Applying Data Mining Techniques in Text Analysis

Helena Ahonen Oskari Heinonen
Mika Klemettinen A. Inkeri Verkamo

{*hahonen,oheinone,mklemett,verkamo*}*@cs.helsinki.fi*

University of Helsinki, Department of Computer Science
P.O. Box 26, FIN–00014 University of Helsinki, Finland

Abstract

A number of recent data mining techniques have been targeted especially for the analysis of sequential data. Traditional examples of sequential data involve telecommunication alarms, WWW log files, user action registration for HCI studies, or any other series of events consisting of an event type and a time of occurrence.

Text can also be seen as sequential data, in many respects similar to the data collected by sensors, or other observation systems. Traditionally, texts have been analysed using various information retrieval related methods, such as full-text analysis, and natural language processing. However, only few examples of data mining in text, particularly in full text, are available.

In this paper we show that general data mining methods are applicable to text analysis tasks under certain conditions. Moreover, we present a general framework for text mining. The framework follows the general KDD process, thus containing steps from preprocessing to the utilization of the results. The data mining method that we apply is based on generalized episodes and episode rules.

We consider preprocessing of the text to be essential in text mining: by shifting the focus in the preprocessing phase, data mining can be used to obtain results for various purposes. We give concrete examples of how to preprocess texts based on the intended use of the discovered results and how to balance preprocessing with post-processing. We also present example applications including search for key words, key phrases and other co-occurring words, e.g. collocations and generalized concordances. These applications are both common and relevant tasks in information retrieval and natural language processing. We also present results from real-life data experiments to show that our approach is applicable in practice.

Keywords: data mining, text, text analysis, applications, experiments

1 Introduction

Recently, we have seen the sudden appearance of very large heterogeneous full-text document collections, available for any end-user. The variety of users' wishes is broad. The user may need an overall view of the document collection: what topics are covered, what kind of documents exist, are the documents related somehow, and so on. On the other hand, the user may want to find a specific piece of information content. At the other extreme, some users may be interested in the language itself, e.g. in word usages or linguistic structures.

A common feature for all the tasks mentioned is that the user does not know exactly what he/she is looking for. Hence, a data mining approach should be appropriate. Surprisingly enough, only a few examples of data mining in text, or *text mining*, are available. The most notable are the KDT and FACT systems [FDK96] used in mining Reuters news articles. Their approach, however, requires a substantial amount of background knowledge, and is not applicable as such to text analysis in general.

In this paper, we show that general data mining methods are applicable to text analysis tasks; we also present a general framework for text mining (Section 2). The framework follows the general KDD process, thus containing steps from preprocessing to the utilization of the results. We also present example applications within the fields of information retrieval and natural language processing (Section 3), and some results from real-life data experiments to show that our approach is applicable in practice (Section 4). Section 5 is a short conclusion.

2 General Framework for Text Mining

In our approach, we consider text as sequential data, in many respects similar to the data collected by sensors or other observation systems. The general knowledge discovery process, adapted to the task of text processing, is represented in Figure 1. The starting point is textual data, and the end product is information describing phenomena that are frequent in the data, e.g., phrases or co-occurring terms. In our approach, this information is presented as *episodes* and *episode rules*, two concepts which will be described in more detail in Section 2.1. In addition to describing the discovery methods, we explain the strategic decisions of the preprocessing and postprocessing phases that are necessary to focus our discovery process.

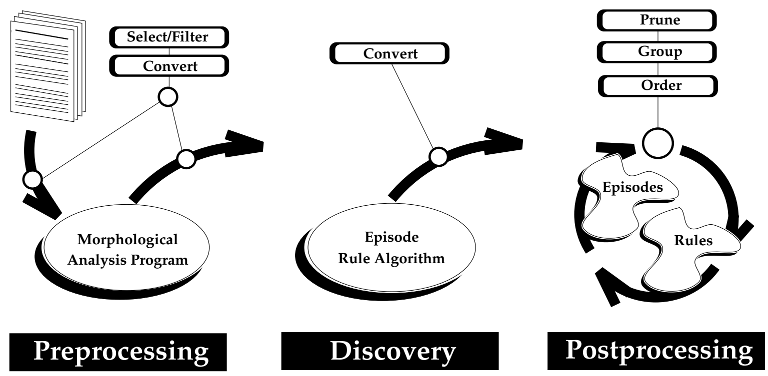

Figure 1: *Knowledge discovery from textual representation into episodes and episode rules.*

