. Most of them are actually not meant for textual data, but rather for any type of sequential data.

We will finally present $MineMFS$, an algorithm for the extraction of MFSs in text. We will also present its limitations and finally introduce our contribution, $MFS\_MineSweep$, a technique that relies upon $MineMFS$ to extract relevant document descriptors from document collections of virtually any size (Section 3.2).

3.1 Maximal Frequent Sequences (MFS)

Let us now introduce the concept of a Maximal Frequent Sequence in further detail. We will then overview the data mining techniques that aim at the extraction of sequential patterns, and notably those that permit to extract Maximal Frequent Sequences.

3.1.1 Definitions

Definition 1 *A sequence $p = a_1 \cdots a_k$ is a* subsequence *of a sequence q if all the items $a_i, 1 \leq i \leq k$, occur in q and they occur in the same order as in p. If a sequence p is a subsequence of a sequence q, we also say that p* occurs *in q and that q is a* supersequence *of p.*

For instance, the sequence *"unfair practices"* can be found in all of the three sentences in Figure 3.1.

The *interestingness* of a subsequence is usually defined with respect to a set of *constraints*, which are assumed to represent some natural restrictions in the domain. In practice, the constraints are also used to reduce computational costs. The most common constraint is the *minimum frequency*. The frequency of a (sub)sequence can be, e.g., the number of text fragments that contain it.

Definition 2 *A sequence p is* frequent *in a set of fragments S if p is a subsequence of at least σ fragments of S, where σ is a given frequency threshold.*

If we assume that the frequency threshold is 2, we can find two frequent sequences in our sample set of sentences: *"congress retaliation against foreign unfair trade practices"* and *"unfair practices"* (Fig. 3.1).

1. The **Congress** subcommittee backed away from mandating specific **retaliation against foreign** countries for **unfair** foreign **trade practices**.

2. He urged **Congress** to reject provisions that would mandate U.S. **retaliation against foreign unfair trade practices**.

3. Washington charged France, West Germany, the U.K., Spain and the EC Commission with **unfair practices** on behalf of Airbus.

Figure 3.1: A set of sentences from the Reuters-21578 collection [Reu87].

As we will see shortly, the special characteristics of text data usually prohibits discovering all frequent subsequences. Instead, the patterns of interest can be restricted to be *maximal frequent subsequences*.

Definition 3 *A sequence p is a* maximal frequent (sub)sequence *in a set of fragments S if there does not exist any sequence p' in S such that p is a subsequence of p' and p' is frequent in S.*

In our example, the sequence *"unfair practices"* is not maximal, since it is a subsequence of the sequence *"congress retaliation against foreign unfair trade practices"*, which is also frequent. The latter sequence is maximal.

In addition to a minimum frequency threshold, we can also set a *maximum frequency threshold*. If we prune away the very frequent words, we can reduce the search space significantly. The disadvantage is naturally that we cannot discover any sequences that contain common words, like verb–preposition pairs.

The *internal density* of subsequences can be influenced by constraining the occurrences of events into a predefined window. The size of a window can be a fixed constant, or some natural structure can be taken into account. For instance, the words in a sequence have to occur within a sentence or a paragraph. This latter constraint can be easily implemented by choosing the representation of a text to be a set of sentences or a set of paragraphs, respectively. We can also define a *maximum gap*, which gives the number

of other words that are allowed in text between the words of a sequence. If the maximum gap is zero, we find n-grams in the most common sense, i.e., sequences of words, where the words occur consecutively. It is also possible to define a *minimum gap*, although it is harder to find any reasons for that in a general case.

A *minimum* and *maximum length* of sequences can also be defined, although both are problematic in practice. Usually the minimum length of interesting sequences is 2. As the number of sequences decreases radically when the length of sequences increases, we would probably lose a significant part of interesting sequences, if we set the threshold even to 3. The set of frequent pairs naturally also contains a load of uninteresting information, and hence, ignoring them is tempting. It seems, however, to be more reasonable to use some other ways to measure the interestingness than the plain length. Setting a maximum length for a sequence may also be problematic, as very long frequent sequences may occur in a text. As we already discussed in section 2.4.3, a maximum length, even high, may easily cause a thousands-fold increase in the number of document descriptors. Whereas if the length is not restricted, the maximal frequent sequences get a chance to be a very compact representation of the regularities in text.

Now that we have defined Maximal Frequent Sequences, and motivated the reasons why they are an efficient text representation, we will focus the rest of this section on how to efficiently extract the set of MFSs of a document collection.

3.1.2 Levelwise Extraction

Given a document collection and a minimal frequency threshold, a natural first idea is to go through the document collection, collect each frequent word, and use the set of all frequent words to produce candidate word pairs (bigrams) and retain only the frequent ones. The process of forming and counting the frequency of $(n+1)$-gram candidates from the set of all frequent n-grams can be repeated iteratively as long as frequent $(n+1)$-grams are found. To obtain the set of all MFSs, the last step is to remove every frequent sequence that is a subsequence of another frequent sequence. This naive approach is very inefficient as such and needs further improvement.

In 1995, Agrawal and Srikant [AS95] introduced the problem of

mining sequential patterns as an advanced subtask of data mining. Its principal aim is thus to support the decision-making of large retail organizations, where typical data consists of customer transactions, that is, database entries keyed on a *transaction id* and each consisting of a *customer id* associated to the list of items that she bought in this very transaction. The problem of mining sequential patterns is a more advanced version of a general problem of data mining, the extraction of interesting *item sets*. Zaki [Zak01] describes the "sequence search space (as) much more complex and challenging than the item set space". An obvious and intuitive explanation is that there is more to deal with. In sequential pattern mining, we aim to exploit the fact that the transaction entries of the databases include a time field that permits to sort the transactions in chronological order and even know the time interval (or distance) that separates them. Association rules describe intra-event patterns, while sequential pattern mining must also discover inter-event patterns. A motivating example of an interesting sequential pattern (imagined in [AS95]) would be that customers typically rent the movie "Star Wars", then "The Empire Strikes Back", and finally "The Return of the Jedi".

In the same paper, Agrawal and Srikant [AS95] present the *AprioriAll* algorithm that only differs from the naive approach presented above because of an intermediary pruning step to remove all $(n+1)$-gram candidates that contain at least one non-frequent n-gram. This permits to avoid a number of useless frequency counts. The remaining necessary counts are executed faster by storing the sequences in *hash-tree* structures. A very close approach by the same authors (in reversed order) is the *GSP* algorithm [SA96], which performs better than *AprioriAll* mostly thanks to slightly more advanced candidate pruning.

In the same spirit, Mannila et al. [MTV95] proposed a technique for discovering frequent episodes that consists in sliding a time window over the input sequence, and finding all the patterns that occur in a user-specified percentage of the windows.

3.1.3 Sequential Pattern Mining Techniques

SPADE. Zaki [Zak01] presented *SPADE*, an advanced technique for the discovery of sequential patterns, which introduced a

number of key improvements. First, it uses a vertical database format, where each sequence is associated with a list of the positions where it occurs. This improves the process of frequency counting. Second, the candidate $(n+1)$-sequences are generated by intersecting each two n-sequences that have a common $(n-1)$-length prefix. A third main improvement is a lattice-theoretic approach that permits to reduce the search space into smaller equivalence classes that can be processed independently in main memory. Unfortunately, although $SPADE$ accelerates frequency counting, it still enumerates all frequent sequences.

DFS_Mine. Using a comparable lattice-theoretic approach, Tsoukatos and Gunopulos [TG01] presented DFS_Mine, a technique that tries to discover $(n+1)$-sequences without enumerating all the frequent sequences of length n. The technique relies on a new way to exploit the fact that every subsequence of a frequent sequence is frequent. It uses the consequent fact that, if a short sequence is not frequent, then every longer sequence that contains it can be discarded. The main idea of DFS_Mine is that, if a n-sequence is found before its $(n-1)$-subsequences have been enumerated, they will not need to be enumerated at all, since they will be known to be frequent. In practice, the generation of candidate $(n+1)$-sequences is done by intersecting a n-sequence with all the frequent items. The algorithm stores two global structures: first, a list of *minimal non-frequent sequences* that is used to prune out all the candidate sequences that contain one of them. Second, the output of the algorithm, a list of *maximal frequent sequences*, which is also used to prune the candidate sequences that are subsequences of a maximal frequent sequence.

DFS_Mine has been developed for spatiotemporal data, where the number of different items is low. Thinking of text, to intersect n-word sequences with all (or many) of the frequent words of the collection is not realistic.

3.1.4 Sequential Patterns and Text

If applied to textual data, all the breadth-first, bottom-up approaches would fail quickly (or actually never end). To extract all the maximal frequent sequences of a document collection, they permit pruning but require to keep in memory all the subsequences

of two distinct lengths. They further require an unreasonable number of passes over the data. Depth-first search takes less memory and permits to calculate all maximal frequent sequences. However, when using data with a large vocabulary (e.g., textual data), the number of items (e.g., words) to be intersected with a given sequence is equally large. So is the number of intersections to be processed.

The specific characteristics of textual data make the applicability of such algorithms rather restricted. As a matter of fact, the domains of application presented in sequential pattern mining have limited-sized vocabulary (the number of *distinct items*), e.g., customer transaction analysis or telecommunication network monitoring [MTV95]. Another main research trend is to mine biosequences, where the size of the vocabulary is even smaller, e.g., there are only 20 amino acids, and there are only 4 molecules containing nitrogen present in the nucleic acids DNA and RNA (designated by the letters A, C, G, and T).

Hence, a major difference between the common data of sequence mining and textual data is the size of the vocabulary. As we have seen in section 2.2, a comprehensive estimate of the number of words in English is two million. We should also observe that this figure does not include numbers and codes, which are often present in real-life documents. Furthermore, one same collection may well contain text fragments in different languages, suggesting that the potential size of the *index term vocabulary* is far greater than two million. Of course, such numbers are never reached in practice, but even a moderate-sized text collection has a considerably large vocabulary. For example, the vocabulary size of the widely known *Brown corpus* is $50,406$ distinct words [KF67]. Even so, this corpus is very small by today's standards, as it contains a total of about one million words for a total size of 7 megabytes, while the collections used in the experiments of this dissertation thesis are much bigger, some of them larger than a gigabyte.

Further, the distribution of words is skewed. There is a small number of words that are very frequent, whereas the majority of words are infrequent. The words with moderate frequency are usually considered the most interesting and most informative. If the very frequent words are removed, the resulting search space is very sparse. The length of the interesting sequences is also skewed.

There is a large number of short sequences, but also very long sequences are possible. An extreme case is, for instance, if the data set contains several copies of the same document.

These special characteristics of textual data have a strong influence on the discovery of interesting sequences in text. Breadth-first, bottom-up approaches such as [MTV95, AS95, SA96] are failing quickly with textual data for a number of reasons. First, they generate a lot of candidates and counting the frequency of occurrence of each of these candidates is slow. Further, in order to answer the question if a candidate occurs in an input sequence, all the n-sequences of the input sequence are conceptually generated and compared to the set of candidates. As frequent sequences can be very long, this is prohibitive.

3.1.5 An Answer: $MineMFS$

Helena Ahonen-Myka developed a method that combines breadth-first and depth-first search, MineMFS [AMD05, AM05]. It extracts maximal frequent sequences of any length, i.e., also very long sequences, and it allows an unrestricted gap between words of the sequence. In practice, however, text is usually divided into sentences or paragraphs, which indirectly restricts the length of sequences, as well as the maximal distance between two words of a sequence. The constraints used in the method are minimum and maximum frequency. Hence, words that are less (respectively, more) frequent than a minimum (respectively, maximum) frequency threshold are removed.

Because the number of distinct words can be relatively high, as can the number of 2- and 3-grams, the sets of frequent pairs and the set of frequent trigrams are computed separately. All the frequent ordered pairs are initially collected. An ordered pair is a 2-gram (A, B) such that the words A and B occur in the same sequence in this order and the pair is frequent in the document collection. Next, all the frequent words that do not belong to a frequent pair are removed. This extra step eases the computation of the frequent ordered 3-grams. As detailed in Algorithm 1, the frequent trigrams are used as seeds for the discovery of longer frequent sequences, while the set of maximal frequent sequences initially contains the maximal frequent pairs, that is, the set of all frequent bigrams that

are not a subsequence of a frequent trigram.

Algorithm 1 *MineMFS.*
Input: G_3*: the frequent* 3*-grams*
Output: Max*: the set of maximal frequent sequences*

```
1.  n := 3
2.  Max := the maximal frequent pairs
3.  While G_n is not empty
4.      For all grams g ∈ G_n
5.          If a gram g is not a subsequence of some m ∈ Max
6.              If a gram g is frequent
7.                  max := Expand(g)
8.                  Max := Max ∪ {max}
9.                  If max = g
10.                     Remove g from G_n
11.             Else
12.                 Remove g from G_n
13.     Prune away grams that are not needed any more
14.     Join the grams of G_n to form G_{n+1}
15.     n := n + 1
16. Return Max
```

As for DFS_Mine, an important principle of $MineMFS$ is that it computes frequent $(n+1)$-sequences without enumerating all frequent n-sequences. The input of the discovery process (i.e., Algorithm 1) is the set of all frequent 3-grams. Its principle is to take a 3-gram and try to combine it with other items in a greedy manner, i.e., as soon as the 3-gram is successfully expanded to a longer frequent sequence, other expansion alternatives are not checked, but only that longer frequent sequence is tentatively expanded again. This expansion procedure is repeated until the longer frequent sequence at hand can only be expanded to infrequent sequences. This last frequent sequence is a maximal one. This step is known as the *expansion step* (line 7 of Algorithm 1).

When all the frequent 3-grams have been processed in this way, those that cannot be used to form a new maximal frequent sequence of size more than 3 are pruned (line 13 of Algorithm 1). The remaining ones are used to produce candidate 4-grams (line 14) that

will be used in a new iteration of the process that relies on 4-gram seeds (line 15). Iterations are repeated until no new maximal frequent sequence can be discovered.

A main strength of $MineMFS$ versus DFS_Mine is the fact that in the expansion step, the choice of items that may be inserted to tentatively expand a n-gram is restricted to the other non-pruned frequent n-grams. Whereas in DFS_Mine, a n-gram is expanded by trying to insert every (or most) frequent items, which is not realistic when the items are as numerous as the distinct words of a document collection.

Finally, by using sophisticated pruning techniques, the *MineMFS* algorithm restricts the depth-first search, which permits to check only a few alternatives for expanding a sequence, even though the size of the vocabulary is large. This permits to extract all the maximal frequent sequences of a document collection in an efficient manner.

The use of minimal and maximal frequency thresholds also permits to reduce the burstiness of word distribution. On the other hand, it causes the miss of a number of truly relevant word associations. For large enough collections, the $MineMFS$ process fails to produce results, unless excessive minimal and maximal frequencies are decided upon, in which case the set of MFSs produced is small and contains mostly non-interesting descriptors.

One reason may be the pruning step, which relies on the heavy process of going through the set of n-grams, and comparing each one of them to every other n-gram with which they can form an $(n+1)$-gram. Numerous checks have to be computed in this step. Another difficulty stems from the expansion step, where it must be checked if a new item can be added between every two adjacent words of a possibly long sequence. The number of possible positions of insertion shall be problematic.

3.2 $MFS\_MineSweep$, Partitioning the Collection to Approximate the MFS set

When we try to extract the maximal frequent sequences of a large document collection, their number and the total number of word

features in the collection pose a clear computational problem and does not actually permit to obtain any result.

To bypass this complexity problem, we present *MFS_MineSweep*, to decompose a collection of documents into several disjointed subcollections, small enough so that the set of maximal frequent sequences of each subcollection can be extracted efficiently. Joining all the sets of MFSs, we obtain an approximate of the maximal frequent sequence set for the full collection. We conjecture that more consistent subcollections permit to obtain a better approximation. This is due to the fact that maximal frequent sequences are formed from similar text fragments. Accordingly, we formed the subcollections by clustering similar documents together using a well-known clustering algorithm.

We will now present and evaluate this new technique that complements the $MineMFS$ algorithm to permit the extraction of more and sharper descriptors from document collections of virtually any size. Its main drawback is the loss of the maximality property, producing a less compact set of content descriptors.

3.2.1 Description and Claims

Our approach relies on the idea to partition the document collection into a set of homogeneous subcollections. The first reason to do this is that $MineMFS$ does not produce any result at all for sufficiently large document collections.

Figure 3.2 describes the steps of $MFS\_MineSweep$. In the first phase, we apply $MineMFS$ on a number of disjoint subcollections, so as to obtain an MFS set corresponding to each subcollection. The second step is to gather the MFS sets of each subcollection to form a set of content descriptors for the whole collection. This gathering operation mainly consists in appending the sets of MFSs, as there is no clear way to join a sequence (maximal frequent in a subcollection) to its subsequence (maximal frequent in another). Only identical sequences can be merged. Thus, the maximality property is lost, and therefore, the content description of our prepartitioning technique is always less or equally compact to that of the MFSs of the whole collection.

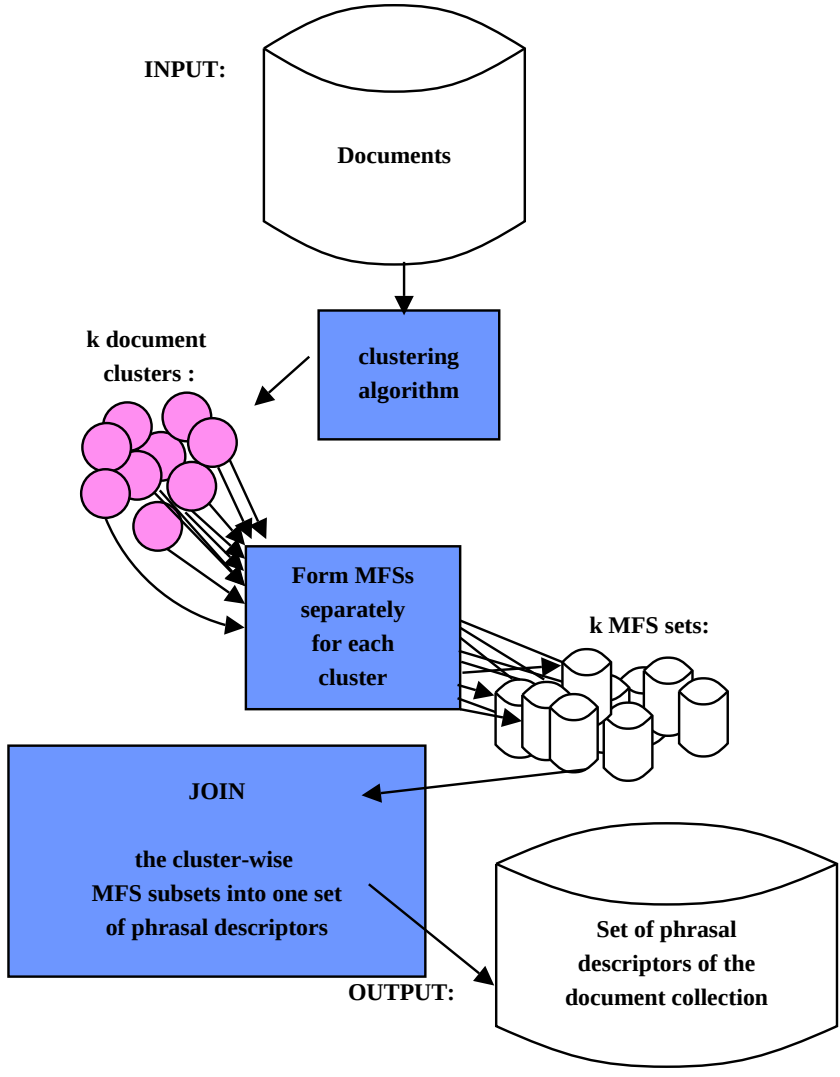

Figure 3.2: The different phases of $MFS\_MineSweep$.

- d_1: Mary had a little lamb whose fleece was white as snow.
- d_2: A radio station called Sputnik broadcasts Russian programs in Saint-Petersburg and Helsinki. It was named after the first satellite ever launched.
- d_3: History changed on October 4, 1957, when the Soviet Union successfully launched Sputnik I. The world's first artificial satellite was about the size of a basketball, weighed only 183 pounds, and revolved around the Earth in about 98 minutes.
- d_4: Everywhere that Mary went, her lamb was sure to go.

Figure 3.3: A collection of four documents.

Hypotheses.

With this technique, we make three main claims that we will try to confirm or disprove in the evaluation. The main motivation for developing $MFS\_MineSweep$ is to obtain results more efficiently, and even to obtain results when using $MineMFS$ directly would fail (*Hypothesis 1: H1*).

Our second claim is that our technique can efficiently obtain a more detailed description of the document collection (*Hypothesis 2: H2*), as we can use looser frequency thresholds. This is easily understood by thinking of an extreme case; if a collection of $|D|$ documents is split into $|D|$ subcollections of size 1 and the minimal frequency is 1, we can obtain the corresponding sets of MFS instantly: each MFS set contains only one sequence of frequency 1, the unique document in the corresponding subcollection. No information is lost, but the content description is probably too large. For example, let us look at the collection of four documents presented in Figure 3.3. By splitting the collection into four subcollections of size 1 and applying $MineMFS$ with unit minimal frequency, each subcollection would be represented by the document it contains, i.e., the document d_1 would be represented by only one 11-gram of frequency 1: "Mary had a little lamb whose fleece was white as snow".

Our third main claim is about the optimal way to form the dis-

jointed subcollections. We conjecture that more consistent subcollections permit to obtain better descriptors (*Hypothesis 3: H3*). The main reason of this train of thought relies on the fact that a collection made of similar documents will contain more interesting maximal frequent sequences than a collection made of dissimilar documents. Again, thinking of extreme cases makes this point easier to see, as a collection where no two documents have a word in common will not contain any frequent sequences, except for the documents themselves (if the frequency threshold is 1).

For example, let us assume that we want to partition the collection of four documents presented in Figure 3.3 into 2 subcollections of 2 documents each, and use a minimal frequency of 2 for extracting MFSs from the subcollections. Only by clustering together the similar documents (d_1, d_4) and (d_2, d_3), will we obtain sequences of words, that is, *phrasal descriptors*. Those descriptors are: "Mary lamb was" for the documents d_1 and d_4, and "Sputnik first satellite launched" for the documents d_2 and d_3. Any other way to partition the collection produces an empty *phrasal description*.

3.2.2 Measuring an MFS-Based Phrasal Description

To confirm or disprove the three hypotheses we just made, we need measures to compare different sets of phrasal descriptors. The quality metrics upon which we want to compare sets of descriptors should be able to evaluate two things:

1. The size of a phrasal text representation.
2. The amount (and density) of information in a phrasal text representation.

In general, the problem of comparing two sets is not an easy one. A large quantity of work in the domains of document clustering and textual classification has proposed measures to compare different ways to partition document sets [SKK00, Seb02]. Unfortunately, we cannot exploit this work to solve our problem, because such techniques rely on the comparison of a given clustering (or classification) to a gold standard. In the general case of textual representation, without aiming at a specific application, there is no obvious way to define a gold standard of the phrasal description of a document collection.

Fortunately, the problem we are facing here is a sub-problem of the above. The sets we need to compare are indeed similar in nature, because they result from various runs of the same extraction technique. For example, a major difficulty in comparing general sequences would be the comparison of long grams to their subgrams. However, in the specific case where all the descriptors are MFS (either of the whole collection or of one of its subcollections), we can simplify the problem by normalizing each descriptor to a set of all its subpairs. This is because the unlimited distance allowed between any two words of an MFS permits to ensure that the assertion "$ABCD$ is an MFS" is equivalent to the assertion: "AB, AC, AD, BC, BD, and CD are frequent bigrams".

