Finding All Maximal Frequent Sequences in Text

Helena Ahonen-Myka
Wilhelm-Schickard-Institut für Informatik,
University of Tübingen,
Sand 13,
D-72076 Tübingen, Germany
helena.ahonen@acm.org

Abstract

In this paper we present a novel algorithm for discovering *maximal frequent sequences* in a set of documents, i.e., such sequences of words that are frequent in the document collection and, moreover, that are not contained in any other longer frequent sequence. A sequence is considered to be frequent if it appears in at least σ documents, when σ is the frequency threshold given. Our approach combines bottom-up and greedy methods, and, hence, is able to extract maximal sequences without considering all the frequent subsequences of them. This is a necessity, since maximal frequent sequences in documents may be rather long. When we have found the set of maximal frequent sequences for each document, we can use these sets as new descriptive representations of documents, and discover further relationships between documents or sequences. For instance, we can find that some sequences (e.g., names of two companies) seem to occur often together in the same documents. We have implemented the algorithm and experimented with a news feed collection.

1 INTRODUCTION

In the last decades, most of the information systems in industry, commerce, administration, education, and science have been transformed into electronic form. This has comprised creating both database and documentation systems to manage both structured and unstructured data. Only recently, the value of this data has been fully recognized, as well as the fact that the potential use of this data may greatly exceed the uses for which the data was actually collected and stored.

The new emerging notion of *knowledge management* includes efforts to utilize the existing data about, e.g., clients, products, and competition. For instance, patterns in client behaviour can be extracted. The research field of *data mining* (or *knowledge discovery in databases*) has in the last few years produced methods for finding patterns and regularities in structured data, mainly in databases. However, knowledge management has even more ambitious goals. One of the major attempts is to make available knowledge, e.g. within a company, that only few employees know but that would benefit a much larger group. Particularly in the large companies the employees are often unaware that solutions for their problems may already be known in some former projects or other working teams.

The latter form of knowledge is hardly stored in a rigid form in databases, but rather in all kind of documents, which can be more or less structured. Unstructured data, particularly free running text, places new demands to the data mining methodology. The usual goal of data mining is to discover some kind of patterns in the data, the representation of which can be, e.g., sets of frequently co-occurring items, or clusters of items that seem to behave similarly in some sense. When the data is structured, it is usually easy to define which is the set of items, i.e. the pieces of data, the occurrence or behaviour of which we are interested in. Regarding to the unstructured data, however, this is not at all obvious.

The more established fields of information retrieval and natural language processing, which naturally also have an important role in knowledge management, have traditionally concentrated on words or phrases. The phrases may be linguistic phrases, usually noun phrases, or statistical phrases, which are most often frequent noun-noun or adjective-noun pairs. In the data mining research, the solution has been to use keywords from a controlled vocabulary (Feldman & Dagan, 1995; Feldman, Dagan, & Klösgen, 1996), names (of people, companies etc.) or rigid technical terms, which usually are noun phrases. We believe that for true knowledge discovery, more versatile phrases are needed. For instance, verb phrases may carry important hints on acts and processes, like in the following sequences of words.

```
bank england provided money market assistance
board declared stock split payable april
boost domestic demand
```

In this paper we present a novel algorithm for discovering *maximal frequent sequences* in documents, i.e., such sequences of words that are frequent in the document collection and, moreover, that are not contained in any other longer frequent sequence. A sequence is considered to be frequent if it appears in at least σ documents, when σ is the frequency threshold given. For instance, we may require the sequences to occur in at least 10 documents.

Our goal is to find a rich computational representation for documents, such that, on the one hand, feature sets for various further forms of analysis can be easily retrieved, but, on the other hand, also a human-readable description of the document can be generated from the representation. One reason to extract maximal sequences, instead of fixed size frequent sequences, e.g. n-grams, is that maximal sequences are both flexible and compact representation. Maximal sequences are flexible, since gaps are allowed in the sequences between words, which is important due to the many variations in the real texts. Maximal sequences also reduce overlapping information in the representation. Assume that some maximal sequence of length 10 occurs frequently in the documents. If we extracted all the 5-grams, 6 grams would be found. If also gaps were allowed, some hundreds of grams were extracted. For some applications, the compact form of maximal sequences is preferred.