2.1 Episodes

Episode rules and *episodes* are a modification of the concept of *association rules* and *frequent sets*, applied to sequential data. (For the basic definitions of association rules, see, e.g., [AMS+96].) Sequential data, such as text, can be seen as a sequence of pairs *(feature vector, index)* where *feature vector* consists of an ordered set of features and *index* contains information about the position of the word in the sequence. It is common practice that the sequence is represented in an increasing order of the indices (corresponding to the order of the words in the original text). A feature can be

- a word; e.g., base form, inflected word form, stem,
- a grammatical feature; e.g., part of speech, case, number,
- a punctuation mark or other special character, or
- an SGML structure tag.

We define a *text episode* as a pair $\alpha = (V, \leq)$, where V is a collection of feature vectors, and $\leq$ is a partial order on V. Given a text sequence S, a text episode $\alpha = (V, \leq)$ *occurs* within S if there is a way of satisfying the feature vectors in V using the feature vectors in S so that the partial order $\leq$ is respected. Intuitively, this means that the feature vectors of V can be found within S in an order that satisfies the partial order $\leq$.

For an occurrence of the episode to be interesting, all feature vectors of the episode must occur close enough in S. What is close enough is defined by giving a limit, the *window size* W, within which the episode must occur. Hence, instead of considering all occurrences of the episode in S, we only examine occurrences within substrings S' of S where the difference of the indices of the feature vectors in S' is at most W. Moreover, since there may be several partially differing occurrences of the episode within the substring S', we restrict ourselves to the distinct *minimal occurrences* of the episode; for a formal definition of a minimal occurrence, see [MT96].

As an example, let us think of a text sequence where the feature vector contains the base form of the word, the part of speech, and the number of the word (when appropriate). The text *knowledge discovery in databases* would now be presented by the sequence

$$(\textit{knowledge}\_\texttt{N\_SG}, 1)\ (\textit{discovery}\_\texttt{N\_SG}, 2)\ (\textit{in}\_\texttt{PP}, 3)\ (\textit{database}\_\texttt{N\_PL}, 4)$$

For a window size of 2, this sequence contains the episode (*knowledge*_N_SG, *discovery* _N_SG), but does not contain the episode (*knowledge*_N_SG, *database*_N_PL).

The most useful types of partial orders are (1) total orders, i.e., the feature vectors of each episode have a fixed order; such episodes are called *serial*; and (2) trivial partial orders, where the order is not significant at all; such episodes are called *parallel*. A typical example of a serial text episode is a *phrase*, consisting of a sequence of related words, with a specific meaning. A typical example of a parallel text episode is a collection of *co-occurring terms* which may describe the contents of a document better than any of the single term.

The *support* of α in S (with respect to a given window size W) is defined as the number of minimal occurrences of α in S. Usually, we are only interested in episodes with a support exceeding a given *support threshold*, meaning that they occur in the sequence frequently enough not to be considered accidental.

We now move on to discuss episode rules. An episode rule gives the conditional probability that a certain episode occurs (within an interval of given window size), given that a subepisode has occurred (within the same or possibly smaller interval). Formally, an *episode rule* is an expression $\beta\,[win_1] \Rightarrow \alpha\,[win_2]$, where β and α are episodes, β is a subepisode of α and win_1 and win_2 are window sizes, with $win_1 \leq win_2$. The *confidence* of the rule is the conditional probability that α occurs, given that β occurs, under the window size constraints specified by the rule. (Since α includes all the feature vectors of β, we usually omit the feature vectors of β when representing the right-hand side of the rule.) An example of an episode rule could be

knowledge, discovery, in [4] $\Rightarrow$ *databases* [5] (85 %),

which tells us that in 85 per cent of the cases, where the three words *(knowledge, discovery, in)* occurred within 4 consequent words, also the word *databases* occurred within 5 words. Usually, we are not interested in rules with a negligible confidence, e.g., less than 20 per cent; hence it is common to select only those rules with a confidence exceeding a given *confidence threshold*.