Without loss of information, we can thus transform each set of phrasal descriptors into a set of comparable items, the frequent bigrams it contains. Let R_D be the phrasal description of a document collection D, and R_d be the corresponding set of phrases describing a document $d \in D$. We can write the corresponding set of word pairs as $bigrams(R_d)$. For $b \in bigrams(R_d)$, we also define df_b as the document frequency of the bigram b. Finally, we define the random variable X over the set $bigrams(R_d)$:

$$\text{for all } b \in bigrams(R_d): \quad p(X = b) = \frac{df_b}{\sum_{y \in \{ \bigcup\limits_{d \in D} bigrams(R_d) \}} df_y},$$

where $\sum_{y \in \{\bigcup_{d \in D} bigrams(R_d)\}} df_y$ is the total number of bigram occurrences resulting from the phrasal description R_D. It can be thought of as the sample size.

Size of the representation of a document collection. The phrasal representation of a document collection can be seen as a set of associations between descriptive n-grams and documents. We define $|R_D|$ as the size of the phrasal representation R_D in a very intuitive way:

$$|R_D| = \sum_{d \in D} |R_d|.$$

Hence, $|R_D|$ is the summation of the size of the phrasal representation of each document $d \in D$, where the size of the phrasal representation of a document is the number of phrasal descriptors it is associated to. In other words, $|R_D|$ is the number of document-phrase associations in the collection representation R_D.

Equivalent quantity of frequent bigrams in the representation. After we have noticed that every MFS can be normalized to a set of frequent bigrams, it appears natural to measure the quantity of information in the description with the number of bigram-document associations that correspond to the description R_D. This value is $bigram\_size(R_D)$, defined as follows:

$$bigram\_size(R_D) = \sum_{d \in D} |bigrams(R_d)|.$$

Hence, the equivalent quantity of frequent bigrams in $R_D|$ is the summation of the size of the equivalent bigram representation of each document in $d \in D$, where the size of the equivalent bigram representation of a document is the number of *distinct* frequent bigrams it is associated to. In other words, $bigram\_size(R_D)$ is the number of document-bigram associations stemming from the collection representation R_D. Observe that many descriptors may contain identical bigrams and represent the same document. To count the number of bigram-document associations is a way to avoid counting redundant information stemming from the long descriptors.

Density of the description. To measure whether the description is loose or dense, we can use the two preceding measures in a very simple way. By computing the ratio between the number of document-bigram associations corresponding to a document representation and the size of that document representation, we obtain a relative measure of the excess number of document-bigram associations that are avoided when they are replaced by longer n-grams:

$$Density(R_D) = \frac{bigram\_size(R_D)}{|R_D|}.$$

For example, a density value of 1.1 means that the bigram representation of R_D contains 10% more associations than the equivalent representation R_D. The higher $Density(R_D)$, the more storage space we save by using R_D instead of frequent pairs only.

3.2.3 Experiments and Results

In this section, we will detail and implement a set of experiments that permit to verify our initial hypotheses, based on the metrics we previously defined.

Table 3.1: INEX. Serial extraction within circa 17 minutes, and corresponding frequency thresholds using $MineMFS$ directly and $MFS\_MineSweep$ on random partitions of size 2, 3 and 5.

Number of partitions (min,max)	Extraction Time	Bigrams
1 [MineMFS] (800,950)	$18min14s$	1,655
2 (725, 900)	16min27s	2,248
3 (550, 700)	16min54s	3,116
5 (500, 600)	16min07s	2,084

$MFS\_MineSweep$ permits extracting descriptors in situations where $MineMFS$ alone fails *(Hypothesis H1)*.

To verify *H1*, the claim that our technique permits to obtain interesting descriptors when a direct use of $MineMFS$ would fail to do so, we apply both approaches to the INEX document collection (Initiative for the Evaluation of XML retrieval[1]), a 494Mb document collection that contains 12,107 English-written scientific articles from IEEE journals. We ignored the XML structure of the documents, to retain only plain text content. We have used a practical time limit of 17 minutes to extract phrasal descriptors, and used identical desktops for all the experiments, with a 2.80 Ghz processor and 1024Mb of RAM. Results are presented in Table 3.1. Applying $MineMFS$ directly, we tried numerous frequency thresholds and failed to compute a non-empty description in less than 17 minutes. The fastest non-null description came short in 18 minutes and 14 seconds, with minimal and maximal thresholds of 800 and 950, resulting in a quantity of information of 1,655 bigrams.

Note, as a general comment on every experiment involving random partitioning in this chapter, that every value resulting from a random partition into n subcollections is actually the average outcome of 10 distinct iterations of the random partitioning and evaluation process. Using our technique, and splitting the collection randomly into disjoint subcollections, we obtained results in less than 17-minute for partition in 2, 3 and 5 subcollections. The amounts of information should be compared carefully, as the num-

[1]available at http://inex.is.informatik.uni-duisburg.de/2005/

Table 3.2: INEX. Extraction times if the MFS extraction is run in parallel for each subcollection.

Partitions (min,max)	Parallel (Average) Extraction Time
1 [MineMFS] (800,950)	$18min14s$ (18min14s)
2 (725, 900)	8min21s (8min14s)
3 (550, 700)	5min44s (5min38s)
5 (500, 600)	3min19s (3min13s)

bers of descriptors are small. We will present an extensive comparison of varying amounts, sizes, and densities of information in the next subsection, where we will confront our second hypothesis using a smaller document collection.

It is important to observe that the extraction of the set of MFSs is an independent process for each distinct subcollection. A profitable alternative, and an easy one to implement, is to run the extraction of the MFS sets in parallel, on distinct computers. If we use a distinct machine for the extraction of each MFS set, the total running time is the running time of the slowest MFS set extraction, plus the time to split the document collection. Based on a set of desktops with a 2.80 Ghz processor and 1024Mb of RAM, the corresponding running times are shown in Table 3.2. Those results are to be compared to the 17-minute time limit under which no direct use of $MineMFS$ could output a non-null description, and to the earlier amounts of information shown in Table 3.1. Thus, we can observe that with 5 desktops, we can obtain results in only 3 minutes and 19 seconds with $MFS\_MineSweep$, whereas no results could be obtained in less than 18 minutes with $MineMFS$. Hence, *Hypothesis H1* is verified: with $MFS\_MineSweep$, we can obtain a phrasal description in situations where $MineMFS$ fails to do so.

To truly place the different values on equal grounds, time-wise, we ran another experiment in which the frequency thresholds were changed so that they approached the 17 minutes time limit for the *parallel* computation of the MFS sets. The results are presented in Table 3.3. It appeared surprisingly difficult to get close to the time limit, as a small increase in the frequency range would sometimes cause a major rise in the extraction time. We will later describe

Table 3.3: INEX. Parallel extraction within circa 17 minutes, and corresponding frequency thresholds using *MineMFS* directly and *MFS_MineSweep* on random partitions of size 2, 3 and 5.

Partitions (min,max)	Extraction Time (avg)	Bigrams
1 [MineMFS] (800,950)	18*min*14*s* (18min14s)	1,655
2 (500, 900)	13min02s (5min08s)	31,247
3 (400, 700)	10min47s (5min31s)	35,844
5 (400, 600)	10min53s (5min53s)	10,895

this phenomenon as *decreasing average times*. The reason is that a small increase in the frequency range can cause the inclusion of many new terms to be taken into account. This fact is unlikely, but it is possible for the MFS extraction of every document collection. It is thus clear that the more extractions that are run, the higher the risk. Hence, the exposure to this phenomenon is linearly aggravated by the number of subcollections we use, *and* by the number of iterations of the experiments. For example, to calculate the values for a partition in 5 subcollections, we had to run 50 distinct MFS extractions (10 random iterations for each subcollection), which on average took 5 minutes and 53 seconds each, but the average slowest extraction for each experiment iteration took almost twice this amount of time: 10 minutes and 53 seconds.

MFS_MineSweep extracts better, but less compact descriptors *(Hypothesis H2)*.

We have verified *H1* and observed that for a document collection that is large enough, *MineMFS* fails to provide an interesting set of descriptors, whereas *MFS_MineSweep* permits to do so. The claim of our second hypothesis *H2* is that, even when *MineMFS* permits to efficiently extract interesting descriptors, we can extract more information using *MFS_MineSweep*, although we then lose the maximality property, subsequently leading to a less compact description.

To verify this, we experiment with a document collection on which both techniques are producing results efficiently, the 16Mb Reuters-21578 newswire collection [Reu87], which originally con-

Table 3.4: Reuters. Serial extraction within circa 5 minutes, and corresponding frequency thresholds using $MineMFS$ directly and $MFS\_MineSweep$ on random partitions of size 2, 3 and 5, 10, 20, 50 and 100.

Parts (min,max)	Time Serial (Parallel)	Bigrams	Descriptors	Density
MineMFS(85,900)	4m35s (4m35s)	147,000	126,000	1.17
2 (90,1000)	4m58s (2m36s)	412,000	397,000	1.04
3 (60,660)	4m50s (1m41s)	405,000	388,000	1.04
5 (50,500)	4m31s (1m00s)	410,000	394,000	1.04
10 (40,300)	4m03s (0m53s)	181,000	169,000	1.07
20 (25,150)	3m51s (1m08s)	117,000	106,000	1.10
50 (20,75)	3m53s (0m53s)	68,000	63,000	1.08
100 (9,35)	4m28s (0m33s)	96,000	86,000	1.11

tains about 19,000 non-empty documents. To place both techniques on equal grounds, we set a practical time limit of 5 minutes, and try to find minimal and maximal frequency thresholds that permit to maximize the amount of information. We then compare the resulting sizes, amounts and densities of information in Table 3.4. The serial time is the time for partitioning the collection into n subcollections added to the total time needed to extract the MFS set corresponding to each subcollection. The parallel time is the total time needed to do this using a distinct computer for each subcollection in parallel. Hence, the serial and parallel time values for the partition in 10 subcollections mean that the partitioning and the extraction of the 10 MFS sets took 4 minutes and 3 seconds, and that the slowest individual MFS extraction, added to the partitioning time, took 53 seconds. As we mentioned earlier, all the values are averaged over 10 iterations of the experiments. The variance is very small, as for example, the number of bigrams for the 20-partitionings varied between 178,000 and 188,000 for an average of 181,000.

Multiplied initial procedures. We may be surprised to observe that the quantity of information stagnates between the 2-, 3-, and 5-partitions, and that it even decreases for larger partitions. The reason is that the extraction of the MFS set of a docu-

ment collection requires a number of initial procedures, as we have seen in Section 3.1.5, such as computing the frequencies of individual words, pairs, and trigrams, and pruning the too frequent and infrequent ones. If we use a larger partition, we must for example reinitialize the count of each word, pair and trigram for each subcollection. The partition of the collection is another operation whose cost grows with the number of subcollections to be produced. In the end, if the extraction time is held constant, the total cost of initial operations grows with the number of subcollections, and the time left for the actual MFS extraction decreases correspondingly. For smaller partitions, i.e., partitions with a small number of subcollections, the more time-demanding preprocessing phase is compensated by the gain induced by *MFS_MineSweep*. For larger partitions, this gain is not sufficient anymore. Another cause for the decreasing performance when the number of subcollections grows is the fact that the average extraction time decreases.

Decreasing average extraction times. Another observation we can make already is that the difference between the parallel and the average time grows with the number of subcollections. This is easy to understand, as the parallel time is in fact the maximal extraction time over the set of all subcollections. It is, hence, a natural fact of statistics that more subcollections are more likely to include one that is problematic with respect to MFS extraction. As a consequence, we need to tighten the frequency range to keep the extraction time of this problematic subcollection below 5 minutes, implying a shorter extraction time for all the other subcollections in the partition. This problem is also aggravated by the fact that we run 10 iterations of each experiment. To obtain values for the 100-partition actually requires finding a frequency range that permits the extraction of 1,000 MFS sets in less than 5 minutes for each of the 10 iterations. None of the 1,000 extractions can be slower.

To lower the impact of the repeated initial procedures, and to take advantage of the independence of the MFS extraction process between subcollections, we will now have a look at the results we obtained when extracting the MFS set of each different document subcollection in parallel. The results are presented in Table 3.5.

MFS_MineSweep **outperforms** *MineMFS*. The first observation about the results in Table 3.5 is that the number of equivalent bigrams is always much higher for *MFS_MineSweep* than it is for

Table 3.5: Reuters. Parallel extraction within circa 5 minutes, and corresponding frequency thresholds using $MineMFS$ directly and $MFS\_MineSweep$ on random partitions of size 2, 3 and 5, 10, 20, 50 and 100.

Parts (min,max)	Time Parallel (Avg)	Bigrams	Descriptors	Density
MineMFS(85,900)	4m35s (4m35s)	147,000	126,000	1.17
2 (70,1000)	4m32s (4m18s)	836,000	814,000	1.03
3 (40,650)	4m31s (4m17s)	1,168,000	1,143,000	1.02
5 (30,500)	4m13s (2m36s)	1,456,000	1,428,000	1.02
10 (25,300)	4m58s (1m48s)	670,000	655,000	1.02
20 (25,175)	3m34s (1m14s)	352,000	355,000	0.99
50 (15,140)	2m52s (1m03s)	1,579,000	2,544,000	0.62
100 (10,75)	2m59s (0m31s)	1,534,000	2,515,000	0.61

$MineMFS$. Slightly confusing is the fact that this improvement is not steady with respect to the number of partitions, as the results are surprisingly weak for the 10- and 20-partitions (670,000 and 352,000 bigrams), compared to the results obtained with the 5- and 50-partitions (1,456,000 and 1,579,000 bigrams). We claim that this is due to the impact of *decreasing average times*, which we will soon demonstrate with further experiments.

The description is less compact. As we predicted in Hypothesis *H2*, we can see that the density of the phrasal representations is decreasing with the number of subcollections. What we did not expect is that the density ratio goes down to values below 1, meaning that the number of equivalent bigrams is less than the number of phrasal descriptors. This steep density decrease expresses more than the loss of the maximality property. A lower density means that the number of descriptors is growing faster than the number of bigrams. When we split the collection into more disjoint subcollections, this means that more and more of the new descriptors we find are only unknown combinations of bigrams that we already found when we split the collection in less partitions. This sharp decrease in density is in fact an indication that the discriminative power of the phrasal description is peaking, and that further augmentations of the number of partitions will be comparatively less

and less worthwhile.

Decreasing average extraction times. This phenomenon is still present. The average extraction time decreases steadily when the number of subcollections grows. The difficulty is now that, to get results for the 100-subcollection for example, we must extract MFS sets from 1,000 different subcollections in less than 5 minutes each. For many cases, a small variation can cause a drastic increase in processing time, sometimes running for a few days before being aborted. As an illustration, the parallel extraction time of the MFS sets for the 100-partition for the frequency range $10-75$ is well below the 5-minute maximum, with 2 minutes and 59 seconds. When we tried the same experiment with the frequency range $10-80$, no less than 8 iterations (out of 10) exceeded the 5-minute limitation.

To remove the impact of the "decreasing average times", we did one more experiment, in which we found a frequency range for every subcollection individually, such that the corresponding MFS extraction time was always between 4 and 5 minutes. This was achieved with a fairly simple heuristic, interrupting the process and decreasing the frequency range when the extraction was too slow, and increasing the frequency range after too fast an extraction. The process stopped after a number of iterations, and the most difficult cases were dealt with manually. The results are presented in Table 3.6.

The number of bigrams increases steadily with the number of subcollections. These results confirm our previous claims. The hypothesis $H2$ is verified, an increase in the number of subcollections is followed by a more exhaustive document description. In addition, the descriptions are less and less compact as the number of partitions grows.

Now that we have verified *H2*, it only remains to check our third hypothesis, and study the effect of splitting the document collection into homogeneous partitions rather than random ones.

The more homogeneous the subcollections, the better the descriptors *(Hypothesis H3)*.

To support *H3*, we use the same newswire collection and compare the size, amount and density of information obtained when splitting the collection into random and homogeneous subcollec-

Table 3.6: Reuters. Corresponding frequency ranges when every subcollection is computed within 4 and 5 minutes using $MineMFS$ directly and $MFS\_MineSweep$ on random partitions of size 2, 3, 5, 10, 20, 50 and 100.

Partitions (min,max)	Bigrams	Descriptors	Density
1 [MineMFS] (85,900)	147,000	126,000	1.17
2 (60-70, 900-1000)	841,000	819,000	1.03
3 (40, 650-715)	1,223,000	1,197,000	1.02
5 (25-30, 400-600)	1,605,000	1,574,000	1.02
10 (5-28, 72-350)	1,453,000	1,466,000	0.99
20 (10-28, 162-385)	1,643,000	2,555,000	0.64
50 (4-20, 60-208)	2,927,000	7,448,000	0.39
100 (3-45, 27-630)	3,570,000	11,038,000	0.32

tions. In the experiments, we formed homogeneous subcollections with the well-known k-means clustering algorithm (see for example [Wil88, DAM02] for more details). We used the publicly available clustering tool implemented by George Karypis at the University of Minnesota[2]. Algorithm 2 shows a *base* k-means algorithm that assumes the number of desired clusters k to be given and relies on the idea to treat documents as data points, as is usual in the vector space model. The main reason for the choice of k-means is its linear time complexity in the total number of documents, but many other document clustering techniques would be equally appropriate [SKK00].

The quality of the phrasal descriptors resulting from homogeneous document partitions is shown in Table 3.7. In this first experiment, we used the same frequency ranges as in Table 3.5. The results are disappointing, whenever they could be obtained at all. We are facing a critical illustration of the discrepancy between parallel and average running times. For example, for the partition in 5 clusters, the maximal running time was 25 hours and 38 minutes, whereas the average running time of the other 4 clusters was only 6 minutes and 4 seconds. The impact of "decreasing average times"

[2]CLUTO, http://www-users.cs.umn.edu/~karypis/cluto/

Algorithm 2 *Base k-means algorithm.*

1. *Initialization:*
   - *k points are chosen as initial centroids*
   - *Assign each point to the closest centroid*
2. *Iterate:*
   - *Compute the centroid of each cluster*
   - *Assign each point to the closest centroid*
3. *Stop:*
   - *As soon as the centroids are stable*

Table 3.7: Reuters. Extraction and corresponding times, using *MineMFS* directly and *MFS_MineSweep* on homogeneous partitions of size 2, 3 and 5, 10, 20, 50 and 100, and with the same frequency ranges as in Table 3.5. We interrupted the MFS extractions that were not completed after a week.

Clusters(min,max)	Time Parallel (Avg)	Bigrams	Descriptors	Density
MineMFS(85,900)	4m35s (4m35s)	147,000	126,000	1.17
2 (70,1000)	16m22s (11m37s)	476,000	479,000	0.99
3 (40,650)	27m46s (15m18s)	405,000	472,000	0.86
5 (30,500)	25h38m (5h12m)	1,237,000	1,291,000	0.96
10 (25,300)	week+ (N/A)	N/A	N/A	N/A
20 (25,175)	46m48s (5m7s)	115,000	81,000	1.43
50 (15,140)	week+ (N/A)	N/A	N/A	N/A
100 (10,75)	week+ (N/A)	N/A	N/A	N/A

is actually lowered by the removal of random factors, since we need to run each experiment only once, in place of several iterations for random procedures. This lowers the chances to encounter a problematic subcollection. But there are actually two more important factors that increase this risk and cause extraction difficulties.

Homogeneity. It was our intention to obtain homogeneous collections. The difficulty to obtain results in a reasonable time is proof that we succeeded in this respect; since similar documents have been gathered together, there are more similar text fragments appearing in the same subcollections, and hence n-gram frequencies are higher, the number of comparisons to be computed is higher, and so on. In a word, $MineMFS$ has more to extract from each subcollection, and this naturally demands more time. A shift in frequency is further likely to affect more items, making homogeneous collections more vulnerable to the "decreasing average times" phenomenon.

Variance in the size of the subcollections. Another consequence of the k-means clustering is that the size of the subcollections varies widely. In the partition in 100 clusters, the number of documents per collection varied from 40 to 670, whereas in all the 10 iterations of random partitioning, these numbers only varied between 150 and 237. When the number of documents in the collections varies so much, it is obviously not appropriate to use the same frequency range for every subcollection.

To account for this and to remove the impact of the "decreasing average times" phenomenon, we proceeded the same way as we did with random partitions and we used a heuristic taking the size of document collections into account, in such a way that we could calculate the MFS set of every subcollection within 4 and 5 minutes. The corresponding results are shown in Table 3.8.

$MFS\_MineSweep$ **outperforms** $MineMFS$**.** What we had observed with random partitions is confirmed with homogeneous collections. We get a more exhaustive description of the document collection if we use $MFS\_MineSweep$ than if we use $MineMFS$ alone.

To permit an easier direct comparison, the quantities and densities of information obtained with random and homogeneous partitions are presented in Table 3.9.

A more compact description. We can observe that when

Table 3.8: Reuters. Corresponding frequency ranges when every subcollection is computed within 4 and 5 minutes using $MineMFS$ directly and $MFS\_MineSweep$ on homogeneous partitions of size 2, 3, 5, 10, 20, 50 and 100.

Clusters (min,max)	Bigrams	Descriptors	Density
1 [MineMFS] (85,900)	147,000	126,000	1.17
2 (40-130, 660-1569)	554,000	568,000	0.97
3 (7-129, 180-1470)	449,000	498,000	0.90
5 (3-55, 47-1224)	995,000	993,000	1.00
10 (5-22, 58-671)	1,255,000	1,280,000	0.98
20 (3-14, 11-682)	1,767,000	1,904,000	0.93
50 (2-37, 5-289)	2,201,000	2,748,000	0.80
100 (2-28, 7-220)	2,932,000	4,597,000	0.64

Table 3.9: Reuters. Quantities and densities of information when every subcollection is computed within 4 and 5 minutes using $MFS\_MineSweep$ on random and homogeneous partitions of size 2, 3, 5, 10, 20, 50 and 100.

Partitions	Random	Homogeneous
2	841,000 (1.03)	554,000 (0.97)
3	1,223,000 (1.02)	449,000 (0.90)
5	1,605,000 (1.02)	995,000 (1.00)
10	1,453,000 (0.99)	1,255,000 (0.98)
20	1,643,000 (0.64)	1,767,000 (0.93)
50	2,927,000 (0.39)	2,201,000 (0.80)
100	3,570,000 (0.32)	2,932,000 (0.64)

the number of partitions rises, the density of the description resulting from homogeneous subcollections decreases slowly, whereas the steep is much sharper for random partitions. We have already seen that a sharp density decrease expresses a fall in the benefits we get from an augmentation of the number of partitions. The fact that the description densities resulting from homogeneous collections remain nearly stable shows that there is room to improve the discriminative power of phrasal descriptions if we partition the document collection in even more clusters.

This is simple to understand. The descriptors extracted from random subcollections are ones that are present all over the collection. Splitting the collection into more subsets permits finding more of those frequent n-grams, formed by the same frequent words, but we reach a point where we only find combinations of the same frequent words originating from different subcollections. On the other hand, homogeneous subcollections permit gathering similar documents together, excluding non-similar documents. Hence, the frequency range can be adapted to extract the specifics of each subcollection. With homogeneous clusters, producing more subsets permits forming more specific subcollections, hence extracting more specific MFSs. In the homogeneous case, increasing the number of subcollections permits embracing more specificities of the document collections, whereas in the random case, it only permits catching more descriptors of the same kind.

Hence, for partitions of the same size, **phrasal descriptors extracted through homogeneous partitions have a stronger discriminative power** than those extracted through random partitions.

