19

Data Mining Meets Collocations Discovery

HELENA AHONEN-MYKA AND ANTOINE DOUCET

In this paper we discuss the problem of discovering interesting word sequences in the light of two traditions: *sequential pattern mining* (from data mining) and *collocations discovery* (from computational linguistics). Smadja (1993) defines a *collocation* as "a recurrent combination of words that co-occur more often than chance and that correspond to arbitrary word usages." The notion of arbitrariness underlines the fact that if one word of a collocation is substituted by a synonym, the resulting phrase may become peculiar or even incorrect. For instance, "strong tea" cannot be replaced with "powerful tea". Acquisition of collocations, a.k.a *multi-word units*, are crucial for many fields, like lexicography, machine translation, foreign language learning, and information retrieval. We attempt to describe the collocations discovery problem as a general problem of discovering interesting sequences in text. Moreover, we give a survey of common approaches from both collocations discovery and data mining and propose new avenues for fruitful combination of these approaches.

19.1 Representation of Text and Interesting Sequences

In this section, we discuss several alternatives for representing text and sequences. Moreover, we define the problem of discovering interesting sequences in text.

Assume a text is split into *a set of text fragments*, e.g., into a set of sentences, paragraphs, or documents. Each fragment is a *sequence*, i.e., an ordered list, of *events*. An event is a non-empty unordered set of *items*. Each

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

item belongs to an *alphabet*. If we are interested in word sequences, each event contains one item only, i.e., a word. Events can, however, also have some inner structure. For instance, we could record for each word token its baseform, part of speech, and some morphological information dependent on the part of speech, like in the following example.

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

In the sample, the first item of an event is an inflected word, the second is the base form, and the fourth is the part of speech. The third item 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 past tense (verbs). Some parts of speech (adverbials, determiners) do not have any specific information. Thus, the third item has a value 'nil' for them.

The *length* of a sequence can be defined as the number of items or as the number of events. Hence, using the first definition, the length of a sequence $\langle sg, det\rangle \rightarrow \langle a\rangle \rightarrow \langle ball, nom, n\rangle$ is six, whereas using the second definition, the length would be three.

In data mining, the sequential pattern discovery problem is usually stated as "*Find all interesting subsequences in the input data.*" If each event contains one item only, the subsequence relation can be defined in the following way.

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

For instance, the sequence $\langle$*unfair practices*$\rangle$ can be found in all of the three sentences in Figure 1. If an event can contain a set of items, it is enough if some of the items occur in the corresponding event in a longer sequence. For instance, $\langle det\rangle \rightarrow \langle nom, n\rangle$ is a subsequence of $\langle sg, det\rangle \rightarrow \langle a\rangle \rightarrow \langle ball, nom, n\rangle$, but $\langle saw, see, past\rangle \rightarrow \langle v\rangle$ is not a subsequence of $\langle saw, see, past, v\rangle$.

The *interestingness* of a subsequence is usually defined with respect to a set of *constraints*, which are assumed to represent some natural restrictions in

1. The **Congress** subcommittee backed away from mandating specific **retaliation against foreign** countries for **unfair** foreign **trade practices**.
2. He urged **Congress** to reject provisions that would mandate U.S. **retaliation against foreign unfair trade practices**.
3. Washington charged France West Germany the U.K. Spain and the EC Commission with **unfair practices** on behalf of Airbus.

FIGURE 1 A set of sentences (Reuters-21578 1987).

the domain. In practice, the constraints are also used to reduce computational costs. The most common constraint is the *minimum frequency*. The frequency of a (sub)sequence can be, e.g., the number of text fragments that contain it.

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

If we assume that the frequency threshold is 2, we can find two frequent sequences in our sample set of sentences: ⟨*congress retaliation against foreign unfair trade practices*⟩ and ⟨*unfair practices*⟩ (Fig. 1).

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

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

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

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

The *internal density* of subsequences can be influenced by constraining the occurrences of events into a predefined window. The size of a window can be a fixed constant, or some natural structure can be taken into account. For instance, the words in a sequence have to occur within a sentence or a paragraph. This latter constraint can be easily implemented by choosing the representation of a text to be a set of sentences or a set of paragraphs, respectively. We can also define a *maximum gap*, which gives the number of other words that are allowed in the text between the words of a sequence. If the maximum gap is zero, we find n-grams in the most common sense, i.e.,

sequences of words, where the words occur consecutively. It is also possible to define a *minimum gap*, although it is harder to find any reasons for that in a general case.

