Extracting Collocations from Syntactically Annotated Corpora

Oliver Sander, Ingrid Fischer
Lehrstuhl für Informatik 2, Universität Erlangen–Nürnberg, Martenstr. 3, 91058 Erlangen
osander@gmx.de, idfische@informatik.uni-erlangen.de
Harald Kirsch
LION bioscience, Waldhofer Str. 98, 69123 Heidelberg
kirschh@lionbioscience.com

Abstract. In this paper the extraction of collocations from biomedical text sources is described. The extraction of uninterrupted collocation candidates is introduced and for interrupted candidates a new technique using suffix tries is developed. It reduces computational complexity, compared to previous approaches to this task. Extraction is further extended to annotated biomedical corpora, which enables the discovery of patterns with mixed attributes like base forms combined with syntactical categories. Finally, different common approaches for evaluating candidates are explained, and their applicability to interrupted and mixed attribute candidates is discussed.

Keywords. Intelligent data analysis, collocations, syntactically annotated corpora

1 Introduction

In the BioPath project[1] methods and tools are developed for information extraction from biomedical scientific texts. The approach used allows fast approximate (shallow) syntactical parsing — or rather detection — of phrases expressing interesting facts about biologically interesting entities. Parsing is based on patterns which capture phrases commonly used to express facts like protein interactions, protein functions or mutations of genes or proteins. Many phrase patterns are necessary to cover the variability of natural language. Identifying the patterns manually turned out to be a major bottleneck.

Taking into account the fact that the finite state automata used for parsing are good at handling a huge number of patterns, two goals of different expected complexity were set:

1. Develop a method able to extract phrases or patterns of frequently used word combinations, so called collocations, from a set of documents.

2. Extend the methods to be able to derive *interrupted* collocations, i.e. frequently used sequences of words interrupted by arbitrary words or phrases.

There are many existing definitions for the term *collocations*. According to a very general, but useful one, a collocation is ”a conventional way of saying things” [MS99]. This turns out to be a very ”friendly” definition, as it does not include specific restrictions. ”Conventional way” can be interpreted differently, so that any phrase that occurs in a text often enough could be considered as a ”conventional way”. Even if this is a bit vague, it matches quite well with our idea of frequently used phrases. Various other definitions can be found in [FA96], [MS99], [Ben89]. Examples of collocations are:

light verbs: *to make a decision, to do a favor*

terminological expressions: *laparoscopic cholecystectomy, endoscopic sphincterotomy*

frozen expressions and idioms: *to pull somebody's leg, to kick the bucket*

Collocations are constructed from so called *tokens*. In normal text, every word is considered a token and a collocation is a sequence of fixed words (aka n-grams or factors) like:

the presence of
be used to

[1]BmBF project 0312385, a cooperation between LION bioscience, Heidelberg, Centrum für Informations- und Sprachverarbeitung (CIS) der Ludwig-Maximilians Universität München and the NLP group of the University of Salford (Uk)

be associate with
on the basis of
the aim of this study is to

We also hope to find what we call *mixed attribute collocations* like

was `adv` *shown to*

were `adv` represents an adverb, e.g. *recently* or *generally*.

An *interrupted collocation* is a set of tokens with an ordering, that occurs within a window of a given size. Opposed to uninterrupted collocations, which are only able to model fixed phrases, interrupted collocations also support token combinations which occur at a distance, with one or more gaps in between. The ordering of tokens is important. For example [CH89] *keep from* is a verb-preposition collocation, that typically occurs with a gap between its components:

keep *you* ***from*** *coming into contact*
keep *the doctor* ***from*** *interfering*
keep *the research project* ***from*** *being investigated*

The dataset used is a subset of MEDLINE[2] abstracts. MEDLINE is a bibliographic database with roughly 11 million abstracts ranging from biomedicine over medicine, nursing to dentistry. The English abstracts were augmented with additional information generated by a statistical Hidden Markov Model tagger. For each word in its surface form (as it appears in the original text) additionally the base form, syntactical category and morphology are stored. For this work 200 MB are available, but for most experiments a subset of 20 MB (approximately 300 000 words, 13 000 sentences) is used. This annotated corpus has the following form:

⟨tok⟩⟨sur⟩There⟨/sur⟩⟨lem cat="adv" mor=":b"⟩there⟨/lem⟩⟨/tok⟩
⟨tok⟩⟨sur⟩have⟨/sur⟩⟨lem cat="v" mor=":I"⟩have⟨/lem⟩⟨/tok⟩
⟨tok⟩⟨sur⟩been⟨/sur⟩⟨lem cat="v" mor=":P"⟩be⟨/lem⟩⟨/tok⟩
⟨tok⟩⟨sur⟩several⟨/sur⟩⟨lem cat="det" mor=""⟩several⟨/lem⟩⟨/tok⟩
⟨tok⟩⟨sur⟩reports⟨/sur⟩⟨lem cat="n" mor=":m"⟩report⟨/lem⟩⟨/tok⟩
. . .

In this notation the surface form (⟨sur⟩...⟨/sur⟩), its base form or lemma (⟨lem⟩ ... ⟨/lem⟩) are stored within the tag ⟨tok⟩...⟨/tok⟩ . Additionally as attributes of the *lem*-tag, the category and morphological information is stored. For our project we only used the base form and the syntactic category of the words following prior experience of LION. The internal corpus representation has the following form:

[('adv','there'), ('v','have'), ('v','be'), ('det','several'),('n','report), ...]

for the sentence start *there have been several reports....*

[2]available online from PubMed at
`http://www.ncbi.nlm.nih.gov/entrez`

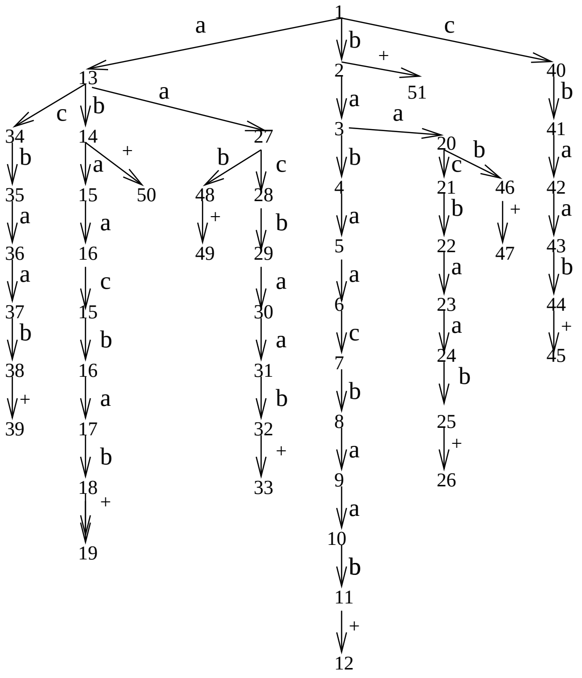

Figure 1 Suffix trie for the text T="babaacbaab"

First the corpus is converted into a suffix data structure, depending on the task a suffix trie or suffix array as explained in section 2. Iterating through these data structures, we can extract recurring phrases and count their frequencies. This step widely differs for uninterrupted, interrupted and mixed attribute collocations. In a next step these candidates have to be evaluated. Using different association measures each candidate is assigned a score, which is used to sort candidates in a rank list (see section 3). This whole process must be seen as semi-automatic, as a help for a human worker. Although appropriate chosen extraction and ranking methods can improve results, still errors concerning recall and accuracy occur. Our result are presented in section 4.

