

Genic Interaction Extraction from MEDLINE Abstracts — A Case Study

Raluca Uricaru, Liviu Ciortuz*
raluca_uricaru@yahoo.com, ciortuz@info.uaic.ro
Faculty of Computer Science, University of Iasi, Romania

Abstract

This paper describes the system that we designed for solving the “Learning Language in Logic” Challenge task 2005 (LLL’05) concerning the extraction of directed genic interactions from sentences in MEDLINE abstracts. We see this task as a classification problem: the system must separate the interacting pairs of genes/proteins from the non-interacting ones, and it achieves this by learning a model from the training data given in the form of positive and negative examples. Each example of gene/protein interactions found in the given MEDLINE abstracts is described linguistically by a set (chain) of syntactic relations, which constitute (most part of) the annotations accompanying the abstracts.

The key task in the present work was the selection of those features that best describe the training and test examples. Such feature include: the ‘root’ and the ‘head’ word of the syntactic chain/tree associated to the example, the (number of) positive and negative examples “close to” a certain instance according to a conveniently defined distance measure, etc. To learn the classification model from the training data, we used different classifiers implemented in the WEKA library (Waikato Environment for Knowledge Analysis): the naive Bayes classifier, Bayes belief networks, radial basis function networks, support vector machines (SVMs) and ADA Boost.

Our best result was 58.7% F-measure on the entire dataset, and it was obtained using the naive Bayes classifier. This result proved to be second best compared to the results reported by the participants at the *LLL’05 Challenge* task.

1 Introduction

Large databases, storing a good proportion of the latest research results in biology and biomedicine, represent a real source of information for biologists. MEDLINE alone, a comprehensive literature database of life sciences and biomedical information, stores more than 11 million summaries of articles, representing an incredible wealth of biological information.¹

*This paper was published in the Scientific Annals of “Al.I.Cuza” University of Iași, the Computer Science series, 2005, pag. 137-152.

¹MEDLINE can be browsed at the Internet address <http://medline.cos.com/>.

Event-based information extraction from such a scientific literature database is still a challenging task, and there is a lack of efficient and accurate systems which could retrieve and process this large amount of information, despite the effort of a number of teams worldwide. This situation is explained in principal by the absence of formal structure in the natural-language narrative in these documents.

Extracting gene interaction is the most popular information extraction task in biology. Gene interactions are generally mentioned in the paper abstract, therefore the full text of the paper is not needed for information extraction. Usually in gene interaction the agent is distinguished from the target of the interaction. The relevant information is mostly local to single sentences [4]. The main exception is due to coreferences, in which case the gene/protein name is mentioned in a sentence and referred to in the form of a pronoun or an hyperonym in the next sentence.

The *LLL'05 Challenge* task focused on extracting gene interactions in *Bacillus subtilis*. This paper describes the system that we created to solve this task. Unfortunately we did not participate at the competition, because we finished the system shortly after the LLL'05 deadline (which was on 27 April 2005). However, the results we obtained are highly competitive with those obtained by the participants. This is why we considered useful to documentation (and eventual publication) of our approach.

The organization of the paper is the following: the remaining part of this first section describes the datasets delivered by the LLL'05 organizers; Section 2 details the proposed solution, focusing on the way we chose the features needed for classification. Section 3 presents the results we obtained using some the WEKA library of machine learning (ML) classifiers, and suggests possible possible ways of continuation for the present work. The last section draws the conclusions.

1.1 Description of the Data

The abstracts extracted from the MEDLINE database have been segmented into sentences. Sentences have been filtered by the *LLL'05 Challenge* organizers in order to retain those that contain at least two gene/protein names and are most probable to denote interactions. MIG-INRA expert biologists have annotated with the XML editor CADIXE hundreds of the interactions and their experimental conditions. For this challenge task, a subset of them was provided as training and test data. The protein/gene names that can play the roles of agent or target of gene interactions in the datasets were recorded in a named-entity dictionary as canonical forms and variants, and the dictionary was provided to the participants. The gene/protein interactions are expressed by means of explicit facts like the binding of a protein on the promoter site of the target gene, or the membership of a protein to a certain regulon family.

