LLL'05 Challenge: Genic Interaction Extraction - Identification of Language Patterns Based on Alignment and Finite State Automata

Jörg Hakenberg HAKENBERG@INFORMATIK.HU-BERLIN.DE
Conrad Plake PLAKE@INFORMATIK.HU-BERLIN.DE
Ulf Leser LESER@INFORMATIK.HU-BERLIN.DE
Knowledge Management in Bioinformatics, Humboldt-Universität zu Berlin, Germany

Harald Kirsch KIRSCH@EBI.AC.UK
Dietrich Rebholz-Schuhmann REBHOLZ@EBI.AC.UK
Rebholz-Group, European Bioinformatics Institute, Hinxton, United Kingdom

Abstract

We present a system for the identification of syntax patterns describing interactions between genes and proteins in scientific text. The system uses sequence alignments applied to sentences annotated with interactions and syntactical information (part-of-speech), as well as finite state automata optimized with a genetic algorithm. Both methods identified syntactical patterns that are generalizations of textual representations of agent-target relations. We match the generated patterns against arbitrary text to extract interactions and their respective partners. Our best system uses finite state automata optimized with a genetic algorithm, and scored an F1-measure of 51.8% on the LLL'05 evaluation set.

1. Introduction

The task for the *Learning Language in Logic (LLL'05)* challenge was to build systems that extract interactions between genes and/or proteins from biological literature. From sentences annotated with agent-target relations and other linguistic information, rules or models had to be learned and were evaluated afterwards (Genic Interaction Extraction Challenge, 2005). For this benchmark, not only the interacting partners had to be extracted, but also the agent-target dependencies had to be resolved correctly.

The task of relation mining in the biomedical domain has been studied widely over the last years. Current research covers protein-protein interactions (Huang et al., 2004; Daraselia et al., 2004), subcellular locations (Stapley et al., 2002), disease-treatment-relations (Rosario & Hearst, 2004), and certain other types. Early systems relied on simple co-occurrence analyses, and such techniques still provide very good recall performance for obvious reasons. Including syntactical information for shallow parsing leads to better results in the identification of co-occurring terms in conjunction with a verbal phrase (Chen & Sharp, 2004), *i.e.*, yields more precise predictions. Currently, systems based on sequence modeling, and pattern- or rule-based extraction provide the best results for detecting protein-protein interactions (Xiao et al., 2005; Huang et al., 2004; Saric et al., 2005).

The relation extraction system we used for the LLL'05 challenge consisted of two major components. The first extracted *syntax patterns* typical for textual descriptions of interactions from labeled examples. These patterns reside on part-of-speech information and markup of genes and selected nouns and verbs. We followed two strategies to learn such patterns from a given training sample. One was to apply a pattern generating algorithm comparable to the one described in Huang et al. (2004) to annotated examples. Subsequent *pairwise alignment* of sentences ultimately yielded patterns with as much support in the training data as possible. The other strategy was to generate *finite state automata* representing patterns, and to optimize these automata on the training sample using a *genetic algorithm* to achieve optimal results.

Appearing in *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

After patterns had been extracted, the second component matched these against arbitrary text to detect interactions.Matching was either based on aligning new sentences with patterns, or on mapping new sentences into FSAs, respectively.

All patterns we extracted from the given examples resembled sentences often used for describing interactions. Typical examples often found in the literature are, for instance,

ykuD is transcribed by SigK RNA polymerase (1)
yfhS is transcribed by E sigma E (2)
ComK regulates the expression of degR (3)
GerE inhibits the transcription of sigK (4)

It can easily be seen that (1) & (2), and (3) & (4) share a similar syntax, respectively. Replacing words with their respective part-of-speech tags, phrases (1) and (2) could be subsumed with the pattern

gene verb:p3s verb:pp preposition protein

This pattern describes every (partial) sentence that contains a gene, followed by a verb (present, third person singular), another verb (past participle), a preposition, and finally a protein[1]. The second verb could be narrowed down even further, by either accepting only verbs seen in the training data, or by compiling a list of possible verbs used for describing interactions. We followed the second idea, and had lists for verbs and nouns, which we refer to as *interaction verbs* and - *nouns* or simply *types* in the following (see Section 2.4.1 and supplementary information).