Maximal frequent sequences can be used, for instance, as content descriptors for documents: a document is represented as a set of sequences, which can then be used to discover other regularities in the document collection. As the sequences are frequent, their combination of words is not accidental and a phrase has a form that is present in many documents, giving a possibility to do similarity mappings for information retrieval, hypertext linking, clustering, and discovery of frequent co-occurrences. A set of sequences, particularly the longer ones, gives also as such a concise summary of the contents of the document. For this use, the sequences can be completed with the words, e.g. prepositions, from the text of the documents.

To the best of our knowledge, no other approaches to find maximal frequent sequences in text have been presented so far, at least no approaches that would meet our requirements. Namely, on the one hand we do not restrict the length of the sequences, and on the other hand we believe that the frequency threshold has to be set rather low to find any interesting sequences. Frequent sequences in text may be very long. For instance in a news collection, the same news story may appear several times with slight variations.

Related research includes discovery of frequent sets (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996) and discovery of sequential patterns (Agrawal & Srikant, 1995; Mannila, Toivonen, & Verkamo, 1995). In our context of textual data, a frequent set would be a set of words that co-occur frequently in documents, i.e., the order of the words is not significant and one word may occur only once in a set. The approaches to discover sequential patterns are usually modifications of the methods for finding frequent sets. Most of these approaches use bottom-up processing: first, the frequent sets, or sequences, of size 1 are found, then longer frequent sequences are iteratively formed from the shorter ones. Finally, also all the maximal sets or sequences are found. The problem from our point of view is that when the maximal sequences are of size, say, 20, the amount of shorter sequences is rather prohibitive. Hence, we cannot afford generating, or even considering, all the subsequences of the maximal sequences. Some approaches exist which try to compute the maximal sets directly, e.g. in (Gunopulos, Khardon, Mannila, & Toivonen, 1997) a randomized algorithm is used. In (Bayardo, 1998) the particular goal is to be able to handle large maximal sets. To summarize, the methods to discover sequential patterns also find maximal frequent sequences, but due to performance reasons they do not succeed with text documents, whereas no other methods for directly finding maximal frequent sequences exist.

The basic idea of our approach is to combine bottom-up and greedy methods. On the one hand, maximal sequences are constructed from shorter sequences, on the other hand a frequent sequence that is not contained in any known maximal sequence is expanded until the longer sequence is not frequent any more.

We have implemented the algorithm in Perl, and experimented with the publicly available Reuters-21578 news collection that contains 19000 small documents. With the frequency threshold 15, a total of 9,625 maximal sequences were found, 7,664 of which were of size 2. The longest maximal sequences had 22 words.

The rest of the paper is organized as follows. Section 2 contains a formal statement of the problem and describes our algorithm to discover maximal frequent sequences. In Section 3 some implementational aspects are discussed. Finally, experimental results are presented in Section 4.

2 FINDING MAXIMAL FREQUENT 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 frequency threshold given.*

Note that we only count one occurrence of a sequence in a document: several occurrences within one document does 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 other sequence* p' *in* S *such that* p *is a subsequence of* p' *and* p' *is frequent in* S.

The goal is to find all maximal frequent subsequences in the set of sequences, i.e., in the document collection. The outline of the method is presented in Algorithms 4, 5, and 6. In the *initial phase* (Alg. 4) we collect all the ordered pairs of words (A, B) such that A and B occur in the same document in this order and the pair is frequent in the document collection.

The maximal frequent sequences are extracted in the *discovery phase* (Alg. 5), which combines bottom-up and greedy approaches. We start from a set of frequent pairs. We take a pair and add an item to it, in a greedy manner, until the longer sequence is no more frequent. In the same way we go through all pairs, but we 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 are 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 sequences, the one pass through the pairs would discover all the maximal sequences. As this cannot be guaranteed, the process continues further.

Algorithm 4 *Initial phase: collect the frequent pairs.*
Input: S*: a set of documents*
Output: $Grams_2$*: all the frequent ordered pairs of* S

    For all the documents $d \in S$
        collect all the ordered pairs within d
    $Grams_2 =$ *all the ordered pairs that are frequent*
        in the set S
    Return $Grams_2$

As a 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 sequences and that are frequent. Always, we can remove the 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.