The training dataset is composed of two subsets of different levels of difficulty. The first subset does not include coreferences neither ellipsis, as the second subset does. The two subsets are accompanied by two kinds of annotations: firstly the sentence and word segmentation and the biological agent-target information, and secondly lemmas (the normalized form of words) and syntactic dependencies (the morpho-syntactic function between two connected words), which were obtained using Link Parser [3] and then checked by hand.

The participants to the challenge were free to use or not the linguistic information. They

were allowed to apply their own linguistic tools. Our system was trained using the enriched (i.e. fully annotated) dataset, on both sets, with and without coreferences.

1.2 Overview of Results Submitted by the Participant Teams

In order to give the reader a right perspective on our results, we present here the results obtained by the groups that submitted papers at the *LLL'05 Challenge*. The information has been extracted from an article written by one of the organizers [2] found on the official site of the ICML05 Workshop.²

The first team, from KMB, University of Berlin and EBI, has applied alignment and finite-state automata technology for generating IE patterns from the LLL'05 dataset [5]. The second group, from the CS department, University of Sheffield, generated candidate patterns from examples parsed by MiniPar and semantically tagged by WordNet and PASBio [8]. The group from HCS Laboratory, University of Amsterdam, has applied the rule induction method Ripper to lexical-semantic-syntactic subtrees obtained by unification of the enriched form of the training examples [9]. The fourth team, from KDLab, University of Brno, has applied the inductive logic programming (ILP) system Aleph on the enriched dataset without coreferences [6]. The group from Biostats and the CS department of the University of Madison used also Aleph on the enriched dataset with and without coreferences wrapped into the ILP system Gleaner [7].

The best results were submitted by the team from ICCS, University of Edinburgh. They applied ILP and Markov Logic methods on the data parsed by the CCG and CCG2sem parsers that build syntactic and semantic paths [10]. The team achieved 52.6% F-Measure on the dataset without coreferences and 54.3% on the dataset with coreferences. After adding explicit clauses which model non- interaction they were able to improve these numbers to 68.4% and 64.7%, respectively.

We achieved 58.7% F-Measure (using both test sets), making our system highly competitive with those presented by the participants at the *LLL'05 Challenge*. Details about our approach will be provided in Section 3.

2 The Proposed Solution

2.1 Extracting Positive and Negative Examples

Positive training examples are simply the interactions provided in the training data. Negative examples are non-interactions: pairs of genes which are not declared as interacting ones in the training data. The test instances are may-be interactions, pairs of genes extracted from sentences in the test files, about which the system has to decide whether they interact or not.

For every training interaction the learning system creates a directed graph of syntactic relations which we will call the *interaction graph*. As it can be seen in the example illustrated by Figure 1, in an interaction graph the nodes are words in the analyzed sentence while an edge between two nodes is represented by the syntactic relation (if any) between the two

²<http://www.cs.york.ac.uk/aig/III/III05/>

words. When the orientation of the syntactic relations is not taken into consideration, we can talk about *syntactic relation chains*.

