EXPLORING POTENTIAL IMPROVEMENTS TO TERM-BASED CLUSTERING OF WEB DOCUMENTS

By

DAMIR ARAČIĆ

A dissertation submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Computer Science

December 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of DAMIR ARAČIĆ find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

_____

ACKNOWLEDGMENTS

I wish to express my gratitude to Dr. Scott Wallace for his invaluable mentorship, and especially for his ability to motivate me. I am grateful to the entire faculty of the School of Engineering and Computer Science for imparting their knowledge and endowing me with the skills I apply to this thesis. Most of all, I am thankful to my parents, for their constant support and unconditional love; I owe them everything.

EXPLORING POTENTIAL IMPROVEMENTS TO TERM-BASED CLUSTERING OF WEB DOCUMENTS

Abstract

by Damir Aračić, MSCS
Washington State University
December 2007

Chair: Scott Wallace

With the size and diversity of the information content on the Web growing at unfathomable pace, retrieval of desired or pertinent information becomes an increasingly difficult task. Particularly difficult is the problem of associating intended meanings of queries to their textual representation in those domains where different meanings might have nearly identical textual representation. This motivates the search for document features, other than words, that are able to express semantic relationships. We present a method for using character patterns in the space of non-words as a document feature to aids in distinguishing semantics of Web documents.

We test the value of such a concept by devising non-word patterns through observation. We then develop an automated method for learning non-word patterns from a corpus of documents. Finally, through a series of document classification experiments, we are able to show the pertinence of non-word patterns in document classification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

The rapidly increasing volume of documents on the World Wide Web makes the classification of those documents and the retrieval of desired content from the Web increasingly difficult. Text-based search engines perform in a satisfactory manner for most queries — namely, those queries where the desired meaning has a unique mapping to the entered query. However, when there are several different meanings, or concepts, associated with the same query (for instance, the way there are at least two meanings associated with the query "jaguar"), the search results will include documents that belong to different concepts, without distinction. If a text-based search engine made an attempt to deliver the search results in a manner that considered all the potential meanings that map to the particular textual query, it would meet with limited success, for the simple reason that it is not always possible to uniquely map classes of meaning to sets of words without overlap; that is to say that identical words might be found represented in significant volume in documents that represent differing concepts.

Thus the problem we are attempting to address is how to improve on classifying documents that are similar text-wise, but distinct in their subject (such as documents about the car Jaguar and documents about the animal jaguar). Our approach is to use features of document text, other than words, that are distinct in their quality from one class of meaning to another. It is our hypothesis that there are patterns of text that exist outside of the realm of words such that distinct sets of these non-word patterns can be isolated for every class of meaning, and that using these patterns of text will improve the classification precision.

Performances of current Web search engines are most often supported by extensive, large database-driven statistics, as well as domain-specific *a priori* information, such as topic hierarchies. In contrast, our approach concentrates on improving *ad hoc* information retrieval methods.

1.1.1 Contributions of the thesis

As it concerns information retrieval, we contribute the realization that there is reasonable semantic value in the space of non-words. We substantiated with experiments that observing content of text documents can result in regular expressions in the space of non-words that improve precision of document classification. In conjunction with this, we also contribute a method for automated extraction of such patterns from text. This supervised pattern learning method is all the more valuable as it does not require any ontological knowledge of the domain it is being applied to. Furthermore, we support our automated approach with experimental results that demonstrate its viability. More generally, as part of our automated pattern learning method, we contribute a dynamic programming algorithm for generalization of regular expressions.

1.2 Information retrieval

The problem of information discovery, or information retrieval, is one of finding desired content amidst quantities of information. In the context of the World Wide Web as a source of widely available information, the problem of information discovery can be rephrased more precisely: given a user attempting to obtain information from the Web, the information discovery problem is the problem of retrieving those web pages that contain the information the user is looking for.

To be able to retrieve information from an information source, the user has to be able to:

- express the nature of the content sought, and
- explore the entirety of the content in the information source.

Expressing the sought content is commonly called forming a *query*; the process of exploring the information source is referred to as conducting *search*. The ability to apply search to some query, as well as all the intricacies involved in executing the search process over the space of the information source, is incorporated into a *search engine*. For any query that the user forms and commits to the search engine, the user will receive a *result*. In the context of the web, the result will be a set of web pages.

Whether the web pages in the result contain the desired information will depend on the several factors, namely:

- expressiveness of the query,
- completeness of the search, and
- semantic relevance of the search to the query.

While expressiveness of queries and completeness of search are important, they are not as complex as the issue of semantic relevance. Semantic relevance of the search to the query measures how well the user-formed query is interpreted by the search. It answers the question how often will the search engine, having ingested the query, return the information the user is trying to obtain, as opposed to information that is irrelevant to the user. The issue of semantic relevance is what concerns us the most and what prompts us to develop our approach throughout this thesis.

1.2.1 Naive approach

Let us consider approaching the problem of information discovery from the simplest perspective: a person (user) needs information about jaguars and inquires some source (another person, information service, the Delphi Oracle, etc.), but this user can only submit the inquiry in written form. Thus the user verbalizes the intent and records it in text:

"I want to find out about jaguars."

Now apply this reasoning to a scenario where the user knows the answer is stored within a corpus of documents, but there is no mediator to provide it. The most obvious approach would be to read all the documents in the corpus and observe which documents are about jaguars. To convert this process into an automated one, a user would build a search engine that examined every document in the corpus and answered the question

"Does this document talk about jaguars?"

for every examined document, thus compiling a list of jaguar-related documents. It is clear that the most difficult question at hand at this point is how does the search engine decide when to answer "yes" and when to answer "no".

The most obvious answer to this question is: whenever there is a sentence found in the document that reads

"This document reads about jaguars.",

answer "yes"; if that sentence is not found, answer "no".

Assuming the users accept the fact that the topic-defining sentence approach is not feasible, they are left with the realization that instead of deciding unmistakably, the search engine will be forced to hypothesize, to some degree of certainty, whether to answer "yes" or "no". In other words, the search engine will need to devise a way to measure its degree of certainty about the document's topic.

1.2.2 Vector space model

The simplest assumption to make about a document on the topic of jaguars is that the text of the document contains the word "jaguar". Furthermore, any two jaguar-related documents will most likely both contain identical jaguar-related words. Thus, for some set of words, two documents both containing that word set are more likely similar than two documents that don't contain it. Also, two documents containing a larger subset of words from that set are likely more similar to one another than to those documents that contain a smaller subset of words.

These observations lead to the formalized representations of documents as vectors. For any set of documents, any document can be expressed as an n-dimensional vector, where n is the size of the entire document set in words (or, the number of unique words across all documents of the set). Two such vector-documents can then be compared using vector operations. For instance, the cosine of the angle between them can be calculated as a direct measurement of their similarity: the more similar

	I	like	love	that	dog	flowers
d_1	1	1	0	0	0	1
d_2	1	0	2	1	1	0

Table 1.1: Term-document matrix built from corpus D.

two documents in n-dimensional space are, the smaller the angle between them and the larger the cosine of that angle.

The vector representation of a set of documents assumes the form of a term-document matrix: a matrix with documents as rows and terms (the set of words across the document set) as columns. Every cell of the term-document matrix corresponds to the occurrence count of a particular term in a particular document. Table 1.1 illustrates the term-document matrix for corpus D containing two documents, d_1 and d_2, with contents "I like flowers" and "I love, love that dog", respectively.

Tf-idf

Consider the mention of jaguar in these two contexts - in this document about melanistic jaguars: "Melanistic jaguars are popularly known as 'black panthers'.", as opposed to this document about the ancient Maya: "The Maya worshiped jaguars." It is obvious that the question arising is how to distinguish a document focusing on jaguars from a document that mentions jaguars casually.

Earliest approaches attempting to answer this question concentrated on the frequency of occurrence of particular keywords (i.e. words, phrases, keyphrases - in general, *terms*) in the document, reasoning that a higher keyword frequency indicated a higher significance to the document content, i.e. a stronger relation to the document's meaning. This *term frequency* of a term t_i in document d_j is defined as

$$(1.1)\quad tf_{i,j} = \frac{n_{i,j}}{\sum_{i=1}^{k_j} n_{i,j}},$$

	I	like	love	that	dog	flowers
d_1	0.33	0.33	0.0	0.0	0.0	0.33
d_2	0.25	0.0	0.5	0.25	0.25	0.0

Table 1.2: Term-document matrix built from corpus D and weighted with tf.

where $n_{i,j}$ is the number of occurrences of the term t_i in document d_j and k_j is the total number of unique terms in d_j. Table 1.2 shows the term-document matrix from Table 1.1 weighted with *tf*.

Consider, however, the scenario where "jaguar" is equally often mentioned in all documents, i.e., where the search engine assigns all documents in the corpus an equal value for term frequency of "jaguar". This, in fact, means that the search result contains the entire corpus of documents, implying that the user needs to read all the documents himself/herself - a practically useless solution that was rejected in the beginning of the discussion. The imminent conclusion is that the goal of the search should not only be to produce the result, but to produce a result that is both manageable in volume and organized in relevance. In fact, solving the problem of organizing the search results with respect to their relevance effectively solves the problem of the result volume, since it directs the user in his/her interpretation of the search result.

The key to relevance ordering of documents in a corpus is the realization that the importance of a term is not only measured by how often it occurs in an individual document, but also how seldom it occurs in the entire corpus as a whole. To illustrate this point, let the user query q consist of terms "jaguar" and "dog". Frequencies for each term in every document will be computed by the search engine. Assume the corpus contains 1) documents that refer to only dogs, and 2) documents that refer to dogs and jaguars (in some joint context); there might or might not be other documents

	I	like	love	that	want	a	dog	and	jaguar	this	not	just	any	flowers
d_1	1	1	0	0	0	0	0	0	0	0	1	0	0	0
d_2	1	0	2	1	0	0	1	0	0	0	0	0	0	0
d_3	1	0	0	0	1	1	1	1	1	0	0	0	0	0
d_4	1	0	0	0	1	0	3	0	0	1	1	1	1	0

Table 1.3: Term-document matrix built from corpus D'.

RANK	DOCUMENT	OCCURRENCES
1	d_4	3
2	d_3	2
3	d_2	1
4	d_1	0

Table 1.4: Order of salience of documents in corpus D' to query q according to the simple occurrence matrix.

that don't refer to either. Further assume there are such documents from 1) where the term frequency of "dog" is higher than the highest combined term frequencies for some document in 2). This is illustrated in the term-document matrix built from corpus $D' = \{d_3, d_4\}$, where $d_3 =$ "I want a dog and a jaguar." and $d_4 =$ "I want this dog; not just any dog, this dog." in Table 1.3).

Assuming that the ordering function sorts the documents in the result from highest term frequency to lowest, this effectively means that the top-ranked result of the search will be a document that doesn't mention any jaguar. However, our intuition suggests that since the number of documents referring to dogs is larger than the number of those referring to jaguars, "jaguar" must be the more important term in the query. Conversely, since "dog" is the term found more often in the entire corpus, its general importance across all documents in the corpus should be smaller. We express this property with *inverse document frequency*; for a document corpus comprising documents $d_j, 1 < j < N$, inverse document frequency for term t_i is given

as

$$(1.2)\ idf_i = log\frac{N}{|\{d_j : t_i \in d_j\}|},$$

where the numerator is the total number of documents in the corpus, and the denominator is the number of documents containing term t_i (i.e. total number of documents where $n_i > 0$).

Since tf grows as the frequency count of a term in a document grows, and *idf* grows as the frequency count of a term in the corpus diminishes, their product is used to filter out common terms — effectively, to give more significance to more unique terms:

$$(1.3)\ tfidf_{i,j} = tf_{i,j} \cdot idf_j$$

Using *tf-idf* to measure relevance prevents the previous issue of "dog" documents coming up as most relevant: even though its tf term is large, it is offset by the *idf* term, which is considerably smaller than its "jaguar" counterpart. Thus the word that is less common in the entire corpus is assigned more significance, which is the desired effect.

The *tf-idf* is applied to the cells of the term-document matrix as a *weight*. That way the importance of terms is either amplified or muffled with respect to their *tf-idf* value.

Stopwords

The observation of the importance of inverse-document-frequency as a relevance measurement can be extended further: some words may be so common in any reasonable corpus of documents that they should not be considered as terms at all. As it's shown

above, a term that is common across the entire corpus doesn't contribute to establishing relevance ordering within the corpus; if a list of words that make for such terms could be compiled, then the items on that list could be ignored by the search engine all together. Such words are commonly dubbed *stopwords*. In the English language entire types of words are especially convenient stopwords, namely articles, adverbs, conjunctions and prepositions, but also some less obvious kinds of words, like pronouns and auxiliary verbs, especially the verb "be" and its forms; "and", "around", "both", "I", "of", "so", "was" are examples of some common stopwords. Although lists of common stopwords have been compiled many times over, the list of stopwords for any two applications might be different: any word that is found with a uniform term frequency across the entire corpus is a candidate for the stopword list and can be included as a stopword without loss of relevance information.