2 Extracting Collocation Candidates

In this project suffix tries have been used as one main data structure. A *suffix* of a text is a part of the text ranging from a certain position to the end. *Suffix tries* are trees, where each suffix of the corpus is a path from of the root to a leaf of the tree (see Figure 1). In tries all edges have only one assigned token. For many applications it is necessary to introduce a *sentinel character*, which is a token that does not occur in the normal set of tokens. This sentinel character should be appended to the original text as a marker for the end. Otherwise suffixes can exist, which are prefixes of other suffixes. These suffixes would appear as a part of a suffix within the tree. It is not distinguishable anymore, if it is a suffix on its own. Moreover

aab
aacbaab
ab
abaacbaab
acbaab
b
baab
baacbaab
babaacbaab
cbaab

Figure 2 Suffix array for the text T="babaacbaab"

it helps to count the number of leaves. In Figure 1 a suffix trie for $T = babaacbaab$ is given. As sentinel character the character $+$ is used.

For further information, the reader is referred to [GI93] and [McC76].

Two variants of suffix tries are also used. The first variant are depth-bounded suffix tries, in which paths do not exceed a given maximal length. As typically n-grams should be extracted only in a given range for n, this is sufficient for our needs. Another alternative is a trie, where paths, which lead directly to leaves are pruned.

In [MM93] suffix arrays are introduced. Instead of storing suffixes as paths in a tree, they are simply stored in lexicographic order (see Figure 2). This also results in groups of suffixes starting with the same prefixes. In terms of space efficiency, suffix arrays beat suffix trees by a factor of 3 to 5. This advantage is opposed to more time-consuming construction and searching.

2.1 Uninterrupted Collocations

In this subsection the extraction of uninterrupted collocations is described. [NM94] introduces a method for efficiently counting n-grams in large corpora. It is based on a lexicographic sorted suffix array of the whole corpus. Suffixes with common prefixes are grouped together, so that by comparing prefixes of array elements, frequencies of prefixes can be extracted. For each suffix the number of tokens it has in common with the succeeding suffix is computed. Nagao's technique leads to a problem called fractional substrings. If e.g. the phrase *the national academy of science of Ukraine* occurs six times in the corpus as well as the phrase *the national academy* it is clear that the second collocation is part of the first and should not be considered on its own. On the other hand, there are nested collocations which occur as part of others, but nevertheless have a meaning on their own, e.g.: *significant difference* as part of *statistically significant difference*. To decide whether a nested collocation can stand on its own or not is a semantic problem not easily handled with statistical approaches alone. Ikehara [ISSU96] ensures that each token occurs in at most one n-gram, thereby completely ignoring nested collocations, while [FA96] describes a method explicitly for nested collocations.