In this paper, we shall give an overview of all methods and algorithms used, present the preprocessing of the data set, and conclude with a discussion of our results and findings.

2. Methods and Algorithms

There were two possibilities to extract and represent patterns from a set of labeled examples. The first generated a set of patterns using pairwise alignments of sentences from the training sample. These alignments were transformed into a pattern. The second method was learning finite state automata from the training sample.

For both approaches, we represented each sentence as a sequence of POS tags instead of tokens. Additionally, we did not take the full sentences, but only the immediate phrases relevant to the description of the interaction. This included a certain boundary around the phrases (we experimented with up to three words to the left and right).

[1]In the following, we do not distinguish between genes and proteins.

Table 1. POS tags and costs for matches/mismatches. Costs shown: Gap - aligning POS tag with gap; Match - exact match; Group - match within group (same first letter); Other - mismatch.

POS Tag	Gap	Match	Group	Other
PTN	-10	+4		-3
IVERB	-10	+3		-3
INOUN	-8	+3		-3
NN/NNP	-8	+2	+1	-1
NNS/NNPS	-7	+2	+1	-1
VB/VBP/VBZ/ VBD/VBN/VBG	-7	+2	+1	-1
IN, CC, TO	-6	+2		-1
'(', ')', ','	-6	+2		-1
RB, RBR, RBS	-1	+2	+1	-1
JJ, JJR, JJS	-1	+2	+1	-1
DT, WDT	-1	+2		-1

2.1. Pairwise Alignment to Generate Patterns

2.1.1. Alignments

Quite a few methods are available to measure the similarity of two (or multiple) sentences. Our method was sequence alignment, which calculates a consensus sequence in addition to the similarity score. This consensus sequence represented all parts (POS tags and their positions) the aligned sequences had in common, and could be used directly to form a pattern. Figure 1 shows an example for two aligned sentences. For the alignment and scoring of sentence pairs, we implemented the local and end-space free alignments (Smith & Waterman, 1981; Needleman & Wunsch, 1970). With local alignment, the aligned sentences could be quite dissimilar overall, but had to contain regions that were highly similar (*e.g.*, a verb phrase or main clause). All other parts could have completely different syntax and semantics. Using the end-space free variation of global alignment, gaps at the beginning or end of a sentence had costs of zero, regardless of length. For calculating the costs for matches, mismatches, and gaps in the alignment, we used a substitution matrix covering the whole alphabet (19 part-of-speech tags). These costs were derived from previous experiments, and are are shown in Table 1. We found that while the exact values were less influential, the overall tendency of rewards and penalties for different sorts of replacements was important. A substitution matrix for an extended alphabet (full Penn Treebank tag-set) can be found in the supplementary information.

```
the oxidized cytochrome c binds    phosvitin and   ->  DT JJ PTN IVERB -- PTN CC
the          profilin    binds to  G-actin   ,     ->  DT -- PTN IVERB IN PTN ,
                                                       --------------------------
consensus sequence of both (partial) sentences     -   DT    PTN IVERB    PTN
```

Figure 1. Example for an alignment and consensus sequence of two POS tags sequences. Sentences are represented using POS tags. CC, conjunction; DT, determiner; IN: preposition; JJ: adjective; PTN, protein/gene; IVERB: verb describing an interaction.