Latent semantic analysis

Known in the context of information retrieval under the alias *latent semantic indexing* (LSI), LSA is a vector space model technique that attempts to represent semantic relationships of document-vectors through dimensional reduction of the term-document matrix into a concept space of the desired size. It assumes the existence of an underlying (hence "latent") structure in word usage that is obscured by variability in word choice, and attempts to eliminate from this variability the semantically irrelevant terms, also known as *noise*. [1]

At the heart of LSA is a particular matrix decomposition, known as the *singular value decomposition* (SVD). SVD is the decomposition of the matrix X, comprising n-dimensional vectors, into matrices U, S and V, such that

$$X = USV^T, \tag{1.4}$$

where U and V are orthonormal matrices and S is a diagonal matrix [2]. (The details of how this is achieved for any particular matrix is beyond the level of detail appropriate to this thesis; see [3] or [1].)

Selecting k largest values from the diagonal matrix S will result in a k-rank approximation of X, reducing all n-dimensional document-vectors to k-dimensions. Remarkably, this statistical reduction corresponds to a semantic interpretation, as LSA effectively reduces the document corpus into a k-sized concept space. However, the reduction will sometimes result in dimensions that have no clear semantic interpretation (e.g. given a corpus D'' comprising documents $d_6 = \{$"dog", "jaguar", "cat"$\}$ and $d_7 = \{$"phone", "line"$\}$, after a 2-rank reduction it is possible a resulting vector might look like this: $d = \{(0.75*$"dog", $0.40*$"phone"), "cat"$\}$, the meaning of the which is simply not interpretable within natural language semantics).

Similar to *tf-idf* weighting, LSA is able to overcome the often conflicting issues of keyword frequency and relevance. Like no other method, LSA can associate documents with queries on the concept level, even when a relevant document contains no keywords the actual query comprises [4]. Especially interesting is the success of LSA in the field of cognitive science: by empirically choosing k 300, LSA is able to achieve results that are curiously similar to human performance when performing tasks such as identifying synonyms [5], deducing the semantics of words that it hasn't been exposed to [6], and even interpreting metaphors [7]. It has also been successfully used in a broad range of applications, from grading essays, to evaluating performance of flight simulators [8].

Stemming

Another improvement to the classification process is *stemming*. Stemming is merely the process of establishing a given word's morphological stem. Absence of a key-

word in its entirety from a document doesn't necessarily mean the absence of some other morphological form of the keyword: it is plausible that if the user's keyword is "cheese", the search should also match occurrences of "cheeses" and "cheesy". Hence stemming is used to semantically empower term comparisons: instead of attempting to match "cheese" with "cheesy" (which would result in failure), the stem of "cheese" and the stem of "cheesy" (both of which are "chees") are compared — and the matching succeeds. It is clear that use of stemming can aid the search process by reducing the number of false negatives in term comparisons and by limiting the dimension growth of the term-document matrix resulting from these false negatives.

Stemming is a particularly interesting improvement. Unlike *tf-idf* and LSA, which are statistical methods, and stopword removal, which is merely an extension of the principles of *tf-idf* weighting, stemming doesn't utilize statistical relationships of terms, documents and corpus as a whole, but attempts to capture semantic relationships expressed through morphological properties of terms.

However, there are always two types of inevitable stemming errors that arise from the inherent irregularity of morphological ties in natural languages: *understemming* occurs when two words that should be reduced to a common stem aren't conflated; similarly, *overstemming* occurs when two words that don't have a common stem are conflated [9]. Stemmers fail to capture proper morphological relationships between words because they do not use a lexicon; thus the result of stemming two words is not their linguistic root, but an artificial stem, which leads to misconflations [10]. In addition to the detachment from linguistics, the utility of stemming can be hindered by polysemy; in our example, "cheesy" could relate to the concept of poor quality (e.g. "cheesy acting"), in which case performing stemming is not a good choice [10, 11]. An even better example: stemming the words "gravitation" and "gravity" is only desirable if the meaning behind "gravity" is within the concept of attraction force, as

opposed to the concept of seriousness [10].

1.2.3 Improvements on the vector-space model

Concepts and methods presented so far in this work represent the general development of the field, i.e. the development of solutions to the problem of discovering content that is relevant with respect to some query. However, further inspection reveals that there is often a disparity between the meaning carried by the query from the perspective of the user and the approach of aforementioned methods to discovering similar content. Consider our original user query, "jaguars"; perhaps the majority of objective observers would agree that the user intention is to inquire about the South American feline predator; a slightly fewer number would say that the query is actually about the British luxury automobile; football fans would say the inquiry is actually about the Jacksonville Jaguars of the NFL; Mac users with good recall would immediately associate the query with the codename of Mac OS X v.10.2; etc. Different minds might differ on the presumed meaning of the query; a great majority might interpret it as one meaning, and only a small minority as another meaning; yet this doesn't say anything about the user's intent.

What this means is that, given a user query and a corpus of documents, interpreting which document is relevant in terms of the words with which the user query is described is a much easier problem than interpreting which document contains information about the meaning of the user query.

This latter problem in particular is very difficult. Somewhat of a converse to the statement of the problem perhaps rings more true: given several different meanings of "jaguars", there is no way to distinguish from the query which one exactly is the one user was referring to. The first reaction might be to attempt to narrow

the query down - include words “animal”, or “car” in the query; however, the more abstract the meaning of the user query is (i.e. the more difficult it is to express its exact meaning in words), the more difficult will it be for information retrieval techniques described so far to produce a satisfying result. Stated more correctly, the more abstract the meaning of the query is, the more often a satisfactory result will be absent, as long as only words are used to establish relevance. A more important difficulty to this problem is that the user might not be familiar with the various meanings of the keyword — perhaps not even familiar with a single meaning, or with the exact number of meanings the keyword represents.

If the meaning of the query can’t be described with words, the solution that imposes itself is that the query needs to contain more than words, i.e. the meaning of the query must be more than the sum of the keywords that comprise it. As the previous section of this paper has mentioned, keywords have been successfully used to establish relevance on the level of textual tokens, so it makes more sense to keep using keywords than abandoning their use altogether. However, the keyword query needs to be augmented with data that transcends textual information in the document.

The assumption essential to solving this problem is that the meaning of the document is reflected in more than what its content reads, i.e. that a document contains characteristics other than text that give information about its meaning. The assumptions are, hence, that characteristics can be extracted from documents such that these characteristics

- are something other than text keywords,
- represent some form of valuable information about the semantics of the document.

Such a document characteristic is a document *feature*. The space of viable so-

lutions to the problem of associating one semantic concept with a query when the concepts share a textual representation varies with the choice of document features to augment the query with. Candidate features with respect to their treatment in academic literature on information retrieval can be classified as follows:

1. domain-independent approaches:
   (a) natural language features
       (e.g. sentence-based summarization)
   (b) structural features
       (e.g. document tag structure representation)
   (c) visual features
       i. image features
          (e.g. histograms of pictures within documents)
       ii. layout features
          (e.g. paragraph size)
2. domain-dependent approaches:
   (a) ontologies

Natural language features

One of the problems of the term-document matrix is the "semantic noise" introduced into the matrix by both its sparseness and the sheer number of keywords prior to dimension reduction. Summarization on the sentence level using natural-processing techniques has been shown to reduce semantic noise and improve precision of document classification. [12] It is plausible to think that summarization can be combined with LSI for even better results.

Structural features

Despite the existence of better structured alternatives, a large majority of documents on the Web are still written in HTML. Representing a document as a *DOM tree*, in a hierarchical fashion based on HTML tags, is a very common occurrence in the field of wrapper induction. [13–15] One application of DOM trees uses LSI as the primary approach, but the keyword frequencies are weighted with respect to the position of the word inside or outside specific tags, or the proximity of the word to specific tags (e.g. the image tag). [16]

Visual features

Because of the multimedia character of today's Web, document semantics are not conveyed exclusively by text. Images, video streams and audio streams are all very common content types, both in scenarios where they serve to support the document text, and in scenarios where the document text supports them. In an existing system, characteristics of document images, namely color histograms and anglograms, are used as additional dimensions to the document-vectors, which are then submitted to latent semantic analysis. [17] In another system, image features were used along textual keywords to create association hypergraphs; hypergraph partitioning resulted in more precise classification than when only keywords were used. [18] There has also been work in discovering similarity between documents containing multimedia objects such as video files by applying different graph traversal techniques in the attribute space of the multimedia objects [19]. Statistical modeling approaches have been applied to the problem of automatic linguistic indexing of pictures for the purpose of image retrieval, where categorized images have been used to train a dictionary of hundreds of concepts automatically [20].

Apart for visual features extracted from objects embedded in document text, there are visual features that stem from the layout of the text itself inside a document. Spacing, font characteristics, paragraph length, number of paragraphs have all been shown learnable with supervision [21], and able to carry semantic value and assist in document classification [22].

Ontologies

All the previous types of approaches use document features without regard for the subject of the document; they are domain-independent. Ontologies, on the other hand, are quite the opposite: they are user-defined schemas that define semantic relations within particular topics. Ontologies define hierarchical relationships between semantic tokens (words, group of words, sentences) and associate them with their domain-specific synonyms via a lexicon particular to the topic. Ontology-based approach to clustering has been shown to improve on text-based approaches when the two are combined [23]. It has also been used for discovering similarity between web services inside a web service repository and has shown improvement on the baseline semantic-based approach when the two are combined [24].

To the aforementioned document features used to improve document classification, our aim is to add another one: non-word patterns. Our quest for document feature extraction takes off in the direction of the principles of morphological analysis of terms, not unlike stemming. In the following chapter we define our non-word patterns, discuss how to observe them within documents and extract them, and measure their impact on document clustering precision. Finally, we present an approach to supervised learning of such patterns from documents and measure the improvement of our approach to the precision of document classification.

Chapter 2

Discovering Patterns

2.1 Introduction

Any corpus of documents, such as a collection of web pages, can be partitioned according to the various topics represented in those documents. Elements of such a partition are *classes* — groups of documents that are conceptually related (most likely, they share a common topic). While this is true for any collection of documents, it is particularly interesting for such collections where different classes map to identical textual representations. For instance, the concepts of Jaguar, the automobile and jaguar, the feline predator, both map to the textual representation "jaguar". These are the cases we are interested in exploring.

2.1.1 Space of non-words

In any document, two types of text are quickly distinguishable: **words**, which are easy to recognize and associate with concepts because their meanings are retrievable via a dictionary; and **non-words**, or everything else that does not satisfy the above criteria. In this space of non-words, meanings of individual tokens of text are most often not

immediately clear. However, some tokens gradually reveal their meaning through repetition — that is, their distribution with respect to the words in a document gives indication of their meaning. Most often these tokens are not mutually identical, but vary in quantity and quality of text that describes (or, characters they comprise). Nevertheless, they are recognizable as similar, because they can be aggregated to take shapes that encompass this variation of quantity and quality that they express. We call these shapes *patterns*.

This idea did not materialize in vacuum, but shares roots with some established methods. In our discussion of stemming as an improvement method to a simple vector-space model, we mentioned that stemming takes advantage of semantic relationships expressed through morphological variations. While stemming limits the extent of analysis of these relationships to (a small set of) rules dictated by spoken language grammars, there is no inherent reason why the principle cannot be extended with new sets of rules, similar to those that govern grammatical correctness.

2.1.2 Pattern properties

Much like certain words are associated with certain meanings, our expectation is that there exist patterns in the space of non-words that are associated with a certain class of documents and not associated with other classes. Thus the patterns that would be considered relevant in the classification process must satisfy two criteria: *mutuality* and *distinction*. Given pattern is said to satisfy the mutuality criterion when that pattern is characteristic of the entire class of documents — that is, when the pattern occurs in significant capacity in most documents of the class. The criterion of distinction is satisfied when there is zero (or negligible) occurrences of the given pattern across classes — that is, when all the documents containing that pattern in

a significant capacity belong to the same class.

Our pattern detection and selection method is based on visual observation of the documents. We attempted to identify tokens inside the text found in the space of non-words, such that these patterns a) can be generalized into patterns, and b) these patterns are characteristic of one class of document and not any others. When a candidate pattern was observed in a document, other documents of the same class were inspected to verify mutuality. If mutuality was established, documents of the other class were inspected to guarantee distinction. If both mutuality and distinction were established, the candidate pattern became a representative pattern of its class.