In the expansion step (Alg. 6) 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 considering alternatives for the expansion.

The basic version of the algorithm proceeds as many levels as is the length of the longest maximal sequence. When long maximal sequences exist in the collection, this can be prohibitive, since on every level the join operation increases the amount of the grams contained in the maximal sequences exponentially. Often, however, it is not necessary to wait until the length of the grams is the same as the length of the maximal sequence, in

Algorithm 5 *Discovery phase.*
Input: $Grams_2$*: the frequent pairs*
Output: Max*: the set of maximal frequent sequences*

```
k := 2
Max := ∅
While Grams_k is not empty
    For all grams g
        If a gram g is not a subsequence of
        some m ∈ Max
            If a gram g is frequent
                max := Expand(g)
                Max := Max ∪ max
                If max = g
                    Remove g from Grams_k
            Else
                Remove g from Grams_k

    Join the grams of Grams_k to form Grams_{k+1}
    k := k + 1
Return Max
```

Algorithm 6 *Expand.*
Input: p*: a sequence*
Output: p'*: a maximal frequent sequence such that* p *is a subsequence of* p'

```
Repeat
    Let l be the length of the sequence p.

    Find a sequence p' such that the length of
    p' is l + 1, and p is a subsequence of p'.

    If p' is frequent
        p := p'

Until there exists no frequent p'

Return p
```

Algorithm 7 *Prune.*
Input: $Grams_k$*: a gram set*
Output: $Grams_k$*: a pruned gram set*

```
For each g = a_1 ··· a_k ∈ Grams_k
    Let LMax_g = {p | p ∈ Max and a_1 ··· a_{k-1}
        is a subsequence of p}
    Let RMax_g = {p | p ∈ Max and a_2 ··· a_k
        is a subsequence of p}
    For each p = b_1 ··· b_n ∈ LMax_g
        LStr_{p,g} = {b_1 ··· b_{i_1 - 1} | a_1 ··· a_{k-1}
            occurs in i_1 ··· i_{k-1} in p}
    For each p = b_1 ··· b_n ∈ RMax_g
        RStr_{p,g} = {b_{i_k + 1} ··· b_n | a_2 ··· a_k
            occurs in i_2 ··· i_k in p}
    LStr_g = {LStr_{p,g} | p ∈ LMax_g}
    RStr_g = {RStr_{p,g} | p ∈ RMax_g}
    For each s_1 ∈ LStr_g
        For each s_2 ∈ 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 ··· a_k ∈ Grams_k
    If g is not marked
        Remove g from Grams_k
```

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 have 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.

The pruning phase proceeds as shown in Algorithm 7. For every gram it is checked how the gram might join existing maximal sequences 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, the supersequences of it cannot be frequent, neither. If a frequent sequence is found, all the grams of it are marked. After all grams are processed, grams that are not marked are removed from the gram set. In the experiments, this pruning approach has been shown to be rather vulnerable for cases, where there exists a set of several maximal sequences that share a long subsequence, since a large amount of subsequences has to checked. Therefore, the development of the algorithm in the near future will concentrate on speeding up this part.

Theorem 8 *The Algorithms* $4-7$ *find all the frequent maximal sequences, w.r.t. a given threshold* σ*.*

Proof

Maximal frequent sequences are frequent sequences that are not contained in any other frequent sequence. This implies that every maximal sequence contains a subsequence that distinguishes the maximal sequence from the other maximal sequences. Let s' *be a distinguishing subsequence for a maximal sequence* s*, with* $|s'| = l'$*. The maximal sequence* s *is found, at the latest, on the level* l'*. On the level* l' *the sequence* s' *occurs in the gram set. The sequence* s' *is not a subsequence of any other maximal sequence, and* s' *is frequent, as all the* *subsequences of a frequent sequence have to be frequent. Hence, the sequence s' is expanded. As the sequence s' is not contained in any other maximal sequence, Algorithm* 6 *returns the maximal sequence s.*