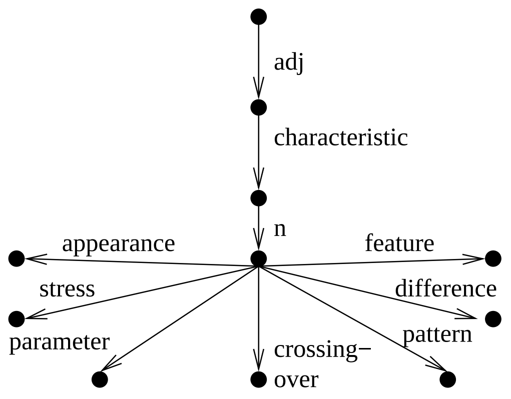

Figure 3 Two stage suffix trie for *adj:chacteristic n:**

2.2 Interrupted Collocations

For interrupted collocations we developed a new approach that searches for recurring subsequences in suffix tries. Subsequences in contrast to substrings allow interruptions. These subsequences can be directly taken as interrupted collocation candidates. We base our work on the idea that *frequent subsequences of length i are generated by extending frequent subsequences of length* $i-1$. Each subsequence of length 1, which can also be seen as an n-gram of length 1 (a single token), corresponds to an edge from the root. By the methods described in subsection 2.1, frequent 1-subsequences can be determined and stored. To compute frequent 2-subsequences, we are considering only the suffixes of frequent 1-subsequences. Given a specific 1-subsequence $\langle a \rangle$ we want to extend, we know all possible suffixes, which are represented by the subtree computed in the previous step. All nodes of this subtree can form a 2-subsequence $\langle a, b \rangle$ if combined with the 1-subsequence. As all descendants are considered, not only direct children, $\langle a, b \rangle$ can correspond to several nodes within the tree. All these nodes have to be stored as suffix representatives of $\langle a, b \rangle$. Again, the subtrees of these nodes represent all possible suffixes of $\langle a, b \rangle$. Opposed to the 1-subsequence, where only one node/subtree had to be stored, now several subtrees have to be considered. For counting, we use leaf-counting. If a subsequence $\langle a, b \rangle$ leads to several nodes, we sum up the counts for all these nodes. Further extension is now straightforward. For an $(i-1)$-subsequence we want to extend, we have a list of corresponding subtrees. For all descendants in all subtrees we are creating i-subsequences and the according node list, representing the suffixes.

2.3 Collocations with mixed Attributes

As for interrupted candidates we are building iteratively larger candidates from frequent smaller ones. A collocation candidate should be extended either by a fully specified token (*category:base form*) or by a token with specified category and arbitrary base form (*category:**). In order to simplify these operation, we are building our suffix trie differently. Instead of assigning tokens to edges, we separate tokens into two stages. The attribute ”category” leads to a new node, from where the attribute ”base form” leads to the second level. This results in a similar tree, but branching is simply split up in a ”category step” and a ”base form step”. See Figure 3 for an example.

3 Scoring and Ranking of Collocation Candidates

Given a list of collocation candidates, we are now facing the task of selecting those, which are really collocations and discarding the others. For each candidate additional information is available. For more detailed information and corresponding formulas see [MS99].

Using frequency to decide if a word combination is a collocation or not, is the simplest technique. The underlying idea states: if a combination is occurring often enough, there must be a reason for that. So the components of the word combination are assumed to have a relation.

Another possibility is to look for candidates that form a collocation that occur often enough and follow a special grammatical pattern. For the pattern ”adjective noun” *systemic treatment, soft tissue, postoperative complication* are collocations in the MEDLINE corpus.

Mutual information is an information theoretic measure, which can also be used as an association measure. Mutual information expresses the amount of information a specific event gives about the occurrence of another event. For our application events are word occurrences, so MI measures how much information a word occurrence gives about another, or the amount of certainty about the following words, a word occurrence gives us. In its original defintion, pointwise mutual information for two specific events x' and y' is given by:

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

[MS99] describes two important problems, which the use of MI imposes. The first problem is that the concept of mutual information is not exactly what we want for our application. Mutual information only measures the strength of consequence, but how often this consequence takes place (determined by the frequency) is not taken into account. The second problem, arises with low frequency events. As noted by many authors, MI fails if the observed frequencies get very low (e.g. lower than 3). In that case the estimated expected frequency gets extremely low, which assigns a higher mutual information score. Although all association measures show problematic behaviour with low frequency data, mutual information seems to be particularly worse. Of course this problem is closely related to the first one. Rare events are more likely to have a stronger association, as the variety of contexts they occur in, is limited by their rarity.

Cost criteria measures for scoring collocations are based on following idea: How much simpler would our corpus get, if we regard collocation candidates as units, represented by a single token. Kita [KKOY94] provides an introduction to a cost criteria measure, which embodies this concept in a simple way.

$$K(\alpha) = (|\alpha| - 1) \cdot f(\alpha)$$

with α being a collocation candidate (a sequence of tokens), $|\alpha|$ being the length of α (number of tokens) and $f(\alpha)$ being the occurrence frequency of α in the whole corpus. The justification of this formula is the following: we are calculating the simplification we would achieve in our corpus, if we regard all appearances of α as a single unit. For each appearance we would have 1 instead of $|\alpha|$ tokens, which is a gain of $|\alpha| - 1$. To get the gain for the whole corpus, we multiply by the number α occurs in the text. This means, the higher the cost is, the better the corresponding collocation is. Moreover, this cost criterion is extended to handle fractional substrings.