Variance in the size of the subcollections. Although the frequency ranges are now adapted to the size of the corresponding subcollection, the variations in the size of the document collections remain problematic. Assume we have two partitions of the 19,000-document Reuters collection in 2 subcollections. One of the partitions has subcollections of size 9,700 and 9,300. The other partition has subcollections of size 1,000 and 18,000. To extract the MFS set of each subcollection in a time between 4 and 5 minutes seems fair enough in the first case. Intuitively, in the second case it does not seem so. We would wish to spend more time extracting the MFS set of the larger subcollection, as it potentially contains

more descriptors.

To solve this problem is very hard. The first approach would be to share the time upon the size of the subcollection. Of the 10 minutes total for the partition, we could assign $\frac{18}{19} = 94.7\%$ to the 18,000-document subcollection and the rest to the other one. But the first problem with this idea is that the parallel extraction time would not be 5 minutes any more, but 94.7% of 10 minutes, that is, 9 minutes 28 seconds. The other major issue is the relative cost of the initial procedures for the smaller subcollection. Only 32 seconds are left for the computation of its MFS set. If this small subcollection was formed by clustering similar documents together, it implies that it is very homogeneous and that there are potentially numerous phrases to be extracted. It also means that the initial procedures will be especially costly, leaving very short time for the actual MFS extraction. This approach means ignoring a majority of the data in the smaller subcollection.

It may be possible to find a good trade-off between assigning equal computation times to each subcollection, and basing those times on the size of the subcollections. This remains, however, an unsolved problem. Although not optimal in terms of the individual homogeneity of the subcollections, a solution could come from an attempt to produce subcollections of closer sizes. Some inspiration can be found in the work of Hearst, and the scatter/gather technique [HP96].

Clustering is safer. Even though the results are inconclusive and we have not been able to clearly prove or disprove hypothesis *H3*, we maintain our preference for the use of homogeneous partitions. The stronger discriminative power of phrases extracted through homogeneous partitioning is the first strong argument, but there are a few others. As opposed to random partitioning, clustering provides *guarantees*. It is more reliable, because it ensures result. The strength of random partitioning is it gives good results and permits MFS extraction in predictable times. But these facts are only true *on average*. The problem if we use random partitioning is that we should, in fact, run several iterations to protect ourselves from an "unlucky" draw. We mentioned earlier that running several random iterations increases the exposure to factors of difficult extraction. Because the extraction of MFS sets from homogeneous subcollections needs to be done only once, it is less costly in

the end. Another issue with averaging numerous iterations is their meaning in an application framework. We can compute an average of different numbers of descriptors, but it is harder to average a document description. If document d was represented 3 times by $gram_A$, and 1 time by $gram_B$, $gram_C$ and $gram_D$, what should be the average document description of d? And what will remain of the maximality property inherent to MFS?

For all these reasons, we think it is safer to use homogeneous subcollections in the $MFS\_MineSweep$ process, although this could be done more efficiently by solving a number of issues. There is intrinsically more to extract from homogeneous subcollections, and this naturally takes longer. The problem of how to distribute the available computation time among different subcollections is a difficult one, as the running times are difficult to predict. This problem could probably be eased if we were able to form homogeneous subcollections of similar sizes.

Conclusion

We established that $MFS\_MineSweep$ is a good complement of $MineMFS$, as it can be used to extract phrasal document descriptors from document collections of virtually any size, whereas $MineMFS$ alone fails to do so, when the document collections are large enough. $MFS\_MineSweep$ further permits obtaining more descriptive results faster, and even more drastically so when running $MineMFS$ for different subcollections in parallel. Our experiments showed that, for a partition of the document set in a sufficient number of subcollections, the discriminative power of the phrasal descriptors extracted by $MFS\_MineSweep$ is higher when the subcollections are homogeneous.

The main drawback of $MFS\_MineSweep$ is the loss of the maximality property inherent to MFSs. This means that, for the same quantity of information, a collection description originating from $MFS\_MineSweep$ is always less or equally compact to one originating from $MineMFS$.

3.3 Conclusion

We have introduced the concept of a maximal frequent sequence, a compact approach to document description. We presented a number of techniques that permit to extract MFSs from sequential data, before covering their weaknesses, when applied to textual data. An efficient solution was introduced with $MineMFS$, although it still fails to produce descriptors efficiently for too large document collections.

We consequently presented $MFS\_MineSweep$, a technique to obtain a better description efficiently, by running $MineMFS$ on homogeneous partitions of the document collection, and joining the results. We introduced measures of quantity, size and density of information to compare results obtained by lone use of $MineMFS$ to those obtained by its use within the $MFS\_MineSweep$ framework.

The general evaluation of *individual* descriptors remains an open problem, however. In numerous real-life applications, it is crucial to be able to rank or weight phrasal descriptors. Basic approaches, such as using the rough length or frequency of the word sequences appear insufficient. In the following chapter, we will present an advanced technique for calculating the probability of occurrence, document frequency, and general-purpose interestingness of discontiguous sequences of any length.

CHAPTER 4

Direct Evaluation of Non-Contiguous Sequences

The main result of this chapter is to present a novel technique for calculating the probability of occurrence of a *discontinued* sequence of n words (actually, of n sequential items of any data type), that is, the probability that those words occur, and that they occur in a given order, regardless of which and how many other words may occur between them. The technique is mathematically sound, computationally efficient, and it is fully language- and application-independent.

Hitherto, this dissertation has focused on the extraction of word sequences as document descriptors. It is, however, certain that techniques based on frequencies, such as $MineMFS$ will extract a number of uninteresting sequences. Frequent words naturally occur together often, whereas the joint occurrence of infrequent words is usually clearer evidence of a meaningful association. The latter are generally more "interesting".

Since the number of phrasal descriptors is often very high, it is desirable to have means to sort them by their level of interestingness. One main advantage of a ranked list over a set of phrasal descriptors is that it permits the end-user to save time by reading through the most important findings first. This is especially important in real-life applications, where time consumes money and is therefore often limited.

However, to rank a list of phrasal descriptors is not trivial. In this chapter, we will present a new technique to compute the exact

probability of a discontinued sequence of items with a very reasonable computational complexity. It relies on the formalization of word occurrences into a Markov chain model. Numerous techniques of probability and linear algebra theory are exploited to offer an algorithm of competitive computational complexity. The technique is further extended to permit the calculation of the *expected document frequency* of an n-words sequence in an efficient manner. The generality of the approach (it suits not only words, but any type of sequential data) ensures language- and domain-independence.

An application of this result is a **fast and automatic technique to directly evaluate the interestingness of word sequences**. This is done by exploiting statistical techniques, of *hypothesis testing*, to evaluate the interestingness of sequences. The idea of such techniques is to account for the fact that word sequences are bound to happen by chance, and to compare how often a given word sequence should occur (by chance) to how often it truly occurs.

4.1 Introduction

The probability of occurrence of words and phrases is a crucial matter in all domains of information retrieval. All language models rely on such probabilities. However, while the probability of a word is frequently based on counting its total number of occurrences in a document collection (collection frequency), calculating the probability of a phrase is far more complicated. Counting the number of occurrences of a multi-word unit is often intractable, unless restrictions are adopted, as we have seen in Chapters 2 and 3, such as setting a maximal unit size, requiring word adjacency or setting a maximal distance between two words.

The evaluation of lexical cohesion is a difficult problem. Attempts at direct evaluation are rare, simply due to the subjectivity of any human assessment, and to the wide acceptance that we first need to know what we want to do with a lexical unit before being able to decide whether or not it is relevant for that purpose. A common application of research in lexical cohesion is lexicography, where the evaluation is carried out by human experts who simply look at phrases to assess them as good or bad. This process permits

scoring the extraction process with highly subjective measures of precision and recall. However, a linguist interested in the different forms and uses of the auxiliary "to be" will have a different view of what is an interesting phrase than a lexicographer. What a human expert judges as uninteresting may be highly relevant to another.

Hence, most evaluation has been indirect, through question-answering, topic segmentation, text summarization, and passage or document retrieval [Vec05]. To pick the last case, such an evaluation consists in trying to figure out which are the phrases that permit to improve the relevance of the list of documents returned. A weakness of indirect evaluation is that it hardly shows whether an improvement is due to the quality of the phrases, or to the quality of the technique used to exploit them. Moreover, text retrieval collections often have a relatively small number of queries, which means that only a small proportion of the phrasal terms will be used at all. This is a strong argument against the use of text retrieval as an indirect way to evaluate the quality of a phrasal index, initially pointed out by Fox [Fox83].

There is a need to fill the lack of a general purpose direct evaluation technique, one where no subjectivity or knowledge of the domain of application will interfere. Our technique permits exactly that, and the current chapter will be showing how.

The technique we introduce permits to efficiently calculate the exact probability (respectively, the expected document frequency) of a given sequence of n words to occur in this order in a document of size l, (respectively, in a document collection D) with an unlimited number of other words eventually occurring between them.

This work tackles a number of challenges. First, it avoids the computational risk inherent to using a potentially unlimited distance between each two words, while not making those distances rigid (we do see "President John Kennedy" as an occurrence of "President Kennedy"). Achieving language-independence and dealing with document frequencies rather than term frequencies are further specifics of this novel approach.

By comparing observed and expected frequencies, we can estimate the interestingness of a word sequence. That is, the more the actual number of occurrences of a phrase is higher than its expected frequency, the stronger the lexical cohesion of that phrase. This evaluation technique is entirely language-independent, as well

as domain- and application-independent. It permits to efficiently rank a set of candidate multi-word units, based on statistical evidence, without requiring manual assessment of a human expert.

The techniques presented in this paper can be generalized further. The procedure we present for words and documents may indeed similarly be applied to any type of sequential data, e.g., item sequences and transactions.

In the next section, we will introduce the problem, present an approximation of the probability of an n words sequence in a document, and then present our technique in full detail before analyzing its complexity and showing how it outperforms naive approaches. In Section 3, we will show how the probability of occurrence of an n words sequence in a document can be generalized to compute its expected document frequency in a document collection, within another very reasonable computational complexity. Section 4 explains and experiments with the use of statistical testing as an automatic way to rank general-purpose non-contiguous lexical cohesive relations. This paper comes to its conclusion in Section 5.

4.2 The Probability of Discontinued Occurrence of an n-Words Sequence

4.2.1 Problem Definition

Let $A_1, A_2, \ldots, A_n$ be n words, and d a document of length l (i.e., d contains l word occurrences). Each word A_i is assumed to occur independently with probability p_{A_i}. This assumption of independent word occurrences is a common simplification in statistical NLP.

Problem: In d, we want to calculate the probability $P(A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_n, l)$ of the words A_1, A_2, ..., A_n to occur at least once in this order, an unlimited number of interruptions of any size being permitted between each A_i and A_{i+1}, $1 \leq i \leq (n-1)$.

More definitions.

Let D be the document collection, and W the set of all distinct words occurring in D. As the probability p_w of occurrence of a word w, we use its term frequency in the whole document collection,

divided by the total number of word occurrences in the collection. One good reason to prefer term frequency versus, e.g., document frequency, is that in this case, the set of all word probabilities $\{p_w \mid \forall w \in W\}$ is a (finite) probability space. Indeed, we have

$$\sum_{w \in W} p_w = 1,$$

and

$$\forall w \in W : \qquad p_w \geq 0.$$

For convenience, we will simplify the notation of p_{A_i} to p_i, and define $q_i = 1 - p_i$, the probability of non-occurrence of the word A_i.

A running example.

Let there be a hypothetic document collection containing only three different words A, B, and C, each occurring with equal frequency. We want to find the probability that the bigram $A \rightarrow B$ occurs in a document of length 3.

For such a simple example, we can afford an exhaustive manual enumeration. There exist $3^3 = 27$ distinct documents of size 3, each occurring with equal probability $\frac{1}{27}$. These documents are:
$\{AAA, \boxed{\text{AAB}}, AAC, \boxed{\text{ABA}}, \boxed{\text{ABB}}, \boxed{\text{ABC}}, ACA, \boxed{\text{ACB}}, ACC,$
$BAA, \boxed{\text{BAB}}, BAC, BBA, BBB, BBC, BCA, BCB, BCC,$
$CAA, \boxed{\text{CAB}}, CAC, CBA, CBB, CBC, CCA, CCB, CCC\}$
The seven framed documents contain the n-gram AB. Thus, we have $p(A \rightarrow B, 3) = \frac{7}{27}$.

4.2.2 A Decent Over-Estimation in the General Case

We can attempt to enumerate the number of occurrences of $A_1 \rightarrow \cdots \rightarrow A_n$ in a document of size l, by separately counting the number of ways to form the $(n-1)$-gram $A_2 \rightarrow \cdots \rightarrow A_n$, given the l possible positions of A_1. For each of these possibilities, we can then separately count the number of ways to form the $(n-2)$-gram $A_3 \rightarrow \cdots \rightarrow A_n$, given the various possible positions of A_2 following that of A_1. And so on until we need to find the number of ways to form the 1-gram A_n, given the various possibilities left for placing A_{n-1}.

In other words, we want to sum up the number of ways to form the n-gram, knowing that A_1 occurs in position pos_{A_1}, with pos_{A_1} varying between 1 and $(l-n+1)$ (the n-gram can only be formed if $pos_{A_1} \leq (l-n+1)$, because $A_2, \ldots, A_n$ must still occur). For each possible position of A_1, we sum up the number of ways to form the n-gram, knowing that A_2 occurs in position pos_{A_2}, with pos_{A_2} varying between $(pos_{A_1}+1)$ and $(l-n+2)$. And so on, until the summation of the number of ways to form the n-gram for each possible position of A_{n-1}, knowing that A_n occurs in position pos_{A_n}, with pos_{A_n} varying between $(pos_{A_{n-1}}+1)$ and l.

This enumeration leads to n nested sums of binomial coefficients:

$$\sum_{pos_{A_1}=1}^{l-n+1} \left(\sum_{pos_{A_2}=pos_{A_1}+1}^{l-n+2} \left(\cdots \sum_{pos_{A_n}=pos_{A_{n-1}}+1}^{l} \binom{l-pos_{A_n}}{0} \right) \right), \tag{4.1}$$

where each pos_{A_i}, $1 \leq i \leq n$, denotes the position of occurrence of A_i.

The following can be proved easily by induction:

$$\sum_{i=k}^{n} \binom{i}{k} = \binom{n+1}{k+1},$$

and we can use it to simplify formula (4.1) by observing that:

$$\begin{aligned} \sum_{pos_{A_i}=pos_{A_{i-1}}+1}^{l-n+i} \binom{l-pos_{A_i}}{n-i} &= \sum_{pos_{A_i}=n-i}^{l-pos_{A_{i-1}}-1} \binom{pos_{A_i}}{n-i} \\ &= \binom{l-pos_{A_{i-1}}}{n-i+1}. \end{aligned}$$

Therefore, leaving further technical details to the reader, the previous nested summation (4.1) interestingly simplifies to $\binom{l}{n}$, which permits to obtain the following result:

$$enum\_overestimate(A_1 \rightarrow \cdots \rightarrow A_n, l) = \binom{l}{n} \cdot \prod_{i=1}^{n} p_i,$$

where $\binom{l}{n}$ is the number of ways to form the n-gram, and $\prod_{i=1}^{n} p_i$ the probability of conjoint occurrence of the words $A_1, \ldots, A_n$ (since we

assumed that the probability of occurrence of a word in one position is independent of which words occur in other positions).

The big flaw of this result, and the reason why it is an approximation only, is that some of the ways to form the n-gram are obviously overlapping. Whenever we separate the alternative ways to form the n-gram, knowing that A_i occurs in position pos_{A_i}, with $1 \leq i \leq n$ and $(pos_{A_{i-1}} + 1) \leq pos_{A_i} \leq (l - n + i)$, we do ignore the fact that A_i may also occur before position pos_{A_i}. In this case, we find and add different ways to form the same occurrence of the n-gram. We do enumerate each possible case of occurrence of the n-gram, but we count some of them more than once, since it is actually *the ways* to form the n-gram that are counted.

Running Example. This is better seen by returning to the running example presented in subsection 4.2.1. As described above, the upper-estimate of the probability of the bigram $A \rightarrow B$, based on the enumeration of the ways to form it in a document of size 3 is: $(\frac{1}{3})^2\binom{3}{2} = \frac{9}{27}$, while the actual probability of $A \rightarrow B$ is $\frac{7}{27}$. This stems from the fact that in the document AAB (respectively ABB), there exist two ways to form the bigram $A \rightarrow B$, using the two occurrences of A (respectively B). Hence, out of the 27 possible equiprobable documents, 9 ways to form the bigram $A \rightarrow B$ are found in the 7 documents that contain it.

With longer documents, the loss of precision due to those cases can be considerable. Still assuming we are interested in the bigram $A \rightarrow B$, we will count one extra occurrence for every document that matches *A*B*B*, where * is used as a wildcard. Similarly, 8 ways to form $A \rightarrow B$ are found in each document matching *A*A*B*B*B*B*.

4.2.3 Exact Probability of a Discontiguous Word Sequence

With a slightly different approach, we can actually reach the exact result. The previous technique exposed *overlapping* ways to form a word sequence. That is why the result was only an overestimate of the desired probability.

In this section, we will present a way to categorize the different sets of documents of size l in which the n-gram $A_1 \rightarrow \cdots \rightarrow A_n$ occurs, with the property that all the sets are disjoint and that no case of occurrence of the n-gram is forgotten. This ensures that we

can calculate $p(A_1 \rightarrow \cdots \rightarrow A_n, l)$ by summing up the probabilities of each set of documents where $A_1 \rightarrow \cdots \rightarrow A_n$ occurs.

A Disjoint Categorization of Successful Documents.

We can split the *successful* documents (those in which the n-gram occurs) of size l, depending on the position from which a successful outcome is guaranteed. For example, and for $l \geq n$, the documents of size l for which success is guaranteed as soon as from position n onwards can be represented by the set of documents E_0:

$$E_0 = \{A_1 A_2 \ldots A_n W^{l-n}\},$$

where as defined earlier W is the set of all words in the document collection, and using regular expression notation, W^{l-n} stands for a concatenation of any $(l-n)$ words of W. Because each word is assumed to occur independently of the others, the probability of a given document is a conjunction of independent events, and therefore it equals the multiplication of the probability of all the words in the document. The probability of the set of documents E_0 is the probability of occurrence of A_1, $A_2 \ldots$, and A_n once, plus $(l-n)$ times any word of W (with probability 1). Therefore,

$$p(E_0) = p_1 \cdot p_2 \ldots p_n \cdot 1^{l-n} = \prod_{i=1}^{n} p_i.$$

Similarly, the documents in which the occurrence of the n-gram is guaranteed as soon as the $(n+1)$-th word can be represented by the set E_1 where, for $1 \leq k \leq n$, $i_k \geq 0$:

$$E_1 = \{\bar{A_1}^{i_1} A_1 \bar{A_2}^{i_2} A_2 \ldots \bar{A_n}^{i_n} A_n W^{l-n-1} \mid \sum_{k=1}^{n} i_k = 1\},$$

where $\bar{A_k}$, $1 \leq k \leq n$, represents any word but A_k. In other words, E_1 is the set of all documents where a total number of 1 word is inserted before each word of the n-gram. The probability of this set of documents is:

$$\begin{aligned} p(E_1) &= (q_1 p_1 p_2 \ldots p_n 1^{l-n-1}) + (p_1 q_2 p_2 \ldots p_n 1^{l-n-1}) + \\ &\quad \cdots + (p_1 p_2 \ldots p_{n-1} q_n p_n 1^{l-n-1}) \\ &= \prod_{i=1}^{n} p_i \sum_{k=1}^{n} q_k. \end{aligned}$$

We can proceed similarly for the following positions after which a successful outcome is guaranteed. Finally, the same idea provides an expression for the set of documents for which the occurrence of the n-gram was not complete before the word in position l (and therefore the last word of the document is A_n):

$$E_{l-n} = \{\bar{A_1}^{i_1} A_1 \bar{A_2}^{i_2} A_2 \dots \bar{A_n}^{i_n} A_n \mid \sum_{k=1}^{n} i_k = (l-n)\}.$$

The set E_{l-n} contains all the possibilities to disseminate exactly $(l-n)$ other words before the words of the n-gram. Its probability of occurrence is:

$$\begin{aligned}
p(E_{l-n}) &= p\left(\left\{\bar{A_1}^{i_1} A_1 \dots \bar{A_n}^{i_n} A_n \mid \sum_{k=1}^{n} i_k = (l-n)\right\}\right) \\
&= p_n \sum_{i_n=0}^{l-n} q_n^{i_n} p\left(\left\{\bar{A_1}^{i_1} A_1 \dots \bar{A_{n-1}}^{i_{n-1}} A_{n-1} \mid \sum_{k=1}^{n-1} i_k = (l-n-i_n)\right\}\right) \\
&= p_n p_{n-1} \sum_{i_n=0}^{l-n} \sum_{i_{n-1}=0}^{l-n-i_n} q_n^{i_n} q_{n-1}^{i_{n-1}} \\
&\qquad p\left(\left\{\bar{A_1}^{i_1} A_1 \dots \bar{A_{n-2}}^{i_{n-2}} A_{n-2} \mid \sum_{k=1}^{n-2} i_k = (l-n-i_n-i_{n-1})\right\}\right) \\
&= \dots \\
&= \prod_{i=2}^{n} p_i \sum_{i_n=0}^{l-n} \cdots \sum_{i_2=0}^{l-n-(i_n+\dots+i_3)} q_n^{i_n} \dots q_2^{i_2} \\
&\qquad p\left(\left\{\bar{A_1}^{i_1} A_1 \mid i_1 = l-n-(i_n+\dots+i_2)\right\}\right) \\
&= \prod_{i=1}^{n} p_i \sum_{i_n=0}^{l-n} \cdots \sum_{i_2=0}^{l-n-(i_n+\dots+i_3)} q_n^{i_n} \dots q_2^{i_2} q_1^{l-n-(i_n+\dots+i_2)}.
\end{aligned}$$

In general, for $0 \leq k \leq l-n$, we can write:

$$p(E_k) = \prod_{i=1}^{n} p_i \sum_{i_n=0}^{k} \cdots \sum_{i_2=0}^{k-(i_n+\dots+i_3)} q_1^{k-\sum_{j=2}^{n} i_j} q_2^{i_2} \dots q_n^{i_n}.$$

The precise formula.

It is clear that the sets E_k, for $0 \leq k \leq (l-n)$, are all disjoint, because in any document, the presence of the n-gram is ensured

from only one position onwards. It is also evident that in any document of size l containing the n-gram, its occurrence will be ensured between the n-th and l-th position. Therefore the sets E_k are mutually exclusive, for $0 \leq k \leq (l-n)$, and their union contains all the documents of size l where $A_1 \rightarrow \cdots \rightarrow A_n$ occurs. Consequently,

$$\begin{aligned} p(A_1 \rightarrow \cdots \rightarrow A_n, l) &= \sum_{k=0}^{l-n} p(E_k) \\ &= \prod_{i=1}^{n} p_i \sum_{k=0}^{l-n} \left(\sum_{i_n=0}^{k} \cdots \sum_{i_2=0}^{k-(i_n+\cdots+i_3)} q_1^{k-\sum_{j=2}^{n} i_j} q_2^{i_2} \ldots q_n^{i_n} \right) \\ &= \prod_{i=1}^{n} p_i \sum_{i_n=0}^{l-n} \cdots \sum_{i_2=0}^{l-n-(i_n+\cdots+i_3)} \sum_{i_1=0}^{l-n-(i_n+\cdots+i_2)} q_1^{i_1} q_2^{i_2} \ldots q_n^{i_n}. \end{aligned}$$

So finally, the formula of the probability of occurrence of a discontiguous sequence of length n in a document of length l is:

$$p(A_1 \rightarrow \cdots \rightarrow A_n, l) = \prod_{i=1}^{n} p_i \sum_{i_n=0}^{l-n} \cdots \sum_{i_1=0}^{l-n-(i_n+\cdots+i_2)} q_1^{i_1} q_2^{i_2} \ldots q_n^{i_n}. \tag{4.2}$$

Running Example. For better comprehension, let us return to the running example:

$$\begin{aligned} p(A \rightarrow B, 3) &= p_a p_b \sum_{i_b=0}^{1} \sum_{i_a=0}^{1-i_b} q_a^{i_a} q_b^{i_b} \\ &= p_a p_b \left(\sum_{i_a=0}^{1} q_a^{i_a} + \sum_{i_a=0}^{0} q_a^{i_a} q_b \right) \\ &= p_a p_b \left(1 + q_a + q_b\right) \\ &= \frac{1}{3} \times \frac{1}{3} \times \left(1 + \frac{2}{3} + \frac{2}{3}\right) \\ &= \frac{7}{27}. \end{aligned}$$

We indeed find the exact result. But we will now see that the direct calculation of Formula 4.2 is is not satisfying in practice because of an exponential computational complexity.

