Discovery of Frequent Word Sequences in Text

Helena Ahonen-Myka

University of Helsinki
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN–00014 University of Helsinki, Finland,
helena.ahonen-myka@cs.helsinki.fi

Abstract. We have developed a method that extracts all maximal frequent word sequences from the documents of a collection. A sequence is said to be frequent if it appears in more than σ documents, in which σ is the frequency threshold given. Furthermore, a sequence is maximal, if no other frequent sequence exists that contains this sequence. The words of a sequence do not have to appear in text consecutively.
In this paper, we describe briefly the method for finding all maximal frequent word sequences in text and then extend the method for extracting generalized sequences from annotated texts, where each word has a set of additional, e.g. morphological, features attached to it. We aim at discovering patterns which preserve as many features as possible such that the frequency of the pattern still exceeds the frequency threshold given.

1 Introduction

We have developed an automatic method for discovering textual patterns that can be used as compact content descriptors of documents [1, 2, 3]. The patterns have the form of a word sequence, i.e., we attempt to extract from the text a small set of word sequences that describe the contents of the document. Word sequences were chosen as a representation, since they have great potential to be a rich computational representation for documents, such that, on the one hand, feature sets for various further forms of analysis (e.g. text classification) can be easily retrieved, but, on the other hand, also a human-readable description of the document (e.g. a summary) can be generated from the representation.

Our discovery method extracts all the maximal frequent word sequences from the text. A sequence is said to be frequent if it appears in more than σ documents, in which σ is the frequency threshold given. Furthermore, a sequence is maximal, if no other frequent sequence exists that contains this sequence. The words of a sequence do not have to appear in text consecutively: a parameter g tells how many other words two words in a sequence can have between them. The parameter g usually gets values $1 - 3$. For instance, if $g = 2$, in both of the following two text fragments:

…President of the United States Bush…
…President George Bush…

a sequence *president bush* would be found (the articles and prepositions are not counted as words).

The ability to extract maximal sequences of any length, i.e., also very long sequences, and the allowance of gaps between words of the sequence distinguish our method from the other methods that could be used for extracting word sequences from text, e.g. the work on sequential patterns by Agrawal and Srikant [4] and the work on episodes by Mannila, Toivonen, and Verkamo [5]. The gaps make the word sequences flexible: the usual variety in natural language can be addressed. In addition to that long maximal sequences are very descriptive, they are also very compact representations. If we restricted the length of the sequences to, e.g., 8 words, and there actually were a frequent sequence of 25 words in the collection, we would find thousands of sequences that only represent the same knowledge as the one maximal sequence.

In this paper, we describe briefly the method for finding all maximal frequent word sequences in text and then extend the method for extracting generalized sequences from annotated texts, where each word has a set of additional, e.g. morphological, features attached to it. We aim at discovering patterns which preserve as many features as possible such that the frequency of the pattern still exceeds the frequency threshold given.

The following application illustrates the problem. A common type of language analysis is to create a *concordance* for some word, i.e., to list all the occurrences of the word in a text corpus, with some left and right context. A human analyst can then study the contexts and try to gather some generalized knowledge about the use of the word. If there are many occurrences, it may not be easy to characterize the contexts, particularly if the word has several senses. For instance, consider the following occurrences of a word *'right'*:

Is that the	right time?
...that things weren't	right between us.
Stay	right here.
They had the	right to strike.

In characterizing the contexts, the analyst might find out that some other words seem to occur frequently together with the word, or that the surrounding words belong to some class of words (e.g. nouns), or that they have a special form (e.g. singular, accusative). We can easily obtain this kind of information for words by using morphological analysis. A morphological analysis attaches each word with, e.g., the base form, number, case, tense, and part of speech. This process is rather fast with current tools. The analysis of a concordance could be automated by discovering generalized sequences, in which the infrequent features are removed, whereas features that appear often — and are probably more significant for the structures studied — are preserved, e.g.:

the	right 'Noun'
be	right between 'Pronoun'

'Verb' right here
the right to 'Verb'

The paper is organized as follows. In Section 2 the method for discovering maximal frequent word sequences is described. In Section 3 the method is extended to find generalized sequences in morphologically annotated texts. Section 4 contains some discussion.