$$K(\alpha) = (|\alpha| - 1) \cdot (f(\alpha) - f(\beta))$$

Here, $f(\beta)$ is the total frequency of larger candidates containing α. For our above example (page 3) ”statistically significant difference” (occuring 19 times) and ”significant difference” (occurring 69 times) this would result in:

$$K(\text{"significant difference"}) = (2-1)\cdot(69-19) = 50$$

$$K(\text{"statistically significant difference"}) = (3-1)\cdot 19 = 38$$

As above, the smaller candidate is not counted if it occurs within the larger one. If this is not desired, [FA96]

Frequency	**Cost Criterion**
candidate	candidate
of:prep the:det	of:prep the:det
in:prep the:det	in:prep the:det
and:cnj the:det	and:cnj the:det
to:prep the:det	to:prep the:det
patient:n with:prep	patient:n with:prep
on:prep the:det	on:prep the:det
for:prep the:det	for:prep the:det
with:prep the:det	with:prep the:det
have:v be:v	have:v be:v
be:v a:det	be:v a:det
effect:n of:prep	effect:n of:prep
in:prep a:det	in:prep a:det
of:prep a:det	of:prep a:det
by:prep the:det	by:prep the:det
to:infp be:v	the:det effect:n of:prep

Table 1 Top 15 candidates for frequency and cost criterion scoring

MI	frequency	candidate
98,71	2	of:prep il-12:n to:infp substitute:v for:prep [. . .]
97,33	2	the:det final:adj product:n be:v confirm:v [. . .]
97,14	2	tannin:n on:prep nutrient:adj utilisation:n [. . .]
97,11	2	pancreatitis:n with:prep diffuse:adj irregular:adj [. . .]
97,10	2	urinary:adj bladder:n and:cnj uretero-seminal:adj [. . .]
96,99	2	dose-response:adj relation:n be:v investigate:v [. . .]
96,96	2	prostatic:adj tissue:n in:prep urinary:adj [. . .]
96,83	2	bladder:n and:cnj uretero-seminal:adj vesicle:n [. . .]
96,50	2	tissue:n in:prep urinary:adj bladder:n and:cnj [. . .]
96,02	2	and:cnj diffusion:n behavior:n of:prep cs:en and:cnj [. . .]

Table 2 Top 15 candidates scored with MI

suggests a further modification, which handles nested collocations in a better way:

$$K(\alpha) = (|\alpha| - 1) \cdot (f(\alpha) - f(\beta)/c(\alpha))$$

where $c(\alpha)$ is the number of different strings β which contain α. So if α occurs only in a single larger candidate string, this formula equals Kita's. But if α occurs in several larger collocation candidates, $c(\alpha)$ raises as well as the $K(\alpha)$ score value, as if the smaller candidate is used in several larger collocations it can be regarded as a unit of its own.

4 Experiments and Results

4.1 Collocational Candidates found

In this subsection extraction results are presented. In table 1 we are comparing the 15 highest ranked candidates for frequency and a cost criterion. The similarities between both result lists are obvious and support the statement that more complicated criteria cannot easily beat frequency.

In table 2 the top 15 candidates scored with MI are shown, whereas in table 3 MI scoring combined with a minimum frequency threshold. Contrasting tables 2 and 3 demonstrates how low frequency can ruin the results. In fact the prevalent frequency of 2 in table 2 might even be an artifact produced by the fact that the same abstract made it two times into MEDLINE.

However, closer examination of table 3 shows what this project is all about and into which direction we want to develop the automatic pattern extraction. The first three lines are covered by the phrase pattern

the n: of this study be to*

being a prototypical example of the patterns we ultimately want to find automatically for several reasons.