Computational Complexity.

Let us observe the steps involved in the computation of $p(E_k)$, $0 \leq k \leq l - n$. To calculate this probability consists in multiplying n values (the p_i's) by a summation of summations. The total number of terms resulting from these nested summations equals the total number of ways to insert k terms in n different positions: n^k. Thus, $p(E_k)$ is the result of multiplying n values by a summation of n^k distinct terms, each individually calculated by k multiplications. Hence, the gross number of operations to calculate $p(A_1 \to \cdots \to A_n, l)$ with Formula 4.2 is:

$$n \sum_{k=0}^{l-n} k n^k.$$

Therefore, the order of complexity of the direct computation of Formula 4.2 is $O(l n^{l-n})$. Consequently, this formula is hardly usable at all, except for extremely short documents and length-restricted n-grams.

4.2.4 Efficient Computation through a Markov Chain Formalization

We found a way to calculate the probability of discontiguous occurrence of an n-words sequence in a document of size l. However, its computational complexity cuts clear any hope to use the result in practice. The following approach permits to reach the exact result with a far better complexity.

An Absorbing Markov Chain.

Another interesting way to formalize the problem is to consider it as a sequence of l trials whose outcomes are $X_1, X_2, \ldots, X_l$. Let each of these outcomes belong to the set $\{0, 1, \ldots, n\}$, where the outcome i signifies that the i-gram $A_1 \to A_2 \to \cdots \to A_i$ has already occurred. This sequence of trials verifies the following two properties:

(i) All the outcomes $X_1, X_2, \ldots, X_l$ belong to a finite set of outcomes $\{0, 1, \ldots, n\}$ called the *state space* of the system. If i

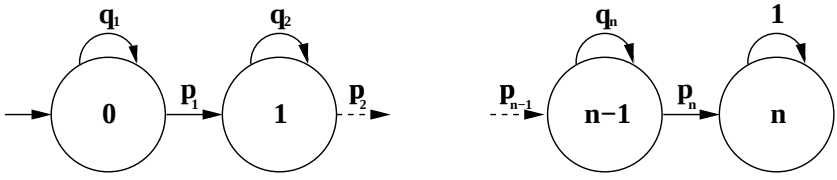

Figure 4.1: The state-transition diagram of the Markov Chain M.

is the outcome of the m-th trial ($X_m = i$), then we say that the system is in state i at the m-th step. In other words, the i-gram $A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i$ has been observed after the m-th word of the document.

(ii) The second property is called the *Markov property*: the outcome of each trial depends at most upon the outcome of the immediately preceding trial, and not upon any other previous outcome. In other words, *the future is independent of the past, given the present*. This is verified indeed; if we know that we have seen $A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i$, we only need the probability of A_{i+1} to determine the probability that we will see more of the desired n-gram during the next trial.

These two properties are sufficient to call the defined stochastic process a (finite) *Markov chain*. The problem can thus be represented by an $(n+1)$-states Markov chain M (see Figure 4.1). The state space of the system is $\{0, 1, \ldots, n\}$ where each state, numbered from 0 to n tells how much of the n-gram has already been observed. Presence in state i means that the sequence $A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i$ has been observed.Therefore, $A_{i+1} \rightarrow \cdots \rightarrow A_n$ remains to be seen, and the following expected word is A_{i+1}. It will be the next word with probability p_{i+1}, in which case a state transition will occur from i to $(i+1)$. A_{i+1} will not be the following word with probability q_{i+1}, in which case we will remain in state i. Whenever we reach state n, we can denote the experience a success: the whole n-gram has been observed. The only outgoing transition from state n leads to itself with associated probability 1 (such a state is said to be *absorbing*).

Stochastic Transition Matrix (in general).

Another way to represent this Markov chain is to write its transition matrix.

For a general finite Markov chain, let $p_{i,j}$ denote the transition probability from state i to state j for $1 \leq i, j \leq n$. The (one-step) stochastic transition matrix is:

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ & & \cdots & \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} \end{pmatrix}.$$

Theorem 4.1 *[Fel68] Let P be the transition matrix of a Markov chain process. Then the m-step transition matrix is equal to the m-th power of P. Furthermore, the entry $p_{i,j}(m)$ in P^m is the probability of stepping from state i to state j in exactly m transitions.*

Our stochastic transition matrix of interest.

For the Markov chain M defined above, the corresponding stochastic transition matrix is the following $(n+1) \times (n+1)$ square matrix:

$$M = \begin{array}{c} \text{states} \\ 0 \\ 1 \\ \\ \\ \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & \ldots & n-1 & n \end{array} \\ \begin{pmatrix} q_1 & p_1 & 0 & \ldots & \ldots & 0 \\ 0 & q_2 & p_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & q_n & p_n \\ 0 & \ldots & \ldots & \ldots & 0 & 1 \end{pmatrix} \end{array}.$$

Therefore, the probability of the n-gram $A_1 \to A_2 \to \cdots \to A_n$ to occur in a document of size l is the probability of stepping from state 0 to state n in exactly l transitions. Following Theorem 4.1, this value resides at the intersection of the first row and the last column of the matrix M^l:

$$M^l = \begin{pmatrix} m_{1,1}(l) & m_{1,2}(l) & \ldots & \boxed{m_{1,n+1}(l)} \\ m_{2,1}(l) & m_{2,2}(l) & \ldots & m_{2,n+1}(l) \\ & & \cdots & \\ m_{n+1,1}(l) & m_{n+1,2}(l) & \ldots & m_{n+1,n+1}(l) \end{pmatrix}.$$

Thus, the result we are aiming at can simply be obtained by raising the matrix M to the power of l, and looking at the value in the upper-right corner. In terms of computational complexity, however, one must note that to multiply two $(n+1) \times (n+1)$ square matrices, we need to compute $(n+1)$ multiplications and n additions to calculate each of the $(n+1)^2$ values composing the resulting matrix. To raise a matrix to the power l means to repeat this operation $l-1$ times. The resulting time complexity is then $O(ln^3)$.

One may object that there exist more time-efficient algorithms for matrix multiplication. The lowest exponent currently known is by Coppersmith and Winograd: $O(n^{2.376})$ [CW87]. This result follows up work by Strassen [Str69], who first beat the naive approach with an $O(n^{2.807})$ algorithm. These results are achieved by studying how matrix multiplication depends on bilinear and trilinear combinations of factors. The strong drawback of such techniques is the presence of a constant so large that it removes the benefits of the lower exponent for all practical sizes of matrices [HJ94]. For Strassen's algorithm, this factor is about 4.537 asymptotically. For our purpose, the use of such an algorithm is typically more costly than to use the naive $O(n^3)$ matrix multiplication.

Linear algebra techniques, and a careful exploitation of the specificities of the stochastic matrix M will, however, permit to perform a few transformations that will drastically reduce the computational complexity of M^l over the use of any matrix multiplication algorithm (it has been proved that the complexity of matrix multiplication cannot possibly be lower than $O(n^2)$).

The Jordan normal form.

Definition: A *Jordan block* J_λ is a square matrix whose elements are zero except for those on the principal diagonal, which are equal to λ, and those on the first superdiagonal, which are equal to unity. Thus:

$$J_\lambda = \begin{pmatrix} \lambda & 1 & & 0 \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda \end{pmatrix}.$$

Theorem 4.2 *(Jordan normal form) [ND77] If A is a general square matrix, then there exists an invertible matrix S such that*

$$J = S^{-1}AS = \begin{pmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_k \end{pmatrix},$$

where the J_i are $n_i \times n_i$ Jordan blocks. The same eigenvalues may occur in different blocks, but the number of distinct blocks corresponding to a given eigenvalue is equal to the number of eigenvectors corresponding to that eigenvalue and forming an independent set. The number k and the set of numbers $n_1, \ldots, n_k$ are uniquely determined by A.

In the following subsection we will show that M is such that there exists only one block for each eigenvalue.

Uniqueness of the Jordan block corresponding to any given eigenvalue of M.

Theorem 4.3 *For the matrix M, no two eigenvectors corresponding to the same eigenvalue can be linearly independent. Following theorem 4.2, this implies that there exists a one-one mapping between Jordan blocks and eigenvalues.*

Proof. Because M is triangular, its characteristic polynomial is the product of the diagonals of $(\lambda I_{n+1} - M)$: $f(\lambda) = (\lambda - q_1)(\lambda - q_2)\ldots(\lambda - q_n)(\lambda - 1)$. The eigenvalues of M are the solutions of the equation $f(\lambda) = 0$. Therefore, they are the distinct q_i's, and 1.

Now let us show that whatever the order of multiplicity of such an eigenvalue (how many times it occurs in the set $\{q_1, \ldots, q_n, 1\}$), it has only one associated eigenvector. The eigenvectors associated to a given eigenvalue e are defined as the non-null solutions

of the equation $M \cdot V = e \cdot V$. If we write the coordinates of V as $[v_1, v_2, \ldots, v_{n+1}]$, we can observe that $M \cdot V = e \cdot V$ results in a system of $(n+1)$ equations, where, for $1 \leq j \leq n$, the j-th equation permits to express v_{j+1} in terms of v_j, and therefore in terms of v_1. That is,

$$\text{for } 1 \leq j \leq n: \; v_{j+1} = \frac{e - q_j}{p_j} v_j = \frac{(e - q_j) \ldots (e - q_1)}{p_j \ldots p_1} v_1.$$

In general (for all the q_i's), v_1 can be chosen freely to have any non-null value. This choice will uniquely determine all the values of V.

Since the general form of the eigenvectors corresponding to any eigenvalue of M is $V = [v_1, v_2, \ldots, v_{n+1}]$, where all the values can be determined uniquely by the free choice of v_1, it is clear that no two such eigenvectors can be linearly independent. Hence, one and only one eigenvector corresponds to each eigenvalue of M. □

Following theorem 4.2, this means that there is a single Jordan block for each eigenvalue of M, whose size equals the order of algebraic multiplicity of the eigenvalue, that is, its number of occurrences in the principal diagonal of M. In other words, there is a distinct Jordan block for every distinct q_i (and its size equals the number of occurrences of q_i in the main diagonal of M), plus a block of size 1 for the eigenvalue 1.

Therefore we can write:

$$J = S^{-1}MS = \begin{pmatrix} \boxed{J_{e_1}} & & & 0 \\ & \boxed{J_{e_2}} & & \\ & & \ddots & \\ 0 & & & \boxed{J_{e_q}} \end{pmatrix},$$

where the J_{e_i} are $n_i \times n_i$ Jordan blocks, corresponding to the distinct eigenvalues of M. Following the general properties of the Jordan normal form, we have:

$$J^l = \begin{pmatrix} \boxed{J^l_{e_1}} & & & 0 \\ & \boxed{J^l_{e_2}} & & \\ & & \ddots & \\ 0 & & & \boxed{J^l_{e_q}} \end{pmatrix}.$$

Also,

$$\begin{aligned} M^l &= \left(SJS^{-1}\right)^l \\ &= \overbrace{\left(SJS^{-1}\right)\cdot\left(SJS^{-1}\right)\ldots\left(SJS^{-1}\right)}^{l \text{ times}} \\ &= S\cdot J\cdot\overbrace{\left(S^{-1}\cdot S\right)\cdot J\cdot\left(S^{-1}\cdot S\right)\cdot J\ldots\left(S^{-1}\cdot S\right)\cdot J}^{(l-1) \text{ times}}\cdot S^{-1} \\ &= S\cdot\overbrace{J\cdot J\ldots J}^{l \text{ times}}\cdot S^{-1} \\ &= S\cdot J^l\cdot S^{-1}. \end{aligned}$$

Therefore, by multiplying the first row of S by J^l (i.e., by raising each Jordan block to the power l), and multiplying the resulting vector by the last column of S^{-1}, we do obtain the upper right value of M^l, that is, the probability of the n-gram $(A_1 \to \cdots \to A_n)$ to appear in a document of size l.

Calculating powers of a Jordan block.

As mentioned above, to raise J to the power l, we can simply write a direct sum of the Jordan blocks raised to the power l. In this section, we will show how to compute $J_{e_i}^l$ for a Jordan block J_{e_i}.

Let us define D_{e_i} and N_{e_i} such that $J_{e_i} = D_{e_i} + N_{e_i}$, where D_{e_i} contains only the principal diagonal of J_{e_i}, and N_{e_i} only its first superdiagonal. That is,

$$D_{e_i} = e_i I_{n_i} = \begin{pmatrix} e_i & & & 0 \\ & e_i & & \\ & & \ddots & \\ 0 & & & e_i \end{pmatrix},$$

and

$$N_{e_i} = \begin{pmatrix} 0 & 1 & & 0 \\ & & \ddots & \\ & & & 1 \\ 0 & & & 0 \end{pmatrix}.$$

Observing that $N_{e_i} D_{e_i} = D_{e_i} N_{e_i}$, we can use the binomial theorem:

$$J_{e_i}^l = (D_{e_i} + N_{e_i})^l = \sum_{k=0}^{l} \binom{l}{k} N_{e_i}^k D_{e_i}^{l-k}$$

Because N_{e_i} is nilpotent ($N_{e_i}^k = 0, \forall k \geq n_i$), we can shorten the summation to:

$$J_{e_i}^l = (D_{e_i} + N_{e_i})^l = \sum_{k=0}^{n_i-1} \binom{l}{k} N_{e_i}^k D_{e_i}^{l-k}$$

Hence, to calculate $J_{e_i}^l$, one can compute the powers of D_{e_i} and N_{e_i} from 0 to l, which is a fairly simple task. The power of a diagonal matrix is easy to compute, as it is another diagonal matrix where each term of the original matrix is raised to the same power as the matrix. $D_{e_i}^j$ is thus identical to D_{e_i}, except that the main diagonal is filled with the value e_i^j instead of e_i. Hence, we have:

$$D_{e_i}^k = \begin{pmatrix} e_i^k & & & 0 \\ & e_i^k & & \\ & & \ddots & \\ 0 & & & e_i^k \end{pmatrix}.$$

To compute $N_{e_i}^k$ is even simpler. Each multiplication of a power of N_{e_i} by N_{e_i} results in shifting the non-null diagonal one row upwards (the values on the first row are lost, and those on the last row are 0's).

The result of $N_{e_i}^k D_{e_i}^j$ resembles $N_{e_i}^j$, except that the ones on the only non-null diagonal (the j-th superdiagonal) are replaced by the value of the main diagonal of $D_{e_i}^j$, that is, e_i^j. Therefore, we have:

$$N_{e_i}^k D_{e_i}^{l-k} = \begin{pmatrix} 0 & e_i^{l-k} & & 0 \\ & & \ddots & \\ & & & e_i^{l-k} \\ 0 & & & 0 \end{pmatrix}.$$

Since each value of k corresponds to a distinct diagonal, the

summation $\sum_{k=0}^{l} \binom{l}{k} N_{e_i}^k D_{e_i}^{l-k}$ is easily written as:

$$J_{e_i}^l = \sum_{k=0}^{l} \binom{l}{k} N_{e_i}^k D_{e_i}^{l-k}$$

$$= \begin{pmatrix} \binom{l}{0} \cdot e_i^l & \dots & \binom{l}{k} \cdot e_i^{l-k} & \dots & \binom{l}{n_i-1} \cdot e_i^{l-n_i+1} \\ & \ddots & & \ddots & \vdots \\ & & \binom{l}{0} \cdot e_i^l & & \binom{l}{k} \cdot e_i^{l-k} \\ & & & \ddots & \vdots \\ 0 & & & & \binom{l}{0} \cdot e_i^l \end{pmatrix}.$$

Conclusion.

The probability of the n-gram $A_1 \to \cdots \to A_n$ in a document of size l can be obtained as the upper-right value in the matrix M^l such that:

$$M^l = SJ^lS^{-1}$$

$$= S \begin{pmatrix} \boxed{J_{e_1}^l} & & & 0 \\ & \boxed{J_{e_2}^l} & & \\ & & \ddots & \\ 0 & & & \boxed{J_{e_q}^l} \end{pmatrix} S^{-1},$$

where the $J_{e_i}^l$ blocks are as described above, while S and S^{-1} are obtained through the Jordan Normal Form theorem (Theorem 4.2). We actually only need the first row of S and the last column of S^{-1}, as we are not interested in the whole matrix M^l but only in its upper-right value.

In the next subsection we will calculate the worst case time complexity of the technique that we just presented. Before that, let us return to the running example presented in subsection 4.2.1.

Running Example.

The state-transition diagram of the Markov Chain corresponding to the bigram $A \to B$ has only three states (see Figure 4.2). The

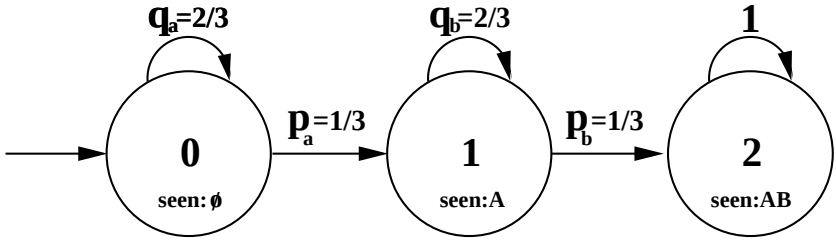

Figure 4.2: The state-transition diagram of the Markov Chain corresponding to our running example.

corresponding transition matrix is:

$$M_{re} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix}.$$

Following Theorem 4.2 on the Jordan normal form, there exists an invertible matrix S_{re} such that

$$J_{re} = S_{re}^{-1} M_{re} S_{re} = \begin{pmatrix} \boxed{J_{\frac{2}{3}}} & 0 \\ 0 & \boxed{J_1} \end{pmatrix},$$

where J_1 is a block of size 1, and $J_{\frac{2}{3}}$ a block of size 2 since $q_a = q_b = \frac{2}{3}$. We can actually write J_{re} as:

$$J_{re} = \begin{pmatrix} \frac{2}{3} & 1 & 0 \\ 0 & \frac{2}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Since we seek the probability of the bigram $A \rightarrow B$ in a document of size 3, we need to calculate J_{re}^3:

$$\begin{aligned} J_{re}^3 &= \begin{pmatrix} \binom{3}{0}(\frac{2}{3})^3 & \binom{3}{1}(\frac{2}{3})^2 & 0 \\ 0 & \binom{3}{0}(\frac{2}{3})^3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{8}{27} & \frac{4}{3} & 0 \\ 0 & \frac{8}{27} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

In the next subsection, we will give further details as to the practical computation of S_{re} and the last column of its inverse S_{re}^{-1}.

For now, let us simply assume they were calculated, and we can thus obtain the probability of the bigram $A \to B$ in a document of length 3 as:

$$\begin{aligned} P(A \to B, 3) &= \overbrace{(1\ 0\ 1)}^{\text{first row of } S} \begin{pmatrix} \frac{8}{27} & \frac{4}{3} & 0 \\ 0 & \frac{8}{27} & 0 \\ 0 & 0 & 1 \end{pmatrix} \overbrace{\begin{pmatrix} -1 \\ -\frac{1}{3} \\ 1 \end{pmatrix}}^{\text{last column of } S^{-1}} \\ &= \begin{pmatrix} \frac{8}{27} & \frac{4}{3} & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -\frac{1}{3} \\ 1 \end{pmatrix} \\ &= \frac{7}{27}. \end{aligned}$$

Our technique indeed obtains the right result. But how efficiently is it obtained? The purpose of the following subsection is to answer this question.

4.2.5 Algorithmic Complexity

The process of calculating the probability of occurrence of an n-gram in a document of size l consists of two main phases: calculating J^l, and computing the transformation matrix S and its inverse S^{-1}.

Below, we will study the worst-case time complexity, but it is interesting to observe that in practice, for a corpus big enough, the number of words equally-weighed should be small. This is especially true since, following Zipf's law, infrequent words are most likely to have equal weights, and they precisely are often pruned during preprocessing.

The following complexity analysis might be easier to follow, if studied together with the general formulas of M^l and the Jordan blocks as presented in the conclusion of Section 4.2.4, on page 95.

Time complexity of the J^l calculation.

Observing that each block J_i^l contains exactly n_i distinct values, we can see that J^l contains $\sum_{1 \le k \le q} n_k = n + 1$ distinct values. Those $(n + 1)$ values are $(n + 1)$ multiplications of a binomial coefficient by the power of an eigenvalue.

The computation of the powers between 0 and l of each eigenvalue is evidently achieved in $O(lq)$, because each of the q distinct eigenvalues needs to be multiplied by itself l times.

For every Jordan block J_i^l, the binomial coefficients to be computed are: $\binom{l}{0}, \binom{l}{1}, \ldots, \binom{l}{n_i-1}$. For the whole matrix J^l, we thus need to calculate $\binom{l}{k}$ where $0 \leq k \leq max_{block}$ and $\max_{block} = \max_{i=1}^{q} n_i$. Observing that $\binom{l}{j+1} = \binom{l}{j}\frac{l-j}{j+1}$, and thus, that $\binom{l}{j+1}$ can be computed from $\binom{l}{j}$ in a constant number of operations, we see that the set $\{\binom{l}{k} \mid 1 \leq k \leq \max_{block}\}$ can be computed in $O(\max_{block})$.

Finally, all the terms of J^l are obtained by $(n+1)$ multiplications of powers of eigenvalues (computed in $O(lq)$) and combinatorial coefficients (computed in $O(\max_{block})$). Note that if $l < n$, the probability of occurrence of the n-gram in l is immediately 0, since the n-gram is longer than the document. Therefore, the current algorithm is only used when $l \geq n \geq \max_{block}$. We can therefore conclude that **the time complexity of the computation of J^l is $O(lq)$.**

Time complexity for computing the transformation matrix and its inverse.

The second phase is to calculate S, the transformation matrix from M to J, and its inverse, S^{-1}.

Calculating the transformation matrix S. Following general results of linear algebra [ND77], the $(n+1)\times(n+1)$ transformation matrix S can be written as:

$$S = \left[S_1 S_2 \ldots S_q\right],$$

where each S_i is an $n_i \times (n+1)$ matrix corresponding to the eigenvalue e_i, and such that

$$S_i = \left[v_{i,1} v_{i,2} \ldots v_{i,n_i}\right],$$

where:

- $v_{i,1}$ is an eigenvector associated with e_i, thus such that $Mv_{i,1} = e_i v_{i,1}$, and

- $v_{i,j}$, for all $j = 2 \dots n_i$, is a solution of the equation $Mv_{i,j} = e_i v_{i,j} + v_{i,j-1}$.