2.1.2. Patterns

Each pattern consisted of two types of information: the pattern itself, *i.e* the sequence of POS tags, and the positions of agent, target, and interaction type in this pattern. A simple example for a pattern would be

$$\begin{aligned} \text{pattern} &= \text{PTN IVERB PTN} \\ \text{relation} &= \{(\text{a}=1, \text{t}=3, \text{i}=2)\}, \end{aligned}$$

matching sentences with the basic structure "protein interaction-verb protein", for instance, "ComK regulates degR". The agent-target dependency is given via the tuple (a,t,i), where the first two digits refer to the positions of agent and target, and the third digit to the position of the interaction type, respectively. Please note that we did not distinguish between genes and proteins, but mapped both entities to the same tag, 'PTN'.

An example is shown in Figure 2. Each pattern p consisted of a POS sequence s, and a (set of) relations, K. Each element of K depicted a single interaction, and the triple (a, t, i) referred to the positions of agent, target, and interaction type within the sequence, respectively. Note that for the positions, we took into account only proteins and interaction nouns or verbs. $(1, 3, 2)$ thus stood for a relation between the agent protein at position 1 and the target protein at position 3, described by the verb at position 2.

2.1.3. Generating Patterns with Alignments

The pattern generating algorithm iterated over all sentence pairs and calculated the best alignment for each pair (see Section 2.1.1). Each respective consensus sequence from the optimal alignment of these two sentences formed a pattern. We counted the occurrences of all such patterns to calculate the support for each pattern in the training data. The maximum number of patterns for n sentences is $n(n-1)/2$, but in practice not all were generated, since a set of different alignments can lead to the same consensus sequence. On the other hand, in case there were multiple optimal alignments, all alignments were taken to form (different and/or identical) patterns. The algorithm in Figure 3 shows the pattern generating algorithm as pseudo-code.

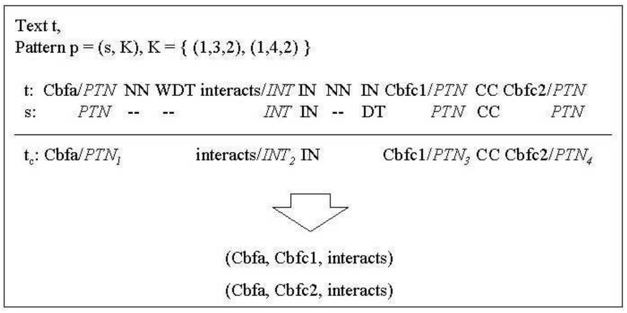

Text t,
Pattern p = (s, K), K = { (1,3,2), (1,4,2) }
t: Cbfa/PTN NN WDT interacts/INT IN NN IN Cbfc1/PTN CC Cbfc2/PTN
s: PTN -- -- INT IN -- DT PTN CC PTN
t_c: Cbfa/PTN_1 interacts/INT_2 IN Cbfc1/PTN_3 CC Cbfc2/PTN_4
(Cbfa, Cbfc1, interacts)
(Cbfa, Cbfc2, interacts)

Figure 2. Text t aligned with pattern p. Consensus sequence is t_c, forming two interactions: agent$_1$ with target$_3$ via type$_2$, and agent$_1$ with target$_4$ via verb type$_2$, respectively.

Our pattern generating algorithm iteratively searched for the most similar sentences, and tried to subsume these with a more general expression, if necessary. As we took alignments for this step, we used the respective consensus sequences for the generalized sequences. This general expression, which we call pattern, was then added to a set. This set functioned as a model to explain the training data. Only patterns with at least two proteins and one interactor (specific verb or noun) were included in the final set. From the final set, all patterns below a minimum support (*i.e.* the number of aligned sentence pairs producing this pattern) were removed.