Consider, for example, a document pertaining to the 'car' meaning of "jaguar". This document will contain the following and similar strings: "XJR-15", "XJ220", "XJ12", "XF" (they happen to represent model numbers). It is clear that these strings can be generalized with a relatively simple regular expression: "[XJRF]2,3-0,1[1250]0,3". Upon inspecting the documents pertaining to the 'animal' meaning of "jaguar", similar strings will not be found. Therefore, we've isolated generalizable pieces of text that are specific to only one class of meaning. Similarly, looking at a document from the 'animal' class of meaning, we might notice strings "cm", "m", "kg", "lbs", generalizable simply as a disjunction: "lbs—kg—cm—m". Although some of these strings (in particular, "m") might be found in the documents from the other class of meaning, their appearance would be sufficiently infrequent within a single document and sporadic across all the documents of that class of meaning that we may conclude we have another meaning-specific pattern. Patterns extracted from our corpus are shown in Table 2.1.

animal	*car*
lbs	mph
cm	[IVX]$^{+}$
in?\.?	[2-5]\.[0-9]L\.*
ft\.?	\[1-9][0-9]*([,][0-9]3)*
k?g\.?	[CJKRSX]$^{+}$-*[0-9]*
	V-*[12468]$^{+}$

Table 2.1: Patterns in both classes of meaning.

2.2 Experiments

The aim of our experiments is to observe whether employing relevant non-word patterns, found in the documents of our corpus, has an effect on the classification precision of our statistical method. To this end we collect a corpus of web documents, process the documents to increase importance of the natural language content, represent the corpus with the vector space model, and perform clustering based on the document/vector similarities.

2.2.1 Data set

The documents comprising our corpus were all collected from a Google search results page for the query jaguar. The first 50 documents were stored and manually labeled according to their topic. While this partition resulted in 12 different classes of meaning, two were dominant ('animal' and 'car'), comprising 35 out of the 50 documents; these would become the effective corpus which our experiments would be performed on. The documents collected were first 50 search results, as displayed by Google, with the exception of the very sparse documents (documents with less than 150 words), which were skipped as they did not contain enough text to facilitate meaningful and consistent classification.

2.2.2 Preparing the document test bed

In our document classification experiments we used the established vector space model, where documents are represented as n-dimensional vectors, where n is the total count of all distinct terms (tokens of text) in the corpus. Value of a particular dimension of a particular vector corresponds to the occurrence count of the term representing that dimension in the given document represented by that vector. The entire corpus is thus represented as a term-document matrix - a matrix with documents for rows and terms for columns (see Table 1.1 for an example). Several steps precede the building of the term-document matrix, namely tag removal, lowercase conversion, stopword removal, and stemming. To make the classification process sensitive to patterns, we added another preprocessing step: all terms recognized as instances of a certain pattern were aggregated into a single, class-specific term, as opposed to representing individual terms. Following the creation of the term-document matrix, term-frequency-inverse-document-frequency (tf-idf) weighting was performed, its goals being the trivialization of those terms characteristic of individual documents, as opposed to individual classes, as well as those terms common to all documents (and classes).

Tag removal

The tag removal steps, as the name indicates, parses the documents and removes HTML/XML tags from consideration as terms. For tags that embed text within opening and closing tag bounds, the embedded text was extracted and treated like any other body of simple text; this was deemed especially necessary for HTML comment tags.

Lowercase conversion

This step included a simple transformation of any term into a semantically equivalent term with all the uppercase letters converted into lowercase. Unlike other preprocessing steps, this conversion was not performed indiscriminately, because a lowercase conversion of a pattern term is not necessarily semantically equivalent to the original pattern term, or, conversely, semantically significant pattern doesn't always have a semantically significant lowercase version. For instance, consider the pattern [CJKRSX]+-*[0-9]*; it will recognize terms starting with some number of uppercase characters. If a term that is matched by this pattern (e.g. "XJ-40") was converted into lowercase (i.e. into "xj-40"), it would no longer be matched by the pattern. The caveat is to always perform lowercase conversion after pattern recognition has already been performed.

Stemming

Stemming was performed with a Python implementation of a Porter Stemmer, the commonly used, *de facto* standard stemmer of the English language, originally designed by Martin Porter in 1979 and described in [25]. Besides stemming all terms that build the term-document matrix, the stemmer was used to reduce to stem all words in the dictionaries/word lists that were used in stopword removal and removal of more arbitrary content from documents.

Stopword removal

Which document terms were considered stopwords was determined by what was deemed a reasonably comprehensive list of stopwords in the English language by the information retrieval faculty of the computing science department of the University

of Glasgow [26]. It includes 319 common words - by in large, pronouns, prepositions, adverbs, numbers and auxiliary verbs.

Removal of arbitrary content

Due to the variety in structure of web documents, preprocessing steps mentioned so far do not thoroughly remove all tokens of text that would intuitively be undesirable for consideration of semantic value. Observation of terms that trickle down through the above preprocessors provides insight on what these tokens might be: embedded Javascript method calls, file path references, HTML non-tag keywords, etc. - they are all removed. Additionally, as we attempt to create the space of non-words discussed at the beginning of this chapter, we will eventually include all words (as recognized by local dictionaries of English words) in this list.

Pattern recognition

Pattern recognition consists of four steps:

1. compiling pattern strings into regular expressions
2. associating regular expressions with document classes
3. matching terms against regular expressions of all classes
    (a) entering the term in the term-document matrix, if not matched
    (b) throwing away the term and increasing the count of the class-representative term in the term-document matrix by the appropriate weight matrix, if matched

The differences between a term-matrix built with pattern sensitivity and a simple term-document matrix are illustrated in Figure 2.1. The mentioned weight w is the

arbitrary significance ratio of pattern-recognized terms to other terms. Whenever a regular term is matched, its occurrence count is increased by one; whenever a pattern term is matched, however, the occurrence count of the unique term representative of the document class associated with that pattern is increased by w.

The function of the pattern recognition is to polarize document semantics — reduce the importance of most term-document matrix dimensions by accumulating the bulk of term occurrences, that would normally register as numerous individual dimensions in the vector, into c large dimensions, where c is the number of document classes. The consequences of this, simply interpreted, are that, for high enough values of w, the answer to the question which document class a document-vector belongs to is reduced from n choices, where n is the number of distinct terms in the entire corpus (cca. 4,000), to c choices (i.e. 2). Naturally, this assumption only holds if our patterns have been chosen correctly (i.e. as prescribed in the discussion about origin and nature of patterns at the beginning of this chapter).

2.2.3 Results

Experiment format

Every experiment in this chapter makes use of some form of patterns. The basic independent variable that experiments differ on is whether our patterns originated in the space of words or the space of non-words.

The experiments can be summarized in the following steps:

1. building a simple term-document matrix (TDMs),
2. building a pattern-sensitive term-document matrix (TDMp), and
3. clustering documents using TDMs and TDMp.

At the end of each test run, the clustering achieved using the simple term-document matrix and the pattern-sensitive term-document matrix (both refered to as the *result*) is compared to the actual distribution of documents across the concept space (known as the *ground truth* or *reference*). The correctness of the result in relation to the reference is expressed as the clustering *precision*. The relationship of the precisions achieved with TDMs and with TDMp for different parameters of the experiment is then graphed.

While our interest lies in the space of non-words, we perform some experiments in the cumulative space of words and non-words (i.e. in the original space of document content). When we look at the space of non-words we are effectively looking at documents that have been pre-processed to exclude terms found in lists of stemmed words in the English language.

Experiment 0.0: Testing the value of patternizing

To begin, we concentrated on verifying the validity of the concepts of mutuality and distinction: if it is possible to extract sets of non-word patterns with the properties of mutuality and distinction, surely an easier task would be to extract sets of words with the same properties. We consider this task easier because the source of these sets need not only be observation, but also *a priori* knowledge of the confronted semantics. If two sets of words — two *word accumulations* — can be obtained, one for each class of meaning, such that those two sets of words satisfied the properties of mutuality and distinction, increasing the relative importance of the words in these sets should, ultimately, polarize the document space in accordance with each document's association with one of the classes. From the low-level perspective, the dimensions in the term-document matrix, represented by words in one of these sets, will be reduced to one dimension. Increasing the importance of this dimension by higher weighting

	car	automobile	vehicle	motor	model	engine	animal	cat	feline	panther	predator	species
d_1	6	2	5	11	18	9	1	0	0	0	1	1
d_2	0	0	0	0	0	0	4	14	6	3	5	3
d_3	3	5	2	7	22	13	0	1	0	0	0	0
d_4	0	0	0	0	1	0	8	9	7	6	6	4

CAR = {car | automobile | vehicle | motor | model | engine}

ANIMAL = {animal | cat | feline | panther | predator | species}

	CAR	ANIMAL
d_1	51	3
d_2	0	35
d_3	52	1
d_4	1	40

Figure 2.1: Equivalent portions of term-document matrices for the same document corpus: simple (up) and sensitive to the illustrated word accumulations (down).

will result in the increase of the classification precision. Figure 2.1 contrasts the appearance of two term-document matrices, one simple, the other sensitive to word accumulations. The word accumulations in the figure are exactly those used in this experiment.

Results of the experiment are shown in Figure 2.2. *Weighted patterns* line represents the clustering precision achieved when each word accumulation was reduced to a single term in the term-document matrix and weighted; the x-axis represents the ratio of the word accumulation weight to regular terms. *No patterns* line is the precision achieved without any term accumulation — that is, with every word in the corpus representing a single term in the term-document matrix.

The graph confirms that the concepts of mutuality and distinction, when applied

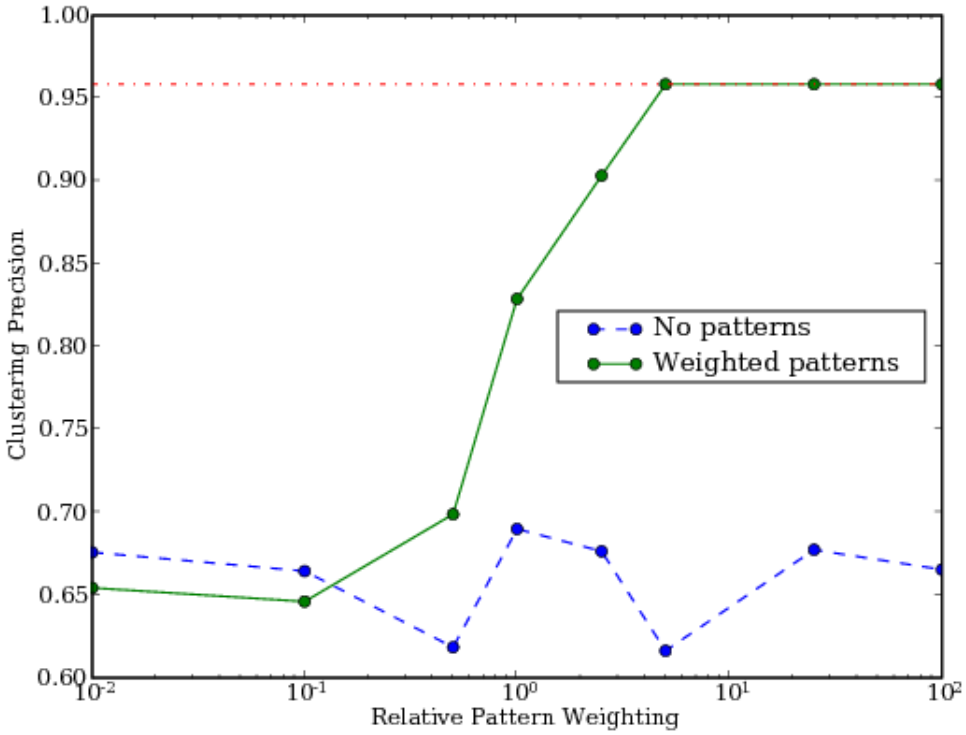

Figure 2.2: Clustering precision increases as more significance is associated with sets of words satisfying the properties of mutuality and distinction.

to the space of words, bear a direct relation to the precision of clustering. Rephrased, in the space of words it is possible to distinguish words related only to one meaning from those related only to the other meaning. This, however, does not come as such a surprise - it is the essence of the ontological approach to information retrieval. What we are interested in next is to find out whether the same applies in the space of non-words.

Experiment 0.1: Testing non-word patterns in non-word space

Our second experiment tests the utility of employing non-word patterns to improve clustering precision. The non-word patterns we use are exactly those described in

Table 2.1. We are expecting results similar to those from our original experiment: the patterns will, because they satisfy the conditions of mutuality and distinction, improve the precision of classification when their significance is amplified over the significance of the rest of the terms.