2 Finding Maximal Frequent Word Sequences

Assume S is a set of documents, and each document consists of a sequence of words.

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

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

Note that we only count one occurrence of a sequence in a document: several occurrences within one document do not make the sequence more frequent.

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

The discovery process is presented in Algorithm 1. In the *initial phase* (steps 1–3) we collect all the ordered pairs, or 2-grams, (A, B) such that words A and B occur in the same document in this order and the pair is frequent in the document collection. Moreover, we restrict the distance of the words of a pair by defining a maximal gap; in our experiments we used a maximal gap of 2, meaning that at most 2 other words may be between the words of a pair.

The maximal frequent sequences are extracted in the *discovery phase* of Algorithm 1 (steps 4–19), which combines bottom-up and greedy approaches. A straightforward bottom-up approach is inefficient, since it would require as many levels as is the length of the longest maximal frequent sequence. When long maximal frequent sequences exist in the collection, this can be prohibitive, since on every level the join operation increases exponentially the number of the grams contained in the maximal frequent sequences. Although the greedy approach increases the workload during the first passes, the gain in efficiency is still substantial.

In the discovery phase, we take a pair and *expand* it by adding items to it, in a greedy manner, until the longer sequence is no more frequent. The occurrences of longer sequences are computed from the occurrences of pairs. All the occurrences computed are stored, i.e., the computation for ABC may help to compute later the frequency for $ABCD$. In the same way, we go through all pairs, but we

Algorithm 1 *Discovery of all maximal frequent subsequences in the document collection.*

Input: *S: a set of documents, σ: a frequency threshold*
Output: *Max: the set of maximal frequent sequences*

```
    // Initial phase: collect all frequent pairs.
1.  For all the documents d ∈ S
2.      collect all the ordered pairs within d
3.  G_2 = all the ordered pairs that are frequent in S
    // Discovery phase: build longer sequences by
    // expanding and joining grams.
4.  k := 2
5.  Max := ∅
6.  While G_k is not empty
7.      For all grams g ∈ G_k
8.          If g is not a subsequence of some m ∈ Max
9.              If g is frequent
10.                 max := Expand(g)
11.                 Max := Max ∪ max
12.                 If max = g
13.                     Remove g from G_k
14.             Else
15.                 Remove g from G_k
16.     Prune(G_k)
17.     G_{k+1} := Join(G_k)
18.     k := k + 1
19. Return Max
```

only try to expand a pair if it is not already a subsequence of some maximal sequence, which guarantees that the same maximal sequence is not discovered several times. When all the pairs have been processed, every pair belongs to some maximal sequence. If some pair cannot be expanded, it is itself a maximal sequence. If we knew that every maximal sequence contains at least one unique pair, which distinguishes the sequence from the other maximal sequence, then one pass through the pairs would discover all the maximal sequences. As this cannot be guaranteed, the process must be repeated iteratively with longer k-grams.

In the expansion step (step 10) of Algorithm 1, all the possibilities to expand have to be checked, i.e., at any point, the new item can be added to the tail, to the front or in the middle of the sequence. If one expansion does not produce a frequent sequence, other alternatives have to be checked. The expansion is greedy, however, since after expanding successfully it proceeds to continue expansion, rather than considers alternatives for the expansion. The choice of items to be inserted is restricted by the k-grams, i.e., also after expansion the sequence is constructed from the existing k-grams.

In the next step, we *join* pairs to form 3-grams, e.g., if there exist pairs AB, BC, and BD, we form new sequences ABC and ABD. Then we make a pass over all these 3-grams, and, as with the pairs, we try to expand grams that are not subsequences of the known maximal sequence and that are frequent. We can always remove those grams that are themselves maximal sequences, since such a gram cannot be contained in any other maximal sequence. The discovery proceeds respectively, variating expansion and join steps, until there are no grams left in the set of grams.