The vectors $v_{i,1} v_{i,2} \dots v_{i,n_i}$ are sometimes called *generalized eigenvectors* of e_i. We have already seen in Section 4.2.4 that the first coordinate of each eigenvector can be assigned freely, and that every other coordinate can be expressed in function of its immediately preceding coordinate. Setting the first coordinate a_1 to 1, we can write:

$$v_{i,1} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_i \\ a_{i+1} \\ \vdots \\ a_{n+1} \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{e_i - q_1}{p_1} \\ \frac{(e_i - q_1)(e_i - q_2)}{p_1 p_2} \\ \vdots \\ \frac{(e_i - q_1) \dots (e_i - q_{i-1})}{p_1 \dots p_{i-1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The resolution of the system of $(n+1)$ linear equations following $Mv_{i,2} = e_i v_{i,2} + v_{i,1}$ permits to write:

$$v_{i,2} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ b_{i+1} \\ b_{i+2} \\ \vdots \\ b_{i+k} \\ b_{i+k+1} \\ \vdots \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \frac{a_1}{p_1} + \frac{e_i - q_1}{p_1} b_1 \\ \frac{a_2}{p_2} + \frac{e_i - q_2}{p_2} b_2 \\ \vdots \\ \frac{a_{i-1}}{p_{i-1}} + \frac{e_i - q_{i-1}}{p_{i-1}} b_{i-1} \\ \frac{a_i}{p_i} \\ \frac{e_i - q_{i+1}}{p_{i+1}} b_{i+1} \\ \vdots \\ \frac{e_i - q_{i+k-1}}{p_{i+k-1}} b_{i+k-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where k is such that $(i + k)$ is the position of second occurrence of e_i on the principal diagonal of M (that is, $q_{i+k} = q_i = e_i$), the position of first occurrence being i.

We can similarly write the other column vectors $v_{i,j}$, where $j = 3 \dots n_i$. Hence, it is clear that each coordinate of those vectors can be calculated in a constant number of operations. Therefore, we can compute each column in $O(n)$, and **the whole matrix S in** $O(n^2)$.

Observe that the value of b_1 is free and that it can be set to 0, without loss of generality. The same is true for the value in the first row of each column vector $v_{i,j}$, where $j = 2 \dots n_i$. In our implementation (notably when applying this technique to the running example on page 95), we made the choice to assign those first row values to 0. This means that the only non-null values on the first row of S are unity, and that they occur on the q eigenvector columns. This will prove helpful when calculating the expected frequency of occurrence of an n-gram by lowering the complexity of repeated multiplications of the first row of S by various powers of the matrix J.

The inversion of S. The general inversion of an $(n+1) \times (n+1)$ matrix can be done in $O(n^3)$ through Gaussian elimination. To calculate only the last column of S^{-1} does not help, since the resulting system of $(n+1)$ equations still requires $O(n^3)$ operations to be solved by Gaussian elimination.

However, some specificities of our problem will again permit an improvement over this general complexity. When describing the calculation of the similarity matrix S, it became clear that the n_i occurrences of e_i on the main diagonal of the matrix M are matched by n_i column vectors whose last non-null values exactly correspond to the n_i positions of occurrence of e_i on the main diagonal of M.

This is equivalent to saying that S **is a column permutation of an upper-triangular matrix**. Let T be the upper-triangular matrix corresponding to the column permutation of S. We can calculate the vector x, equal to the same permutation of the last column of S^{-1}, by solving the triangular system of linear equations

that follows $TT^{-1} = I_{n+1}$:

$$\begin{pmatrix} T_{1,1} & T_{1,2} & \dots & T_{1,n+1} \\ 0 & T_{2,2} & & T_{2,n+1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & T_{n+1,n+1} \end{pmatrix} \quad \overbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \end{pmatrix}}^{x = \text{last column of } T^{-1}}$$

$$= \quad \overbrace{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}}^{\text{last column of } I_{n+1}} .$$

The solution x is calculated by backward substitution:

$$\begin{cases} x_{n+1} &= \frac{1}{T_{n+1,n+1}} \\ x_n &= -\frac{1}{T_{n,n}}(T_{n,n+1}x_{n+1}) \\ \vdots & \\ x_1 &= -\frac{1}{T_{1,1}}(T_{1,2}x_2 + T_{1,3}x_3 + \dots + T_{1,n+1}x_{n+1}) \end{cases}$$

This way, the set of $(n+1)$ triangular linear equations can be solved in $O(n^2)$. It only remains to apply the reverse permutation to x to obtain the last column of S^{-1}.

Finding the permutation and its inverse are simple sorting operations. Thus, **the whole process of computing the last column of the transformation matrix S is $O(n^2)$.**

Conclusion

To obtain the final result, the probability of occurrence of the n-gram in a document of size l, it only remains to multiply the first row of S by J^l, and the resulting vector by the last column of S^{-1}. The second operation takes $(n+1)$ multiplications and n additions. It is thus $O(n)$.

The general multiplication of a vector of size $(n+1)$ by an $(n+1) \times (n+1)$ square matrix takes $(n+1)$ multiplications and n additions for each of the $(n+1)$ values of the resulting vector. This

is thus $O(n^2)$. However, we can use yet another trick to improve this complexity. When we calculated the matrix S, we could assign the first row values of each column vector freely. We did it in such a way that the only non-null values on the first row of S are unity, and that they occur on the q eigenvector columns. Therefore, to multiply the first row of S by a column vector simply consists in the addition of the q terms of index equal to the index of the eigenvectors in S. That operation of order $O(q)$ needs to be repeated for each column of J^l. The multiplication of the first row of S by J^l is thus $O(nq)$.

The worst-case time complexity of the computation of the probability of occurrence of an n-gram in a document of size l is finally $\max\{O(lq), O(n^2)\}$. Clearly, if $l < n$, the document is smaller than the n-gram, and thus the probability of occurrence of the n-gram therein can immediately be said to be null. Our problem of interest is hence limited to $l \geq n$.

Therefore, following our technique, **an upper bound** of the complexity for computing **the probability of occurrence of an n-gram in a document of size l is** $O(ln)$. This is clearly better than directly raising M to the power of l, which is $O(ln^3)$, not to mention the computation of the exact mathematical Formula 4.2, which is only achieved in $O(ln^{l-n})$.

4.3 The Expected Frequency of an n-Words Sequence

Now that we have defined a formula to calculate the probability of occurrence of an n-gram in a document of size l, we can use it to calculate the expected document frequency of the n-gram in the whole document collection D. Assuming the documents are mutually independent, the expected frequency in the document collection is the sum of the probabilities of occurrence in each document:

$$Exp\_df(A_1 \rightarrow \cdots \rightarrow A_n, D) = \sum_{d \in D} p(A_1 \rightarrow \cdots \rightarrow A_n, |d|),$$

where $|d|$ stands for the number of word occurrences in the document d.

4.3.1 Naive Computational Complexity

We can compute the probability of an n-gram to occur in a document in $O(ln)$. A separate computation and summation of the values for each document can thus be computed in $O(|D|ln)$, where $|D|$ stands for the number of documents in D.

In practice, we do improve the computational efficiency by counting the number of documents of same length and multiplying this number by the probability of occurrence of the n-gram in a document of that size, rather than reprocessing and summing up the same probability for each document of equal size. But as we are currently considering the *worst case* time complexity of the algorithm, we are facing the *worst case* situation in which every document has a distinct length.

4.3.2 Better Computational Complexity

We can achieve better complexity by summarizing everything we need to calculate and organizing the computation in a sensible way. Let $L = \max_{d \in D} |d|$ be the size of the longest document in the collection. We first need to raise the Jordan matrix J to the power of every distinct document length, and then to multiply the (at worst) $|D|$ distinct matrices by the first row of S and the resulting vectors by the last column of its inverse S^{-1}.

The matrix S and the last column of S^{-1} need to be computed only once, and as we have seen previously, this is achieved in $O(n^2)$, whereas the $|D|$ multiplications by the first row of S are done in $O(|D|nq)$. It now remains to find the computational complexity of the various powers of J.

We must first raise each eigenvalue e_i to the power of L, which is an $O(Lq)$ process. For each document $d \in D$, we obtain all the terms of $J^{|d|}$ by $(n+1)$ multiplications of powers of eigenvalues by a set of combinatorial coefficients computed in $O(\max_{block})$. The total number of such multiplications is thus $O(|D|n)$, an upper bound for the computation of all combinatorial coefficients. The worst case time complexity for computing the set $\{ J^{|d|} \mid d \in D\}$, is thus $\max\{O(|D|n), O(Lq)\}$.

Finally, **the computational complexity for calculating the expected frequency of an n-gram in a document collection**

D **is** $\max\{O(|D|nq), O(Lq)\}$, where q is the number of words in the n-gram having a distinct probability of occurrence, and L is the size of the longest document in the collection. The improvement is considerable, compared to the computational complexities of the more naive techniques, in $O(|D|ln^{l-n})$ and $O(|D|ln^3)$.

4.4 Direct Evaluation of Lexical Cohesive Relations

In this section, we will introduce an application of the expected document frequency that fills a gap in information retrieval. We propose a direct technique, language and domain-independent, to rank a set of phrasal descriptors by their interestingness, *regardless of their intended use*.

4.4.1 Hypothesis Testing

A general approach to estimate the interestingness of a set of events is to measure their statistical significance. In other words, by evaluating the validity of the assumption that an event occurs only by chance (the *null hypothesis*), we can decide whether the occurrence of that event is interesting or not. If a frequent occurrence of a multi-word unit was to be expected, it is less interesting than if it comes as a surprise.

To estimate the quality of the assumption that an n-gram occurs by chance, we need to compare its (by chance) expected frequency and its observed frequency. There exists a number of statistical tests, extensively described in statistics textbooks, even so in the specific context of natural language processing [MS99]. In this paper, we will base our experiments on the *t-test*:

$$t = \frac{Obs\_df(A_1 \rightarrow \cdots \rightarrow A_n, D) - Exp\_df(A_1 \rightarrow \cdots \rightarrow A_n, D)}{\sqrt{|D|Obs\_DF(A_1 \rightarrow \cdots \rightarrow A_n)}}$$

MFSs are very appropriate non-contiguous lexical units to be evaluated through our technique, since they are built with an unlimited gap and no length ceiling. The extraction algorithm also has the advantage of providing each MFS with its document frequency.

To compare the observed frequency of MFSs to their expected frequency is thus especially meaningful, and we will hence be able to sort the set of MFSs according to their statistical significance.

4.4.2 Experiments

Corpus

For experiments we used the publicly available Reuters-21578 newswire collection [Reu87], which originally contains about 19,000 non-empty documents. We split the data into 106,325 sentences. The average size of a sentence is 26 word occurrences, while the longest sentence contains 260.

Using a minimum frequency threshold of 10, we extracted 4,855 MFSs, distributed in 4,038 2-grams, 604 3-grams, 141 4-grams, and so on. The longest sequences had 10 words.

The expected document frequency and the t-test of all the MFSs were computed in 31.425 seconds on a laptop with a 1.40 Ghz processor and 512Mb of RAM. We used an implementation of a simplified version of the algorithm that does not make use of all the improvements presented in this paper.

Results

Table 4.1 shows the overall best-ranked MFSs. The number in parenthesis after each word is its frequency. With Table 4.2, we can compare the best-ranked bigrams of frequency 10 to their worst-ranked counterparts (which are also the worst-ranked n-grams overall), noticing a difference in quality that the observed frequency alone does not reveal.

It is important to note that our technique permits to rank longer n-grams amongst pairs. For example, the best-ranked n-gram of a size higher than 2 lies in the 10^{th} position: *"chancellor exchequer nigel lawson"* with t-test value 0.02315, observed frequency 57, and expected frequency $0.2052e-07$.

In contrast to this high-ranked 4-gram, the last-ranked n-gram of size 4 occupies the $3,508^{th}$ position: *"issuing indicated par europe"* with t-test value 0.009698, observed frequency 10, and expected frequency $22.25e-07$.

Table 4.1: Overall 10 best-ranked MFSs and the corresponding expected and observed frequencies (in the columns "exp" and "obs", respectively).

t-test	n-gram	exp	obs
0.03109	los(127) angeles(109)	0.08085	103
0.02824	kiichi(88) miyazawa(184)	0.09455	85
0.02741	kidder(91) peabody(94)	0.04997	80
0.02666	morgan(382) guaranty(93)	0.20726	76
0.02485	latin(246) america(458)	0.65666	67
0.02432	orders(516) orders(516)	1.54953	66
0.02431	leveraged(85) buyout(145)	0.07198	63
0.02403	excludes(350) extraordinary(392)	0.79950	63
0.02389	crop(535) crop(535)	1.66546	64
0.02315	chancellor(120) exchequer(100) nigel(72) lawson(227)	0.2052e-07	57

Table 4.2: The 5 best- and worst-ranked bigrams of frequency 10.

t-test	n-gram	exp	obs
9.6973-03	het(11) comite(10)	0.6430-03	10
9.6972-03	piper(14) jaffray(10)	0.8184-03	10
9.6969-03	wildlife(18) refuge(10)	0.0522-03	10
9.6968-03	tate(14) lyle(14)	0.1458-03	10
9.6968-03	g.d(10) searle(20)	0.1691-03	10
8.2981-03	pacific(502) security(494)	1.4434	10
8.2896-03	present(496) intervention(503)	1.4521	10
8.2868-03	go(500) go(500)	1.4551	10
8.2585-03	bills(505) holdings(505)	1.4843	10
8.2105-03	cents(599) barrel(440)	1.5337	10

Now, let us attempt to compare our ranking, based on the expected document frequency of discontiguous word sequences to a ranking obtained through a well-known technique. We must first underline that such a "ranking comparison" can only be empirical, since our standpoint is to focus on general-purpose descriptors. It is therefore, by definition, impossible to assess descriptors individually as interesting and not.

Evaluation through Mutual Information. The choice of an evaluation technique to oppose is rather restricted. A computational advantage of our technique is that it does not use distance windows or the distances between words. A consequence is that no evaluation technique based on the mean and variance of the distance between words can be logically considered. Smadja's z-test [Sma93] is then out of reach. Another option is to apply a statistical test, using a different technique for the calculation of the expected frequency of the word sequences. We decided to opt for pointwise mutual information, as first presented by Fano [Fan61] and applied to collocations discovery by Church and Hanks [CH90], an approach we already discussed in Section 2.3.2.

The rank of all word pairs is obtained by comparing the frequency of each pair to the probability that both words occur together by chance. Given the independence assumption, the probability that two words occur together by chance is the multiplication of the probability of occurrence of each word. And pointwise mutual information is thus calculated as follows:

$$I(w_1, w_2) = log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}.$$

If $I(w_1, w_2)$ is positive, and thus $P(w_1, w_2)$ is greater than $P(w_1) \times P(w_2)$, it means than the words w_1 and w_2 occur together more frequently than chance. In practice, the mutual information of all the pairs is greater than zero, due to the fact that the maximal frequent sequences that we want to evaluate are already a selection of statistically remarkable phrases.

As stated by Fano [Fan61], the intrinsic definition of mutual information is only valid for bigrams. Table 4.3 presents the best 10 bigrams, ranked by decreasing mutual information. Table 4.4 shows the 5 best- and worst-ranked bigrams of frequency 10 (again, the worst ranked bigrams of frequency 10 are also the worst ranked

Table 4.3: Mutual Information: the 10 best bigrams.

Bigram	Frequency	Mutual Information
het(11) comite(10)	10	17.872
corpus(12) christi(12)	12	17.747
kuala(14) lumpur(13)	13	17.524
piper(14) jaffray(10)	10	17.524
cavaco(15) silva(11)	11	17.425
lazard(16) freres(16)	16	17.332
macmillan(16) bloedel(13)	13	17.332
tadashi(11) kuranari(16)	11	17.332
hoare(15) govett(14)	13	17.318
ortiz(16) mena(14)	13	17.225

overall).

We can observe that, for the same frequency, the rankings are very comparable. Where our technique outperforms mutual information is in ranking together bigrams of different frequencies. It is actually a common criticism against mutual information, to point out that the score of the lowest frequency pair is always higher, with other things equal [MS99]. For example, the three best-ranked MFS in our evaluation, *"Los Angeles", "Kiichi Miyazawa"* and *"Kidder Peabody"*, which are among the most frequent pairs, rank only 191^{st}, 261^{st} and 142^{nd} with mutual information (out of $4,038$ pairs).

Mutual information is not defined for n-grams of a size longer than two. Other techniques are defined, but they usually give much higher scores to longer n-grams, and in practice, rankings are successions of decreasing size-wise sub-rankings. A noticeable exception is the measure of mutual expectation introduced by Dias (see [DGBPL00a], and our overview in Section 2.3.2).

Compared to the state of the art, the ability to evaluate n-grams of different sizes on the same scale is one of the major strengths of our technique. Word sequences of different size are ranked together, and furthermore, the variance in their rankings is wide. While most of the descriptors are bigrams ($4,038$ out of $4,855$), the 604 trigrams are ranked between the 38^{th} and $3,721^{st}$ overall positions. For the 141 4-grams, the position range is $10-3,508$.

Table 4.4: Mutual Information: the 5 best and worst bigrams of frequency 10.

Bigram	Frequency	Mutual Information
het(11) comite(10)	10	17.872
piper(14) jaffray(10)	10	17.524
wildlife(18) refuge(10)	10	17.162
tate(14) lyle(14)	10	17.039
g.d(10) searle(20)	10	17.010
pacific(502) security(494)	10	6.734
present(496) intervention(503)	10	6.725
go(500) go(500)	10	6.722
bills(505) holdings(505)	10	6.693
cents(599) barrel(440)	10	6.646

4.5 Conclusion

We presented a novel technique for calculating the probability and expected document frequency of any given non-contiguous lexical cohesive relation. We first calculated an exact formula to reach this result, and observed that it is not usable in practice, because of an exponential computational complexity. We then found a Markov representation of the problem and exploited the specificities of that representation to reach linear computational complexity. The initial order of complexity of $O(ln^{l-n})$ was brought down to $O(ln)$.

We further described a method that compares observed and expected document frequencies through a statistical test as a way to give a direct numerical evaluation of the intrinsic quality of a multi-word unit (or of a set of multi-word units). This technique does not require the work of a human expert, and it is fully language- and application-independent. It permits to efficiently compare n-grams of different length on the same scale.

A weakness that our approach shares with most language models is the assumption that terms occur independently from each other. In the future, we hope to present more advanced Markov representations that will permit to account for term dependency.

CHAPTER 5

Exploratory Application to Document Retrieval

The previous chapter presented a technique to evaluate the quality of phrasal descriptors directly. We will now take a closer look at their use in the framework of an information retrieval application, namely document retrieval. First of all, we will define essential concepts of document retrieval in Section 5.1 before providing a short summary of related techniques to exploit phrases in the retrieval task (Section 5.2).

We will then present the last contribution of this monograph in the form of a novel technique to compute the phrase-based similarity of documents (Section 5.3). To understand and evaluate the impact of this contribution, we formulate in Section 5.4 a number of hypotheses and questions together with the definition of a set of experiments that are expected to provide the corresponding answers. Finally, the results are presented and discussed in Section 5.5.

5.1 Basic Concepts of Document Retrieval

5.1.1 The Document Retrieval Task

The task of document retrieval consists of selecting a set of documents in a collection, in response to a user's request.

The user initially formulates her information need, as a question in natural language, for example, or as a set of keywords or

keyphrases. We refer to the formulation of an information need as a *topic*.

The task of a document retrieval system is then to transform the topic into a machine-interpretable *query*. Depending on the document retrieval system, this query can be of different types, e.g., if the system is based on the vector space model, the topic will be transformed into a vector.

This transformation of a topic permits to compare it to the documents of the collection. With respect to a given query, the documents of a collection can be assigned similarity values by which they may be sorted. The answer of a document retrieval system to a user-defined topic is a list of documents, ranked by corresponding similarity values (also called *Retrieval Status Value (RSV)*) in decreasing order.

In large collection testbeds, the evaluation of a system is based on its combined performance versus a number of queries. The set of answers corresponding to a set of topics is called a *run*. Document retrieval systems are evaluated through the quality of the runs they produce. In the following section, we will outline the main measures for the evaluation of document retrieval, with an emphasis on the ones we will use in the rest of this chapter.

5.1.2 Evaluation of Document Retrieval Systems

The effectiveness of the set of documents returned to a topic is generally measured upon judgments of which documents are truly relevant to the user's information need and which are not. Given a topic and an associated set of *relevance assessments*, i.e., a list of which documents of the collection were judged as relevant by a domain expert, we can define a number of evaluation measures.

A document selected by the retrieval system as relevant to a given topic is called a *positive*. Based on the relevance assessments, we further qualify such a retrieved document as a *true positive (TP)* if it was truly assessed as relevant, and as a *false positive (FP)* if it was not. Symmetrically, a document that is not returned by the system is called a *negative*. It is a *true negative (TN)* if it was judged as irrelevant, and a *false negative (FN)* if it should have been returned by the system. The concepts of positive and negative documents, true and false, are summarized in Figure 5.1.2.

Documents	relevant	not relevant
retrieved	True Positives	False Positives
not retrieved	False Negatives	True Negatives

Figure 5.1: True and false, positive and negative documents.

Recall and Precision. Given the number of truly relevant documents, i.e., the sum of the number of true positives and false negatives, we can compute a ratio of exhaustiveness by dividing the number of relevant documents found by the total number of relevant documents. This measure is the *recall*:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

In practice, the number of false negatives is often an estimation only, as it is too time-demanding for a domain expert to assess every single document of a collection as relevant or irrelevant. Among the documents retrieved, the ratio of relevant ones is the *precision* of the retrieval:

$$\text{Precision} = \frac{TP}{TP + FP}.$$

We may observe that returning all the documents of the collection is an easy way to ensure 100% recall. But the precision is then low. In a similar fashion, to get high precision ratios, a safe heuristics is to return a limited number of documents, the ones with the very highest similarity measures, with respect to the topic. Subsequently weak is the recall. Efficient evaluation of document retrieval systems is in fact based on combinations of recall and precision values. For example, the *F-measure* [VR79] of a ranked list of documents is the harmonic mean of recall and precision:

$$\text{F-measure} = \frac{2 \times recall \times precision}{recall + precision}.$$

The F-measure is null when either recall or precision is null, and it is equal to 1 if and only if both recall and precision are equal to 1.

Precision at k. A common way to obtain a measure of precision that also accounts for recall is to look only at the k best ranked documents. This measure looks at the performance at the top of

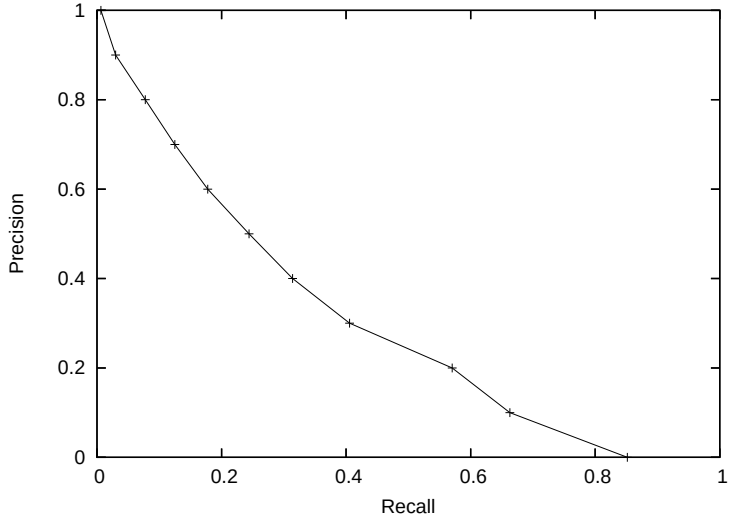

Figure 5.2: A recall-precision graph.

the list and is intuitive of the behavior of users, as they scan the ranked list from top to bottom, and actually in many practical applications, for example web searching, they only scan through the very first results of a ranked document list, until they find a relevant document. However, such measures are typically less useful for information needs where exhaustivity is important, i.e., to see as many of the relevant documents as possible.