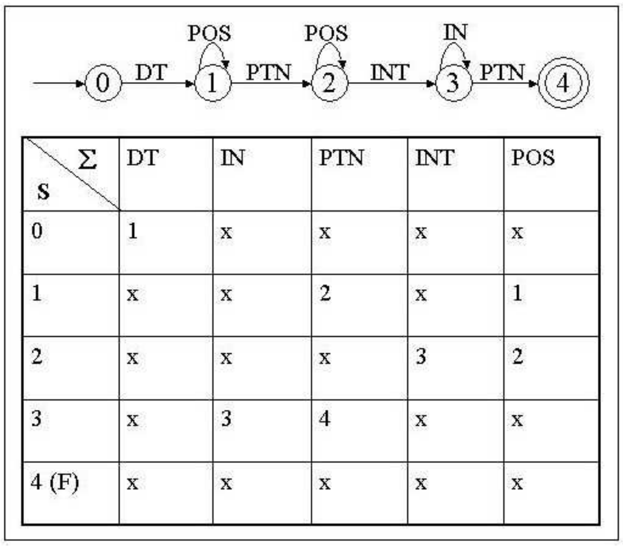
2.1.4. Applying Patterns to Arbitrary Text

We studied different methods to apply the generated patterns to arbitrary text: alignments, finite state automata, hidden Markov modeling, and hand-crafted rules (Plake et al., 2005a; Plake et al., 2005b). For the LLL'05 challenge, we used local and end-space-free alignment.

The alignment worked just like described in Section 2.1.1. Figure 2 shows an example for a pattern, its information, its alignment with a text and the resulting interactions. The positions for agent, target, and interaction type (the latter was not necessary for the

```
Input: set of annotated sequences, $S$;
       threshold $d$
  for all $s_i, s_j \in S$, $i \neq j$ do
    $c_{i,j}$ = consensus($s_i, s_j$)
    $K = (pos_a, pos_b, pos_i)$ = positions of
      partners $a, b$, and interactor $i$ in $c_{i,j}$
    $p = (c_{i,j}, K)$
    if $p \notin P$ then
      add $p$ to $P$; $occ_p = 1$
    else
      $occ_p$++
    end if
  end for
  remove all $p \in P$ with $occ_p < d$
Output: set of patterns, $P$
```

Figure 3. Pseudocode of the pattern generating algorithm.



S \ Σ	DT	IN	PTN	INT	POS
0	1	x	x	x	x
1	x	x	2	x	1
2	x	x	x	3	2
3	x	3	4	x	x
4 (F)	x	x	x	x	x

Figure 4. Mealy-FSA and its transition matrix; s: number of states, Σ: alphabet of POS tags. For example, from state 1 a transition is possible either to state 2 using 'PTN', or back to state 1 using any other, non-listed POS tag.

LLL'05) in the new sentence were deduced from the information stored with each pattern. This was needed to solve the dependency between two proteins and for cases with multiple proteins in one sentence.

2.2. Learning Patterns with Finite State Automata

For the second approach to generate and represent patterns describing protein-protein interactions, we used *Mealy finite state automata* (FSAs). In these automata, each position in a sentence (*i.e.* part-of-speech tag) was represented by a transition from one state to another. Transitions thus modeled the sequence of POS tags in a sentence. The FSA has a dedicated *start state*, depicting the position before the first tag in a phrase. We stored each transition possible from each state to another in a matrix. For each state in the FSA, such a matrix contained all transitions possible to a new state using the next POS tag in the sequence. Figure 4 shows an example, where columns represent the alphabet (=POS tags), and rows the states, respectively. Each cell denoted whether a transition from the current using a particular POS tag was possible or not, and to which new state this transition would lead.

It was possible to further generalize FSAs. By introducing word gaps of variable length between states (*i.e.* between positions in a sentence), the FSA contained not only sentences fitting phrases exactly, but allowed for insertions (for example, short subordinate clauses or expressions in brackets). See Figure 5 for examples. Intuitively, a more "strict" FSA (no word gaps) would yield more precise results, while a "loose" FSA (word gaps of infinite length) would gain a higher recall, by fitting more sentences.