Algorithm 2 *Prune.*
Input: G_k*: a gram set*
Output: G_k*: a pruned gram set*

For each $g = a_1 \cdots a_k \in G_k$
    Let $LMax_g = \{p \mid p \in Max \text{ and } a_1 \cdots a_{k-1} \text{ is a subsequence of } p\}$
    Let $RMax_g = \{p \mid p \in Max \text{ and } a_2 \cdots a_k \text{ is a subsequence of } p\}$
    For each $p = b_1 \cdots b_n \in LMax_g$
        $LStr_{p,g} = \{b_1 \cdots b_{i_1 - 1} \mid a_1 \cdots a_{k-1} \text{ occurs in } i_1 \cdots i_{k-1} \text{ in } p\}$
    For each $p = b_1 \cdots b_n \in RMax_g$
        $RStr_{p,g} = \{b_{i_k + 1} \cdots b_n \mid a_2 \cdots a_k \text{ occurs in } i_2 \cdots i_k \text{ in } p\}$
    $LStr_g = \{LStr_{p,g} \mid p \in LMax_g\}$
    $RStr_g = \{RStr_{p,g} \mid p \in RMax_g\}$
    For each $s_1 \in LStr_g$
        For each $s_2 \in RStr_g$
            $s_{new} = s_1.g.s_2$
            If s_{new} *is not a subsequence of a maximal sequence*
                For each frequent subsequence s *of* s_{new}
                    If s *is not a subsequence of a maximal sequence*
                        Mark all grams of s
For each $g = a_1 \cdots a_k \in G_i$
    If g *is not marked*
        Remove g *from* G_k

Often it is not necessary to wait until the length of the grams is the same as the length of a maximal sequence, in order to remove a gram from the set of grams. After a discovery pass over the set of grams, every gram is a subsequence of at least one maximal sequence. Moreover, any new maximal sequence that can be generated has to contain grams either from at least two maximal sequences or two grams from one maximal sequence in a different order than in the existing maximal sequence. Otherwise a new sequence would be a subsequence of an existing maximal sequence.

This motivates the *pruning* phase of the algorithm, which proceeds as follows. For every gram it is checked how the gram might join existing maximal sequence to form new sequences. If a new candidate sequence is not a subsequence of some existing maximal sequence, all subsequences of the candidate sequence are considered in order to find new frequent sequences that are not contained in any maximal sequence, remembering that if a sequence is not frequent, its supersequences cannot be frequent. If a frequent sequence is found, all its grams are marked. After all grams are processed, grams that are not marked are removed from the gram set.

3 Finding Maximal Frequent Generalized Feature Sequences

We now move on to consider annotated documents, like in the sample collection of four documents in Figure 1. We assume again that S is a set of documents, but now each document consists of a sequence of feature vectors.

```
* Document 1
i       i       nom     pron
saw     see     past    v
a       a       sg      det
red     red     abs     a
ball    ball    nom     n
and     and     nil     cc
a       a       sg      det
green   green   abs     a
ball    ball    nom     n
* Document 2
the     the     nil     det
red     red     abs     a
ball    ball    nom     n
was     be      past    v
small   small   abs     a
* Document 3
the     the     nil     det
green   green   abs     a
ball    ball    nom     n
was     be      past    v
big     big     abs     a
* Document 4
he      he      nom     pron
saw     see     past    v
the     the     nil     det
balls   ball    nom     n
as      as      nil     adv
well    well    nil     adv
```

Fig. 1. A sample document collection.

In the sample, the first feature is an inflected word, the second is the base form, and the fourth is the part of speech. The third feature varies based on the part of speech of the word. For instance, 'nom' means nominative (nouns, pronouns), 'abs' an absolute (adjectives; as opposite to comparatives and superlatives), and 'past' means a past tense (verbs). Some parts of speech (adverbials, determiners) do not have any special information. Thus, the third feature has a value 'nil' for them.

We model each feature vector as an ordered set of feature values. In order to simplify the problem, though, we make some restrictions. First, we need the following notation.

Definition 4. *Let $r[u_i, \ldots, u_j]$, $1 \le i \le k$, $i \le j \le k$, be the set of occurrences of the feature vectors u in the document collection for which $\{u_i, \ldots, u_j\} \subset u$.*

The constraints are the following.