The method that we have used to discover frequent episodes and episode rules in our data is described in [MT96]. This method allows us to discover serial and parallel episodes of a given support threshold and episode rules of a given confidence threshold for a collection of window sizes with a fixed upper limit.

2.2 Preprocessing the Data

As we saw in Figure 1, the process of obtaining useful information also relies on preprocessing the data before the discovery phase, and on postprocessing the results after the discovery phase. In particular, the preprocessing phase is crucial to the efficiency of the process, since according to the results in different domain areas and applications, preprocessing can require as much as 80 per cent of the total effort [Man96].

There are certain special aspects in the preprocessing of textual data. Text consists of words, special characters, and structural information. The preprocessing required depends heavily on the intended use of the results. Typically, the data is homogenized by replacing special characters and structural information (e.g., SGML tags) with symbols. Punctuation marks and structural information often need to be handled separately. Some of them may be ignored entirely, some of them may require special treatment, e.g., to find and process the sentence breaks which may be considered to represent a longer distance between two consecutive words. Another reason for special treatment of these characters is to replace them with symbols that give a better visual effect in the discovered results.

Preprocessing may involve some amount of natural language analysis. Morphological analysis gives us detailed information of the data. We may use this analysis to generalize the data, e.g., by replacing words by their parts of speech, which allows us to identify constructs such as *(preposition, noun)* instead of combinations of specific words.

Filtering of the data is used to focus our discovery phase, to limit the number of results so that we are not overwhelmed by uninteresting rules, and to decrease the processing effort needed in the discovery phase. Pruning can be used in two distinct ways. We may prune entire feature vectors, i.e., drop uninteresting items such as articles, prepositions, non-informative verbs (e.g., *be*), or punctuation marks, or select only some class of words (e.g., nouns) to be examined. We may also focus on some features of each word, e.g., by leaving only the base form of the word, or only part of the morphological information.

2.3 Postprocessing the Results

Postprocessing is essential to permit focusing our study after we have seen what kind of rules and episodes are found in the data. Typically, the user may be interested in a word or a set of words, and only the rules that contain these words are included, while all others are filtered out. If discrimination between several documents is needed the frequency of phrases is compared to the others in the collection to evaluate, if the phrase is considerably more frequent in this document than in general.

Many of the pruning decisions may be done either before or after the discovery phase. It is often advisable to defer these to the postprocessing phase instead of heavy pruning in the preprocessing phase, if we do not have a clear idea of what kind of regularities we are looking for. Instead of pruning entire feature vectors during preprocessing (e.g., dropping all prepositions) we can ignore all episodes and rules containing those feature vectors. However, during the postprocessing we cannot efficiently prune the single features included in the feature vector. Likewise, if we end up with a collection of very specialized rules instead of more general ones, we need to have a way of combining these using some predefined hierarchies of the features. On the other hand, with large collections of documents, efficiency has to be taken into account. In cases where we know what features we are not interested in, preprocessing must be preferred, to limit the size of the search space and the time requirement of the discovery phase.

As each feature vector is handled as one entity in the discovery phase, it is not possible to study the relation of distinct features in the co-occurring words, e.g., *noun* followed by *plural*. Additionally, co-occurrence of, e.g., two parts of speech may not become visible, if this need is not taken into account during preprocessing. The phenomenon itself might be frequent enough, but if the parts of speech are combined with, e.g., word stems, any distinct combinations may not occur frequently enough to be considered. On the other hand, if the support threshold is set very low, the result contains mostly uninteresting rules. Therefore, choosing a clear focus affects the result remarkably.

Efficient postprocessing requires tools to enable sorting and filtering the episodes and rules produced by the discovery phase. Based on the fixed representation of the episodes and rules, much of the postprocessing can be performed by general tools such as *grep* and *sort*.

3 Applications

3.1 Information Retrieval Tasks

In information retrieval — or more specifically text retrieval — key words and key phrases are commonly used to boost query processing [Sal88, LSJ96]. Consider a common information retrieval task: The user expresses his/her information needs, e.g., by giving a query, and the system executes the search by matching the query with the documents. With large collections simply scanning the text is not feasible. Hence, a set of representative key words must be selected and attached to the documents. For this purpose, single-term key words may be too broad to be used alone. Phrases consisting of sequences of related words carry a more specific meaning than the single terms included in the phrases.