We encoded the transition matrix described above in a single array. The value in each cell was transformed into a binary representation, depicting the index of a new state, when the transition using a particular POS tag was possible, or zero otherwise. In addition, we added a bit to this cell value. Whenever this bit was set to one, the transition led not only to a next state, but in addition, this state was an end-state. Table 2 shows an example for the binary representation of state 3 from Figure 4. In addition, the positions for agent(s), target(s), and interaction type(s) in the sentences had to be encoded. Taken together, this led to a binary representation of FSAs, which we refer to as a *genome*. In such a genome, all positions for the states, and all positions for transitions and end-states were fixed. Translations from a matrix to a genome and back thus were unambiguous.

The task now was to find one or more good FSAs encoded by such genomes. We first started with a set of random genomes, representing a *population*, where each individual encoded an FSA with six states[2]. Initially, most of the individual members of a population would encode a valid, but certainly a "bad performing" matrix. We optimized this population on the training sample using a genetic algorithm, see (Plake et al., 2005b). For this approach, we could use either preci-

[2]The number of states was evaluated in further experiments; data not shown.

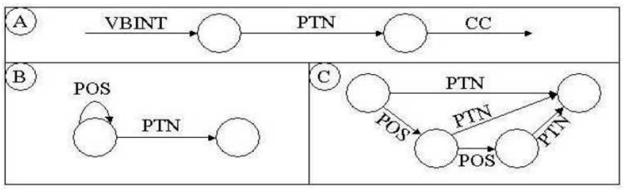


Figure 5. FSAs with word gaps. (A) shows the original transitions between two states, with a protein tag in the middle. In (B), the FSA was extended with an additional transition from the left state, using any POS tag, to the same state. This transition could be used multiple times. (C) shows an extension for a maximum of two, instead of multiple, optional POS tags.

Table 2. Binary representation of state 3 from the matrix in Figure 4. The table shows all possible transitions, and to which new state each transition leads. Using 'PTN', the transition from 3 to 4 would lead to an end-state.

POS tag	new state	end-state?	possible transition
DT	000	0	"no transition possible"
IN	011	0	"possible to state 3"
PTN	100	1	"to state 4 ⇒ end-state"
INT	000	0	"no transition possible"
POS	000	0	"no transition possible"

sion, recall, or f-measure as a *fitness function* to measure the performance of every individual FSA in the population. Choosing the best 25% of a population to form the basis for a new generation, we filled the missing 75% using *recombinations*, *cross-overs*, and *mutations* of this *parents*. Ultimately, this led to an FSA (the "best" genome in the population) covering at least some positive examples[3] from the training sample. As soon as no further improvements could be achieved, we removed all training sentences covered by this FSA. On the remaining sample, we started over with a completely new random population. We followed this *separate-and-conquer strategy*, until no positive sentences were left in the training sample, or a certain number of FSAs was found.

2.2.1. Applying Finite State Automata

In order to match interactions and to learn patterns based on FSAs, we had to foresee in our patterns that they match variability in the text. This meant that starting at the first tag in each phrase, a sequence of states including repetition of transitions had to be foreseen in our FSAs, ultimately leading to an end-state with the last tag. To parse complete sentences, we tested all sub-phrases starting at the first word, the second word, the third word, and so on. If the language of an FSA *covered* a phrase, then index positions similar to the ones described in 2.1.1 defined the interaction's partners and type.

2.3. Data Sets

For generating patterns, we used two corpora in two separate runs. All contained sentences found in MedLine citations, together with markup for all gene/protein names, all interactions and their respective types (*i.e.* agent-target relations). The first corpus was the training data set provided with the LLL'05 challenge, consisting of 55 sentences with 103 genic interactions. No negative examples were included in this corpus. The other corpus had annotated protein-protein interactions, and consisted of 1000 sentences with 256 interactions in 174 sentences. This second corpus was derived from the BioCreAtIvE task 1A data set (Hirschman et al., 2005), in which gene and protein names were marked. Further annotations for protein-protein interactions based on this markup were added manually.

For the evaluation of strategies and methods, we generated patterns from the 1000 sentences, and applied them to the 55 sentences, on which we could measure the performance. The final test set consisted of 86 sentences containing genic interactions, and was provided with the LLL'05 challenge.