It remains to be shown that the sequence s' occurs on the level l', i.e., the constituents of s' are not removed on the lower levels. In the initial phase, all frequent pairs of documents are included, hence also all the pairs of s'. In the join phase, the pairs of s' are combined to form longer subsequences of s'. Grams are removed for three different reasons. Grams that are not frequent are removed, as well as grams that themselves are maximal sequences. As the subsequences of s' have to be frequent, and as s cannot contain any other maximal sequence, the constituents of s' cannot be removed for these reasons. In the pruning phase, the grams that cannot be constituents of new maximal sequences are removed. As all the grams are subsequences of some existing maximal sequences, a new maximal sequence has to combine grams either from two different maximal sequences or from one maximal sequence in a different order. Algorithm 7 *checks all these possibilities to combine maximal sequences in a new way, and finds all the frequent sequences that are not contained in the existing maximal sequences. Hence, it also finds and marks all the constituents of s', which means that they are not removed.*

3 IMPLEMENTATION

In this section we sketch some solutions that we have used to implement the maximal frequent sequence discovery algorithm. First, to make the discovery more efficient, we have restricted the maximal distance of two consecutive items in a sequence, i.e., when collecting the pairs in the initial phase, we only consider words that have at most two other words between them. This principle reduces the amount of pairs in the initial phase, which would otherwise need a lot of space and time. Furthermore, we do not end up extracting such undesirable small sequences, the parts of which have a distance, say, 100 words. The meaning of this kind of sequences would be most probably rather empty.

The major data structures used in the implementation include:

- a hash table that stores for each pair its exact occurrences in text,
- a hash table that stores for each prefix the grams that have this prefix,
- a hash table that stores for each suffix the grams that have this suffix,
- a hash table that stores for each pair the indexes of maximal sequences within which it is a subsequence,
- an array of maximal sequences.

Also the documents in which a gram occurs are attached to the gram. Originally, only the exact occurrences of pairs within the documents are stored. During the execution of the algorithm, however, all the computations of occurrences of longer sequences are saved. When a sequence is expanded by adding words to both the ends of it, the occurrences of the longer sequence are always a subset of the occurrences of the shorter sequence. Hence, the computation of new occurrences can be restricted to the old occurrences. Also, when words are added in the middle of the sequence, the old occurrences can be partially utilized.

As for each gram to be processed we have to know whether the gram is already contained in some maximal sequence, we need a fast way to check the fact. When a maximal subsequence is found, we mark up all the pairs it contains with the index of the subsequence in the table of maximal sequences. When we have to decide for some sequence whether it is contained in any maximal sequence, we can pick up fast the sequences in which the first pair of the sequence is contained. Then we check in regard to all these sequences, whether the entire sequence is included in them.

4 EXPERIMENTS

We have implemented the maximal sequence discovery algorithm in Perl. For experiments we have used the publicly available Reuters-21578 news collection[1], which contains about 19000 documents. The average length of the documents is 135 words. Originally, the documents contain 2,5 million words. After stopword pruning against a list of 400 stopwords, the amount of words is reduced to 1,3 million words, which are instances of 50000 distinct words.

The list of stopwords contains, e.g., single letters, pronouns, prepositions, some for the Reuters collection common abbreviations (e.g., pct, dlr, cts, shr), and some other common words that do not have any content-bearing role in text phrases. On the other

[1] www.research.att.com/~lewis/reuters21578.html

hand, many common verbs are not stopwords, since they may add some necessary context into phrases (e.g. call, bring, and sell). In addition to stopwords, all numbers are removed from the documents. Stopword pruning is necessary, since the efficiency of the algorithm may be ruined if there are a lot of words that appear several times in one document. Frequency of words as such is not a problem, although they naturally increase the size of data, and sequences containing very frequent words only may not be very interesting.