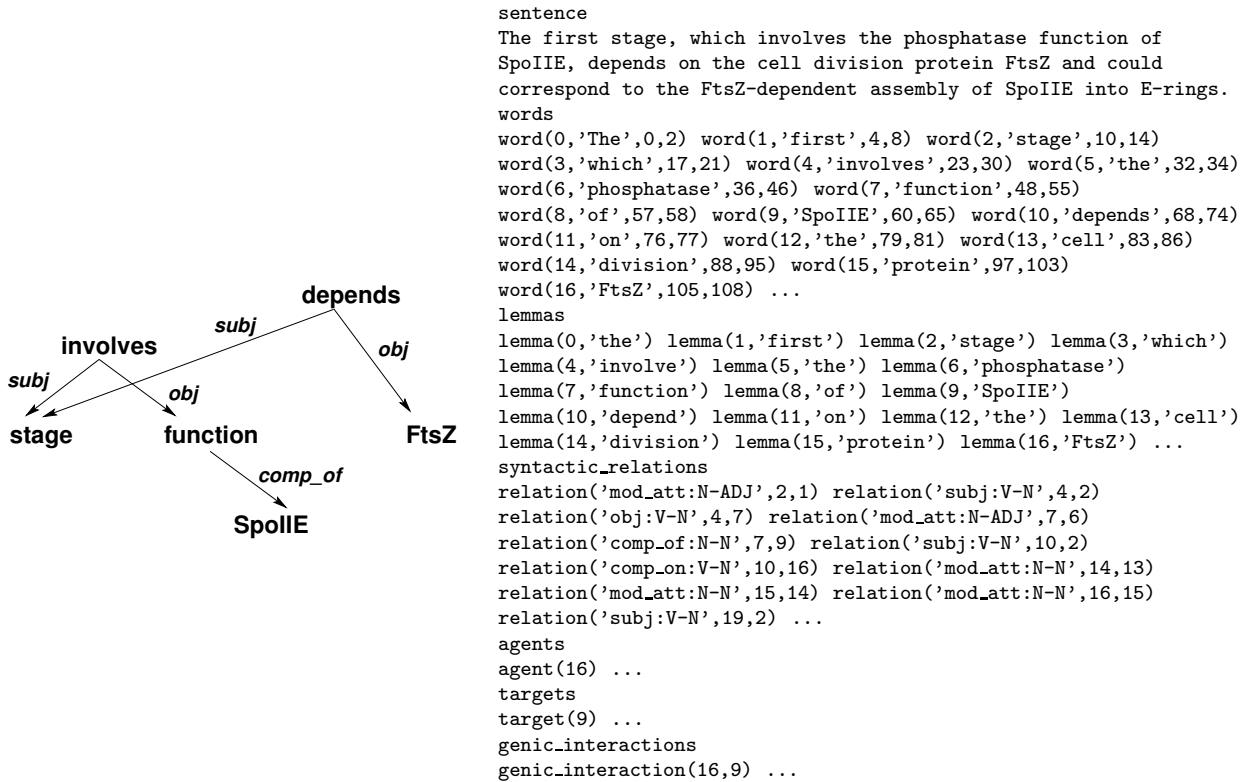


Figure 1: A sample multi-rooted genetic interaction graph. It was extracted from the annotations provided to the sentence *The first stage which involves the phosphatase function of SpoIIE, depends on the cell division protein FtsZ.*

For every test pair of genes we will also try to obtain an interaction graph, i.e. to find the syntactic relation chain(s) that connect the two genes in the given sentence, so to be able afterward to decide if the two genes interact or not. To get the interaction graph, first all the chains between the two genes (the agent and the target) are identified, and then the shortest chain is preferred. If there is no syntactic chain that connects the two genes in the analyzed sentence, we decide that there is no interaction between those genes. Unfortunately in the current version of our system we do not solve coreferences (anaphora) so we are not able to find a path between the two genes if one of them is not referred directly by its name.

However, when processing the training data in all those cases in which we found no syntactic relation chain connecting the two actors of a designated genetic interaction, this was due to the presence of an anaphora, and we added manually (as a supplement to the given training annotation) a new syntactic relation (named *coreference*), which solved the anaphora. Solving the anaphora (and automatically creating coreference relations) should be part of our system in a future version.

When, inside a syntactic relation chain, a link changes the orientation we say that we found a *root*. A root is thus an “inflexion” point, a node having the in-degree or the out-degree 2 in a syntactic chain. As shown in Figure 1, there could be multi-rooted syntactic

relation graphs. There is also possible that there is no change of orientation/direction in a syntactic relation chain. An example of this kind appears in Figure 2. The way we choose a root for such chains is explained in Section 2.2.