We tested different combinations of data sets for training and testing, and different methods for applying patterns to the respective test set. For the final solution, we learned patterns from our own corpus of 1000 sentences and combined them with patterns learned from the 55 training sentences.

2.4. Preprocessing

In order to apply our system to the training and test data, we had to perform several pre-processing steps. First of all, we transformed the data into XML format, including a proper tokenization. The indices of words in the word-list provided differed slightly from the corresponding token's index. Our tokenization included punctuation marks, brackets, and hyphens. We split expressions like *'sigma(K)-dependent'* into three tokens, *'sigma(K) - dependent'*.

2.4.1. Named Entity Recognition

A full list of all gene and protein names and synonyms or spelling variants appearing in the corpus was provided with the data. Named entity recognition could

[3]A positive example contains at least one interaction.

Table 3. Slight modifications of the dictionaries and data sets.

- removed blanks	E sigma 27 (29, 43); E sigma A (D,E,F,G,H); sigma 29 (32, 70)
- removed symbols	sigma-43; Spo0A-P, Spo0A˜P; SpoIIAA-P; alpha-amylase; PBP4*; PhoP˜P
- removed symbols & blanks	pro-sigma E (K)
- added to dictionary	lacZ (ID 8169223-5,10767540-2); orf10 (10481082-2)
- altered in dictionary	sigma 27 = sigK according to 2492118-2
- added '.' at end of sentence	ID 10400595-1

thus be reduced to an exact matching of sentences against the dictionary. However, we introduced some minor modifications to cope with blanks, symbols, and overlapping gene names (*'sigma E'* versus *'E sigma E'* and *'pro-sigma E'*). We normally tackle the NER task using machine learning approaches based on support vector machines (Hakenberg et al., 2005). We did not distinguish between genes and proteins, neither in NER nor in patterns. In the literature, names referring to genes or proteins often are quite similar or even used as synonyms, and the exact meaning most times is hard to resolve.

Our pattern composition is based on *part-of-speech tags* annotated for each token. For POS-tagging, we used the TnT-Tagger, trained on the Wall-Street-Journal corpus (Brants, 2000), generating tags from the Penn Treebank tag-set (Santorini, 1990). In addition, we needed markup for tokens describing interactions, referred to as either *interaction nouns* or *verbs* (such as *'activation'*, or *'inhibits'*). We performed this step by combining POS-tag and word stem (Porter, 1980) to find an entry in fixed term-lists for nouns and verbs. The tokens in these lists were selected by experience gathered in previous studies, and additionally included suggestions from Temkin and Gilder (2003) as well (see supplementary information). We restricted the tags used in this approach to the ones shown in Table 1.

2.4.2. Dictionary and Canonical Forms

For the LLL data sets, we transformed every protein/gene name to its respective canonical form, as provided with the data. We subsequently matched every spelling variant of protein/gene names occurring in the corpus to its corresponding canonical form, starting with the longest synonyms to avoid errors in overlapping names. In addition, we altered some entries in the dictionary to deal with these problems. In general, in the refined dictionary, canonical forms did not have blanks or symbols (see Table 3). We added two names (lacZ and orf10), because they occurred in the corpus, but not in the dictionary.

Table 4. Patterns extracted from our corpus; table shows only patterns with a support of ≥30. Dependencies refer to different possibilities for agent/target relations; A: first protein in the sentences, B: second, C:third.