Recall-Precision curve. Another approach to estimate the quality of a list of retrieved documents is to plot a *recall-precision graph*. A typical such graph is shown in Figure 5.2. The graph is drawn by extrapolation from a number of data points. Typical data points are measures of precision at every 10% of recall, i.e., at recall 0, 0.1, 0.2,. . . , and 1. For example, the precision measure corresponding to a 40% recall tells the ratio of relevant documents the user needs to go through before she has seen 40% of all the relevant documents.

A subsequent popular measure of the quality of a ranked list of documents is the *average precision* over a number of points of recall. For example, for the data points at every 10% of recall, we talk about 11-*point average precision*. Reading the ranked document list from top to bottom, we can also calculate the precision each time a true positive is encountered. By averaging all those precision values

together, we obtain a popular measure, the *mean average precision (MAP)*.

Different measures for different information needs. In some applications, such as searching for information on the web, users typically look at the first 10 results only. In this case, an appropriate measure is the precision at 10. Values at the left-hand side of the recall-precision graph are most important.

There are cases, however, when finding every bit of relevant information is crucial. Typical such domains are patent validation, where every similar existing patent must be carefully checked, or jurisprudence searches in the judicial domain. A lawyer does not want to miss a case that presents strong similarities to the one she is working on. To find the retrieval system that is most appropriate to her information need, precision at 10 is an inappropriate evaluation measure. We rather wish to know the proportion of relevant documents she had to go through before reading all the relevant documents (or more realistically 80 or 90% of them). In this case, it is values at the right-hand side of the recall-precision graph that matter most. And we may wish to compute an evaluation measure based on this very part of the graph, for example the 11-point average precision for recall values above 50%, e.g., 0.5, 0.55, 0.6, ..., and 1.

5.1.3 Document Retrieval and Multiple Languages

To illustrate and support the fact that the contributions of this thesis are applicable to any human language, we are about to present experiments on a few languages that are radically different in nature, namely, English, Japanese, Korean, and Chinese. Since we have no knowledge of the last three languages, the ability to experiment with them is good support for our early plan to develop techniques that do not require knowledge about the language of the documents at hand. We will now briefly clear up terminology issues and explain why our work may not consensually qualify as "multilingual".

Following the general definition of the adjective "multilingual" (using or having the ability to use several languages), we can certainly claim that all the techniques we presented in this dissertation are multilingual. However, this may cause some confusion, as the

meaning of the term "multilingual" has shifted within the overlapping research communities of information retrieval and computational linguistics. It is indeed often misused for "cross-lingual". Cross-Lingual Information Retrieval (CLIR) is a subdomain of document retrieval of which the goal is to support queries in one language against a collection in other languages [AAB+03]. The typical use case is truly serving multilingual *users*, as to obtain documents in all the languages she can understand, a user only needs to query in one of them. This is opposed to *monolingual retrieval*, where the answers to queries formulated in one language are documents in the same language. This denomination unfortunately makes no distinction between techniques that are specifically conceived for one language, and techniques that are well suited and applied to any.

Hence, all the techniques we presented in this dissertation, as well as the one that remains to be presented in Section 5.3 can duly be called multilingual. However, to avoid misleading researchers in the field, and to follow the practice of its sublanguage, a more adequate qualification of our work would be "monolingual techniques for any language"

5.2 Previous Attempts to Use Phrases in Document Retrieval

As opposed to words, the higher content specificity of phrases is a strong motivation for their extraction. Word sequences in document retrieval should hence be more precise index terms than individual words. The potential improvement of using phrases in document retrieval is supported by the behavior of users. In an analysis of the query log of the Excite search engine (more than 1.5 million queries), Williams et al. [WZB04] found that 8.4% of the queries contained explicit phrases, that is, they included at least two words enclosed in quotes. Even more interestingly, the authors found it beneficial to treat 40% of the queries without quotation marks as phrases rather than independent words. Consequently, there is no doubt that an efficient technique to use phrases may bring solid improvement to document retrieval applications.

Work on the use of phrases in IR has been carried out for more than 25 years. Early results were very promising. However, unexpectedly, the constant growth of test collections caused a drastic fall in the quality of the results. In 1975, Salton et al. [SYY75] showed an improvement in average precision over 10 recall points between 17% and 39%. In 1989, Fagan [Fag89] reiterated the exact same experiments with a 10 Mb collection and obtained improvements from 11% to 20%. This negative impact of the collection size was lately confirmed by Mitra [MBSC97] over a 655 Mb collection, improving the average precision by only one percent. Turpin and Moffat [TM99] revisited and extended this work to obtain improvements between 4% and 6%.

A conclusion of this related work is that phrases improve results in high levels of recall, but are globally inefficient for the n first ranked documents. According to Mitra [MBSC97], this low benefit from phrases to the best answers is explained by the fact that phrases promote documents that deal with only one aspect of possibly multi-faceted queries. For example, a topic of TREC-4 is about "problems associated with pension plans, such as fraud, skimming, tapping or raiding". Several top-ranked documents discuss pension plans, but no related problem. This problem, as we already mentioned in Section 2.4.4, was termed by Mitra as one of *inadequate query coverage*.

In our opinion, this does not contradict the idea that adding document descriptors accounting for word order must permit to improve the performance of IR systems. In Section 2.4, we expanded on the need to improve the ways to account for phrases in document representations. Another difficult problem is to find efficient ways to benefit from those phrases. Related work has shown the need for another way to combine phrase and word term descriptors [SK98]. This need was illustrated by work of Lewis [Lew92] and Vechtomova [Vec05], who both decided to involve human experts in the process. Both obtained small improvement, suggesting that the techniques to exploit the extracted phrases can also be perfected.

There are various ways to exploit phrase descriptors. The most common technique is to consider phrases as supplementary terms of the vector space, using the exact same technique as for word terms. In other words, phrases are thrown into the bag of words. However, according to Strzalkowski and Carballo [SC96], using a

standard weighting scheme is inappropriate for mixed feature sets (such as single words and phrases). In such cases, the weight given to the least frequent phrases is considered too low. Their specificity is nevertheless often crucial in order to determine the relevance of a document [Lah00]. In weighting the phrases, the interdependency between a phrase and the words that compose it is another difficult issue to account for [SSW+98].

An advanced matching technique was recently introduced by Vechtomova [Vec05]. Its contribution is to address the problem of overlapping phrases, in a way that accounts for the relative positions of occurrence of the words they contain. The problem of overlapping phrases occurs for phrases of more than *two* words. Given a query phrase ABC, it is the question of how to evaluate a document that contains the phrase ABC and a document that contains the phrases AB and BC separately.

For each query phrase, a pass through the document collection is done, to retain every occurrence of terms of the query phrase and their original positions in the document. Terms that form the keyphrase or one of its sub-phrases are gathered into so-called "windows". Each window is weighed upon the idf of the words that compose it and the distance that separated them originally:

$$WindowWeight(w) = \sum_{i \in w} idf_i \times \frac{n}{(span + 1)^p},$$

where i is a word occurring in the window w, $span$ is the distance between the first and last word of the window, and p is a tuning parameter, arbitrarily set to 0.2 in [Vec05]. The score attributed to each document is calculated as the sum of the weights of the phrases it contains, where the weight of a phrase a in a document is defined as follows:

$$PhraseWeight(a) = \frac{(k + 1) \times \sum_{w=1}^{n} WindowWeight(w)}{k \times NF + n},$$

where n is the number of windows w extracted for the phrase a, k is a phrase frequency normalization factor, arbitrarily set to 1.2 in [Vec05], and NF is a document length normalization factor:

$$NF = (1 - b) + b \times \frac{DocLen}{AveDocLen},$$

where *DocLen* and *AveDocLen* are the document length and the average document length in the corpus (number of words), and b is a tuning constant, set to 0.75.

A major drawback is the computational complexity of this process. In this method, there is no static phrase index that gives a phrasal representation of the document collection. It is only at query-time that a representation of the collection is built that only contains the terms of the query. Such heavy processing in response to a query is quite problematic, as users usually expect to obtain results promptly.

In practice, the method has only been used for re-ranking the 1,000 best documents returned to a query by a vector space model relying on single word features. The results demonstrate a performance improvement in terms of average precision, which is unfortunately not statistically significant. They also confirm a common observation when using phrases for document retrieval: compared to the use of single word features only, improvement is observed at high recall levels, while the impact is negative at lower levels.

In the following section, we will introduce a new technique for computing phrase-based document similarity. We will then apply it to document retrieval.

5.3 An Advanced Phrase-Matching Technique

5.3.1 Problem Definition and Goals

Problem definition. Given a set of sequences that describe the documents of a collection, how can we determine to what extent the sequence $p_1 \dots p_n$, issued from the document collection, corresponds to the sequence $q_1 \dots q_m$, found in a user query? And how can we subsequently rank the documents according to how well we think they answer to the query?

We propose an approach that consists in comparing a set of descriptive phrases extracted from the document collection, to a set of *keyphrases* from the query. Given a query, every document receives a reward for every sequence it contains that matches a keyphrase of the query. This bonus generally differs for each different phrase. Note that from here onwards, the term *keyphrase* will be used to

refer to a phrase found in a query.

A base weight. The most informative lexical associations should notably be promoted, using statistical information such as their term and inverted document frequency.

Longer matches are better matches. Further, it is natural to wish that longer matches should be better rewarded. If a query contains the keyphrase "XML structured information retrieval", the most appropriate documents are those whose descriptors contain this exact sequence, followed by those containing a subsequence of size 3 (e.g., "structured information retrieval"), and finally by documents containing a subpair of the keyphrase (e.g., "structured information" or "information retrieval").

Adjacency should not be required. Clearly, a phrasal descriptor containing the pair "XML retrieval" has a relationship with the keyphrase "XML structured information retrieval". This illustrates the fact that natural language is richer in variety than only recurrent adjacent word sequences.

But adjacency is generally a stronger indicator. We should, however, bear in mind the general rule that the more distant two words are, the less likely they are to be related. And the degree to which the relatedness of two words is affected by distance certainly varies greatly with different languages.

Inverted usage. An extension of the previous comments about word adjacency is that we should also try to take into account the fact that words might as well occur in inverted order, while still not necessarily being adjacent. For example, a phrase "retrieval of XML" triggers interest with respect to the earlier keyphrase "XML structured information retrieval".

Jones and Sinclair [JS74] give the example of the pair "hard work", where throughout their document collection, the words "hard" and "work" are occurring together in distinct order, and with a variable distance between them. Of course, in English, not all collocations are this relaxed, and others are exclusively rigid, for example the pair "Los Angeles" is very unlikely to occur in a different order, or with other words inserted. They term those two types of collocations as *position dependent* and *position free* collocations. By attributing a positive score to matches and ignoring misses, we can get around this problem. If we look for phrasal document descriptors containing "Angeles Los" or for the occurrence of "Los"

and "Angeles" separated by other words, and we fail to find any, it will not worsen the retrieval performance. Whereas finding that a document about "retrieval of XML" is relevant to a query about "XML retrieval" is evidently better than failing to observe it.

In the next subsection, we will introduce our approach to the problem. It aims at taking into account all the observations above in a sensible way.

5.3.2 Document Score Calculation

Our approach exploits and combines two complementary document representations. One is based on single word terms, in the vector space model, and the other is a phrasal description, taking the sequential nature of text data into account.

Once documents and queries are represented within those two models, a way to estimate the relevance of a document with respect to a query remains to be found. We must sort the document list with respect to each query, which is why we need to compute a *Retrieval Status Value (RSV)* for each document and query. Below, we will explain how we calculate two separate RSVs, one for a word features vector space model and one for our phrasal description. In a later step, we aggregate these two RSVs into one single relevance score for each document with respect to a query.

Word features RSV

Our first document representation is a standard vector space model, of which all features are single words. It represents a baseline model that our goal is to perfect by the addition of sequential information from our second document model.

The index term vocabulary W is as simple as can be, it includes every word found in the document collection, without preselection. Further, the words are left in their original form, morphological conflation techniques being left aside. This guarantees generality, as this can be done in an equally simple way for document collections written in any language.

Hence, as we have seen in detail in section 2.2, each document is represented by a $\|W\|$-dimensional vector filled in with a weight standing for the importance of each word token with respect to

the document. To calculate this weight, we use a term-frequency normalized version of term-weighted components, as described by Salton and Buckley [SB88], that is:

$$tfidf_w = \frac{tf_w \cdot \log \frac{|D|}{df_w}}{\max(tf) \cdot \sqrt{\sum_{w_i \in W} \left(tf_{w_i} \cdot \log \frac{|D|}{df_{w_i}} \right)^2}}$$

where tf_w and df_w are the term and document frequencies of the word w, $|D|$ is the total number of documents in the collection D, and $\max(tf)$ is the (term-) frequency of the most (term-) frequent word, i.e., $\max(tf) = \max_{w_i \in W} tf_{w_i}$.

The vector space model offers a very convenient framework for computing similarities between documents and queries. Among the number of techniques to compare two vectors, we chose cosine similarity because of its computational efficiency. By normalizing the vectors, which we do in the indexing phase, $cosine(\overrightarrow{d_1}, \overrightarrow{d_2})$ indeed simplifies to the vector product $(d_1 \cdot d_2)$.

We have already expanded on the weaknesses and the amount of information that such a simple model cannot catch. This is why we will complement this model with a phrasal one, bringing sequential information into the document model, and aiming to carry it on into document retrieval.

Phrasal RSV

The core of techniques for the extraction of phrasal document descriptors was covered in Chapters 2 and 3. Once such a technique has been applied and the phrasal descriptors have been extracted, an index is easily built for the document collection. The only task is to associate each document with the list of all phrasal descriptors that it contains. This can involve an extra step when the retrieval granularity does not match the extraction granularity. For example, MFSs are often extracted from a sentence collection, whereas text retrieval applications often seek longer document fragments.

Once this first step has been achieved, and a set of n-grams is attached to each document, it remains to define a procedure to match a phrase describing a document and a keyphrase.

Our approach consists in decomposing keyphrases of the query into key pairs. Each of these pairs is bound to a score representing

its inherent *quantity of relevance*. Informally speaking, the quantity of relevance of a key pair tells how much it makes a document relevant to contain an occurrence of this pair. This value depends on a basic measure of the importance of the pair (its *base weight*, which can be its inverted document frequency, for example) combined with a number of modifiers, meant to take into account the distance between two words of a pair, to penalize their possible inverted usage, and so on.

Definitions. Let D be a document collection and $K_1 \dots K_m$ a keyphrase of size m. Let K_i and K_j be two words of $K_1 \dots K_m$. We define the quantity of relevance associated to the key pair K_iK_j as:

$$Q_{rel}(K_iK_j) = Base\_Weight(K_iK_j, D) \cdot Integrity(K_iK_j),$$

where $Base\_Weight(K_iK_j, D)$ represents the general importance of K_iK_j in the collection D. A possible measure of this kind is the statistical significance of the pair, or its specificity, measured in terms of inverted document frequency:

$$idf(K_iK_j, D) = \log\left(\frac{|D|}{df(K_iK_j)}\right),$$

Integrity Modifier. When decomposing the keyphrase $K_1 \dots K_m$ into pairs, the *Integrity Modifier* of the key pair K_iK_j is defined as the combination of a number of modifiers:

$$Integrity(K_iK_j) = adj(K_iK_j) \cdot inv(K_iK_j) \cdot dup(K_iK_j).$$

Non-adjacency penalty. $Adj(K_iK_j)$ is a score modifier meant to penalize key pairs formed from non-adjacent words. Let $d(K_i, K_j)$ be the distance between K_i and K_j, that is, the number of other words appearing in the keyphrase between K_i and K_j ($d(K_i, K_j) = 0$ means that K_i and K_j are adjacent). We define:

$$adj(K_iK_j) = \begin{cases} 1, & & \text{if } d(K_i, K_j) = 0 \\ \alpha_1, & 0 \le \alpha_1 \le 1, & \text{if } d(K_i, K_j) = 1 \\ \alpha_2, & 0 \le \alpha_2 \le \alpha_1 & \text{if } d(K_i, K_j) = 2 \\ \dots & & \\ \alpha_{m-2}, & 0 \le \alpha_{m-2} \le \alpha_{m-3}, & \text{if } d(K_i, K_j) = m-2 \end{cases}$$

Accordingly, the larger the distance between the two words, the lower a quantity of relevance is attributed to the corresponding

pair. In the experiments, we set only a base value of non-adjacency penalty *adj_pen* that is raised to the power of the distance between the two words of the key pair. In other words, $\alpha_{d(K_i,K_j)} = adj\_pen^{d(K_i,K_j)}$. In practice, choosing the example value of 0.9 for *adj_pen* means that the base matching quantity awarded to documents containing K_iK_j is lowered by 10% for every other word occurring between K_i and K_j in the original keyphrase.

A further possibility is to define a maximal distance between two words by setting, for example, $\alpha_k = 0$, for k greater than a given maximal distance threshold. As we have seen, a maximal distance of 5 was suggested for English document collections [JS74, Sma93, DGBPL00b].

Inversion penalty. $Inv(K_iK_j)$ is another score modifier used to penalize key pairs K_iK_j that occur in the opposite order in the original keyphrase:

$$inv(K_iK_j) = \begin{cases} 1, & \text{if } K_i \text{ occurs before } K_j. \\ inv\_pen \leq 1, & \text{otherwise.} \end{cases}$$

Evidently, the non-adjacency and inversion penalties are strongly language- and domain-dependent. The less relative word positions matter, the lower those penalties should be. For a theoretical document collection where relative word positions have no importance, we should have $inv\_pen = 1$ and, for $0 \leq l \leq (m-2)$, $\alpha_l = 1$. Observe, that for such a theoretical document collection, a bag of words representation would be optimal, and the use of phrasal descriptors would actually not bring any supplementary information.

Duplication bonus. A result of the creation of non-adjacent and inverted key pairs is that the list of pairs representing a query may contain a number of duplicates. Rather than incrementing a corresponding number of matching quantities, we decide to remove the duplicates, and keep one occurrence of the key pair together with its highest associated matching quantity. This highest matching quantity is further increased by $dup(K_iK_j)$, a relative weight increase awarded to those pairs occurring several times in the original keyphrase.

Maximal matching distance. Observe that the question of which parts of a document descriptor can be matched with a pair was left open. If the phrasal descriptors are maximal frequent sequences, it is a sensible option to allow for an unlimited gap between

Table 5.1: Quantity of relevance stemming from various indexing phrases with respect to a keyphrase query $ABCD$. Bw stands for $Base\_Weight$.

Document	Description	Quantity of relevance
d_1	AB	$Bw(AB)$
d_2	ACD	$Bw(CD) + \alpha_1 Bw(AC) + \alpha_2 Bw(AD)$
d_3	AFB	$Bw(AB)$
d_4	ABC	$Bw(AB) + Bw(BC) + \alpha_1 Bw(AC)$
d_5	ACB	$Bw(AB) + \alpha_1 Bw(AC) + \alpha_1 \cdot inv\_pen \cdot Bw(CB)$

each two words of the descriptor, because by definition, if $ABCD$ is frequent, then so are AB, AC, AD, BC, BD, and CD. In the general case, however, we allow for the possibility to use a maximal matching distance max_d. We try to match two words of a phrasal descriptor against a key pair only if there is no more than max_d other words occurring between them.

Example. To illustrate these definitions, let us have a look at the decomposition of the keyphrase $ABCD$. It is decomposed into 12 tuples (pair, integrity modifier):

$(AB, 1)$, (AC, α_1), (AD, α_2),$(BC, 1)$, (BD, α_1), $(CD, 1)$, (BA, inv_pen), $(CA, \alpha_1 \cdot inv\_pen)$, $(DA, \alpha_2 \cdot inv\_pen)$, (CB, inv_pen), $(DB, \alpha_1 \cdot inv\_pen)$, (DC, inv_pen).

Let us compare this keyphrase to the documents d_1, d_2, d_3, d_4 and d_5, represented respectively by the phrasal descriptors AB, AC, AFB, ABC and ACB. The maximal matching distance max_d is set higher than 1. The corresponding quantities of relevance brought by matching part of the keyphrase $ABCD$ are shown in table 5.1.

Assuming equal $Base\_Weight$ values, we observe that the quantities of relevance form an order matching the desirable properties that we had wished for in Section 5.3.1. The longest matches rank first, and matches of equal size are untied by relative word positions (adjacency and inversion). Moreover, non-adjacent matches (AC and ABC) are not ignored as in many other phrase represen-

tations [MBSC97].

Aggregated RSV

In practice, a query may not contain any keyphrases, and a document may not be represented by any phrasal descriptors. However, there can of course be correct answers to these queries, and those documents must be relevant to some information needs. Also, all documents containing the same matching phrases get the same phrasal RSV. If the phrasal description is small, it is necessary to find a way to break ties. The cosine similarity measure based on word features is very appropriate for that.

Another response could have been to decompose the pairs into single words and form document vectors accordingly. However, this would not be satisfying, because a relevant word term may not occur frequently enough to belong to any phrasal descriptor. An even more important category of missed words is that of the words that are individually frequent but do not frequently co-occur with other words. The loss would be considerable.

This is the reason to compute another RSV using a basic word-feature vector space model. To combine both RSVs to one single score, we must first make them comparable by mapping them to a common interval. To do so, we used *Max Norm*, as presented by Lee [Lee95], which permits to bring all positive scores within the range [0,1]:

$$New\ Score = \frac{Old\ Score}{Max\ Score}$$

Following this normalization step, we aggregate both RSVs using a linear interpolation factor λ representing the relative weight of scores obtained with each technique (similarly as in [MKR02]).

$$Aggregated\ Score = \lambda \cdot RSV_{Word\_Features} + (1 - \lambda) \cdot RSV_{Phrasal}$$

The evidence of experiments with the INEX 2002 collection and MFS phrasal descriptors [Dou04] showed good results when weighting the single word RSV with the number of distinct word terms in the query (let a be that number), and the phrasal RSV with the

```
<Keywords>
 "concurrency control"
 "semantic transaction management"
 "application" "performance benefit"
 "prototype" "simulation" "analysis"
</Keywords>
```

Figure 5.3: Topic 47

number of distinct word terms found in keyphrases of the query (let b be that number). Thus:

$$\lambda = \frac{a}{a+b}$$

For example, in Figure 5.3, showing topic 47 of the INEX collection, there are 11 distinct word terms and 7 distinct word terms occurring in keyphrases. Thus, for this topic, we have $\lambda = \frac{11}{11+7} \approx 0.61$.

5.4 Experimental Framework

We will now present our practical approach to the evaluation of MFSs as content descriptors in the application domain of document retrieval.

5.4.1 Open Questions and Protocol of the Experiments

In the experiments, we will apply MFSs and our novel matching technique to document retrieval. Crucially, these are two contributions whose impact should be evaluated separately. To decide whether the results are influenced by the intrinsic quality of MFSs as indexing terms for document retrieval, or whether the results are due to the matching technique, we will need to answer the following two crucial questions:

- (Q1) Does our matching technique permit effective improvements in the use of MFSs in document retrieval?
- (Q2) Are MFS good indexing terms for document retrieval?