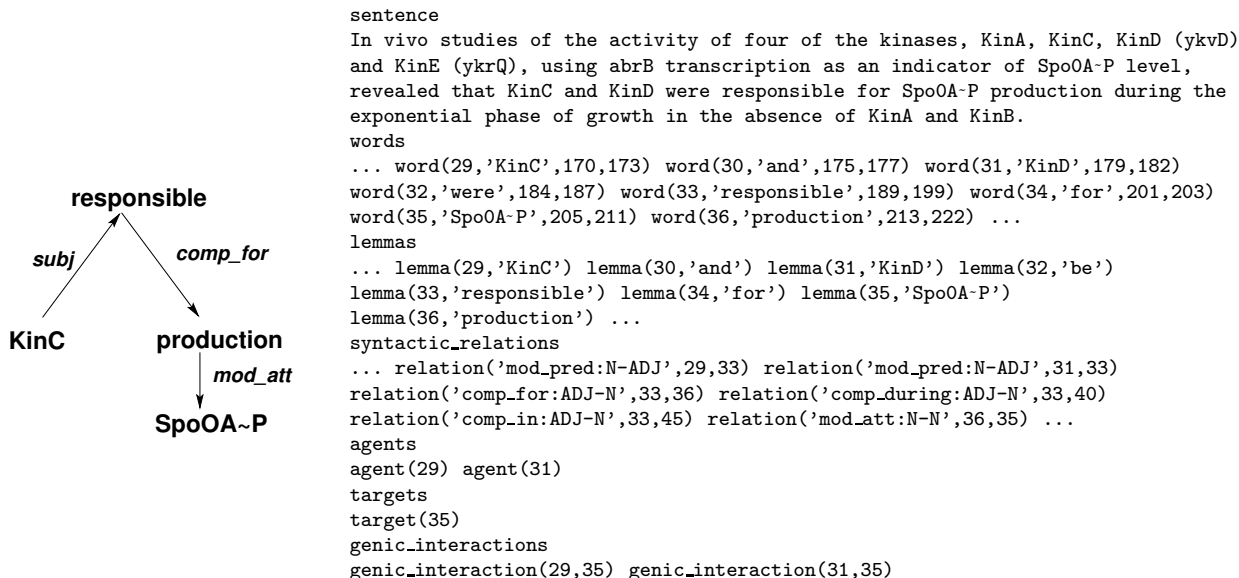


Figure 2: A genic interaction tree without change of direction, extracted from the annotations provided to the sentence *KinC and KinD were responsible for SpoOA-P production*.

Choosing the negative examples can be a tricky matter. For a given sentence s and two gene names A and B found in that sentence the *relative distance* between A and B can be defined as the number of gene names that occur between A and B in s , plus one. We observed that the distance between most interacting genes in the training sentences was of one unit. Therefore, to generate negative examples, we started by selecting all the non-interacting pairs of genes in the training sentences for which the relative distance was greater than one. Then, as we wanted to get a roughly similar distribution of positive and respectively negative examples, and because designating the agent and the target of an interaction was part of the *LLL'05 Challenge*, we also added to the list of negative examples some of the gene pairs that were designated as interacting, but swapping the agent with the target.

2.2 Selected Features

This section presents the features we used in order to classify the instances, detailing the way we computed the values of the most important features. The first of such features, the so-called *number of “close” neighbors* for an *interaction graph* is the subject of Section 2.2.1, while the *main root*, the *head word* or *head expression* are presented in Section 2.2.2. Section 2.2.3 will enumerate the other features that our system uses, up to 23 features in total.

2.2.1 Computing the Number of “Close” Neighbors for an Interaction Graph

The number of neighbors for an instance was proved to be the most important feature for classifying instances as positive or negative examples of interacting genes. The notion of

distance between two structures represented as graphs is not an usual one, so we just have to accommodate this notion to our specific problem.

Given the *interaction graphs* of two gene pairs we check whether they have the same number of roots, their respective roots have the same (part of speech) POS tags and the orientation of the paths between two consecutive roots is the same for both graphs. We consider that two interaction graphs that differ in at least one of these conditions are far from being similar and therefore we will say that they are not “close neighbors”. In such a case, the distance we assign to the two graphs is a certain maximum value, chosen empirically, not allowing these graphs to be treated as close neighbors in subsequent computations.

Otherwise — i.e. after the application of the above step the two interaction graphs remain candidates for being close neighbors — the next step is to associate a certain string to each one of them, and then to define the distance between the two graphs as the Levenshtein distance between the two associated strings. The strings basically codify the path between a (agent or target) gene and the closest root in the given interaction graph, and between two consecutive roots in case the graph is a multi-rooted one. Each string is obtained using the following simple and natural method: we start the string with the code of the POS tag of the agent word and the agent word itself. Then for every syntactic relation (corresponding to an edge of the path), we add to the resulting string the code of the relation, the code of the POS tag of the target word in that relation, and the target word itself. Finally, for the two given interaction graphs we compute the Levenshtein distance between the two created strings and thus obtain the distance between the two graphs. An interaction graph is considered to be a “close neighbor” of another interaction graph if the distance between them is less than a certain value that was experimentally chosen.