A set of phrases can be regarded as a content descriptor that should distinguish the document from other documents in the collection. In addition to simple queries, content descriptors can be used for various text classification tasks. For instance, documents can be clustered according to their similarity, e.g., to visualize a large document collection [CKPT92].

Although indexing and selecting key words are well-studied within information retrieval, new challenges have been recently set by the sudden appearance of very large heterogeneous full text document collections. Lewis and Spärck Jones [LSJ96] consider compound key terms as one essential possibility to improve the quality of text retrieval in this new situation. They also emphasize the need of exhaustive experimenting to determine what the precise form of these compound terms should be, and how they should be selected and weighted relative to their constituents.

The preprocessing required for discovering key phrases is fairly straightforward. The grammatical features of the words are not used, and typically we are interested either in the original word or in its base form (e.g., *processing* or *process*). The structural information and punctuation marks are usually dropped, but they may affect the gaps in the indexing scheme, e.g., it is often desired that the words in a phrase occur in the same sentence. Common function words (prepositions, articles, etc.) are pruned.

A phrase is considered interesting according to its discriminating ability. Additionally, according to Salton [Sal88], parts of an interesting phrase should have different frequencies: the phrase head should have a frequency exceeding a stated threshold, while the other components should have a medium or low frequency; common function words should not be used. Such analysis is possible using some *grep*-like tool by selecting appropriate conditions for the left-hand and right-hand sides of the rule.

3.2 Natural Language Processing

Another application area is analysing the linguistic features of the text. We have considered three natural language processing applications:

1. discovering grammatical rules,
2. discovering collocations, and
3. constructing generalized concordances.

The *grammatical rules* that we consider here are any rules describing dependencies between linguistic features attached to words. For instance, we may want to study the structure of sentences by discovering the ordered sequences of parts of speech. The preprocessing requires leaving only the selected morphological features while the actual word forms are pruned. Depending on the focus of the study, entire feature vectors of some words and punctuation marks may be pruned as well. Postprocessing may include sorting and grouping of rules according to some features. Some examples of rules can be found in Section 4.

Collocations are recurrent combinations of words corresponding to arbitrary word usages. Unlike typical phrases used in information retrieval, collocations often contain prepositions and inflected words. Smadja [Sma93] distincts three types of collocations:

1. predicative relations; e.g., frequent predicate–object pairs like *make–decision*,
2. rigid noun phrases; e.g., *The Dow Jones average of 30 industrial stocks*, and
3. phrasal templates that may contain empty slots; e.g., *The average finished the week with a net loss of* [*NUMBER*].

In addition to the linguistic interest, collocations may be useful in retrieval tasks. It has been shown [Ril95] that some types of collocations are domain-dependent and, hence, good indicators of the topics covered by the document. What kind of collocations are considered interesting depends on the intended application. If they are used as content descriptors in information retrieval, their discriminating ability is one criterion.

A widely used tool for examining word usages in some collection of texts is constructing *concordances*: all the occurrences of a given word in the collection are listed together with the context, i.e., the words appearing immediately before and after the word. If the collection is large, however, concordances can provide too much data. One way to group different word uses or to rank them in order of importance is to sort concordance lines according to the collocations surrounding the word [Bib93]. We consider an even more advanced approach, so-called *generalized concordances*: frequent patterns that may contain both words and grammatical features. Preprocessing may drop all sentences that do not contain the given word. Another possibility is to remove the episodes or episode rules not containing the word in the postprocessing phase. As the features of a feature vector are handled as one entity, the current discovery method does not produce concordances including both word forms and grammatical features. For instance, if we are study the word *discovery*, then a pattern

```
knowledge /discovery/ in [N]
```

cannot be produced. However, the instances of this pattern, e.g.

```
knowledge /discovery/ in databases,
```

can be found, again, using some *grep*-like tool, given that original word forms and parts of speech are included in the feature vector.

4 Experiments

To survey the usefulness of knowledge discovery methods and the discovered knowledge in the context of text documents, we have made experiments with real data sets. We first briefly describe the data sets and the conversion of the original data into a suitable format for the analysis. Then, we present the experiments and their results.