A *minimum* and *maximum length* of sequences can also be defined, although both are problematic in practice. Usually the minimum length of interesting sequences is 2. As the number of frequent sequences decreases radically when the length of sequences increases, we would probably lose a significant part of interesting sequences, if we set the threshold even to 3. The set of frequent pairs naturally also contains a load of uninteresting information, and hence, ignoring them is tempting. It seems, however, to be more reasonable to use some other ways to measure the interestingness than plain length. Setting a maximum length for a sequence may also be problematic, as very long frequent sequences may occur in text. If we set a length threshold to, say, 10, but there is a frequent sequence of length 22 in a text, the discovery process has to output all the 10-subsequences of this long sequence. This would mean outputting thousands of subsequences that actually only represent one sequence. If length is not restricted, the maximal frequent sequences get a chance to be a very compact representation of the regularities in text.

Discovery of interesting sequences in a text is influenced by the special characteristics of textual data. The alphabet size can be 50,000-100,000 words even in a moderate size text collection, which is high compared to alphabets in other application fields, e.g., there are 20 amino acids only. The distribution of words is skewed. There is a small number of words that are very frequent, whereas the majority of words are very infrequent. The words somewhere in the middle, i.e., words with moderate frequency, are usually considered the most interesting and most informative. If the very frequent words are removed, the resulting search space is very sparse. Also the length of the interesting sequences is skewed. There is a large number of short sequences, but also very long sequences are possible. An extreme case is, for instance, if the data set contains several copies of the same document.

19.2 Collocations Discovery

Collocations discovery has been an active research field for decades and various techniques have been explored. Three major types of approaches can be recognized (Schone and Jurafsky 2001): 1) segmentation-based, 2) word-based and knowledge-driven, and 3) word-based and probabilistic. Segmentation-based approaches have focused on identifying words in phonetic streams or in languages that do not include word delimiters. Word-based, knowledge-driven approaches use more or less advanced linguistic filtering, whereas word-based, probabilistic approaches attempt to find collocations using word combination probabilities. Many word-based techniques,

naturally, combine the both approaches.

Choueka et al. (1983) define a collocation simply as "a sequence of adjacent words that frequently appear together". In their approach the sequences are theoretically of any length but were limited to size 6 in practice, due to computational reasons. They experimented on a corpus of 11 million words from the New York Times archive and found thousands of common expressions such as "home run", "fried chicken", and so on. The criteria used to qualify or reject a sequence as a collocation is simply based on a frequency threshold, which makes the results dependent on the size of the document collection.

The technique presented by Mitra et al. (1987) for extracting syntactical phrases is based on a part-of-speech analysis of the document collection. A set of part-of-speech tag sequence patterns are predefined to be recognized as useful phrases. All maximal sequences of words accepted by this grammar form the set of phrases. For instance, a sequence of words tagged as "verb, cardinal number, adjective, adjective, noun" constitutes a phrase of length 5. Every subsequence occurring in this same order is also extracted, with an unlimited gap (e.g., the pair "verb, noun"). This technique defines no minimal frequency threshold.

Church and Hanks defined a collocation to be "a pair of correlated words" (Church and Hanks 1990), that is, as a pair of words that occur together more often than by chance. The correlation is measured by *pointwise mutual information* I:

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

where $P(x)$ and $P(y)$ are the probabilities of occurrence of the words x and y, and $P(x, y)$ is the probability that both occur simultaneously. Simultaneous occurrence can be defined in several ways. Two words may occur together, when they are adjacent and in a given order, while a more relaxed definition may require the words to occur within a given window or in the same document in any order.

Building on the work of Choueka et al., Smadja (1993) proposed a hybrid technique that uses statistical measures to find candidate sequences and syntactical analysis to extract collocations. In the statistical phase, the *z-score* of a pair is calculated by computing the average frequency of the words occurring within a 5-word radius of a given word (either forward or backward) and then determining the number of standard deviations above the average frequency for each word pair. Pairs with a *z-score* below a threshold parameter are pruned away. Syntactic filters are then applied to get rid of those pairs which are not considered to be true lexical collocations. For instance, if a candidate is a noun-verb pair, it is accepted only if it is identified either as

a subject-verb or as a verb-object collocation. Following the identification of word pairs, the collocation set is recursively extended to longer sequences, by searching for the words that co-occurred significantly often together with a collocated sequence identified earlier.