Before we proceed to testing non-word patterns on our document corpus, we pause to test the fundamental soundness of our non-word patterns. Because these patterns exist in the space of non-words (which is to say that none of the terms that are aggregated into patterns exist in the standard English dictionary), we preprocessed the documents to **exclude all words**. Every document was effectively reduced to a version stripped of entries that would be found in a standard English dictionary; also excluded were any word compounds, such as hyphenates. The main reason for performing this step is to check whether our patterns indeed fully exist in the space of non-words. We are specifically concerned that the regular expressions representing our patterns might inadvertently match some terms that we would consider to reside in the space of words. If we tested the non-word patterns on the corpus as it is (in the cumulative space of words and non-words) and achieved some increase in clustering precision, we wouldn't be able to say whether that occurred because our patterns did what we expected them, or if the patterns actually accidentally amplified the significance of some words, due to some oversight in our formulation of patterns. On the other hand, if we are able to achieve precision when clustering documents in the non-words space, we would be satisfied that our patterns indeed reside in the space of non-words.

Results of our second experiment are presented in Figure 2.3. *Weighted patterns* in this case pertain not to word accumulations, but to non-word patterns. More specifically, they pertain to those terms from the corpus that are matched by the regular expressions representing the extracted non-word patterns (see Table 2.1 for

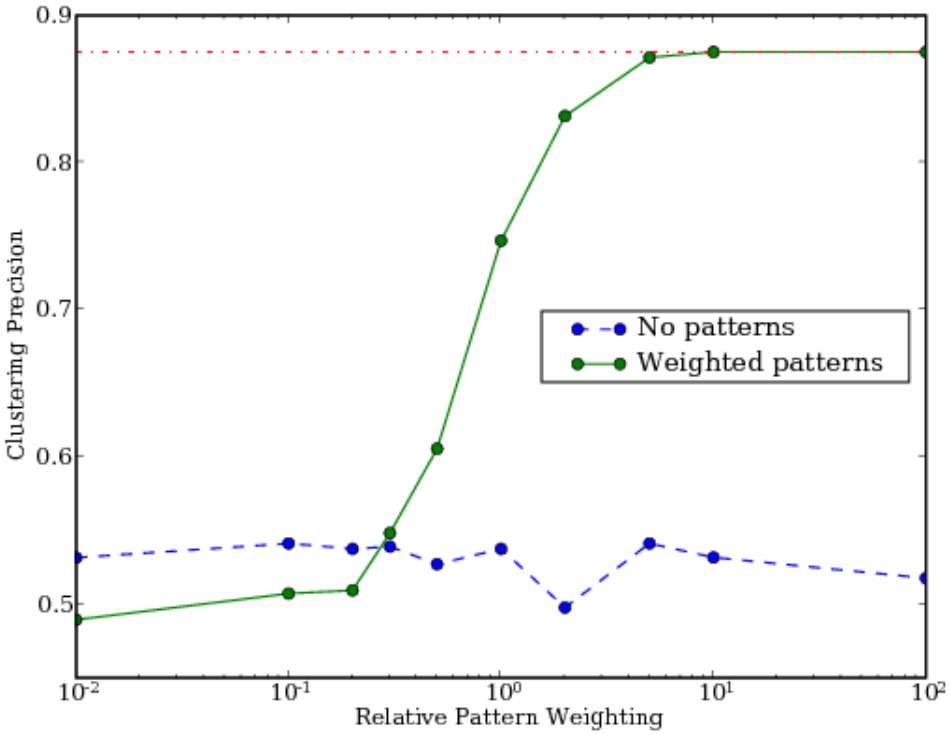

Figure 2.3: The precision of clustering documents residing in the space of non-words increases as more significance is associated with non-word patterns satisfying the properties of mutuality and distinction.

the exact non-word patterns used in this experiment).

This experiments confirms that we can indeed apply concepts of mutuality and distinction in the space of non-words to observe a relationship to classification precision. It also confirms that our patterns reside in the space of non-words to a degree that is sufficient to achieve classification improvements. The last experiment of this chapter extends our idea to its natural conclusion, as we examine whether our non-word patterns have an effect on clustering precision of documents that predominantly comprise words.

Experiment 0.2: Testing the value of non-word patterns

We finally attempt to discover whether applying the non-word patterns to the original corpus, comprising fully-worded documents, as retrieved from the web, yields a classification precision benefit. We know, from the previous experiment, that these patterns with properties of mutuality and distinction can have an impact on the classification precision of non-word documents — now we need to find out whether the volume of words and their semantic capacity is overwhelming for the non-word patterns to again affect the clustering precision, or if we will achieve positive results again.

Considering the lessons of our previous experiment, we expected to see precision improvement in this experiment as well. However, having also in mind our assumption about relative semantic value of words compared to non-words, in this experiment we expected the highest precision achievable to be lower than the highest precision achievable in our original experiment. As can be seen from Figure 2.4, our expectations are realized: the clustering precision grows with the weight until it reaches the highest possible precision. When we compare the highest possible precisions of Figure 2.2 and Figure 2.4, we see that the highest precision achieved using word accumulations is higher than the highest precision achieved using non-word patterns. This confirms our assumptions about the semantic values of words and non-words.

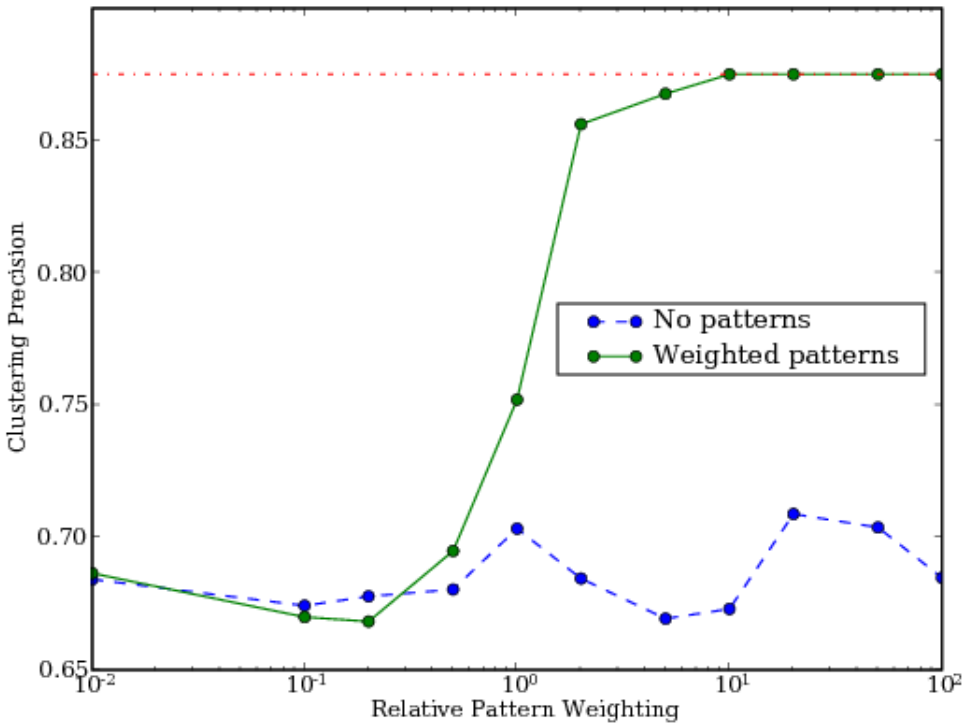

Figure 2.4: Clustering precision increases as more significance is associated with non-word patterns, but does not increase as high as when word aggregations are used (as they were in the original experiment).

Chapter 3

Pattern Learning

3.1 Introduction

We have established that our non-word patterns aid in classifying documents in our corpus closer to the ground truth. However, consider the way we created the patterns: we had to visually examine most, if not all, document in each document class and attempt to identify candidate patterns by first observing the text of the document and then comparing it to other documents. Obviously, this approach does not scale well for large corpora, especially if the concept space is large. We would like to automate the pattern creation process, so that it doesn't require any *a priori* knowledge of patterns. As the end result, we would like to be able to collect a corpus of documents, estimate its ground truth through uninformed clustering (clustering on the simple term-document matrix), process the documents in a way that allows us to infer a set of non-word patterns for each concept, and use those patterns in the document clustering. Finally, we hope that the learned non-word patterns will have a similar effect to their hand-made counterparts and improve the clustering precision.

3.2 Overview

The approaches of this chapter and the previous chapter share the common thread of operating in the space of non-words; all the documents are stripped of dictionary-like entries. When we talk about documents, terms or patterns from now on, we are talking exclusively about constructs that reside in the space of non-words.

Because we met with success using hand-made patterns in the previous chapter, we feel that the fundamental principles behind our rationale for building the hand-made patterns were sound. Thus it would be beneficial to our automated process of learning patterns if we were able to quantitatively express these principles and apply them in our implementation. In fact, these quantitative descriptions constitute the essence of our pattern learning approach.

Devising our hand-made patterns required two key ingredients: observation and intuition.

Intuition

In the process of deciding what terms should be included in the forming of our observed patterns we relied on the intuition that some terms were more important than others for distinguishing documents of different classes one from another - that is, that some terms carried more semantic value than others. We speculated that such terms

- occurred with relatively high frequency within the majority of documents of a single document class, and
- occurred with negligible (or at least less significant) frequency in other classes of documents.

We called these properties *mutuality* and *distinction*, respectively. To quantitatively describe our intuition about mutuality and distinction, we implemented measures of term salience. We generally assumed that the salience of a term for a particular document class was related to the probability of that term occurring in that document class and not occurring in others. Once we are able to measure a term's salience to any document class, we would be able to dictate which terms could and couldn't participate in the pattern learning process for a given document class.

Observation

When we first set out looking at the corpus documents, we were observing text tokens that weren't immediately interpretable like words are, but formed meaning through 1) repetition (they were found throughout individual documents and throughout individual document classes) and 2) mutual similarity (they comprised a small set of characters, had similar length, similar distribution of lowercase to uppercase characters, etc.). This is, for instance, how we identified textual representations of Jaguar models in the documents within the concept of *car*: we observed repeated occurrences of strings featuring the set of characters {'X', 'J', 'R', 'C', 'S', 'K'}, followed by two or three numeric characters that are sometimes preceded by a hyphen. We can summarize the cumulative intricacies of what we observed as

1. observing the character-wise similarity of tokens, and

2. observing how this similarity was reflected in non-identical tokens.

These similarities mostly pertained to the variety of type of characters and their distribution within tokens. Observing this led us to describing these similar, non-identical strings of characters with regular expressions. We attempted to emulate

our imprecise observation by implementing an aggregation process, the purpose of which is to join strings of text (i.e. document terms) into aggregate representations. Assuming that the patterns of our aggregation implementation are at least as nearly as good as the patterns of our observation, we expect that submitting a set of terms that satisfy mutuality and distinction properties to the aggregation process will result in an aggregate representation that engulfs this set of terms (in our case, a regular expression) and itself satisfies the same properties.

We can now define what we understand by the phrase "pattern learning" in terms of salience and aggregation. *Pattern learning* is the process in which regular expressions are inferred from the lists of document class-specific salient terms through the process of joining the terms into aggregate representations.

3.3 Aggregation

In the process of aggregation, an aggregate term is created from two simple terms; the aggregate term represents the two participant strings in the form of a regular expression. The aggregation occurs on character level: given a set of character transformations, obtained by comparing two simple strings with some string metric, simple characters are molded into *aggregate characters*. In the context of aggregation, aggregate characters, or aggregate *atoms*, sustain an is-part-of relationship with *aggregate terms*; this relationship corresponds to the is-part-of relationship of characters to strings.

There are two basic character transformations: *union* and *iteration*. The union is characterized by a transformation of one character into another (e.g. 's' → 't' = 's' OR 't') and interpreted as the choice between the two characters. The string representation of union is '[]' (i.e. s → t = [st]). The iteration, in the narrowest

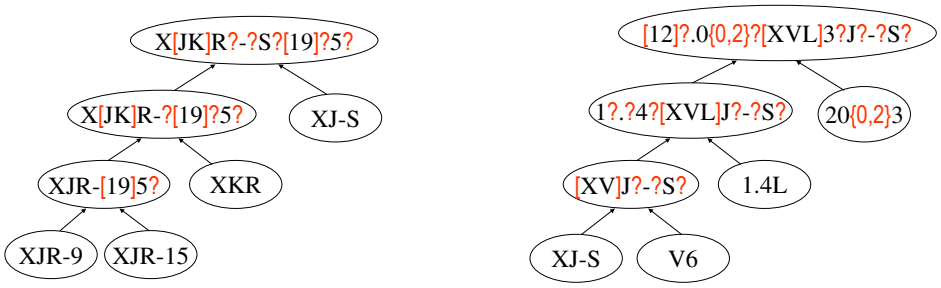

Figure 3.1: Cumulative aggregate term of several simple terms.

meaning, is characterized by a transformation of one character into a non-character (e.g. 's' → ' ' = 's' OR ' '). The string representation of such an iteration is particularly interesting: s → None = s{0,1}, or s? for short. The meaning of this aggregate atom can be thought of as a choice between presence and absence of the simple character 's'. Iteration, in its broader meaning, is any repetition of characters. For example, the transformation s{2} → s = s{1,2} comprises two iterations in the broader sense, s{2}, which represents no more and no less than two repetitions of the simple character 's'; and s{1,2}, which represents the choice between one or two iterations of 's'. Of course, s itself is an implicit iteration of 's' (s = s{1,1}). Iterations can comprise not just simple characters, but unions as well; for instance, the interpretation of aggregate atom [kbr]{1,2} is the choice between one or two characters from the set {'k', 'b', 'r'}.