Pattern	Support	Dependencies
PTN IVERB PTN	1405	A→B, B→A, A↔B
PTN IVERB DT PTN	258	A→B, A↔B
PTN IVERB IN PTN	173	A→B, B→A
PTN INOUN PTN	138	A→B, B→A
PTN INOUN IN PTN	116	A→B, B→A
PTN IVERB IN DT PTN	46	A→B
PTN RB IVERB PTN	45	A→B, A↔B
INOUN IN PTN IN PTN	35	A→B, B→A
PTN IVERB NN PTN	35	A→B
PTN IVERB PTN PTN	30	A→B, A→C

Table 5. Performance on different types of interactions; all without linguistic information, without co-refs. Numbers in brackets give the total number of existing interactions (FN), and predicted interactions (FP), respectively.

	action	bind	regulon	nothing	all
TP	19	7	2	0	28
FN	17 (36)	5 (12)	2 (4)	0 (0)	24 (52)
FP	10 (29)	4 (11)	1 (3)	13 (13)	28 (56)

3. Results and Discussion

From our corpus of 1000 sentences, we were able to extract 148 patterns (see Table 4). The pattern with the highest support was "PTN IVERB PTN" – 1405 different alignments produced this sequence. The second best pattern, "PTN IVERB DT PTN" had a support of 258 only. These numbers included multiple optimal alignments for pairing two sentences. Table 4 shows the ten patterns with the highest support in the training data.

We decided to send the prediction (supposedly) having the highest F1-measure. This solution scored an F1 of 51.8% (precision of 50.0% at 53.8% recall; see Tables 5 and 6).

For the challenge, we did not use the co-references and linguistic information as provided with the corpus. Error analysis (for predictions on the training data) revealed that our system often predicted the in-

Table 6. Results from the evaluation for different methods and corpora. Upper corpus for training, lower for test; C_{1000}: corpus of 1000 sentences; C_{55}: genic_interaction_data; C_{86}: basic_test_data.

Method	Corpus	Pre	Rec	F1
Alignment	C_{1000} C_{55}	57.1	7.8	13.7
Alignment	C_{55} C_{86}	63.6	8.1	14.4
Mealy (2 FSAs)	C_{1000} C_{55}	64.3	17.5	27.5
Mealy (2 FSAs)	C_{1000} C_{86}	42.9	9.2	15.2
Mealy (4 FSAs)	C_{55} C_{86}	28.1	31.4	29.6
Mealy (5 FSAs)	$C_{1000+55}$ C_{86}	**50.0**	**53.8**	**51.8**

teracting partners correctly, but erred on the direction of the interaction, *i.e.* it interchanges agent and target. Using linguistic information, it might be possible to resolve some of these dependencies. Syntactic relations could help to generate more specific patterns containing whole phrases instead of single tags ("expression of rsfA"). For this specific phrases, certain positions could even be restricted to particular tokens, instead of general POS tags. Combinations of verbs and prepositions, for instance, contain linguistic information clearly stating the exact role of a gene/protein appearing before or after this verb phrase ("was phosphorylated by"). Grouping particular interaction types together (*e.g.*, "phosphorylation" and "methylation" refer to the creation of bonds, while "reduction" and "repression" imply an inactivation) and restricting patterns to these particular types might further help to exploit nomenclature usages.

We saw that the performance of our approach is clearly dependent on the size of the training set. Our system was able to detect only interactions for which it encountered a quite similar example in the training data. Using only 55 sentences for generating patterns proved insufficient, as many relations in the test set did not resemble any of these.

Our training corpus of 1000 sentences contained 256 different protein-protein interactions, but these were in general quite dissimilar from the genic interactions in the LLL data. This difference clearly had an impact on the performance. For instance, our examples did not include any descriptions of regulon family memberships at all. Most of our examples describe actions and bindings, and our system performed better in these categories (see Table 5).

Statistical analyses of the corpora revealed that the pattern protein-interactor-protein (PIP) was used in 78.1% of all cases, and IPP accounted for 18%, leaving 3.9% for PPI. In 58% of the relations, the agent was mentioned before the target.

The results of the system presented here were not overly satisfying. On our own corpus, alignments scored a precision of 91% or a recall of 65%, and Mealy-FSAs yielded 79% precision or 88% recall. These results could be confirmed on the IEPA corpus (Ding et al., 2002), which contains protein-protein interactions as well.

Supplementary Information

For supplementary information, please visit `http://www.informatik.hu-berlin.de/~hakenber/publ/suppl/`.