A more recent approach of Dias et al. (2000) generates directly all the n-grams, in practice 7-grams, and attempts then to identify their subsequences that are collocations. A sequence is considered to be a collocation, if its words are tighter associated than the words in any of its sub- or supersequences. The association measure used is based on the average expectation of one word occurring in a given position knowing the occurrence of the other $n-1$ words, also constrained by their positions. Also the frequency of a sequence is taken into account. This approach does not need any global thresholds.

Most of the approaches for collocations discovery find rather short sequences only. This is linguistically motivated by experimental evidence that most of the lexical relations associate words separated by at most five other words (Smadja 1993, Dias et al. 2000). As some other languages than English may not share this property, and as some applications, like text summarization, may benefit also from longer interesting sequences, it would be important to have methods that can find longer collocations as well. The approaches described above, however, cannot be straightforwardly extended to find longer sequences, when applied to large corpora. The performance bottleneck can be, at least partially, traced back to processing collocations for each word separately. In the next section we introduce several methods from the data mining community which might solve the problem by processing contexts of many words in parallel, and hence, reducing overlapping work.

19.3 Discovery of Maximal Frequent Sequences

In the spirit of some previous data mining algorithms (Mannila et al. 1995, Srikant and Agrawal 1996), one might suggest the breadth-first, bottom-up approach in Algorithm 1 for the discovery of maximal frequent word sequences. Given the specific characteristics of textual data, the applicability of this algorithm is rather restricted. It generates a lot of candidates and counting of their frequency is slow. In order to answer the question of whether a candidate occurs in an input sequence, all the k-sequences of the input sequence are conceptually generated and compared to the set of candidates. If frequent sequences can be longer than 10 words, this becomes prohibitive.

Some recent methods (Zaki 2001, Tsoukatos and Gunopulos 2001) have proposed ways around this problem. SPADE (Zaki 2001) accelerates frequency counting by using more efficient data structures, but it still enumerates all frequent sequences. DFS_MINE (Tsoukatos and Gunopulos 2001) uses a depth-first approach and, hence, avoids enumerating all subsequences. A

Algorithm 1 *Bottom-up*
Input: a set of sequences (e.g. a set of sentences), a frequency threshold
Output: a set of maximal frequent sequences

1. *Collect all the items of the input sequences, count them, and select the frequent ones.*
2. *Build candidate sequences of length $k+1$ from frequent sequences of length k*
3. *Prune a candidate if some subsequence is infrequent.*
4. *Count the occurrences of the candidate sequences in input and select sequences that are frequent.*
5. *If sequences left: Go to 2.*
6. *Choose the maximal frequent sequences.*

candidate $k+1$-sequence is formed by intersecting a k-sequence with all frequent items. The method has been developed for spatiotemporal data, where the number of different items is much less than in textual data. Intersecting frequent word sequences with all (or many) words is not reasonable.

As we have seen, discovery of maximal frequent sequences in text cannot rely on enumerating all the frequent sequences. Although breadth-first search enables more pruning, it is not feasible, as all subsequences are processed. Depth-first search makes direct computing of maximal frequent sequences possible, but it may have to consider several sequences which are not frequent in the text. We have developed a method MineMFS (Ahonen 1999a,b) that combines breadth-first and depth-first processing. It extracts maximal frequent sequences of any length, i.e., also very long sequences, and it allows an unrestricted gap between words of the sequence. In practice, however, text is usually divided into sentences, which restricts the length of sequences, and gaps as well.

The constraints used in the method are minimum and maximum frequency. Hence, words that are less frequent than a minimum frequency threshold and words that are more frequent than a maximum frequency threshold are first removed. Then, we collect all the ordered pairs, or 2-grams, (A, B) such that words A and B occur in the same sentence in this order and the pair is frequent in the sentence collection.