3.3.1 Edit distance

We now gain insight into the genesis of character transformations presented in 3.3. The learning of non-word patterns occurs through a process of aggregation, where pairs of non-word terms are replaced by minimal regular expressions that match the pairs' terms. In making the decision which pairs are good aggregation candidates and

which aren't, the choice of a good term similarity metric is essential, as aggregating overly diverse terms would lead to an overly high degree of generalization. Figure 3.1 shows aggregation sequences of some good (left) and bad (right) candidate terms.

As we mentioned earlier, our knowledge of patterns comes from observing character-wise diversity, or similarity, of terms. Through our observation, we established a hierarchy of important character-wise properties that shape our understanding of term similarity:

1. number and distribution of identical characters,
2. number and distribution of case-equivalent characters, and
3. ratio and distribution of characters of particular type.

We believe that these properties, in the given order of relevance, contain the essence of term similarity. Hence we adopt them as foundation for our term similarity metric, making certain that they are addressed in this order of importance.

We start with the number and distribution of identical characters. It is convenient to look at this property from the standpoint of *edit distance*. Edit distance is a string metric that expresses string similarity through edit operations required to transform one string into another. One explanation of our property of number and distribution of identical characters is this: if we are trying to transform one term into another by inserting, deleting and substituting characters, the less insertions, deletions and substitutions we need to make, the more similar the terms are. Using edit distance as our basic approach we can still pay attention to the rest of our important term similarity properties by introducing operation selection bias that is related to case-equivalence, character type and term length.

		x	j	-	4	0
	0	1	2	3	4	5
x	1	0	1	2	3	4
k	2	1	1	2	3	4
-	3	2	2	1	2	3
r	4	3	3	2	2	3

Table 3.1: Sample edit distance matrix for two simple strings.

We adopted the Levenshtein distance [27,28] as the preferred metric for calculating the edit distance between two terms. After implementing the commonly-used Levenshtein distance algorithm, we make significant modifications that allow the metric to consider all aforementioned properties of term similarity.

Unmodified Levenshtein distance

This edit distance algorithm utilizes a bottom-up, dynamic programming approach. It examines two strings, s and t, and makes use of a m-by-n matrix d, where $m = l(s)+1$, $n = l(t) + 1$, and $l(x)$ is the length of term x. It performs a character-by-character examination between the strings, and at each step (i.e. for each character pair) makes decisions about the appropriate (i.e. lowest cost) edit operation for that pair of characters: deletion, insertion, or substitution. [27, 28]

At any step of the algorithm, $d[i][j]$ contains the minimal number of edit operations to transform the substring $s[1 \ldots i]$ into the string $t[1 \ldots j]$. For any i, j, the contents of cell $d[i][j]$ are reached by selecting the minimum of three values: $d[i-1][j]$ + cost of deletion; $d[i][j-1]$ + cost of insertion; and $d[i-1][j-1]$ + cost of substitution. The completion of the main loop leaves the answer — the minimal edit distance between s and t — at $t[m][n]$. [27, 28]

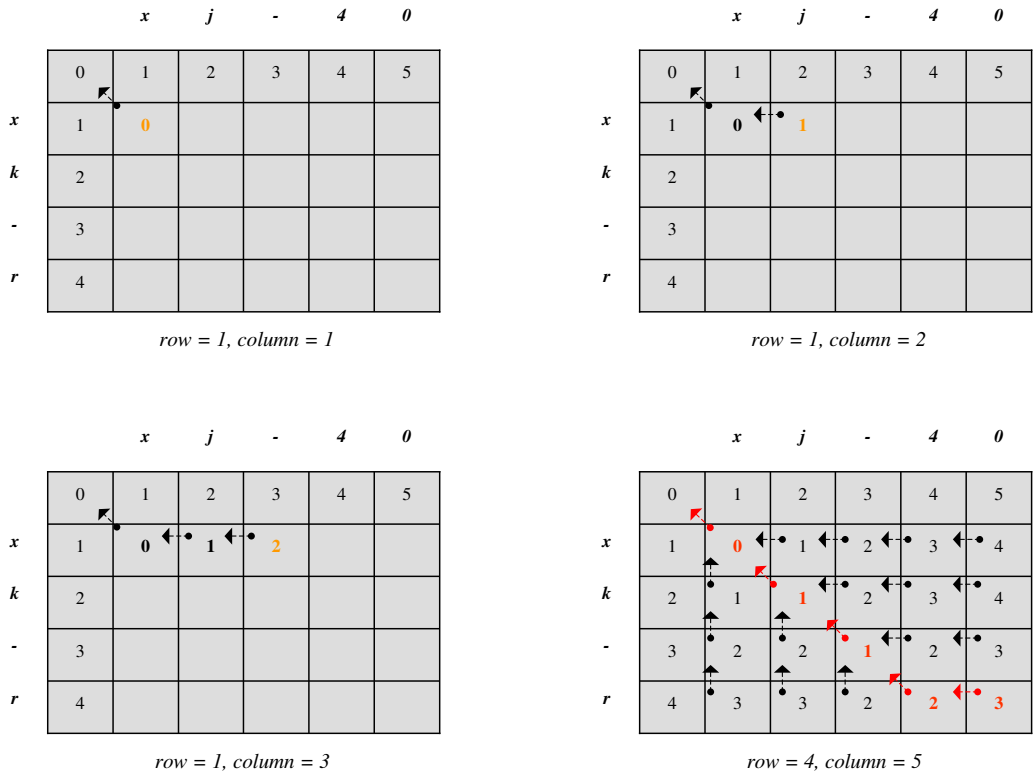

Figure 3.2: Gradually building the edit distance matrix.

Edit operations history

When making the selection of the minimum value to assign to $d[i][j]$, the (relative) "direction" of the selected edit operation (i.e. the indices of the cell that corresponds to the selected edit operation) must be recorded. For instance, if substitution was the lowest cost operation, $d[i-1][j-1]$ should be stored as the direction of the edit operation). When the algorithm has finished, the edit operation history makes it possible to retrace the sequence of edit operations for any cell. When the edit operations history is extracted for cell $d[m][n]$ what is obtained is the lowest-cost sequence of edit operations required to transform one string into the other. [28] This lowest-cost sequence is directly related to the format of the aggregate string: a substitution signals that the necessary aggregate form to match both characters is a union, while

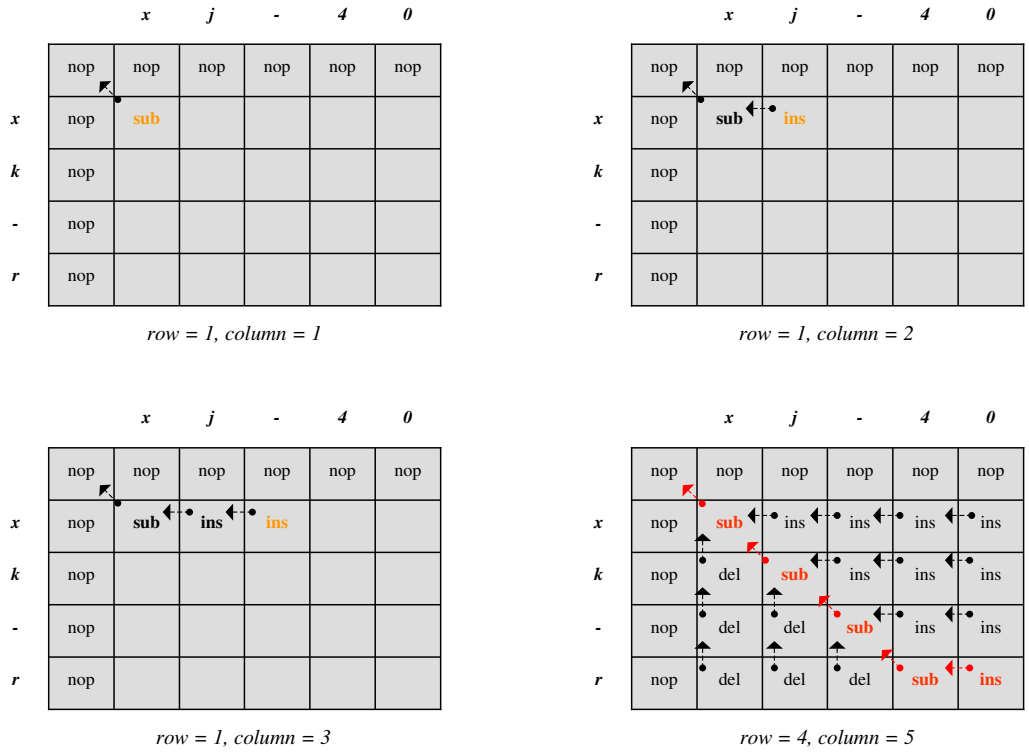

Figure 3.3: Gradually building the edit operation trace matrix.

deletion or insertion signal that the aggregate should be an iteration of the character in s or t, respectively, with a lower iteration bound of zero.

Treatment of aggregate terms

The pattern learning is a multiple iteration process, where the following steps occur on each iteration:

1. edit distances are calculated between all candidate term pairs,
2. the single term pair with the lowest edit distance is aggregated,
3. the aggregate term is added to the term candidate list and pair terms are removed from consideration in subsequent iterations.

This means that the edit distance algorithm must be able to properly treat aggregated terms as input as well as terms that have a simple string representation. To facilitate this, we considered how our dynamic programming algorithm should behave differently when encountering aggregate terms as opposed to simple terms.

We made a general observation that it is sufficient to consider the lower bounds of iteration as relevant in considering the costs of insertion and deletion of an aggregate character. To illustrate this point, consider the aggregate character 1{0,1} as it is being examined by the edit distance algorithm: the cost of inserting/deleting this character is 0, because its lower iteration bound indicates that the simple character 1 can occur once or not occur at all in the term. Similarly, for the aggregate term \$19{2}, the cost of inserting/deleting is four, the sum of the lower iteration bounds of all the aggregate characters (1+1+2); in contrast, for the term \$19{1,2}, the cost is only three (1+1+1).

While the lower iteration bound rule applies universally to insertion and deletion, we need a more detailed explanation of how to deal with substitution. For aggregate atoms a and b, such that they comprise differing simple characters, the substitution cost is equivalent to the sum of the costs of deletion of one and insertion of the other atom:

$$cost_{substitution}(a, b) = cost_{deletion}(a) + cost_{insertion}(b). \tag{3.1}$$

For aggregate atoms that comprise identical simple characters, we observe that

- for *overlapping ranges*, substitution costs nothing;
- for *non-overlapping ranges*, we must calculate the cost by considering all iteration bounds.

The notions of overlapping and non-overlapping ranges pertain to the iteration ranges spanned by the iteration bounds. Two aggregate atoms, a and b, are said to have overlapping ranges if

$\{a.l \ldots a.u\} \subset \{b.l \ldots b.u\}$,

where $x.l$ and $x.u$ are the lower and upper iteration bounds, respectively, of atom x. When this is not the case, the atoms are said to have non-overlapping ranges. An example of overlapping range is y{1,2} and y{0,3}; a similar non-overlapping range is g{1,2} and g{3}.

We calculate the substitution cost as the difference of the higher lower iteration bound and the lower upper iteration bound between the two atoms, or:

(3.2) $cost_{substitution} = min(|a.u - b.l|, |b.u - a.l|)$.

Because the interpretation of any iteration atom it is the choice in the range $r = it.u - it.l$ iterations of the comprised character $it.c$, we reason that for two atoms, a and b, and their ranges, r_a and r_b, where $r_a < r_b$, any choice of $a.c$ with r_a iterations is already anticipated by some choice of $b.c$ with r_b iterations. Thus we consider that an appropriate estimate of the substitution cost of the two atoms is the number of edit operations required to bring their iteration ranges into overlap. Certainly this is least expensively done by expanding the narrower range on the side closest to the lower iteration bound of the atom with the broader range, or collapsing the broader range on the side closest to the upper iteration bound of the atom with the narrower range. This is exactly what formula 3.2 tells us.

Character type and case consideration

When applying intuition to reasoning about patterns, it becomes clear that it is not sufficient to merely consider the number of edit operations as a measure of similarity.

		x	j	-	4	0
	0	1	2	3	4	5
x	1	0	1	2	3	4
k	2	1	1	2	3	4
-	3	2	2	1	2	3
r{2}	5	4	4	3	3	4

Table 3.2: Sample edit distance matrix for two aggregate concatenations.