Abbreviations

FSA, finite state automaton; NER, named entity recognition; PGA, pattern generating algorithm; POS, part-of-speech (tag).

Acknowledgments

This work is supported by the German Federal Ministry of Education and Research (BMBF) under grant contract 0312705B. The Knowledge Management in Bioinformatics Group is a member of the Berlin Center for Genome Based Bioinformatics (BCB). Funding for the Rebholz group is provided by the Network of Excellence "Semantic Interoperability and Data Mining in Biomedicine" (NoE 507505). JH is additionally supported by the German Foreign Exchange Service (DAAD), reference number D/05/26768.

References

Brants, T. (2000). TnT - a statistical part-of-speech tagger. *Proc 6th Applied NLP Conference*. Seattle, USA.

Chen, H., & Sharp, B. M. (2004). Content-rich biological network constructed by mining PubMed abstracts. *BMC Bioinformatics*, *5*, 147.

Daraselia, N., Yuryev, A., Egorov, S., Novichkova, S., Nikitin, A., & Mazo, I. (2004). Extracting human protein interactions from medline using a full-sentence parser. *Bioinformatics*, *20*, 604–611.

Ding, J., Berleant, D., Nettleton, D., & Wurtele, E. (2002). Mining MEDLINE: Abstracts, Sentences, or Phrases? *Pacific Symposium on Biocomputing* (pp. 326–337). Kaua'i, Hawaii, USA.

Genic Interaction Extraction Challenge (2005). Learning Language in Logic Workshop (LLL). http://genome.jouy.inra.fr/texte/LLLchallenge/.

Hakenberg, J., Bickel, S., Plake, C., Brefeld, U., Zahn, H., Faulstich, L., Leser, U., & Scheffer, T. (2005). Systematic Feature Evaluation for Gene Name Recognition. *BMC Bioinformatics*, *6*, S9.

Hirschman, L., Yeh, A., Blaschke, C., & Valencia, A. (2005). Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, *6*, 1.

Huang, M., Zhu, X., Hao, Y., Payan, D. G., Qu, K., & Li, M. (2004). Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, *20*, 3604–3612.

Needleman, S., & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, *48*, 443–53.

Plake, C., Hakenberg, J., & Leser, U. (2005a). Learning Patterns for Information Extraction from Free Text. *Proc Workshop des Arbeitskreises Knowledge Discovery*. Karlruhe, Germany.

Plake, C., Hakenberg, J., & Leser, U. (2005b). Optimizing Syntax Patterns for Discovering Protein-Protein Interactions. *Proc ACM Symposium for Applied Computing, Bioinformatics track*. Santa Fe, USA.

Porter, M. (1980). An algorithm for suffix stripping. *Program*, 130–137.

Rosario, B., & Hearst, M. (2004). Classifying semantic relations in bioscience texts. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, ACL*. Barcelona, Spain.

Santorini, B. (1990). *Part-of-speech tagging guidelines for the Penn Treebank Project* (Technical Report). MS-CIS-90-47, University of Pennsylvania.

Saric, J., Jensen, L., Ouzounova, R., Rojas, I., & Bork, P. (2005). Large-scale Extraction of Protein/Gene Relations for Model Organisms. *Proc Symp on Semantic Mining in Biomedicine* (p. 50). Hinxton, UK.

Smith, T., & Waterman, M. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, *147*, 195–197.

Stapley, B., Kelley, L., & Sternberg, M. (2002). Predicting the sub-cellular location of proteins from text using support vector machines. *Proceedings of the Pacific Symposium on Biocomputing* (pp. 374–385).

Temkin, J. M., & Gilder, M. R. (2003). Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, *19*, 2046–2053.

Xiao, J., Su, J., Zhou, G., & Tan, C. (2005). Protein-Protein Interaction Extraction: A Supervised Learning Approach. *Proc Symp on Semantic Mining in Biomedicine* (pp. 51–59). Hinxton, UK.