- The pattern has a fixed part which is large enough

MI	frequency	candidate
36,04	32	the:det aim:n of:prep this:det study:n be:v to:infp
35,28	14	the:det purpose:n of:prep this:det study:n be:v to:infp
34,48	12	the:det objective:n of:prep this:det study:n be:v to:infp
32,72	17	play:v an:det important:adj role:n in:prep
32,58	14	95:adj %:n confidence:n interval:n
31,75	33	aim:n of:prep this:det study:n be:v to:infp
30,94	14	purpose:n of:prep this:det study:n be:v to:infp
30,37	11	there:adv be:v no:det significant:adj difference:n
30,26	13	objective:n of:prep this:det study:n be:v to:infp
30,09	11	national:adj academy:n of:prep science:n
29,92	10	palladin:n institute:n of:prep biochemistry:n
29,44	13	of:prep this:det study:n be:v to:infp evaluate:v
28,80	33	the:det aim:n of:prep this:det study:n be:v
28,09	15	the:det purpose:n of:prep this:det study:n be:v
27,76	21	play:v an:det important:adj role:n

Table 3 Top 15 candidates with frequency ≥ 10 scored with MI

for the pattern to be very precise in matching, i.e. the chance that it will spuriously match in the wrong context is almost negligible.

- The pattern has a variable part (the wildcard) ”in the middle”, and thereby allows to recover this part of the sentence as a separate phrase without risk of getting the border of the phrase wrong.
- As a side effect, phrases found for the wildcard have a high probability to be synonyms of each other. The finding is one step in the direction of recovering semantic information automatically from text. The words *objective*, *aim* and *purpose* are indeed listed as synonyms in Merriam-Websters’s collegiate thesaurus[3].

The pattern *the n:* of this study be to* is a mixed collocation where the wildcard can be substituted by nouns like *objective*. However the place of the wildcard syntactically allows full noun phrases. If we allow the wildcard to model arbitrary gaps (interrupted collocations), we can even cover noun phrases. While we are not yet in the position to find such collocations fully automatically, the first results to this regard are reported below.

For interrupted and mixed collocations scoring is much more difficult, so we stick with frequency. Despite its simplicity, several authors report very good results using pure frequency scoring. [KE01] concludes, that for scoring PP-verb (prepositional phrase - verb) collocations, ”none of the association measures is significantly better than mere co-occurrence frequency”.

However, frequency was not able to recover *the *:n of this study be to* as a mixed collocation. Only after applying a heuristic along the lines of:

1. exactly one baseform-wildcard,
2. at least 3 realisations of the pattern in the corpus,
3. a hand chosen noun of interest (e.g. *study*)

the collocation was found early in the result list. Taking *disease* as the noun of interest, the pattern *the *:n of the disease*, realised with `diagnosis`, `pathophysiology` or `course`, can be found on rank 10 of the result list. Applying the pattern to texts, the word realising the wildcard can be used to categorize texts according to what (diagnosis, pathophysiology, course) is reported about a disease. Therefore the pattern is particularly useful for Information Extraction. Ranked first in this same result list is *of *:adj disease* realised with *basic* or *infectious*. It shows that typical simple syntactic constructs can be recovered as mixed collocations.

In table 4 the results for interrupted collocations are given. Here not only frequency but also a certain linguistic pattern is used: verb preposition. As an example of how the gaps can be filled, consider the phrases *report cases/details/sightings/incidents of*. The collocations listed are not as specific as the one described above simply because they contain only a few words, but they nevertheless represent patterns which help to create patterns for information extraction.

Manning [MS99], [Kre00a] remarks that the combination of a statistic technique combined with simple linguistic knowledge can produce interesting results.