- **WVSM**, the baseline, a retrieval run that follows the vector space model as described in Section 2.2, relying on word term features only.
- **VSM-AdjBig**, VSM with words and adjacent bigrams: a retrieval run that follows the vector space model as described in Section 2.2, except that all the adjacent bigrams of the document collection are added to word features in the same $tfidf$ weighting scheme. Adjacent bigrams are thrown into the bag of words, so to say.
- **MFS-Big**, VSM with all the bigrams occurring in MFSs: In this run, all the bigrams occurring in an MFS are added to the vector space. For example, with an MFS $ABCD$, the bigrams AB, AC, AD, BC, BD and CD are thrown into the bag of words.
- **MFS-Adv**, advanced use of MFSs: This run applies the technique we presented in Section 5.3.

Figure 5.4: Description of the runs to be produced for the experiments of Chapter 5.

To answer these questions efficiently, we will need to produce a consequent number of runs, listed in Figure 5.4. This set of four runs per collection permits to give answers to the questions Q1 and Q2.

To answer Q1, about the performance of our phrase-matching algorithm (MFS-Adv), we can notably measure the results of MFS-Adv against the use of phrasal descriptors as a set of frequent pairs used to augment the vector space (MFS-Big). Naturally, we will also compare those two approaches to the word features baseline (WVSM).

To study the inherent quality of MFSs as content descriptors and answer Q2, we can compare the two previous results (MFS-Adv and MFS-Big) and compare them to a straight addition in the word-term vector space of all the adjacent bigrams of the document collection (VSM-AdjBig).

5.4.2 Tuning our Matching Technique

Although we expect that the techniques presented in this dissertation can be applied to any language and any type of document, we can make conjectures about document collections for which our phrase-based similarity measure will typically perform better and worse. The following hypotheses are to be verified in the experiments.

- H1: Because our matching technique can account equally for multi-word units whose words occur at various relative positions, we believe that it will bring higher improvement for languages where the relative positions of words are less important (*hypothesis H1*). A corresponding family of languages is known as that of agglutinative languages. The least importance of relative positions is due to the fact that word-modifying morphemes are typically *agglutinated* to the corresponding word, meaning that changing its position seldom changes its role in the sentence. Typical agglutinative languages are, e.g., Turkish, Finnish and Japanese. In the opposite situation, where relative word positions are most important, we do not expect great performance from our matching technique. This situation is that of isolating languages, such as Chinese, Vietnamese, Samoan, or to a lesser extent, English.

- H2: The number of multi-word units that are regularly and consistently used throughout a document collection is generally known to be greater if that collection is specialized. Typically, more multi-word units tend to occur in a technical document than in a newspaper article. Our second hypothesis (H2) is that the improvement brought by our technique will be greater for a more specialized document collection.

As we have seen in Section 5.3, our matching technique functions with a number of parameters to be applied to the key phrases, namely, inversion and non-adjacency penalties, duplication bonus, and maximal matching distance. In this dissertation, we will present a few experiments to determine suitable parameter values for each document collection. Naturally, in real-life applications, this would

Table 5.2: MFS-Adv. The five different runs of the advanced matching technique and their different parameter values for maximal distance (max_d), inversion (inv_pen) and non-adjacency penalty (adj_pen).

	max_d	inv_pen	adj_pen
Adj_Baseline	0	0	not def.
Balanced	5	0.5	0.8
No_Inv	5	0	0.8
Dist_pen	5	0.5	0.2
Max_d	10	0.5	0.8

not always be possible. We can, however, give guesses on what would be good parameters, depending on the nature of the document collection.

The same train of thoughts that led us to formulating hypotheses H1 and H2 also leads us to thinking that agglutinative languages and specialized collections will benefit from a higher maximal distance than isolating languages and general collections. To inflict a lower penalty to pairs occurring in inverse order or with many other words between them should similarly benefit agglutinative languages and specialized collections, rather than isolating languages and general collections. To verify these assumptions, we will run our advanced matching technique for each collection with 5 different sets of parameters. The corresponding five runs are described in Table 5.2. "Adj_Baseline" rejects inversion, and only considers adjacent words of the key phrase. The run "Balanced" is meant to integrate some of each spice: each parameter is represented with a reasonable general value. Each of the last three runs emphasizes one of the three parameters, as compared to the run "Balanced". For example, "Dist_pen" emphasizes the distance penalty because it lowers the weight of pairs formed from distant words, by setting adj_pen to 0.2 instead of 0.8.

To perform the set of experiments needed, we will now introduce two appropriate document collections, upon which our techniques will be applied.

5.4.3 Presentation of the Document Collections

After summarizing the open questions and hypotheses to which we shall answer with experiments, it appears clearly that we need document collections written in different languages, possibly representing radically different types of human languages. To check whether our technique, as we expect, is better performing for specialized than general document collections, we also need representative corpora of these kinds.

Two appropriate collections are the NTCIR collection[1], and the INEX collection[2]. The INEX collection, which we introduced earlier (see Sections 2.3.1 and 3.2.3), is a collection of computer science journal articles written in English. The NTCIR collection contains news-feed documents in four distinct languages, namely, English, Japanese, Chinese and Korean. The corresponding collections will permit to confirm or disprove the domain-independence claim we made about our technique, by comparing the results we obtain with scientific and news-feed articles in English, i.e., specialized and non-specialized terminology. Since Chinese is a typical isolating language, and Japanese a typical agglutinative one, we will also be able to measure the performance evolution of our technique versus radically different languages.

NTCIR

With the aim to promote information retrieval research on East Asian languages, the Japanese National Institute of Informatics (NII) has made a number of collections of newspaper articles available in English, Japanese, Chinese and Korean under the acronym NTCIR, standing for "NII Test Collection for IR systems". Since these collections are meant for evaluating the performance of document retrieval systems, they are provided with a set of topics and associated manual relevance assessments. A sample topic in English is shown in Figure 5.5. Our experiments will only use the concept element (<CONC>) that gathers keywords relevant to the topic. As a general rule, keyphrases are comma-separated, which

[1]details available at http://research.nii.ac.jp/ ntcadm/index-en.html
[2]details available at http://inex.is.informatik.uni-duisburg.de/2005/

```
<TOPIC>
<NUM>013</NUM>
<SLANG>CH</SLANG>
<TLANG>EN</TLANG>
<TITLE>NBA labor dispute</TITLE>
<DESC>To retrieve the labor dispute between the two
parties of the US National Basketball Association at
the end of 1998 and the agreement that they reached.
</DESC>
<NARR>
<REL>The content of the related documents should
include the causes of NBA labor dispute, the relations
between the players and the management, main
controversial issues of both sides, compromises after
negotiation and content of the new agreement, etc. The
document will be regarded as irrelevant if it only
touched upon the influences of closing the court on
each game of the season.</REL>
</NARR>
<CONC>NBA (National Basketball Association), union,
team, league, labor dispute, league and union,
negotiation, to sign an agreement, salary, lockout,
Stern, Bird Regulation.</CONC>
</TOPIC>
```

Figure 5.5: An NTCIR topic in English.

Table 5.3: Number of documents and fully assessed topics in the NTCIR-3 collection, per language.

Language	Documents	Topics
Chinese	381,681	42
Japanese	220,078	42
Korean	66,146	30
English	22,927	30

simplifies greatly their extraction from the topics.

In the experiments, we used the NTCIR-3 document collections, about which statistics are summarized in Table 5.3. A sample of a Japanese document is shown in Figure 5.6.

INEX

The document collection of the Initiative for the Evaluation of XML retrieval (INEX[3]) is a 494Mb collection of 12,107 English-written computer science articles from IEEE journals. Naturally, much research has been directed at using the deep XML structure of this collection, but in the following experiments, we will ignore the XML markup and only use the collection for plain text document retrieval, since our purpose here is to find a corpus with specialized terminology to confront to the non-specialized English corpus of NTCIR.

We carried out experiments based on the set of 30 topics and corresponding assessments of the 1^{st} INEX initiative. We have only used the Keyword element of each topic, of which an example was shown earlier in Figure 5.3. A sample document was also shown in Figure 2.2. Following the requirements of XML retrieval, the assessments follow a 2-dimensional scale, where one dimension represents relevance and the other represents exhaustivity. For our purpose, we ignored the exhaustivity values and used solely the relevance assessments.

The inaccuracy of the recall base needs to be underlined. The evaluation procedure was meant for XML retrieval systems, and

[3]available at http://inex.is.informatik.uni-duisburg.de/2005/

<DOC>
<DOCNO>JA-980101001</DOCNO>
<LANG>JA</LANG>
<SECTION>１面</SECTION>
<AE>無</AE>
<WORDS>742</WORDS>
<HEADLINE>［社告］「第３９回毎日芸術賞」決まる</HEADLINE>
<DATE>1998-01-01</DATE>
<TEXT>
　第３９回毎日芸術賞（１９９７年度）の受賞者が決まりました。この賞は当年度、優れた芸術活動をした個人・団体に贈るもので、各分野の多数の専門家のご意見を参考に毎日新聞社が選定しました。
……
</TEXT>
</DOC>

Figure 5.6: A sample Japanese document of the NTCIR collection.

thus the set of true positives contains different types of XML elements. Since we only return articles, the maximal recall we can reach is far below one, and the results we will present will seem very weak, compared to the NTCIR collection results. This is only due to the rough use of an XML retrieval assessment as a document retrieval one.

5.5 Results

5.5.1 Generalities

An important point of the contributions of this dissertation is the development of language- and domain-independent techniques. This is put in practice in the following experiments. We have used no list of stopwords, and have applied no stemming. The only exception we made to this rule is in fact applicable to all languages: sentences are delimited by punctuation. We, hence, used every item in the text as a feature, with the exception of punctuation marks (e.g., periods, commas, parentheses, exclamation and question marks).

Encoding

For characters of the Roman alphabet, words can be defined as space-separated units. This approach does not function for most Asian characters, where there may be no whitespace at all. Korean is in fact an exception, but to keep generality, we opted for a very low-level approach, based on character encodings. In the main character settings (e.g., EUC or BIG5 encodings), Asian characters are represented by multiple bytes, while Roman characters are represented by a single byte. Simple regular expressions applied to the hexadecimal code of a text permit to detect the beginning and the end of a character.

We used this bit of knowledge to build the document features on which we based all of our experiments. What we will later refer to as "a word" is obtained as follows.

- Characters encoded with multiple bytes are represented by a concatenation of the hexadecimal code of the corresponding bytes.
- Characters encoded on a single byte are concatenated to adjacent characters that are also encoded on a single byte.
- The subsequent units are space-separated.

The input of our technique is a document representation that consists of a space-separated sequence of characters from Asian alphabets and of strings of characters from the Roman alphabet. The corresponding representation of the document sample of Figure 5.6 is shown in Figure 5.7. We can verify the presence of full stops and observe that the two numbers occurring in the original document ("39" and "1997") are not replaced by hexadecimal code and concatenated (on the second and third line of Figure 5.7). This is because they are formed of characters encoded on a single byte.

MFS extraction

We applied $MFS\_MineSweep$ to all document collections with the following settings. First, following the recommendation of Chapter 3, we used a sentence-level granularity for the MFS extraction. In other words, preprocessing of the MFS extraction has split the

```
a1ce bcd2 b9f0 a1cf a1d6 c2e8 39 b2f3 cbe8 c6fc b7dd
bdd1 bede a1d7 b7e8 a4de a4eb a1a1 c2e8 39 b2f3 cbe8
c6fc b7dd bdd1 bede a1ca 1997 c7af c5d9 a1cb a4ce bcf5
bede bcd4 a4ac b7e8 a4de a4ea a4de a4b7 a4bf . a4b3
a4ce bede a4cf c5f6 c7af c5d9 a1a2 cda5 a4ec a4bf b7dd
bdd1 b3e8 c6b0 a4f2 a4b7 a4bf b8c4 bfcd . c3c4 c2ce
a4cb c2a3 a4eb a4e2 a4ce a4c7 a1a2 b3c6 caac ccee a4ce
c2bf bff4 a4ce c0ec cce7 b2c8 a4ce a4b4 b0d5 b8ab a4f2
```

Figure 5.7: A sample of the representation of the Japanese document shown in Figure 5.6.

Table 5.4: Number of sentences and clusters in the application of $MFS\_MineSweep$ to the NTCIR-3 and INEX collections.

Collection	Sentences	Clusters
NTCIR3-Chinese	5,697,191	114
NTCIR3-Japanese	3,140,092	62
NTCIR3-Korean	617,947	12
NTCIR3-English	510,798	10
INEX (English)	4,823,813	96

collection into sentences. The sentences were then clustered into homogeneous subcollections using k-means, where the value of k was uniformly decided to be 1 per $50,000$ sentences. The following number of clusters for each collection is shown in Table 5.4.

A weakness and a consequence of our focus on generality is that the detection of sentence boundaries was often faulty. Full stops (e.g., periods, exclamation and question marks) often do not truly indicate a sentence boundary, as in "e.g." or "George W. Bush". More advanced heuristics permit to detect sentence boundaries more accurately, but they are essentially language-dependent.

Table 5.5: MFS-Adv. Summary of Mean Average Precision for the five different variations of MFS-Adv.

	Adj_Baseline	Balanced	No_Inv	Dist_pen	Max_d
NTCIR-CH	**0.1885**	0.1818	0.1837	0.1846	0.1820
NTCIR-JP	0.2232	0.2154	**0.2246**	0.2190	0.2189
NTCIR-KR	0.1370	**0.1498**	0.1477	0.1378	**0.1499**
NTCIR-EN	0.2186	0.2180	**0.2208**	0.2162	0.2180
INEX(EN)	**0.04370**	0.04193	0.04193	0.04193	0.04193

5.5.2 Results and Discussion

Tuning the matching parameters

For each collection, our novel matching technique will be applied to the MFS-based collection representation to produce one retrieval run (MFS-Adv). This requires finding good parameter values for each collection. We have computed the five runs described in Table 5.2 for each collection, and we will use the results to determine the score of MFS-Adv, and to verify the Hypotheses H1 and H2, claiming that our technique should do best for agglutinative languages and specialized collections, as opposed to isolating languages and general collections. The hypotheses further suggested that agglutinative languages and specialized collections should benefit more from raising the maximal distance or lowering the distance and inversion penalties than isolating languages and general collections. This is what we will check with the five runs presented in Table 5.2, whose corresponding results are now given in Table 5.5.

The confirmation of our assumptions is clear for Chinese, whose isolating nature is shown by the best performance observed when only adjacent non-inverted pairs are considered. As compared to the "Balanced" parameter values, suppressing inverted pairs and penalizing distance more heavily are both beneficial. The only feature for which we cannot confirm our assumptions is the augmentation of the maximal distance. The results are then very similar to those of the "Balanced" run.

The same idea is confirmed with NTCIR-KR, where the agglutinative nature of the Korean language is shown by the domination

of the runs in which few restrictions are applied on relative word positions. Using adjacent non-inverted pairs only (0.1370) and emphasizing the distance penalty (0.1378) perform far worse than the other three attempts. Increasing the maximal distance permitted the best performance (0.1499), but the improvement over the balanced parameter set was not significant. Surprisingly, allowing for the inversion of the word pairs affected the results negatively. We mentioned earlier, in Section 5.5.1, that a particularity of the Korean language is that words are space-separated. To preserve generality, we chose to ignore this fact. The negative impact of allowing for inverted (character) pairs may well be a consequence of this decision. Inverting *words* should not be a problem in an agglutinative language, but the only outcome of inverting individual characters may be to destroy the words and create non-sense. Worse, these inversions may accidentally form other words.

Japanese is a very typical agglutinative language, yet we observed the same phenomenon. The run that does not account for inverted pairs is the best-performing of all. The second best is obtained with adjacent non-inverted pairs. However, we could verify that allowing for a longer distance is beneficial for the Japanese collection, as with other things equal, we obtained better results with a maximal distance of 10 (0.2189) than with a maximal distance of 5 (0.2154).

When varying the parameter values, it turns out to be impossible to study the evolution of the results for the two English collections for the simple reason that there is nearly no evolution. The reason is that the queries are typically much shorter. In practice, there is no difference in using a maximal distance of 5 or 10 words, because none of the queries are long enough. The other parameter variations produce insignificant differences.

Now that we have determined suitable parameter values for our matching technique for each document collection, we can present a summary of our results of this chapter (Table 5.6). The results will be further analyzed in the following sections.

Table 5.6: Summary of Mean Average Precision for our experiment set.

	WVSM	VSM-AdjBig	MFS-Big	MFS-Adv
NTCIR-CH	0.1705	**0.1955**	0.1327	*0.1885*
NTCIR-JP	0.2151	**0.2808**	0.1480	*0.2246*
NTCIR-KR	0.1707	**0.2334**	0.1049	*0.1499*
NTCIR-EN	0.2555	**0.2836**	*0.2692*	0.2208
INEX(EN)	0.04193	**0.05298**	*0.04935*	0.04370

Better results for agglutinative languages and specialized collections (*Hypotheses* $H1$ and $H2$)

Agglutinative and isolating languages (H1). For the four NTCIR collections, if we compare the results obtained with the word term vector space model (column WVSM) to those obtained with our technique (column MFS-Adv), we can notice that our technique provides better results for Chinese and Japanese, while it is beaten for English and Korean. Hypothesis H1 was that our technique would perform better for agglutinative languages than for isolating languages. Chinese and Japanese respectively are often cited as very typical of the isolating and agglutinative families of languages. Additionally, English is considered isolating and Korean agglutinative.

Hence, our results do not confirm H1, as we obtained an increase in MAP for both Chinese (+10.6%) and Japanese (+4.4%), while the outcome was a decrease for both English (-13.6%) and Korean (-12.2%). Our results thus contradict $H1$. They neither indicate a better performance for agglutinative nor for isolating languages.

Specialized and general collections (H2). By similarly opposing the differences between the MAP results of the word terms vector space model (WVSM) and of our technique (MFS-Adv) for the specialized INEX collection and the NTCIR English news-feed collection, we can observe that only the INEX collection obtains better results with MFS-Adv (+4.2%). The specificity of the collection truly seems to make a difference, as opposed to the MAP decrease observed with the English NTCIR collection (-13.6%).

H2 is therefore confirmed, as we obtain **better performance**

for the specialized collection.

5.5.3 Impact of our Matching Technique (Q1)

Looking at Table 5.6, we can extend the comments we made as we verified the hypotheses H1 and H2. As compared to the word term vector space model (WVSM), the impact of our matching technique was beneficial for NTCIR-CH (+10.%), NTCIR-JP (+4.4%) and the collection of English-written computer science articles of INEX (+4.2%). On the other hand, the retrieval of NTCIR-KR (-12.2%) and NTCIR-EN (-13.6%) was more successful with a word-based vector model.

As mentioned in the protocol of the experiments, to truly evaluate the impact of our technique and not the impact of MFSs as descriptors for document retrieval, we should actually compare the results of MFS-Adv to those of MFS-Big. MFS-Big is the approach where the adjacent bigrams occurring in the set of phrasal descriptors are added as extra dimensions of the vector space model. The comparison of our technique to MFS-Big shows a decrease for both English collections, -11.4% for the INEX collection and -18.0% for NTCIR-EN. A very clear improvement is, however, observed for all three Asian languages. For Japanese, the MAP improvement is as high as +51.2%. Comparably high benefits are observed for Chinese (+42.0%) and Korean (+42.3%).

The main difference between the way we processed the English and Asian document collections is that we formed words in the English collection, while we worked at the character level for the three Asian collections. This difference of granularity may be a good explanation for the clear improvement brought by our technique in one case, and for the harm it did in the other. This would indicate that the benefit of MFS-based descriptors is linked to the granularity of the items at hand, with same-sized sequences of small items being more useful than those of large items. In other words, a sequence of 5 characters would be more beneficial than a sequence of 5 words, because a sequence of 5 words is too specific. Consequently, our technique permits a higher improvement versus a 2-gram baseline, when the grams represent smaller items, e.g., characters rather than words. Unfortunately, our poor knowledge of Asian languages does not permit guaranteeing the validity of this explanation.

It is important to note that despite our initial goal of separating both cases, the evaluation of the matching technique is very related to that of MFSs for document retrieval. We have indeed been able to try different parameters for the exploitation of the topics, but we have always used the same sets of phrasal descriptors, stemming from *MFS_MineSweep*. One must bear in mind that, even when only adjacent pairs of words are selected from the phrasal description, the original distance that separated them in the original documents is only limited by the length of the sentence of origin.

5.5.4 Quality of MFSs as Indexing Terms for Document Retrieval (Q2)

For all the collections, the best MAP is always obtained by VSM-AdjBig, the experiment where all the adjacent word pairs of the collection are represented by dimensions of the vector space together with the word terms.

The use of MFS for document retrieval therefore appears questionable. We should, however, underline that the difference between the best use of MFSs and VSM-AdjBig is most often decent. For Chinese and the two English collections, it is only -3.6%, -5.1% and -6.9% respectively. Moreover, this difference is compensated by the use of a far lower number of descriptors. The numbers of feature terms used in WVSM and VSM-AdjBig are shown in Table 5.7 together with the number of distinct bigrams of the MFS-based phrasal descriptions. The percentages indicate the increase of terms, compared to the word-based vector space. Observe that, for WVSM and VSM-AdjBig, the number of features is the number of dimensions of the vector space.

For the three Asian languages, the number of features is about 3 times higher for VSM-AdjBig than it is for MFS-Adv. The difference is 15- to 25-fold for English. This high number of features has a subsequent cost that cannot be neglected.

The use of MFSs in document retrieval may then be thought of as a compact representation of collections, performing a bit worse, but more efficient to exploit.

Table 5.7: Number of features, and percentage of augmentation compared to using only word terms.

	WVSM	VSM-AdjBig	MFS-Adv
NTCIR-CH	56,093	3,238,957 (+5774%)	1,014,109 (+1810%)
NTCIR-JP	40,182	1,103,594 (+2746%)	424,134 (+1056%)
NTCIR-KR	19,876	411,717 (+2071%)	176,623 (+889%)
NTCIR-EN	81,861	1,363,445 (+1566%)	49,914 (+61%)
INEX(EN)	423,195	5,697,211 (+1246%)	328,892 (+78%)

5.5.5 Results in the Context of Related Work

To relate our results to comparable related work is difficult for the simple reason that there is very little comparable work. For the INEX collection, we have returned full documents, whereas the aim of the INEX initiative is to retrieve XML elements. As we mentioned earlier, this means notably that most of the recall base was out of reach of our system. In the first INEX initiative, the results of the runs that we presented in Table 5.6 would have placed 18^{th} (WVSM), 9^{th} (VSM-AdjBig), 13^{th} (MFS-Big) and 16^{th} (MFS-Adv) positions out of 49 official runs submitted. Ironically, in this first edition of the XML retrieval initiative, returning full documents permits to obtain fairly good performance. This is good motivation for the need to perform research in XML retrieval, but this result is also due to a weakness of the evaluation system in use in INEX at the time. It took neither the length of returned documents nor the amount of overlap in submitted runs into account. The only consequence of this evaluation flaw for full document retrieval is unduly low recall values, but this does not affect the relative comparison of mean average precision values. The position of most of our runs in the first third of the ranking shows the coherence of our experimental benchmark in the real world. In addition, most of the other approaches used word stemming, stoplists, relevance feedback, and so on. All those techniques permit performance improvement, but we left them aside to preserve generality.

The NTCIR is the only evaluation forum for the retrieval of some of the languages it addresses. Hence, a number of purely language-

Table 5.8: Mean average precision and relative comparison with a comparable system presented in the NTCIR-3 workshop.