In the discovery part (Algorithm 2), maximal frequent sequences are discovered directly, expanding each k-sequence that is not a subsequence of any known maximal sequence, until the sequence is not frequent any more. After all k-sequences have been processed, those k-sequences that cannot be used to construct any new maximal sequences are pruned away. The remaining k-

Algorithm 2 *MineMFS.*
Input: G_2*: the frequent pairs*
Output: Max*: the set of maximal frequent sequences*

1. $k := 2$
2. $Max := \emptyset$
3. *While* G_k *is not empty*
4. *For all grams* $g \in G_k$
5. *If a gram* g *is not a subsequence of some* $m \in Max$
6. *If a gram* g *is frequent*
7. $max :=$ *Expand(*g*)*
8. $Max := Max \cup max$
9. *If* $max = g$
10. *Remove* g *from* G_k
11. *Else*
12. *Remove* g *from* G_k
13. *Prune away grams that are not needed any more*
14. *Join the grams of* G_k *to form* G_{k+1}
15. $k := k + 1$
16. *Return* Max

sequences are joined to form the set of $k + 1$-sequences (e.g. AB and BC would produce ABC), and the process is repeated. In our approach the set of k-sequences restricts the depth-first search. Although the alphabet size is large, we have to check only a few alternatives for expanding a sequence. As we do not enumerate all subsequences, we do not have to restrict the length of the sequences.

For experiments the publicly available Reuters-21578 newswire collection (Reuters-21578 1987), which contains about 19,000 non-empty documents, has been used. The average length of the documents is 135 words. Originally, the documents contain 2.5 million words. We have implemented the MineMFS algorithm in Perl and performed experiments using several values for a minimum frequency threshold and a maximum frequency threshold. For instance, with minimum threshold 5 and maximum threshold 200, we found 25,000 frequent 2-sequences, 6,100 3-sequences, 3,700 4-sequences, and so on. In this case, the longest frequent sequences had 20 words. Respectively, with minimum threshold 15 and maximum threshold 500, we found 9,000 2-sequences, 1,600 3-sequences, 650 4-sequences, and finally one sequence of 15 words.

19.4 Discussion

Data mining research often concentrates on developing efficient algorithms with simple frequency-based interestingness measures only. Hence, a new avenue could be to embed interestingness considerations of collocations discovery methods into some data mining algorithms. We have also identified a clear need for more general approaches: both data mining and collocations discovery methods are often developed with some specific data type, task, or language in mind, and, hence, they are not easily applicable to new areas. In the data mining community this need has recently led to the birth of a new research field: *local patterns detection* (Hand et al. (2002)). Its main objective is to develop a theoretical base that would make it possible to identify which methods and interestingness measures are useful for what purposes. In this same spirit we hope, one day, to be able to develop a collocations discovery method that can be easily tuned for any wish of a potential user, let it be common verb–object relations for a foreign language learner, the most significant topical phrases for information retrieval, or domain-specific terms for a machine translation system.

References

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

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

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

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

Dias, Gaël, Sylvie Guilloré, Jean-Claude Bassano, and José Gabriel Pereira Lopes. 2000. Combining linguistics with statistics for multiword term extraction: A fruitful association? In *Proceedings of Recherche d'Informations Assistee par Ordinateur 2000 (RIAO 2000)*.

Hand, David, Niall Adams, and Richard Bolton, eds. 2002. *Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK, September, 2002*. Lecture Notes in Artificial Intelligence. Springer.

Mannila, Heikki, Hannu Toivonen, and A. Inkeri Verkamo. 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.

Mitra, Mandar, Chris Buckley, Amit Singhal, and Claire Cardie. 1987. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200–214.

Reuters-21578. 1987. Text Categorization Test Collection, Distribution 1.0. http://www.daviddlewis.com/resources/testcollections/reuters21578.

Schone, Patrick and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In L. Lee and D. Harman, eds., *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108.

Smadja, Frank. 1993. Retrieving collocations from text: Xtract. *Journal of Computational Linguistics* 19:143–177.

Srikant, Ramakrishnan and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology, EDBT'96*, pages 3–17.

Tsoukatos, Ilias and Dimitrios Gunopulos. 2001. Efficient mining of spatiotemporal patterns. In *Proceedings of the SSTD 2001*, no. 2121 in Lecture Notes in Computer Science, pages 425–442.

Zaki, Mohammed J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning* 42(1):31–60.