[3] `http://www.m-w.com/cgi-bin/thesaurus`

frequency	candidate
12	v:consist prep:of
11	v:report prep:of
10	v:treat prep:by
7	v:use pep:of
7	v:perform prep:in
6	v:use prep:for
6	v:present prep:of
6	v:find prep:in
6	v:achieve prep:of
5	v:treat prep:with
5	v:treat prep:of
5	v:show prep:of
5	v:range prep:to
5	v:range prep:from
5	v:occur prep:in

Table 4 Results for interrupted collocations

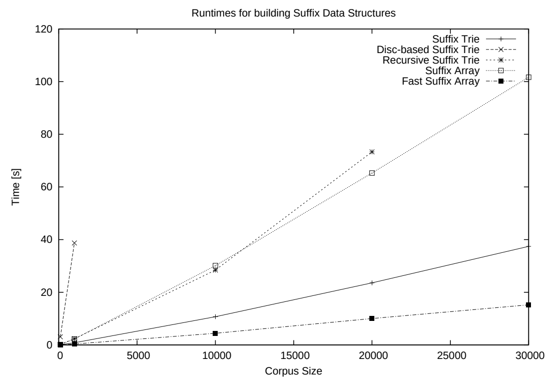

Figure 4 Runtime for constructing the necessary data structures

4.2 Performance of different algorithms

In this section the scalability of the different approaches is examined. Runtimes are measured depending on different variables. The goal of these performance tests is to evaluate differences in scalability between different approaches and implementations. All experiments have been executed on a 500-MHz Intel Celeron with 192 MB of main memory. As programming language Python [Lut96] was used. The problem sizes have been chosen, that no swapping was necessary. For timing experiments, the average of ten experiment runs have been used. As can be seen, for all problem sizes, that fit into main memory, the computational times are within an acceptable range. All experiments can be executed in a few minutes. The problematic bottle-neck seems to be memory consumption.

The following implementations are available:

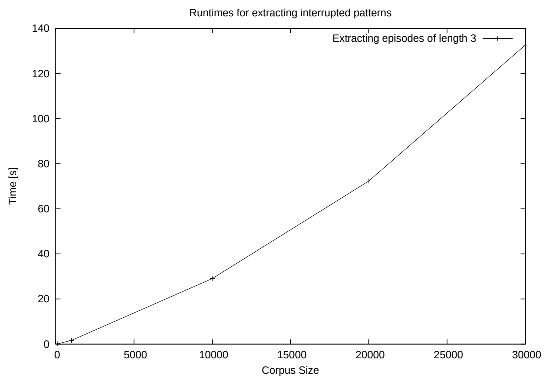

Figure 5 Runtime for extracting interrupted collocations of length three

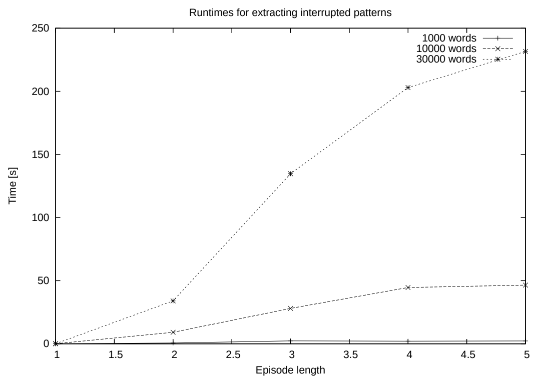

Figure 6 Runtime for extracting interrupted collocation from a certain number of words

Suffix Trie is an implementation using two hashtables, one for edges and one for nodes.

Recursive Suffix Trie uses a recursive definition of node objects which contain references to child nodes.

Disc-based Suffix Trie is an attempt to make usage of larger corpora possible. As an extension of the suffix trie, the hash tables containing edges and nodes, are stored in disc based dictionaries.

Suffix Array is a straightforward implementation of Nagao's suffix arrays. As suffix references, indexing into the corpus is used.

Fast Suffix Array builds the table of suffixes with the suffixes themselves, not using references. This takes more memory space, but allows for faster sorting, as no dereferencing has to be done for each comparing step.