Also important is to consider the overall quality of the string with respect to the types of characters it comprises. (Character type, in this context, refers to property of being alphabetic, numeric or something else.)

Our observed, hand-made patterns, (intuitively) always have one of the following characteristics:

- patterns comprise characters of a single type, or
- patterns are composed of (non-trivial) substrings that comprise characters of a single type.

Patterns that observe the first characteristic are simple to understand (e.g. "lbs", "[IVX]+", "[12][09][1-9]2"); examples of patterns with the second characteristic are the following: "[CJKRSX]+-*[0-9]*", "V-*[12468]+", "[2-5]$\dot{[}$0-9]L$\dot{;}$". Essentially, our observed patterns can always be divided into non-trivial substrings of a single character type. Thus we assumed that the learned patterns should have the same property.

In terms of our aggregates, common substrings correspond to union atoms: at a particular character in a string, union atoms reflect commonness between terms, while zero-iteration atoms reflect their disparity. That is why we believe that the best way to ensure that our learned patterns possess the properties mentioned above is to heavily favor union atoms to comprise characters of the same type. We conclude one more thing from the knowledge about our hand-made patterns: alphabetic or numeric

characters account for substrings of identical character types, while non-alphanumeric characters appear as substring delimiters.

To direct the aggregation process into producing such aggregates, we implemented a type-sensitive substitution cost as part of our edit distance algorithm. Thus, when the substitution cost of two aggregate terms is considered, not merely their bounds of iteration are considered, but the quality of the characters they comprise. The basic principles guiding our hierarchy of bias toward creation of union atoms are that

1. creation of unions of alphanumeric characters with non-alphanumeric characters should be highly prohibitive, and
2. creation of unions between alphabetic and numeric characters should be discouraged.

The details of our implementation decisions, as they pertain to character type and case, can be summarized as follows, in the decreasing order of union creation bias:

1. characters are identical
2. characters are lowercase identical
3. characters are of the same type
    (a) characters are alphabetic or numerical
    (b) characters are neither alphabetic or numerical
4. characters are of different types
    (a) characters are alphabetic and numerical
    (b) characters are alphabetic/numerical and other

3.3.2 Term salience

As we have shown in previous experiments, some non-words carry more semantic significance than others. Generally, the more concentrated the occurrences of the non-word are within one cluster, the more that non-word is semantically salient. In particular, we assume that the salience of a particular non-word is directly related to the probability of that non-word appearing in one document class and not appearing in other classes. This probability is expressed with the formula

(3.3) $Q_i = P_i/(P_n - P_i)$,

where Q_i is the salience of the non-word for document class i, P_i is the probability of the non-word appearing in class i, and P_n is the probability of the non-word appearing in any class (i.e. of the non word appearing at all in the document corpus). This measure of significance allows us to compile a list of most salient terms per document class; our assumption is that the best candidate non-words for learning patterns will be found in this list.

An important property of the salient terms lists is that the terms are sorted by their *alphabetness*, which is to say that they are sorted by the inverse ratio of the number of alphabetical characters they comprise to the length of the term. Because we are working in the space of non-words, we reason that "better" non-words are those non-words with less alphabetic characters, as alphabetic characters are the single constituent type of dictionary words. It is our expectation that this inverse alphabetness sorting will polarize the terms in a way where the most interesting non-words - in our minds, those non-words that don't contain alphabetical characters - will be pushed to the top of the list.

As we described earlier, to create patterns out of salient non-word terms, our aggregation process is used: it transforms a pair of similar terms into the regular

expression form that represents both of those terms; both terms are matched by this aggregate term. The main concern of the aggregation is the question of an appropriate aggregation level. In our first attempt at learning patterns (referred to as Experiment 1.0), we used salient terms lists of variable length s and allowed the aggregation process to run over the entire list, iteratively choosing pairs of terms to aggregate, until a chosen number of iterations was complete. Since we could not detect any improvement in document clustering, we turned to analysis of the created salient terms lists and the learned patterns for hints on improvement. Our conclusion was that the learned patterns entertained a very high degree of generalization, effectively matching an overwhelming number of terms, regardless of document class.

To minimize this risk in future experiments, we consider the string/character differences between different salient non-word terms within a single document class and attempt to group similar terms together prior to aggregating them. We use edit distance as the measurement of inter-non-word similarity within a document class. Given an edit distance threshold, nearest neighbor clustering is performed on the set of salient non-words within each document class; the clustering results in an arbitrary number of non-word clusters for each document class. Because each cluster contains non-words that are much more similar to each other than to non-words in other clusters, we estimate that the danger of over-generalizing with our aggregation process is smaller, and, equally important, that the risk of under-generalizing is negligible. Thus we proceed to run the aggregation within each cluster, until all terms within the cluster have been reduced to a single regular expression.

Length consideration

We return briefly to the discussion of edit distance. In the original implementation of the Levenshtein distance, edit distances of very short strings would result in very low

edit distances. Because we believe that term length is related to term salience, and that longer terms are more salient than shorter terms, we addressed this by adjusting the implementation of the edit distance algorithm to include scaling the edit distance by the sum of the lengths of the strings. To illustrate this point, consider s_1 = "1", t_1 = "2", and s_2 = 'truck", t_2 = "track". In the original implementation, edit distances of both pairs are equal: $d(s_1, t_1) = d(s_2, t_2) = 1$; in the scaled implementation, however, $d(s_1, t_1) = 0.5$, while $d(s_2, t_2) = 0.1$.

Nearest neighbor clustering

The clustering of salient terms is performed with an implementation of the k-nearest neighbor (kNN) algorithm. In kNN, a data point (i.e. term) is classified based on the *simple vote* of k already classified data points. The vote in this context is the cluster associated with each of the k classified points; simple means that simple majority rule applies (one point, one vote). Given an unclassified data point p, first its proximity to all other classified points (i.e. points already associated with a cluster) is calculated; then the information about cluster association of the closest k of these points is collected. Finally, whichever cluster is associated with the most points in this k-sized set is the cluster to associate p with. [29]

We alter the basic implementation of kNN, so that k is actually variable within a single run of the algorithm, as all classified data points participate in the vote. When considering a data point p, instead of polling the votes of the closest k neighbors, the distance of existing clusters from p is calculated; p is then added to the cluster with the lowest distance from p. For cluster c with data points p_i, $1 < i < n$, the distance of c from p is the average of distances between p and p_i:

$$d(c, p) = \frac{\sum_{i=0}^{n} d(p_i, p)}{n} \tag{3.4}$$

We label this altered implementation of k-nearest neighbors kNN_{avg}. Figure 3.4 illustrates the repositioning of the cluster centroid as new points are added; this is the desired consequence of using the average of edit distances of points currently in the cluster to the newly added point.

Algorithm 3.3.1: NEAREST NEIGHBOR CLUSTERING($terms, metric, threshold$)

```
clusters ← {}
clustercount ← 1
clusters[clustercount].add(terms[0])
terms.remove(terms[0])
for each term ∈ terms
do { eds ← {}
     for each c ∈ clusters
     do { eds[c] ← 0.0
          ed ← 0, count ← 0
          for each otherterm ∈ clusters[c]
          do { ed ← ed + metric(term, otherterm)
               count ← count + 1
          if ed/count ≤ threshold
            then eds[c] ← ed/count
     if eds = EMPTY
       then { clustercount ← clustercount + 1
              clusters[clustercount].add(term)
       else cluster ← minimum(eds)
     clusters[cluster].add(term)
```

The pseudocode of our nearest neighbor approach is illustrated in Algorithm 3.3.1.

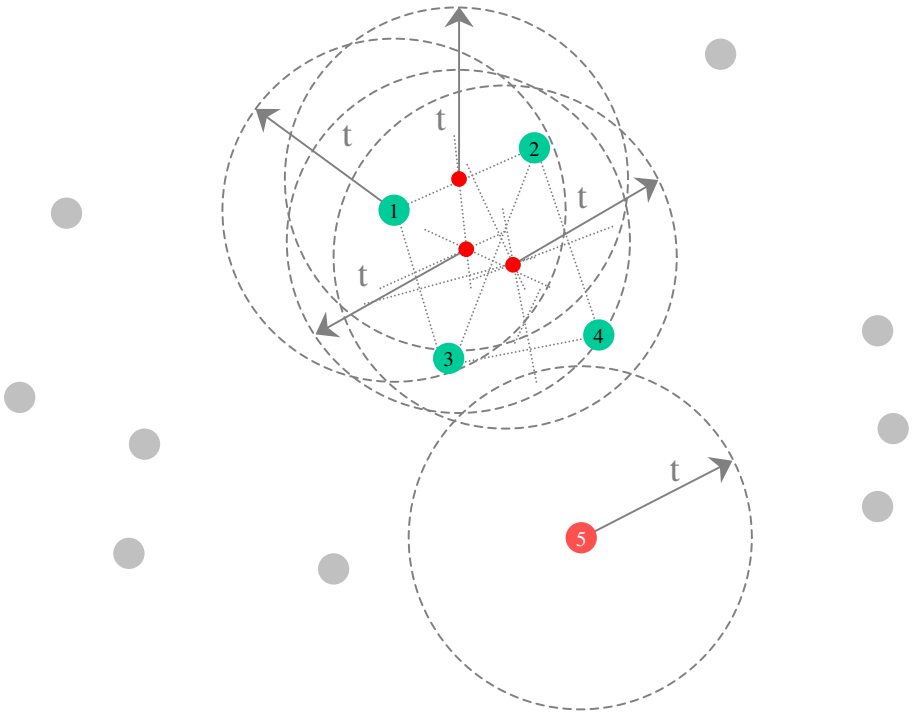

Figure 3.4: Nearest neighbor clustering using average edit distance.

3.4 Experiments

3.4.1 Expectations

When patterns have been learned for all document classes, they can be used in term-document matrix construction to make it document class-sensitive - just like our observed, hand-made patterns were used in our original set of experiments. The expected result is that the precision of the document clustering process will increase to some degree when compared to the clustering precision of not using any patterns.

Because the pattern clustering is unsupervised (the number of resulting clusters is unknown), different edit distance thresholds t will result in different cluster arrangements. For practical (runtime) purposes, not all salient non-words are selected to be

considered for pattern learning; instead, only a subset of s salient non-words represents each document class (we can afford to do this because the salient non-words list is sorted in what we believe to be a grouping-friendly manner).

These variables, s and t, become the parameters of our experiment: our document clustering is conducted for varying values of these variables. We expect that most choices of s and t will result in overly generalized patterns that will not improve the document clustering precision. We also expect that there will be a small subset of parameter values that will result in useful aggregations and aid in the document clustering. We speculate that most of the useful patterns will be produced for smaller values of t and larger values of s. We expect useful patterns for smaller values of t because the possibility of overly generalizing is minimized in those instances. Why we expect larger values of s to be more useful requires a closer look at the end of this section.

The pseudo-code for the experiment is described in Algorithm 3.4.1. There are defined sets of possible values for every variable in each experiment. Namely, C is a dictionary where each key represents a document classes c and value of each key is the list of all terms in that class, sorted by salience; S is a fixed set of integer values that salient terms list size s might be assigned; T is a fixed set of floating point values that salient terms clustering threshold t might take on; and W is a fixed set of floating point values allowed for pattern weight w. For every document class c, s top salient terms are selected and clustered with our nearest neighbor algorithm, the result of which are salient terms clusters $lpcs$. Each cluster lpc in $lpcs$ is fully aggregated to produce a single pattern lp, which is then added to the dictionary of patterns lpc under the key c. We use this dictionary of patterns to build the pattern-sensitive term-document matrix tdm and perform clustering of document-vectors in tdm.

Algorithm 3.4.1: PRECISION W/ LEARNED PATTERNS(C, S, T, W)

```
global classes, salientwords
for each w ∈ 𝒲
do { for each s ∈ 𝒮
     do { for each t ∈ 𝒯
          do { for each c ∈ 𝒞
               do { salients[c] ← c[1...s]
                    lpcs ← nearestneighbor(salients[c])
                    for each lpc ∈ lpcs
                    do { lp ← aggregatefully(lpc)
                         lps[c].add(lp) }
               tdm ← buildtdm(lps)
               clusters ← kmeans(tdm)
               precision ← compare(classes, clusters)
               output (precision) } } }
```

Expectation of useful values of s

In general, we believe it nearly impossible to predict the improvements to the document clustering by the choice of s: because some values of s might correspond better to natural groups of salient terms, as it pertains to our model of edit distance, we are prepared that there might be no obvious pattern to predicting useful values of s. However, we do have an intuitive notion of the consequences of sorting the salient terms list by alphabetness as it pertains to distribution of alphabetic characters in the

entire body of non-words. We expect a Gaussian distribution of terms with respect to ratio of alphabetic characters; on the other hand, we sort the salient terms by inverse ratio of alphabetic characters. This means that between selecting a smaller value s_i and the larger s_{i+1}, the similarity of newly introduced terms of size $s_j = s_{i+1} - s_i$ will contribute more significantly to the overall similarity of the salient terms set the larger s_i is. Effectively, for larger s, we expect the salient terms set to become more compact in terms of character similarity as measured by our implementation of edit distance.