- Each feature vector has k feature values, i.e. $u =< u_1, \ldots, u_k >$, and the *ith* value of each vector, $1 \le i \le k$ represents a comparable generalization level within all the vectors.
- Dropping a feature value from the beginning of a feature vector generalizes the feature vector. That is, $r[u_i, \ldots, u_k] \subseteq r[u_j, \ldots, u_k]$, in which $1 \le i \le k$, $i \le j \le k$.

The second condition holds for many interesting linguistic features, like the ones in our sample. However, this condition does not hold, if we consider at the same time features like *number, case, and gender* for nouns, or *tense, mood and aspect* for verbs, since these features are on the same generalization level.

Note that the second condition could not be replaced by using taxonomies. That is, there is not necessarily an ISA relationship between feature values u_i and $u_{i'}$, $i < i' \le k$. For instance, from a feature vector $< saw, see, past, v >$ we cannot deduce that all the words with a base form 'see' are in the 'past' tense.

The definitions for word sequences can be updated for the generalized case in an obvious way.

Definition 5. *A sequence $p = a_1 \cdots a_k$ is a* g-subsequence *of a sequence $q = b_1 \cdots b_n$, if there exists a sequence of feature vectors $s = b_{j_1} \cdots b_{j_k}$ such that b_{j_i} occurs before $b_{j_{i+1}}$ in q, $1 \le i \le k$, and $a_i \subseteq b_{j_i}$ for each i.*

Definition 6. *A sequence p is* g-frequent *in S if p is a g-subsequence of at least σ documents of S, where σ is a given frequency threshold.*

Definition 7. *A sequence p is a* maximal g-frequent (sub)sequence *in S if there does not exist any sequence p' in S such that p is a g-subsequence of p' and p' is g-frequent in S.*

We can apply Algorithm 1 for the generalized case by modifying the initial phase. In the initial phase, all the frequent ordered pairs are collected. In Algorithm 1, two words that have at most g other words between them are collected, the frequencies of pairs are counted, and the frequent pairs are selected. The main difference, when we consider generalized sequences, is that, even if two feature vectors u and v that occur close enough may not be frequent in the collection, a pair (u', v'), where $u' \subseteq u$ and $v' \subseteq v$, may be. Hence, the initial phase of Algorithm 1 has to be enhanced by discovering which subsets of the feature vectors form frequent pairs.

The straightforward way would be to collect for each pair of feature vectors $(< u_1, \ldots, u_n >, < v_1, \ldots, v_m >)$ all the pairs $(< u_i, \ldots, u_n >, < v_j, \ldots, v_m >)$, in which $1 \leq i \leq n$ and $1 \leq j \leq m$ and count the frequencies. There are, however, some possibilities for optimization. The Lemma 9 below expresses a similar constraint as the 'Apriori trick', which is used in the discovery of association rules [6].

Lemma 1. *If a feature value f does not occur in at least σ documents, any sequence of feature vectors containing f is not g-frequent.*

The following lemma uses the knowledge of the features, as given in the conditions above.

Lemma 2. *Let $u =< u_1, \ldots, u_n >$ and $v =< v_1, \ldots, v_m >$ be feature vectors. If a pair $(< u_i, \ldots, u_n >, < v_j, \ldots, v_m >)$ is not g-frequent, also any pair $(< u_{i'}, \ldots, u_i, \ldots, u_n >, < v_{j'}, \ldots, v_j, \ldots, v_m >)$, in which $1 \leq i' \leq i$ and $1 \leq j' \leq j$, is not g-frequent.*

Based on the lemmas above, in the initial phase, the infrequent feature values are pruned from the feature vectors. Moreover, for each ordered pair of feature vectors, we can first collect all the pairs of suffixes. The frequencies of the pairs of suffixes can then be used to guide the creation of combinations in the following way.