The results when constructing the different data structures can be found in Figure 4. The favorites are the suffix trie implementation using hash tables and the fast suffix array. But we can use the suffix array for extraction of uninterrupted collocation candidates only, for interrupted candidates and candidates with mixed attributes, suffix tries are needed.

In Figures 5 and 6 runtime for extracting collocational candidates for interrupted collocations is shown. As this approach was newly developed, the runtime is of special interest. In Figure 5 episode length (in this case three) versus corpus size is shown where as in Figure 6 the opposite relation is depicted.

5 Conclusions and Future Work

In this paper we presented an approach of extracting collocations from biomedical syntactically annotated corpora. Unfortunately exact evaluation of extracted candidates in terms of recall and accuracy is not possible in this work. Manually crafted references have to be available to be compared with the automatically extracted suggestions. This task is extremely time consuming and requires extensive lexicographic knowledge and experience. So for evaluation, we have to restrict our demands to a more subjective review of results. Since even in the long run it can not be expected that any method will produce 100% viable pattern candidates, for practical applications of pattern finding a method which generates a decent number of useful pattern candidates fast and easy is sufficient.

Now that the tools to formulate and run patterns are in place, the focus of Biopath moves to pattern writing. Consequently, the result of the work reported here becomes more important. After mixed interrupted collocation candidates can be generated easily, different scoring schemes must be evaluated and heuristic methods need to be tested in order to produce collocations which can be immediately used as patterns. A lot of papers have been published on scoring, see e.g. [Kre00b], [EK01] and [KE01]. Finally the complete corpus with all its information should be taken into account.

Acknowledgements

The authors thank Dr. Mark Minas for his helpful comments concerning the data structures used.

References

[Ben89] Morton Benson. The structure of the collocational dictionary. *International Journal of Lexicography, Oxford*, 1989.

[CH89] Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th. Annual Meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, B.C., 1989. Association for Computational Linguistics.

[EK01] Stefan Evert and Brigitte Krenn. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, 2001.

[FA96] K.T. Frantzi and S. Ananiadou. Extracting nested collocations. In *Proceedings from the 16th International Conference on Computational Linguistics (COLING-96)*, pages 41–46, 1996.

[GI93] Roberto Grossi and G. Italiano. Suffix trees and their applications in string algorithms. In *Proc. 1st South American Workshop on String Processing (WSP 1993)*, pages 57 –76, 1993.

[ISSU96] Ikehara, Satoru, Shirai, and Uchino. A statistical method for extracting uninterrupted and interrupted collocations from very large corpora. In *16th International Conference on Computational Linguistics: COLING-96*, pages 574–579, 1996.

[KE01] Brigitte Krenn and Stefan Evert. Can we do better than frequency? a case study on extracting pp-verb collocations. In *Proceedings of the ACL Workshop on Collocations*, Toulouse, France, 2001.

[KKOY94] K. Kita, Y. Kato, T Omoto, and Y. Yano. A comparative study of automatic extraction of collocations from corpora: Mutual information vs. cost criteria. In *Journal of Natural Language Processing, Vol.1, No.1*, pages 21–33, 1994.

[Kre00a] Brigitte Krenn. Collocation mining: Exploiting corpora for collocation identification and representation. In *Proceedings of KONVENS 2000*, 2000.

[Kre00b] Brigitte Krenn. Empirical implications on lexical association measures. In *Proceedings of The Ninth EURALEX International Congress*, 2000.

[Lut96] M. Lutz. *Programming Python*. O'Reilly and Associates, 1996.

[McC76] E.M. McCreight. A space-economical suffix tree construction algorithm. 1976.

[MM93] Udi Manber and Gene Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. ComptitLq*, 22(5):935–948, 1993.

[MS99] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing, Collocations*. The MIT Press, Cambridge, Massachusetts, 1999.

[NM94] M. Nagao and S. Mori. A new method of n-gram statistics for large number of n and automatic extraction of words and phrases from large text data of japanese. In *COLING-94*, pages 611–615, 1994.