3.4.2 Results

Experiment format

Each of the following experiments consists of four steps:

1. building a simple term-document matrix (TDMs),
2. learning patterns from TDMs,
3. building a pattern-sensitive term-document matrix (TDMp), and
4. clustering documents using both TDMs and TDMp.

At the end of each test run, the clustering achieved using the simple term-document matrix and the pattern-sensitive term-document matrix (both known as the *result*) is compared to the actual distribution of documents across the concept space (known as the ground truth or *reference*). As our goal for this thesis is a proof of concept regarding non-word patterns, we simplified the process to use informed ground truth (the actual distribution of documents across classes, collected with the corpus) rather than a simple clustering estimate. The correctness of the result in

VARIABLE	SYMBOL	VALUES
pattern clustering algorithm	kNN	{None, avg, agg}
secondary sorting key	ssk	{None, abic, abness}

Table 3.3: Experiment variables and their allowed values.

relation to the reference is expressed as the clustering *precision*. Finally, we graph the relationship of the precisions achieved with TDMs and with TDMp for different parameters of the experiment.

Our patterns are learned exclusively from the space of non-words, as we exclude terms found in the standard English dictionary from participating in building TDMs. We, however, test the effectiveness of the learned patterns in the natural space of document text tokens (which is the cumulative space of words and non-words); in other words, TDMp is built from words and non-words alike. This is consistent with our approach in experiments 0.0 and 0.2.

There is a limited set of properties that are varied across the experiment space. At the highest abstraction level, experiments differ on whether we employed pattern clustering prior to aggregation, or not. Furthermore, in experiments that employ pattern clustering, the clustering method slightly varies between experiments. Secondary sorting key to the salient terms list is another variable: some experiments employ it, some don't; those that do possibly employ different keys. Table 3.3 illustrates these variables and their possible values.

As we mentioned earlier, the size of the salient terms list, s, is varied within every experiment, and the pattern clustering threshold, t, is varied within every experiment that employs pattern clustering; in experiments without pattern clustering, there is variable aggregation limit, l. We add to these the pattern weighting, w, already used in our experiments with observed patterns; it dictates the participation ratio of pattern-recognized terms to other terms in the term-document matrix.

Finally, our general expectation from the set of experiments to follow is that we will be able to reproduce, to some degree, the effect that our hand-made patterns had on document clustering. In more abstract terms, we hope to confirm that it makes sense seeking out the semantic value of text tokens in the space of non-words and contemplating class-specific patterns of non-words.

The following is the look at the body of pattern learning experiments we conducted. To begin, we present our original attempt at pattern learning, for completeness and to motivate many of our implementation decisions.

Experiment 1.1: Verifying the value of learning by using controlled salient terms lists

To stifle the over-generalization of our learned patterns, we introduced two new steps to the experiment: secondary sorting key to the salient terms lists, and clustering of salient terms. To verify the general sanity of these new implementation decisions, we add a preprocessing step to the creation of salient terms lists: we only include those terms that are matched by our observed patterns.

Setup: Since we have shown (in experiments 0.0–0.2) that our observed patterns (Table 2.1) contribute to the precision of document clustering, we expect that patterns learned from a list of terms that are matched by those observed patterns will more likely improve document clustering than patterns learned from a more diverse list of terms. Showing this to be incorrect would force an adjustment in what we expect from our approach.

The problem with our salience ordering is that an overwhelming number of terms are exclusive to one document class, effectively tying their salience probability at $Q = 1.0$. The secondary sorting key aims to break those ties. The key is consistent

with our thinking in the space of non-words, as we believe that, within the space of non-words, terms that are less similar to words are more salient. The main attribute of words (think dictionary entries) is that they consist solely of alphabetic characters; hence, we measured a non-word term's dissimilarity to a word by the ratio of its alphabetic characters to its non-alphabetic characters: the smaller this ratio, the more salient the term. Since only the s top terms are selected to make the salient terms lists, it's important that the most salient terms are at the very top of the list; our belief was that this secondary sorting key will ensure exactly that.

Second, clustering of the salient terms was applied in the pattern learning process. Instead of cutting the aggregation off after a fixed number of iterations and aggregating over the entire salient terms list, salient terms are first clustered and the aggregation is performed over each cluster. As we said earlier, because the clustering is cut off at a given edit distance threshold, t, we are able to control the degree of generalization.

Results: As we expected, filtering the salient terms lists to only include terms matched by our observed patterns resulted in patterns that achieved a higher clustering precision (69%) than was achieved by clustering with a simple term-document matrix (54%). While the standard deviations of both precisions were very high (7% and 10%, respectively), we believed the experiment to be an encouraging result to continue in similar fashion.

Experiment 1.2: Measuring the value of learned patterns, part I

Setup: We followed by extending experiment 1.1 to its natural conclusion: omitting the step of filtering salient terms that are not matched by observed patterns. The salient terms lists include top s salient terms (in particular, $s \in \{25, 50, 75, 100, 125,$

$150, 200, 250\}$); salient terms clustering is performed for edit distance threshold t (in particular, $t \in \{0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.75, 1.0\}$); and terms matched by the learned patterns are added to the term-document matrix with weight w (in particular, $w \in 1000000$; w stands so high to ensure that any impact on the precision of document clustering, however small it might be, is observed).

Results: We observed statistically relevant increases in document clustering precision for several couplings of s and t (in particular, $s \in \{150, 200\}, t \in \{0.15, 0.2\}$; see graphs in Figure 3.5). Consistent with the expectations discussed at the beginning of this section, most couplings of s and t did not afford worthwhile increases in document clustering precision; however, as we expected, there were couplings that did. Furthermore, as we expected, most couplings occurred for the lowest values of t, confirming our suspicion that over-generalization prevents utility for larger values of t. Our expectations that useful values of s will be larger rather than smaller is partially confirmed; we discuss the anomalies to our expectations regarding the choice of s in section 4.2.1.

Experiment 1.3: Measuring the value of learned patterns, part II

Setup: This experiment builds on the success of experiment 1.2. We attempt to find out if slight changes in our pattern learning process can expand the space of useful couplings of s and t. The changes consist in details of our k-nearest neighbors algorithm implementation: instead of performing aggregation over each cluster once the salient terms have been clustered, the aggregation is performed gradually, as every cluster is represented by the intermediate aggregation of its current terms - a cluster *centroid* of sorts. Whenever a new term is added to a cluster, it is aggregated with the centroid and this new aggregate term effectively becomes the new cluster centroid.

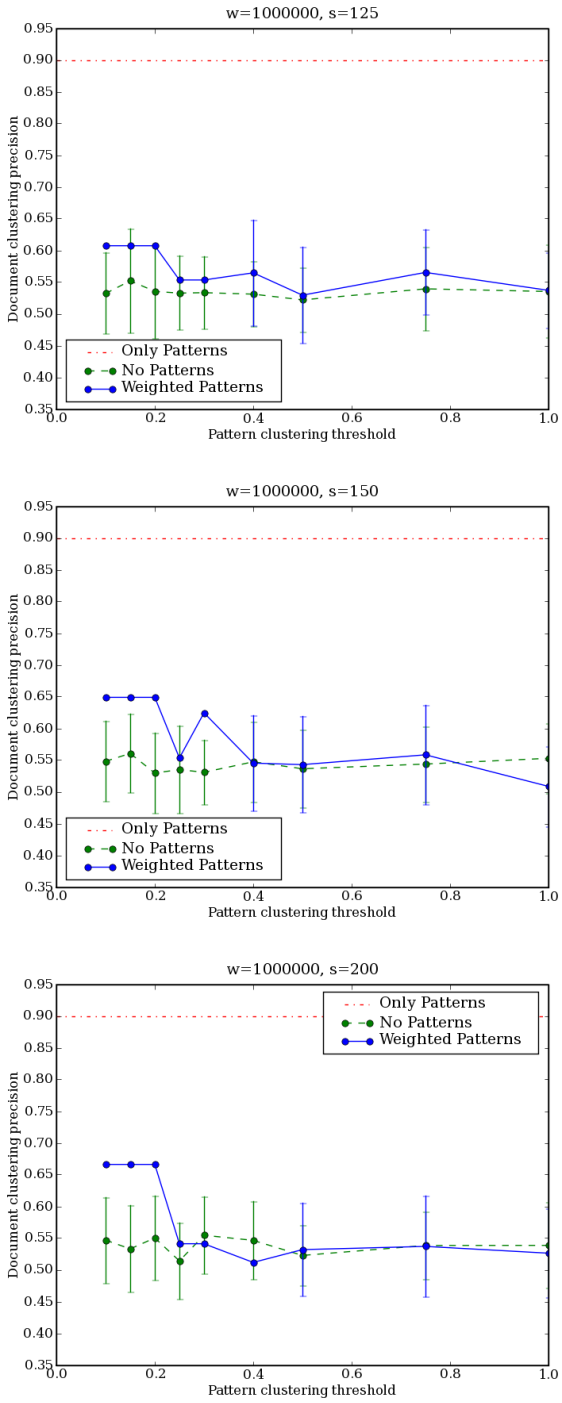

Figure 3.5: Statistically relevant improvements to document clustering precision for various values of s. Notice the peaks for $t = 0.1, 0.15, 0.2$.

The distance of any cluster c with centroid c_i from the term/point p is thus the edit distance of p to c_c:

(3.5) $d(c, p) = d(c_i, p)$,

where c_i is the aggregate term of i terms associated with cluster c so far. We label this implementation of k-nearest neighbors kNN_{agg}.

Results: Again we observe statistically relevant increases in document clustering precision. While all the couplings of s and t from experiment 1.2 are repeated, we discover one new coupling (see Figure 3.6). We discuss these particular results in the future work section of the last chapter.

Experiment 1.4: Finding the minimal significant pattern weights

Setup: In all of our previous experiments we used weight $w = 1000000$, as we deemed it high enough to reveal even the slightest clustering precision improvement our patterns would cause. For our last experiment we set out to empirically establish minimum weights w for which our learned patterns affect the document clustering precision. We varied s and t within the values of useful couplings discovered in experiments 1.2 and 1.3. Starting from $w = 1000000$, we decreased and increased w as we saw fit, observing the impact of the weight on the document clustering precision.

Results: Finally, we determined the range of values of w between which the document clustering precision is positively affected before it reaches a plateau (see graphs in Figure 3.7). This is consistent with the behavior we recorded in experiments 0.0–0.2; however, the maximum achieved precisions are significantly lower than in those experiments, indicating that our learned patterns, although good enough to show statistically relevant improvement, are not as good as our hand-made patterns.

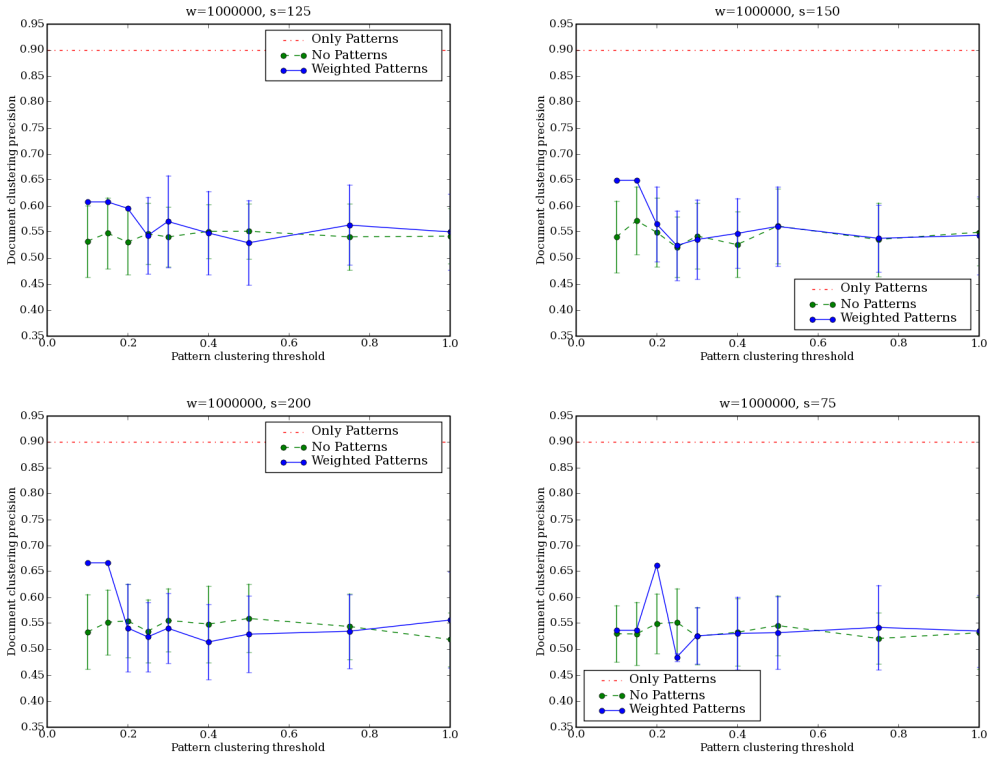

Figure 3.6: Statistically relevant improvements to document clustering precision for various values of s, now using a slightly different implementation of the nearest neighbor algorithm.