2.2.2 Selecting the Head Word/Expression and the Main Root for an Interaction Graph

An usual practice when working with parse trees in classification problems is to use the *head word* of the tree as a feature [1]. For the genic interaction problem, we define the head word as the key word defining the interaction in the analyzed phrase. For example, the interaction graph in Figure 1 has “depends” as the head word. We naturally extend the widely used definition for head word to *head expression*, as being a collocation that constitutes the key part of the phrase documenting the interaction.

As in our approach the roots play an important part in defining the interaction graphs, we considered a new feature to serve towards classification: the *main root*. For example, the interaction graph in Figure 1 has “depends” also as main root besides being the head word. However it is not always true that the head word coincides with the main root.

We used *boosting*-like approach to recursively discover/learn the values of these two features in the interaction graphs. We started by extracting the root words from the smallest graphs associated to training instances (for which it can be assumed that the head word coincides with the main root), creating thus a list of *candidate* head (and main root) words. Then, we identified heads and main roots for larger graphs. When we look for a candidate word in the created list of candidates, first we compare the lemma (morphological stem) of the word that we need to check for the head (main root) status with the lemma of each word in the list. If we were not able to find it in the list, we extend the search to the related forms

of the word lemmas using WordNet.³

In the case of graphs like the one in Figure 2, for which we cannot find a root in the usual manner, we noticed that it was characterized by the presence of a “mod_pre” relation. The phrase had the following (maybe simplified) form:

subject “be” adjective/noun

In such a case, we chose the second element of the “mod_pred” relation to be the main root of the interaction graph.

Head expressions, for instance “*acts as repressor*”, “*is member of the ... regulon*”, “*is under the control of*” proved to be described by certain patterns of relations (or relations chains) of the form `comp_as(V, N)`, `med_pred(N, N)`, `med_pred(N, ADV)`...

2.2.3 Other Features

Most of the following features needed to be computed after computing the values for the features described above:

- number of nodes (words) in the interaction graph; the number of roots in the interaction graph; the number of words between the gene agent and the gene target, in the sentence documenting the interaction;
- the position in the sentence of the agent relative to the target;
- a boolean value that specifies whether the main root is a verb or not;
- the type of the syntactic relation between the main root and the next/previous word in the syntactic chain; the type of the syntactic relation between the head word and the next/previous word in the syntactic chain;
- the POS tag of the word before/after the main root; the POS tag of the word before/after the head word;
- the direction of the syntactic relation between the main root and the word after/before it; the direction of the syntactic relation between the head word and the word after/before it.

3 Genic Interaction Extraction as a Classification Task: Methods, Results, and Possible Improvements

Using the features presented in the precedent section, we tried several machine learning algorithms implemented in the freely available WEKA library to learn a classification model and then predict the values for the test instances. To compute the final precision, recall and F-measure scores we used the program offered by the organizers of the *LLL'05 Challenge*.⁴

³WordNet is a semantic lexicon of the English language: <http://wordnet.princeton.edu/>.

⁴<http://genome.jouy.inra.fr/texte/LLLchallenge/scoringService.html>

Classifier	Precision	Recall	F-Measure
SVM	50.9	63.8	56.6
AdaBoost	69.4	49.3	57.7
BayesNet	54.3	60.2	57.1
NaiveBayes	55.3	62.6	58.7
RBFNet	56.0	61.4	58.6

Table 1: Final Results

We compared the results obtained by WEKA on our data using nominal attributes versus those obtained using numeric attributes. The last ones were far better, so we didn't use the nominal attribute version anymore.

3.1 Preprocessing and Postprocessing the Data

In order to improve the classification results we applied some simple preprocessing and postprocessing rules.

Preprocessing. After the first classification results on the test data have been obtained, we observed that many pairs of genes in coordinative relation were classified as true interactions. We decided not to consider anymore such genes as potentially interacting genes, and this step helped us to obtain better precision scores.