If a pair of suffixes $(< u_i, \ldots, u_n >, < v_j, \ldots, v_m >)$ is frequent, in which $1 \leq i \leq n$ and $1 \leq j \leq m$, all the pairs $(< u_{i''}, \ldots, u_n >, < v_{j''}, \ldots, v_m >)$, in which $i \leq i'' \leq n$ and $j \leq j'' \leq m$, are frequent and, hence, are added to the set of 2-grams. Moreover, the pairs $(< u_i, \ldots, u_n >, < v_{j'''}, \ldots, v_m >)$, in which $1 \leq i \leq n$ and $j \leq j''' \leq m$, and the pairs $(< u_{i'''}, \ldots, u_n >, < v_j, \ldots, v_m >)$, in which $i \leq i''' \leq n$ and $1 \leq j \leq m$, may contain frequent pairs. Hence, these pairs have to be collected and the frequent ones are added to the set of 2-grams.

All the g-frequent pairs are given as input to the discovery phase, as described by Algorithm 1. This means that one location in text may contribute to several pairs in the set, as we can see in the following example. If we assume that the frequency threshold is 2 and the pairs "I saw" and "he saw" occur in the text, the following pairs are g-frequent.

```
<nom, pron> <saw, see, past, v>
<nom, pron> <see, past, v>
<nom, pron> <past, v>
<nom, pron> <v>
<nom> <saw, see, past, v>
<nom> <see, past, v>
```

```
<nom> <past, v>
<nom> <v>
<pron> <saw, see, past, v>
<pron> <see, past, v>
<pron> <past, v>
<pron> <v>
```

All the subsets are necessary, since in longer sequences more general feature vectors may be needed to make a sequence frequent. Each subset is indexed by a unique integer, which is passed as input to the discovery phase. Hence the discovery phase cannot use the knowledge that some subset is a generalization of some others. After the discovery phase (as in Algorithm 1), the resulting sequences are not maximal g-sequences, but for each g-frequent sequence also all the more general variations of the sequence are returned. For instance, a sequence

```
<nom, pron> <saw, see, past, v> <det> <ball, nom, n>
```

also produces 23 other sequences, e.g.

```
<pron> <saw, see, past, v> <det> <nom, n>
<pron> <v> <det> <n>
```

As the maximal g-frequent sequences are a compact representation, the more general sequences should be pruned away from the result. Compared to the discovery phase, this pruning can be done fast and efficiently. Moreover, it is rather easy to find all the more general and shorter subsequences that occur more frequently. In our sample document collection (Fig. 1), the following maximal generalized frequent sequences can be found. All of them have a frequency 2, i.e., they occur in two documents.

```
<det> <green, green, abs, a> <ball, ball, nom, n>

<det> <red, red, abs, a> <ball, ball, nom, n> <abs, a>

<the, the, nil, det> <abs, a> <ball, ball, nom, n> <was, be, past, v> <abs,a>

<nom, pron> <saw, see, past, v> <det> <ball, nom, n>
```

4 Discussion

Both the basic method for finding maximal frequent word sequences and the extension for finding generalized sequences have been implemented in Perl. The basic method has been tested using, e.g., the Reuters-21578 news collection, and the method is designed — and further developed — to cope with large document collections. It is not clear, however, how far we can come with the extension. At the moment, it has been tested with a toy example only, and analysing any larger collection would probably need development of more efficient data structures.

The special characteristics of the linguistic features includes the large variation of the frequency of the features. It seems to be that in any text collection, however large, half of the words occur only once, while features like the part of speech form closed sets with a rather small number of distinct values. Hence, discovering patterns based on the frequency, treating all the features uniformly, may not be adequate.

Acknowledgements

This work is supported by the Academy of Finland (project 50959; DoReMi - Document Management, Information Retrieval, and Text Mining).

References

[1] Helena Ahonen. Knowledge discovery in documents by extracting frequent word sequences. *Library Trends*, 48(1):160–181, 1999. Special Issue on Knowledge Discovery in Databases.

[2] Helena Ahonen. Finding all maximal frequent sequences in text. In *ICML99 Workshop, Machine Learning in Text Data Analysis*, Bled, Slovenia, 1999.

[3] Helena Ahonen-Myka, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Finding co-occurring text phrases by combining sequence and frequent set discovery. In Ronen Feldman, editor, *Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, pages 1–9, Stockholm, Sweden, 1999.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, March 1995.

[5] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215, Montreal, Canada, August 1995.

[6] 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.