We have performed experiments with a frequency threshold 15. The hardware environment was Sun Enterprise 450 with 1 GB of main memory. In the initial phase, 12,071 frequent pairs were found. The pairs had altogether 427,775 occurrences, whereas there were in the average 23 frequent pairs in one document. Additionally, there were 406 documents without any frequent pairs. In Table 1 can be seen some performance figures of the execution with the frequency threshold 15. Before the first pass over the set of grams, 2.36 min was needed to construct the initial data structures. As can be seen from the table, most of the time is used on the first level (actually level 2), when also most of the maximal sequences, also the longer ones, are extracted. The second hard phase is the first pruning phase, on the level 4, basically due to a handful of maximal sequences that share a long common part. The amounts of maximal frequent sequences of various sizes are shown in Table 2.

In order to give some impression on the quality of sequences produced, in the following, all the sequences starting with either *acquire* or *acquired* can be seen. The words are not stemmed, since the inflections may carry some important meaning. However, in most cases stemming would probably combine sequences that have identical meanings. Moreover, some low frequent variations would together exceed the frequency threshold.

```
acquire additional
acquire american
acquire assets
acquire bank
acquire co
acquire common shares
acquire company
acquire control
acquire corp
acquire inc
acquire interest
acquire outstanding shares
acquire share
acquire stock
acquired assets
acquired company
acquired december
acquired group
acquired immune deficiency syndrome
acquired interest
acquired shares corp total outstanding
acquired shares inc total outstanding
acquired stake
acquired stock
acquired year
```

Typical group of sequences are names of persons, companies and institutions, which are generally rather rigid compositions, and hence, appear mostly in the same form:

```
bundesbank president karl otto poehl
european monetary system ems
```

Particularly among the smallest sequences, there are sequences that do not carry very specific information content, like the following.

```
expects higher
expects complete
```

However, also these sequences may be useful, when they are considered as part of the set of sequences for one document. From the other extreme, the following sequence was one of the longest sequences found.

```
federal reserve entered u.s. government securities
market arrange customer repurchase agreements fed
spokesman dealers federal funds trading fed began
temporary indirect supply reserves banking system
```

As the exact occurrences of sequences can be easily found, it is possible to complete the sequences using the words from the source documents, and hence, to make the sequences more readable. This kind of post-processing might be useful when sequences are considered as summaries of documents.

5 CONCLUSION

We have presented an algorithm to discover the maximal frequent sequences in a set of documents. The approach combines bottom-up and greedy methods to avoid generating all the frequent subsequences of the maximal frequent sequences. This is a necessity with text documents, since the maximal sequences can be very long. We have implemented the algorithm and experimented with the Reuters-21578 news collection. The quality of sequences is very reasonable and offers a good starting point for a broad spectrum of various further applications.

Table 1: Performance Figures of the Phrase Discovery Phase.

Level	2	3	4	5	6	7	8	9
Pass time over the set of grams (min)	26.12	2.33	0.40	0.04	0.01	0.03	0.02	0.00
Join time (min)	3.44	0.24	0.02	0.02	0.01	0.01	0.01	0.00
Prune time (min)	-	-	38.16	4.38	2.27	2.03	0.47	0.06
Grams in the set	12,071	4,451	2,934	343	361	501	373	67

Table 2: The Number of Maximal Phrases of Various Lengths.

Length	2	3	4	5	6	7	8	9	10	11	12
$\sigma = 15$	7,664	1,320	353	146	65	17	8	4	13	12	13

Length	13	14	15	16	17	18	19	20	21	22	23
$\sigma = 15$	5		1	1		1				2	

Acknowledgments

This work has been supported by the EC/TMR Marie Curie research training grant #ERBFMBICT972821.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. & Verkamo, A. I. (1996). 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.

Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. In *International Conference on Data Engineering*, March 1995.

Bayardo, R. (1998). Efficiently mining long patterns from databases. In *Proc. of the 1998 ACM SIGMOD Conference on Management of Data*, pages 85–93, 1998.

Feldman, R. & Dagan, I. (1995). Knowledge discovery in textual databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 112–117, Montreal, Canada, August 1995.

Feldman, R., Dagan, I. & Klösgen, W. (1996). 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.

Gunopulos, D., Khardon, R., Mannila, H. & Toivonen, H. (1997). Data mining, hypergraph transversals, and machine learning. In *Proc. of the 1997 ACM Conference on Principles of Database Systems*, 1997.

Mannila, H., Toivonen, H. & Verkamo A. I. (1995). 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.