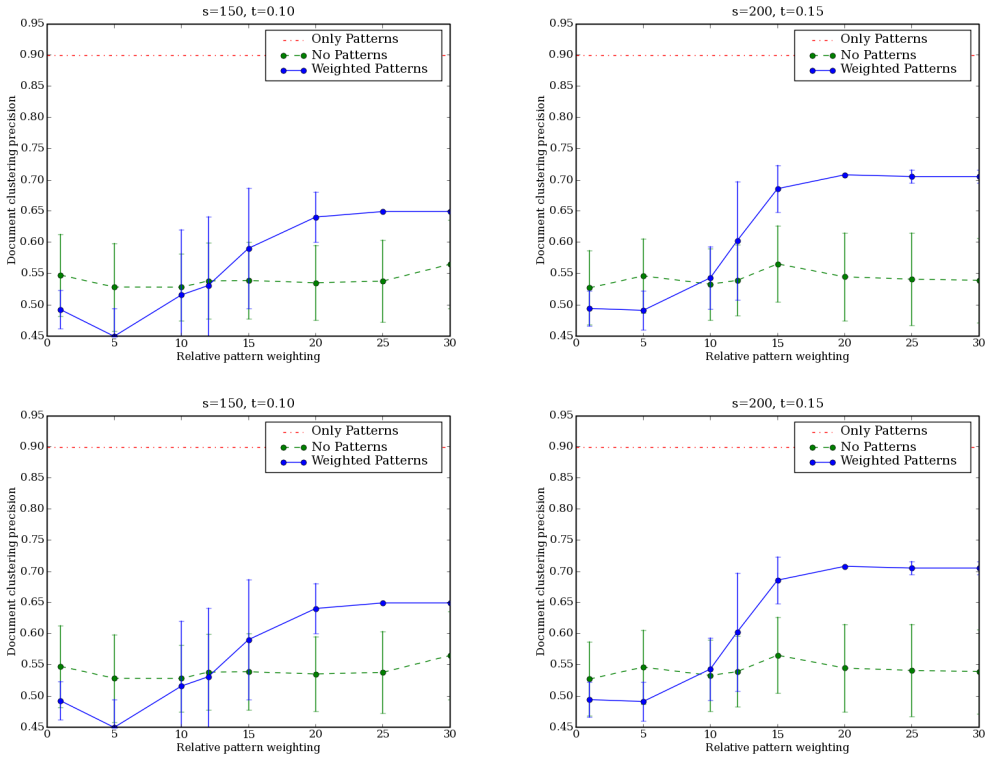

Figure 3.7: Precision of document clustering using learned patterns. As the significance weight of pattern-matched terms is increased, precision increases, until it reaches a plateau.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

In our quest for a new document feature to aid in term-based clustering of web documents, we turned to examining semantic value of text through analysis of its morphological properties. We were determined to utilize parts of document text that are not directly linked with meaningful interpretations, like words are, but can be semantically interpreted by observing their repetition and variability throughout the content whole. We analyzed text tokens in this space of non-words in order to obtain some generalized knowledge about how we can decide which non-word tokens are useful for distinguishing between textually related, but semantically different concepts.

As the result, we developed a process that allows us to learn such patterns of non-words that can be successfully used in aiding document classification. We verified, through a series of experiments, that the conclusions of our learning indeed brought intermediate success, as they accounted for statistically relevant increases in clustering precision for certain values of our experiment parameters.

4.2 Future work

The most obvious future direction to take with our general approach is expanding the analysis to several larger morphologically related and semantically diverse corpora of web documents; as of now, we've only demonstrated the existence of useful non-word patterns for one concept space.

4.2.1 Unexpectedly useful values of s

What remains to be addressed is a couple of particular couplings of our experiment parameters. In figures 3.5 ($s = 150$) and 4.1 isolated peaks in precision can be observed — peaks that are not preceded by high precision for smaller values of t (on their immediate left). We expected the improvements to clustering precision to come for the smaller values of t, because that is where the risk of overly generalizing is smaller. However, in these two graphs we see the precision increase for smallest values of t, then diminish, and then rise again for a single value of t. When discussing expectations related to the choice of our parameters, s and t, we anticipated that predicting the behavior of our pattern learning as a function of s would be difficult. We then offered the reason for this: particular values of s might correspond better to natural edit distance-wise grouping of salient terms than others. If a particular s didn't correspond well to this grouping, some resulting salient term clusters might create useless patterns, or, even worse, detrimental patterns — patterns that match significant volume of terms across document class boundaries. Future work should include inspecting the differences in the number and size of salient term clusters in the pattern learning process for anomalous values of s, as well as smaller and larger values in its vicinity. This would provide answers to the question what the properties are of salient terms clusters between two experiment runs, where one run shows

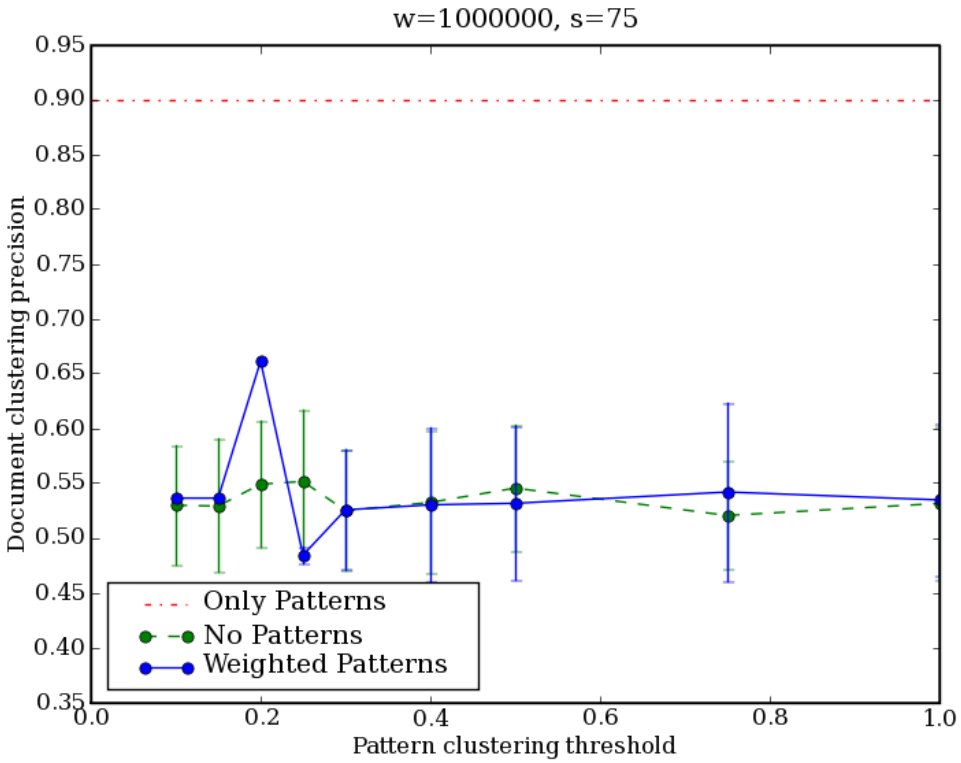

Figure 4.1: Precision peak anomaly. We would expect precision utility only for small and **continuous** values of t.

precision improvement and the other one doesn't — namely, do the added salient terms reinforce clusters from the previous run, or do they create new clusters, some of which are possibly detrimental, due to across-class aggregation.

Another issue that future work might inspect is the absence of precision peaks for values larger than $s = 200$. From the pattern of precision peaks in the graphs, we would expect that at $s = 250$ we might find a peak for values of t smaller than 0.1. This, however, does not occur. Examining even larger sizes of salient terms lists than $s = 250$ should be conducted; we were discouraged from using such large values by the computational cost.

Bibliography

[1] M. W. Berry and R. D. Fierro, "Low-rank orthogonal decompositions for information retrieval applications," *Numerical Linear Algebra with Applications*, vol. 3, pp. 301–327, Apr 1996.

[2] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 289–297, Morgan Kaufmann, Jul 1999.

[3] R. Baeza-Yates and B. Ribeiro-Neto, *Latent Semantic Indexing Model*, pp. 44–45. ACM Press, Addison Wesley, 1999.

[4] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[5] T. K. Landauer, P. W. Folz, and D. Laham, "Introduction to latent semantic analysis," *Discourse Processes*, vol. 25, pp. 259–284, 1998.

[6] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[7] W. Kintsch, "Predication," *Cognitive Science*, vol. 25, no. 2, pp. 173–202, 2001.

[8] P. Wiemer-Hastings, "Latent semantic analysis," in *Encyclopedia of Language and Linguistics*, Elsevier, second ed., 2004.

[9] C. D. Paice, "Method for evaluation of stemming algorithms based on error counting," *Journal of the American Society for Information Science*, vol. 47, pp. 632–649, Dec 1996.

[10] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 191–203, ACM, Jun-Jul 1993.

[11] M. Sinka and D. Corne, *A Large Benchmark Dataset for Web Document Clustering*, vol. 87 of *Frontiers in Artificial Intelligence and Applications*, pp. 881–890. IOS Press, 2002.

[12] R. Mihalcea and S. Hassan, "Using the essence of texts to improve document classification," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Sep 2005.

[13] J. Wang and F. H. Lochovsky, "Data extraction and label assignment for web databases," in *Proceedings of the 12th International Conference on World Wide Web*, pp. 187–196, ACM, May 2003.

[14] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards automatic data extraction from large web sites," in *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 109–118, Morgan Kaufmann, Sep 2001.

[15] B. Liu, R. Grossman, and Y. Zhai, "Mining web pages for data records," *Inteligent Systems*, vol. 19, pp. 49–55, Nov–Dec 2004.

[16] M. L. Cascia, S. Sethi, and S. Sclaroff, "Combining textual and visual cues for content-based image retrieval on the web," in *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries*, pp. 24–28, IEEE Computer Society, June 1998.

[17] R. Zhao and W. I. Grosky, "Narrowing the semantic gap: Improved text-based web document retrieval using visual features," *IEEE Transactions on Multimedia*, vol. 4, pp. 189–200, Jun 2002.

[18] H. H. Malik and J. R. Kender, "Clustering web images using association rules, interestingness measures, and hypergraph partitions," in *Proceedings of the 6th International Conference on Web Engineering, ICWE 2006*, pp. 48–55, ACM, Jul 2006.

[19] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proceedings of the 10th Annual International ACM-KDD Conference on Knowledge Discovery and Data Mining*, pp. 653–658, ACM, Aug 2004.

[20] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Transactions on Pattern Analysis and Machine Inteligence*, vol. 25, pp. 1075–1088, September 2003.

[21] S. Klink, A. Dengel, and T. Kieninger, "Document structure analysis based on layout and textual features," in *Proceedings of the 4th IAPR International Workshop on Document Analysis Systems, DAS2000*, pp. 99–111, 2000.

[22] D. Malerba, M. Ceci, and M. Berardi, "Xml and knowledge technologies for semantic-based indexing of paper documents," in *Proceedings of the 14th Inter-*

national Workshop on Database and Expert Systems Applications, pp. 256–265, Springer, Sep 2003.

[23] A. Hotho, S. Staab, and G. Stumme, "Wordnet improves document clustering," in *Proceedings of the 26th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, ACM, Jul–Aug 2003.

[24] T. Syeda-Mahmood, G. Shah, R. Akkiraju, A.-A. Ivan, and R. Goodwin, "Searching service repositories by combining semantic and ontological matching.," in *Proceedings of the IEEE International Conference on Web Services*, pp. 13–20, IEEE Computer Society, Jul 2005.

[25] M. F. Porter, *An Algorithm for Suffix Stripping*, pp. 313–316. Morgan Kaufmann, 1997.

[26] T. I. R. G. at University of Glasgow. `http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/`.

[27] B. W. Paleo, "An approximate gazetteer for gate based on levenshtein distance," in *Proceedings of the 12th ESSLLI Student Session*, pp. 197–207, Oeistein E. Andersen, August 2007.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Longest Common Subsequence*, pp. 350–356. MIT Press, McGraw-Hill, second ed., 2003.

[29] Y. H. Li and A. K. Jain, "Classification of text documents," *The Computer Journal*, vol. 41, no. 8, pp. 537–546, 1998.