4.1 Data Sets and Preprocessing

In the experiments we used a random sample of 14 documents taken from a large collection of Finnish legal texts, originally in SGML format. The sizes of the documents varied between 2 500 and 60 000 words (see Table 1). The number of words includes punctuation marks; the number of real words is about 20% lower and the number of words and their interpretations is about 20% higher, respectively. The SGML tags, used only in the discovery of structural dependencies (Section 4.4), are not included in the word counts.

After cleaning the data, the statutes were fed to a morphological analyser program (FINTWOL[1]), which gives us the base form and the morphological analysis of each word. Note that FINTWOL only looks at one word at the time and does not try to disambiguate using the word context. An example of the output is

[1] FINTWOL is a product of Lingsoft, Inc.

Statute		Phrases		Co-occ. terms		Parts of speech		Morphology		Structure	
#	Words	Epis.	Rules	Epis.(1)	Epis.(2)	Epis.	Rules	Epis.	Rules	Epis.	Rules
1	4 273	39	3	37	118	964	2 234	600	807	266	775
2	38 970	428	130	437	2 362	7 756	23 759	25 164	75 303	1 678	2 837
3	2 622	29	5	28	61	531	1 121	871	2 317	857	1 958
4	19 682	276	83	345	1 532	4 746	13 768	11 579	32 322	1 354	2 870
5	6 427	89	11	74	194	1 512	3 758	2 213	4 977	999	2 010
6	61 559	812	371	1 090	4 911	12 014	39 035	61 816	206 237	3 148	5 800
7	5 815	80	14	90	311	1 790	4 576	1 656	3 397	727	1 553
8	20 756	321	217	343	1 601	4 316	12 316	12 051	35 971	1 219	2 222
9	3 997	43	10	47	113	869	2 025	523	707	887	2 042
10	5 491	56	9	57	186	1 221	2 950	2 209	5 905	946	2 059
11	7 169	122	66	114	376	1 291	3 133	4 475	13 843	1 279	2 387
12	3 445	32	4	35	124	947	2 279	727	1 255	671	1 394
13	4 576	62	12	70	191	1 116	2 688	757	1 038	939	2 060
14	5 239	48	9	61	110	1 132	2 721	816	1 303	729	1 519

Table 1: *The test material and results. The number of co-occurring terms is taken from the test with a gap between sentences; without a gap the figures are about 20% higher. With co-occurring terms, figures (1) and (2) refer to the two different experiments described in Table 2.*

```
rikoslain rikoslaki N GEN SG
```

which tells us that the word which occurred in the text is *rikoslain*, its base form is *rikoslaki* (in English, Criminal Act), and the word is noun (N), genitive (GEN) and singular (SG). However, a word may have several interpretations, even being inflections of separate base forms.

After the preprocessing phase, the data format consists of one word, its morphological information and index per line. Multiple interpretations of the words are presented on consecutive lines. The selected features are concatenated as one entity (see examples in Section 2.1).

4.2 Phrases and Co-occurring Terms

In our preliminary experiments (see [AHKV97]), we selected certain interesting parts of speech and considered two simple test cases, the discovery of *phrases* and *co-occurring terms*. The latter case is particularly important in Finnish, where the word order is, in general, very flexible. We also wanted to compare the results between the analysis of the words with and without separating successive sentences. The former was implemented by adding enough space or null events between sentences; i.e., we increased the index between the sentences corresponding the maximum window size (see Table 2).

For phrase discovery, we selected only nouns, proper nouns, and adjectives. The search for co-occurring terms, on the other hand, was first carried through with nouns and proper nouns, and then with nouns, proper nouns, verbs, and adjectives. In the experiments, the words not included in the selected set were bypassed by increasing the index by 1. We also used different window sizes for episodes and produced both parallel and serial episodes. See Table 2 for the exact parameters used in the experiments.