Collection	NTCIR-CH	NTCIR-JP	NTCIR-KR	NTCIR-EN
Juang et al.	0.2403	0.2674	0.2675	N/A
VSM-AdjBig	0.1955	0.2808	0.2334	0.2836
Difference	-18.6%	+5.0%	-12.7%	N/A

dependent techniques are indirectly evaluated in the NTCIR forum. Other low-level techniques that we did not use also seem to permit considerable improvement. For instance, it is a common preprocessing of the Chinese collection to convert it from "Traditional Chinese" to "Simplified Chinese", a simplified character set developed by the Chinese government in the 1950s to try to eliminate unnecessary variations. In Japanese, there exist three main scripts that are commonly dealt with differently, namely *Kanji*, *Hiragana* and *Katakana*. And last but not least, the main focus of NTCIR is cross-language information retrieval, a task we do not address in this dissertation. The very few papers that presented generalist techniques did use longer topic parts and were not quite language-independent, with the exception of a technique presented by Juang and Tseng [JT03]. They identically applied a phrase-extraction technique, originally built for Chinese, to the Japanese and Korean collections. This method has been tuned for Chinese and, reportedly, the character associations it extracts permit to form words with an accuracy of 86%. In a sense, even though this technique is truly language-independent, it still benefited from the knowledge of its authors, at least in the Chinese language. They could check what their technique extracted and improve it to obtain the extraction of meaningful units most often corresponding to words. We, on the other hand, have no idea what our descriptors represent, besides maximal frequent sequences of characters from homogeneous partitions of the collection. The outcome, using the same topic part and evaluation measure as we did, is shown in Table 5.8. Their results demonstrate comparable effectiveness in all three languages. An illustration of the orientation of their method is that it is precisely for Chinese that their results outperform ours very clearly (-18.6%).

For Japanese, our version of the vector space model that includes words and adjacent bigrams actually obtains a better performance (+5.0%). Unfortunately, they did not report any experiment with the English data set, where the Chinese-oriented origins of the technique may not have benefited in the same way. Given our level of ignorance of the languages at hand, we believe that the performance of our system compares honorably.

5.5.6 Conclusion

We presented a novel technique for measuring the similarity of phrasal document descriptors and combining it to word-based vector space similarity measures. After a short review on the use of phrases in information retrieval, we applied our technique to the problem of document retrieval, where we compared the MFS-based phrasal representations of documents to sets of keyphrases describing user needs.

Due to a number of adjustable parameters, our method allows accounting for occurrences of the words of a phrase over a longer span, or in a different order. These usages may be gradually penalized, as compared to an exact phrase occurrence, i.e., adjacent words occurring in the same order. This approach permits taking a wide variation of word usages into account.

It notably deals with the problem of overlapping phrases, as described by Vechtomova [Vec05]. She states the problem of overlapping phrases as the fact that, given a query ABC, a document containing the exact match ABC and a document containing AB and BC separately both obtain the same score in numerous approaches of the state of the art. A subsequent issue is that the weight of the word B becomes artificially heavier than that of A and C, because B is present in both pairs AB and BC. Our technique permits eradicating this problem, since it can also take the pair AC into account. Hence, the distance one between A and C in the first document (with ABC) ensures that it gets a better score than the second document (with AB and BC). Another consequence is that the weights of A and C are increased along with that of B, avoiding to unbalance the individual term weights within the phrase. A weakness, however, remains with this approach: the word terms that belong to a long phrase appear in numerous sub-

pairs, and hence their artificial weight increase is more important than that of a word occurring in a shorter phrase. Notably, the weight of individual word terms that do not occur in a keyphrase is made lower in comparison to that of word terms occurring in a keyphrase. A solution would be to normalize the weight of terms upon the number and size of the phrases they occur in. This problem is not straightforward, as was recently suggested by work of Robertson et al. [RZT03] who suggested subtracting the individual weight of words that occurred redundantly in keyphrases and obtained very disappointing results.

Our experiments suggested that MFSs may not be very appropriate for document retrieval, or that a way to exploit them remains to be found. When it comes to our novel similarity measure, the experiments showed mixed results. As compared to throwing all descriptors in a bag of words, it is more efficient, as it uses less features. The quality of the results was further greatly improved for the NTCIR collections in Chinese, Japanese and Korean, with encouraging amelioration ranging between +42% and +51%. This suggests that exploiting languages at a character level may well be the appropriate way for applying our technique with worthwhile improvement. The nature of most Asian alphabets is well suited therefore.

One must underline the better overall performance of representing every word and adjacent bigram by a dimension of the vector space. Given the far better performance of our matching technique for taking advantage of MFSs in the three Asian collections, a natural continuation of this work will be to experiment on our similarity measure with better performing phrasal descriptors.

CHAPTER 6

Conclusions

In this thesis, we have considered the use of the sequential nature of text for document representations. A central point of our work is a voluntary choice of generality. We developed techniques that are suited for document collections of any type and that may be written in any language.

The first result of our work is the development of an efficient technique for the extraction of a compact set of word sequences from text. Built upon $MineMFS$, an existing technique for the extraction of maximal frequent sequences from text, we proposed $MFS\_MineSweep$, an improvement of this technique that is based on splitting the original document collection into homogeneous partitions. The outcome is that we can obtain more exhaustive results faster. Further, a drawback of $MineMFS$ is that it fails to produce any descriptor at all for large document collections, whereas our contribution permits to extract phrasal descriptors out of a collection of virtually any size.

The following contribution of this thesis permits filling a gap in the current research on multi-word units and their use for information retrieval applications. We presented efficient algorithms for computing the probability and expected frequency of occurrence of a given sequence of items. One application of this technique is the direct evaluation of sequences, notably word sequences, obtained by interestingness measures that are calculated by comparing the expected and observed document frequencies of a sequence. The more the hypothesis that a sequence occurs by pure chance is wrong, the more that sequence is interesting with respect to the corpus. Our

technique offers an efficient alternative to the current evaluation methods of word sequences. These techniques are indeed essentially task-based, relying on time-consuming manual assessments, as is the case in lexicography, or embedded within an application framework as is usually done in information retrieval. The weakness of indirect evaluation is that it remains difficult to decide whether any result stems from the quality of the phrases or from the way they were used. The evident benefit of a direct evaluation technique such as ours is that its results are easier to interpret, as neither intervenes a separate application, nor a subjective human judgment.

Our last contribution results from exploratory work on an attempt to use MFS-based phrasal descriptors in document retrieval. We developed a new phrase-based similarity measure and experimented with it with MFS-based descriptors on a number of document collections, written in four different languages. Our experiments did not demonstrate a result improvement for the use of MFS-based descriptors, as opposed to a straightforward utilization of adjacent bigrams. Our phrase-based similarity measure, however, clearly outperformed the incorporation of bigram features in the vector space model for document collections written in three Asian language collections. An under-performance was, however, observed with English collections, defined at the word level. Since the benefit in using maximal frequent sequences was not demonstrated for document retrieval, a natural continuation of this work is to experiment our phrase-based similarity measure with other phrasal descriptors, and check whether this provides comparable improvement as it did with MFS-based descriptions. The significance of the amelioration obtained with the Asian language document collections allows for some optimism in this case.

References

[AAB+03] James Allan, Jay Aslam, Nicholas Belkin, Chris Buckley, Jamie Callan, Bruce Croft, Sue Dumais, Norbert Fuhr, Donna Harman, David J. Harper, Djoerd Hiemstra, Thomas Hofmann, Eduard Hovy, Wessel Kraaij, John Lafferty, Victor Lavrenko, David Lewis, Liz Liddy, R. Manmatha, Andrew McCallum, Jay Ponte, John Prager, Dragomir Radev, Philip Resnik, Stephen Robertson, Roni Rosenfeld, Salim Roukos, Mark Sanderson, Rich Schwartz, Amit Singhal, Alan Smeaton, Howard Turtle, Ellen Voorhees, Ralph Weischedel, Jinxi Xu, and ChengXiang Zhai. Challenges in information retrieval and language modeling: report of a workshop held at the center for intelligent information retrieval, university of massachusetts amherst, september 2002. *SIGIR Forum*, 37(1):31–47, 2003.

[AM05] Helena Ahonen-Myka. Mining all maximal frequent word sequences in a set of sentences. In *Proceedings of the 2005 International Conference on Information and Knowledge Management, poster session, October 31 - November 5, 2005, Bremen, Germany*, pages 255–256. ACM, 2005.

[AMD05] Helena Ahonen-Myka and Antoine Doucet. Data mining meets collocations discovery. In *Inquiries into Words, Constraints and Contexts*, pages 194–203. CSLI Publications, Center for the Study of Language and Information, University of Stanford, 2005.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip Yu and Arbee Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[BM05] Holger Bast and Debapriyo Majumdar. Why spectral retrieval works. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA, 2005. ACM Press.

[BP03] Satanjeev Banerjee and Ted Pedersen. The design, implementation, and use of the Ngram Statistic Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, February 2003.

[BPSM+04] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Francois Yergeau. Extensible Markup Language (XML) 1.0 - W3C recommendation 04 february 2004. Technical Report REC-xml-20040204, 2004.

[Cai02] Guoray Cai Cai. Geovsm: An integrated retrieval model for geographic information. In *GIScience '02: Proceedings of the Second International Conference on Geographic Information Science*, pages 65–79, London, UK, 2002. Springer-Verlag.

[CH90] Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[CH03] Nick Craswell and David Hawking. Overview of the TREC-2003 Web Track. In *Proceedings of TREC-2003*, Gaithersburg, Maryland USA, November 2003.

[CH04] Nick Craswell and David Hawking. Overview of the TREC-2004 Web Track. In *Proceedings of TREC-2004*, Gaithersburg, Maryland USA, November 2004.

[CKN83] Yaacov Choueka, Shmuel T. Klein, and E. Neuwitz. Automatic retrieval of frequent idiomatic and collocational expressions in a large corpus. *Journal for Literary and Linguistic computing*, 4:34–38, 1983.

[Cla05] Charles L. A. Clarke. Controlling overlap in content-oriented xml retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 314–321, New York, NY, USA, 2005. ACM Press.

[Cry00] David Crystal. *Language Death*. Cambridge University Press, first edition, 2000.

[Cry03] David Crystal. *The Cambridge Encyclopedia of the English Language*. Cambridge University Press, second edition, 2003.

[CSW97] Michal Cutler, Yungming Shih, and Meng Weiyi. Using the structure of html documents to improve retrieval. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (NISTS'97)*, 1997.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6, 1987.

[DALP03] Antoine Doucet, Lili Aunimo, Miro Lehtonen, and Renaud Petit. Accurate retrieval of XML document fragments using EXTIRP. In *Proceedings of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX 2003)*, ERCIM Workshop Proceedings, Schloss Dagstuhl, Germany, 2003.

[DAM02] Antoine Doucet and Helena Ahonen-Myka. Naive clustering of a large xml document collection. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 81–87, Schloss Dagsuhl, Germany, 2002.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DGBPL00a] Gaël Dias, Sylvie Guilloré, Jean-Claude Bassano, and José Gabriel Pereira Lopes. Combining linguistics with statistics for multiword term extraction: A fruitful association? In *Proceedings of Recherche d'Information Assisteé par Ordinateur (RIAO 2000)*, 2000.

[DGBPL00b] Gaël Dias, Sylvie Guilloré, Jean-Claude Bassano, and José Gabriel Pereira Lopes. Extraction automatique d'unités complexes: Un enjeu fondamental pour la recherche documentaire. *Traitement Automatique des Langues*, 41(2):447–472, 2000.

[Dou04] Antoine Doucet. Utilisation de séquences fréquentes maximales en recherche d'information. In *Proceedings of the 7th International Conference on the Statistical Analysis of Textual Data (JADT-2004)*, pages 334–345, Louvain-La-Neuve, Belgium, March 2004. JADT-2004.

[Est16] Jean-Baptiste Estoup. *Gammes Sténographiques*. Institut Sténographique de France, Paris, fourth edition, 1916.

[FA96] Katerina Frantzi and Sophia Ananiadou. Extracting nested collocations. In *Proceedings of the 16th conference on Computational linguistics (COLING)*, pages 41–46, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[Fag89] Joel L. Fagan. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, 40:115–132, 1989.

[Fan61] Robert M. Fano. *Transmission of Information: A statistical Theory of Information.* MIT Press, Cambridge MA, 1961.

[FAT98] Katerina Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii. The c-value/nc-value method of automatic recognition for multi-word terms. In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 585–604. Springer-Verlag, 1998.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley Publications, third edition, 1968.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* The MIT Press, 1998.

[FGKL02] Norbert Fuhr, Norbert Gövert, Gabriella Kazai, and Mounia Lalmas, editors. *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany, December 9-11, 2002*, 2002.

[FLM03] Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors. *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany, December 15-17, 2003*, 2003.

[FLMS05] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors. *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2005.

[Fox83] Edward A. Fox. Some considerations for implementing the smart information retrieval system under unix. Technical Report TR 83-560, Department of

Computer Science, Cornell University, Ithaca, NY, September 1983.

[GJ05] Raymond G. Gordon Jr., editor. *Ethnologue: Languages of the World*. SIL International, fifteenth edition, 2005.

[Hav03] Taher H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

[HJ94] Roger A. Horn and Charles R. Johnson. *Topics in matrix analysis*. Cambridge University Press, New York, NY, USA, 1994.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM Press.

[IV98] Nancy Ide and Jean Véronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40, 1998.

[JS74] Stuart Jones and John Mc Hardy Sinclair. English lexical collocations: A study in computational linguistics. *Cahiers de Lexicologie*, 24:15–61, 1974.

[JT03] Da-Wei Juang and Yuen-Hsien Tseng. Uniform indexing and retrieval scheme for chinese, japanese, and korean. In Keizo Oyama, Emi Ishida, and Noriko Kando, editors, *Proceedings of the Third NTCIR Workshop on Evaluation of Information Retrieval, Automatic Text Summarization and Question Answering*, pages 137–141. National Institute of Informatics (NII), 2003.

[Kar00] Jussi Karlgren. *Stylistic Experiments for Information Retrieval*. PhD thesis, Stockholm University, 2000.

[KF67] Henry Kucera and Nelson Francis. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, Rhode Island, 1967.

[KLdV04] Gabriella Kazai, Mounia Lalmas, and Arjen P. de Vries. The overlap problem in content-oriented xml retrieval evaluation. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 72–79, New York, NY, USA, 2004. ACM Press.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[KRS05] Jaap Kamps, Maarten De Rijke, and Börkur Sigurbjörnsson. The importance of length normalization for xml retrieval. *Information Retrieval*, 8(4):631–654, 2005.

[Lah00] Timo Lahtinen. *Automatic Indexing: an approach using an index term corpus and combining linguistic and statistical methods*. PhD thesis, University of Helsinki, 2000.

[Lee95] Joon Ho Lee. Combining multiple evidence from different properties of weighting schemes. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 180–188. ACM Press, 1995.

[Lew92] David Dolan Lewis. *Representation and learning in information retrieval*. PhD thesis, University of Massachusetts at Amherst, 1992.

[Luh57] Hans Peter Luhn. A statistical approach to mechanical encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[MBSC97] Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200–214, 1997.

[MKR02] Maarten Marx, Jaap Kamps, and Maarten de Rijke. The university of amsterdam at inex 2002. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 24–28, Schloss Dagsuhl, Germany, 2002.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge MA, second edition, 1999.

[MTV95] Heikki Mannila, Hannu Toivonen, and Inkeri Verkamo. Discovering frequent episodes in sequences. In *KDD*, pages 210–215, 1995.

[ND77] Ben Noble and James W. Daniel. *Applied Linear Algebra*, pages 361–367. Prentice Hall, second edition, 1977.

[NJ02] Andrew Nierman and H.V. Jagadish. Evaluating Structural Similarity in XML. In *Fifth International Workshop on the Web and Databases (WebDB 2002), Madison, Wisconsin*, 2002.

[otI95] United States Department of the Interior. *U. S. Geological Survey, Geographic Names Information System: Data Users Guide 6*. Reston, VA, USA, 4th printing, revised edition, 1995.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[PWCB00] Gordon W. Paynter, Ian H. Witten, Sally Jo Cunningham, and George Buchanan. Scalable browsing for large collections: a case study. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 215–223, New York, NY, USA, 2000. ACM Press.

[QLZ+05] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, Zheng Chen, and Wei-Ying Ma. A study of relevance propagation for web search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 408–415, New York, NY, USA, 2005. ACM Press.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[Reu87] Reuters-21578. Text categorization test collection, distribution 1.0, 1987.
http://www.daviddlewis.com/resources/testcollections/reuters21578.

[RLHJ99] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification - W3C recommendation 24 december 1999. Technical Report REC-html401-19991224, 1999.

[RZT03] Stephen E. Robertson, Hugo Zaragoza, and Michael Taylor. Microsoft cambridge at trec-12: Hard track. In *TREC*, pages 418–425, 2003.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[SB88] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information*

Processing and Management: an International Journal, 24(5):513–523, 1988.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[SC96] Tomek Strzalkowski and Jose Perez Carballo. Natural language information retrieval: TREC-4 report. In *Text REtrieval Conference*, pages 245–258, 1996.

[SDGPL99] Joaquim Ferreira da Silva, Gaël Dias, Sylvie Guilloré, and José Gabriel Pereira Lopes. Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units. In *Proceedings of the 9th Portuguese Conference on Artificial Intelligence*, pages 113–132. Springer-Verlag, 1999.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Survey*, 34(1):1–47, 2002.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Tech*, 27:379–423, 623–656, 1948.

[Sim89] John Simpson, editor. *Oxford English Dictionary*. Oxford University Press, second edition, 1989.

[Sin97] Amitabh K. Singhal. *Term Weighting Revisited*. PhD thesis, Cornell University, 1997.

[SJ72] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[SJ99] Karen Spärck Jones. *Natural Language Information Retrieval*, pages 1–24. Dordrecht: Kluwer Academic Publishers, 1999.

[SK98] Alan F. Smeaton and Fergus Kelledy. User-chosen phrases in interactive query formulation for information retrieval. In *Proceedings of the 20th BCS-IRSG Colloquium*, 1998.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar. A Comparison of Document Clustering Techniques. In *Proceedings of KDD 2000, Workshop on Text Mining*, 2000.

[Sma93] Frank Smadja. Retrieving collocations from text: Xtract. *Journal of Computational Linguistics*, 19:143–177, 1993.

[SSW+98] Tomek Strzalkowski, Gees Stein, G. Bowden Wise, Jose Perez Carballo, Pasi Tapanainen, Timo Jarvinen, Atro Voutilainen, and Jussi Karlgren. Natural language information retrieval: TREC-6 report. In *Text REtrieval Conference*, pages 164–173, 1998.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerical Mathematics*, 13:354–356, 1969.

[Sus93] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *CIKM '93: Proceedings of the second international conference on Information and knowledge management*, pages 67–74, 1993.

[SWY75] Gerard Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11):613–620, 1975.

[SY73] Gerard Salton and C.S. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29:351–372, 1973.

[SYY75] Gerard Salton, C.S. Yang, and Clement T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975.

[SZ03] Azadeh Shakery and ChengXiang Zhai. Relevance propagation for topic distillation uiuc trec 2003 web track experiments. In *Proceedings of TREC-2003*, pages 673–677, Gaithersburg, Maryland USA, November 2003.

[TG01] Ilias Tsoukatos and Dimitrios Gunopulos. Efficient mining of spatiotemporal patterns. In *SSTD '01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pages 425–442, London, UK, 2001. Springer-Verlag.

[TM99] Andrew Turpin and Alistair Moffat. Statistical phrases for vector-space information retrieval. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 309–310, 1999.

[TVBK04] Takaaki Tanaka, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors. *Second ACL Workshop on Multiword Expressions: Integrating Processing*, 2004.

[Vec05] Olga Vechtomova. The role of multi-word units in interactive information retrieval. In *Proceedings of the 27th European Conference on Information Retrieval, Santiago de Compostela, Spain*, pages 403–420, 2005.

[Vos98] Piek Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998.

[VR79] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Department of Computer Science, University of Glasgow, 1979.

[Wil88] Peter Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[WP94] Allison Gyle Woodruff and Christian Plaunt. GIPSY: Georeferenced information processing SYstem. Technical Report S2K-94-41, 25, 1994.

[WWX+05] Lee Wang, Chuang Wang, Xing Xie, Josh Forman, Yansheng Lu, Wei-Ying Ma, and Ying Li. Detecting dominant locations from search queries. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 424–431, New York, NY, USA, 2005. ACM Press.

[WZB04] Hugh E. Williams, Justin Zobel, and Dirk Bahle. Fast phrase querying with combined indexes. *ACM Transactions on Information Systems*, 22(4):573–594, 2004.

[YBLS83] Clement T. Yu, Chris Buckley, K. Lam, and Gerard Salton. A generalized term dependence model in information retrieval. *Information Technology: Research and Development*, 2(4):129–154, 1983.

[YS00] Jeonghee Yi and Neel Sundaresan. A classifier for semi-structured documents. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts*, pages 340–344, 2000.

[Zak01] Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Mach. Learn.*, 42(1-2):31–60, 2001.

[ZK01] Ying Zhao and George Karypis. Criterion functions for document clustering. Technical report, Department of Computer Science and Engineering, University of Minnesota Twin Cities, 2001.

[ZTMFE97] Chengxiang Zhai, Xiang Tong, Natašha Milić-Frayling, and David A. Evans. Evaluation of syntactic phrase indexing. In *Proceedings of the 5th Text Retrieval Conference, TREC-5*, pages 347–358, 1997.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1993-1 E. Ukkonen: On-line construction of suffix-trees. 15 pp.

A-1993-2 Alois P. Heinz: Efficient implementation of a neural net α-β-evaluator. 13 pp.

A-1994-1 J. Eloranta: Minimal transition systems with respect to divergence preserving behavioural equivalences. 162 pp. (Ph.D. thesis).

A-1994-2 K. Pohjonen (toim./ed.): Tietojenkäsittelyopin laitoksen julkaisut 1992–93 – Publications from the Department of Computer Science 1992–93. 58 s./pp.

A-1994-3 T. Kujala & M. Tienari (eds.): Computer Science at the University of Helsinki 1993. 95 pp.

A-1994-4 P. Floréen & P. Orponen: Complexity issues in discrete Hopfield networks. 54 pp.

A-1995-1 P. Myllymäki: Mapping Bayesian networks to stochastic neural networks: a foundation for hybrid Bayesian-neural systems. 93 pp. (Ph.D. thesis).

A-1996-1 R. Kaivola: Equivalences, preorders and compositional verification for linear time temporal logic and concurrent systems. 185 pp. (Ph.D. thesis).

A-1996-2 T. Elomaa: Tools and techniques for decision tree learning. 140 pp. (Ph.D. thesis).

A-1996-3 J. Tarhio & M. Tienari (eds.): Computer Science at the University of Helsinki 1996. 89 pp.

A-1996-4 H. Ahonen: Generating grammars for structured documents using grammatical inference methods. 107 pp. (Ph.D. thesis).

A-1996-5 H. Toivonen: Discovery of frequent patterns in large data collections. 116 pp. (Ph.D. thesis).

A-1997-1 H. Tirri: Plausible prediction by Bayesian inference. 158 pp. (Ph.D. thesis).

A-1997-2 G. Lindén: Structured document transformations. 122 pp. (Ph.D. thesis).

A-1997-3 M. Nykänen: Querying string databases with modal logic. 150 pp. (Ph.D. thesis).

A-1997-4 E. Sutinen, J. Tarhio, S.-P. Lahtinen, A.-P. Tuovinen, E. Rautama & V. Meisalo: Eliot – an algorithm animation environment. 49 pp.

A-1998-1 G. Lindén & M. Tienari (eds.): Computer Science at the University of Helsinki 1998. 112 pp.

A-1998-2 L. Kutvonen: Trading services in open distributed environments. 231 + 6 pp. (Ph.D. thesis).

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. thesis)

A-2005-1 T. Mielikäinen: Summarization Techniques for Pattern Collections in Data Mining. 201 pp. (Ph.D. thesis)