The results from discovering phrases and co-occurring terms were rather similar. Both approaches produced reasonable results; in fact the main difference was some increase or decrease in the term/phrase frequencies. With co-occurring terms, the same effect occurred between the results obtained with or without a gap between the sentences. Ex-

Discovery of	Episode type	Selected parts of speech	Distinct symbols/ document	Episode support	Episode rule conf.	Gap between sentences	Window sizes
Phrases	serial	N, PROP, A	365..2693	10	0.2	3(+1)	1..3
Co-occurring Terms	parallel	(1) N, PROP (2) N, PROP, V, A	293..1904 457..3109	10	—	(a) 10(+1) (b) 1	1..10
Parts of Speech	serial	all (no words)	17..21	10	0.2	5(+1)	1..5
Morphology	serial	all (no words)	352..1245	10	0.2	5(+1)	1..5
Structure	serial	— (tags only)	34..62	10	0.2	10(+1)	1..10

Table 2: *The test parameter values. Note the abbreviations for nouns (N), proper nouns (PROP), adjectives (A), and verbs (V). For "parts of speech" and "morphology", "all" means that only special characters have been filtered out.*

amples of the phrases[2] we found are the terms *teollinen käsittely* (industrial processing) and *vesioikeus päätös* (Water Rights Court judgement) in Figure 2.

```
(a)  37:   teollinen (A)                 (c)  44:   vesioikeus (N)
           käsittely (N)                            päätös (N)

(b)  IF    teollinen (A)                 (d)  IF    vesioikeus (N)
     THEN  käsittely (N)                      THEN  päätös (N)
     WITH  [0]  [1]  0.0000 (0/38)            WITH  [0]  [1]  0.0000 (0/558)
           [0]  [2]  0.9737 (37/38)                 [0]  [2]  0.0681 (38/558)
           [0]  [3]  0.9737 (37/38)                 [0]  [3]  0.0735 (41/558)
```

Figure 2: *Examplary results from phrase discovery: episodes and episode rules from the Chemical Act (a and b; statute #9) and Water Rights Act (c and d; statute #6).*

The episodes and episode rules can be studied separately, but also in parallel: we can first search for frequent episodes and then study them more carefully by looking at the rules. For instance, consider the examples in Figure 2. If we take the episode (a) that occurs in the Chemical Act (statute #9) rather frequently then by looking at the rule (b) we can conclude that the phrase *teollinen käsittely* is not only a common phrase, but in practice the term *teollinen* always implies an immediate occurrence of the term *käsittely*. On the contrary, with an equally frequent episode (c) in the Water Rights Act (statute #6), the rule (d) tells us that *vesioikeus* is actually quite rarely immediately followed by *päätös*. This kind of analysis can be completed by looking at all rules that have either *vesioikeus* on the left-hand side or *päätös* on the right-hand side.

4.3 Morphological Dependencies

To complement the experiments with phrases and co-occurring terms, we expanded our approach to cover more detailed text and language analysis. We first filtered out entire feature vectors containing special characters and then — unlike in the earlier experiments — considered only the morphological information. The parameters of our two test cases, denoted as *parts of speech* and *morphology*, are shown in Table 2.

Consider the rules in Figure 3, taken from the National Pension Act (statute #8); similar phenomena hold in other statutes, too. In Finnish, an adjective is most often followed by a noun (Figure 3a); the opposite situation occurs much more rarely (Figure 3b). Moreover, the adjective attribute usually takes the case and the number of the headword noun (Figure 3c); the other possibilities (e.g. Figure 3d) are more infrequent.

[2]Note that the Finnish phrases are given without inflections and are not always proper phrases as such.

```
(a)  IF    A
     THEN  N
     WITH  [0]  [1]  0.0199 (45/2267)
           [0]  [2]  0.6608 (1498/2267)
           [0]  [3]  0.8090 (1834/2267)
           [0]  [4]  0.8646 (1960/2267)
           [0]  [5]  0.8972 (2034/2267)
(b)  IF    N
     THEN  A
     WITH  [0]  [1]  0.0095 (63/6610)
           [0]  [2]  0.1601 (1058/6610)
           [0]  [3]  0.2696 (1782/6610)
           [0]  [4]  0.3540 (2340/6610)
           [0]  [5]  0.4156 (2747/6610)
```

```
(c)  IF    A POS NOM SG
     THEN  N NOM SG
     WITH  [0]  [1]  0.0000 (0/198)
           [0]  [2]  0.3384 (67/198)
           [0]  [3]  0.4192 (83/198)
           [0]  [4]  0.4444 (88/198)
           [0]  [5]  0.4545 (90/198)
(d)  IF    A POS NOM SG
     THEN  N GEN SG
     WITH  [0]  [1]  0.0000 (0/198)
           [0]  [2]  0.0909 (18/198)
           [0]  [3]  0.1212 (24/198)
           [0]  [4]  0.1869 (37/198)
           [0]  [5]  0.2576 (51/198)
```

Figure 3: *Examplary results: parts of speech (a and b) and morphology (c and d).*

Since the Finnish language is morphologically rich, a plenty of dependencies are found in analysing large text documents — many of them much more complex than, e.g., in Figure 3.

4.4 Structural Dependencies

We also made preliminary experiments with the SGML structure using the tag information. First, we processed the data as before, by increasing the index on every word or symbol, but selecting only SGML tags. In the second experiment, we preprocessed the input data so that only SGML tags were considered; i.e., the index was increased only in case of SGML tags.[3]

The first experiment is useful in finding local phenomena, i.e., tags that occur close to each other in the text. The second approach can be used in detecting the document structure (document type definition). For example, consider the following simple example. In the first experiment we found the rule in Figure 4a which tells that the *chapter* start tag is followed by a *heading* start tag within 6 words. A rule found in the second experiment (Figure 4b) verifies the structure: *chapter* is always followed by both *heading* start and end tags within 9 words. The rule in Figure 4c complements our knowledge with information about the lengths of the *heading* elements, typically under 10 words.

```
(a)  IF    <CHAPTER>
     THEN  <HEADING>
     WITH  [0]   [6]  0.3871 (12/31)
(b)  IF    <CHAPTER>
     THEN  <HEADING>
           </HEADING>
     WITH  [0]   [5]  0.2258 (7/31)
           [0]   [7]  0.9355 (29/31)
           [0]   [9]  1.0000 (31/31)
```

```
(c)  IF    <HEADING>
     THEN  </HEADING>
     WITH  [0]   [3]  0.2111 (19/90)
           [0]   [4]  0.5667 (51/90)
           [0]   [5]  0.6889 (62/90)
           [0]   [6]  0.8111 (73/90)
           [0]   [7]  0.8556 (77/90)
           [0]   [8]  0.8778 (79/90)
           [0]   [9]  0.8889 (80/90)
           [0]  [10]  0.9111 (82/90)
```

Figure 4: *Examples of structural dependencies.*

[3]Note that in Table 1, the number of episodes and rules is given according to the results of the first experiment; in the second experiment the number on frequent episodes varied between 3 548 and 53 167, whilst the number of rules was from 8 540 to 105 383.

5 Conclusion

In this paper we showed that general data mining methods are applicable to text analysis tasks. Moreover, we presented a general framework for text mining. The framework follows the general KDD process, thus containing steps from preprocessing to the utilization of the results.

We gave concrete examples of how to pre- and postprocess texts based on the intended use of the discovered results. We also presented example applications from information retrieval and natural language processing and demonstrated the applicability of our approach with experiments on real-life data.

In further analysis, we plan to survey whether the results can be used in improving the overall accessibility of documents, and which tools are needed to make the analysis of a presumably large collection of episodes and episode rules more efficient.

References

[AHKV97] Helena Ahonen, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Mining in the Phrasal Frontier. Technical Report C-1997-14, University of Helsinki, Department of Computer Science, February 1997.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, California, USA, 1996.

[Bib93] Douglas Biber. Co-occurrence patterns among collocations: a tool for corpus-based lexical knowledge. *Computational Linguistics*, 19(3):531–538, 1993.

[CKPT92] Douglass R. Cutting, David Karger, Jan Pedersen, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference*, pages 318–329, Copenhagen, Denmark, June 1992.

[FDK96] R. Feldman, I. Dagan, and W. Klösgen. Efficient algorithms for mining and manipulating associations in texts. In *Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*, Vienna, Austria, April 1996.

[LSJ96] David D. Lewis and Karen Spärck Jones. Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101, 1996.

[Man96] Heikki Mannila. Data mining: machine learning, statistics, and databases. In *Proceedings of the 8th International Conference on Scientific and Statistical Database Management*, pages 1–6, Stockholm, Sweden, 1996.

[MT96] Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference* *on Knowledge Discovery and Data Mining (KDD'96)*, pages 146–151, Portland, Oregon, USA, August 1996. AAAI Press.

[Ril95] Ellen Riloff. Little words can make a big diffrence for text classification. In *Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 130–136, Seattle, Washington, USA, July 1995.

[Sal88] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1988.

[Sma93] Frank Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177, 1993.