Postprocessing. We observed that many gene pairs were considered to interact in both directions, i.e. the training data made the classifiers unable to determine which one of the two genes under consideration was the agent gene and which one was the target gene. This is a serious problem because it lowers the precision score radically. We tried several rules to help the system choose among the two syntactic relation trees obtained by ignoring other roots than the main root in the interaction graphs, and the following one proved useful: Choose the tree which has the highest number of positive close instances in the training data. In those cases in which the two trees have the same number of positive close instances, we took a simple approach: choose the one that has the target's position preceding that of the agent in the documenting sentence. (We observed when checking the other possibility around that the F-measure score lowers by 1-2%.)

3.2 Our Results

The final version of our system contains all the features presented in Section 2 and the preprocessing and the postprocessing steps explained above. Table 1 presents the results obtained by the system when running different classifiers.

The best F-measure score obtained was 58.7%, using the *Naive Bayes* classifier. If we make a comparison between the results of several classifiers, we observe that the Support Vector Machine (SVM) produced the best recall, but in this case the precision was much lower. Apart the *AdaBoost* meta-algorithm, all the other classifiers produced a higher recall than precision. *AdaBoost* obtained a 69.4% precision, much higher compared to that of the SVM, which was only 50.9%. The most balanced results between precision and recall were

obtained by the *RBF Network* classifier, for which the F-measure was only 0.1% lower than the best, obtained by the *Naive Bayes* classifier.

3.3 Directions of Improvement

The most important part of our genic interaction extraction system is the conception of the set of features used in classification. This is the means to overcome the drawback consisting of the flat representation of the syntactic relation trees. Our opinion is that there is enough space for improvement in this core part of the system.

Remember that we computed the number of close neighbors of an interaction graph by counting all the interaction graphs in the training data that were at a certain distance from it or closer. As one can easily remark, there is a problem which appears when comparing the distance between very small trees versus very large ones. It would be better to use a certain correlation between these distances and the size of interaction graphs. What we mean by this is that for small graphs/trees we should use a smaller maximum distance, while the larger the graphs/trees are, the bigger the maximum distance should be.

Moreover, we should use an improved version the Levenshtein distance in order to give different weights to the edges, depending on which level they are in the graph. Solving the anaphora occurrences is also an important issue that needs to be done to obtain better results.

The participating teams at the *LLL'05 Challenge* that obtained the best results used additional training data besides what was distributed to the participants by the *LLL'05 Challenge* organizers. It would be interesting to see the way our system can benefit from such an improvement in the size of the training dataset.

The results we obtained are very encouraging but we must have in mind that training data were carefully selected in order to have a simple underlying biological model. Also, the syntactic relations between words, after being computed by the LinkParser have been corrected by hand. What we have to do next is to extend the training datasets so to becomes more representative of the real data, as found in MEDLINE abstracts, leaving the syntactic parsing partially incorrect as it is when produced by automatic methods.

4 Final Conclusions

This paper presented our approach for solving the *Genic Interaction Extraction* LLL'05 challenge task, namely how to extract protein/gene interactions from abstracts in the MEDLINE bibliography database.

Most of the groups that participated at the challenge have used only the training dataset without coreferences and obtained high recall and low precision. Quite surprisingly, the two groups that used both training sets (with and without coreferences), obtained very similar scores when using the the basic version and respectively the linguistically enriched version of the two training datasets.

The final version of our system employs all the 23 features presented in Section 2 and also the preprocessing and the postprocessing phases presented in Section 3. The best F-measure

score that we obtained was 58.7%, using the *Naive Bayes* classifier. The best was obtained by *ADA Boost*, 69.4%, while the best recall was due to a Support Vector Machine, 63.8%.

Our results are comparable to the best results obtained by the participant teams at the challenge task. However, we have argue that there is still significant space left for improvement in solving such a task.

References

- [1] A. Moschitti. A Study on Convolution Kernels for Shallow Semantic Parsing. *Proceedings of the ACL Conference*, pages 335–342, 2004.
- [2] C. Nédellec. Learning Language in Logic - Genic Interaction Extraction Challenge. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 31–37, 2005.
- [3] D. Temperley. An Introduction to the Link Grammar Parser. 1999. <http://www.link.cs.cmu.edu/link/dict/introduction.html>.
- [4] J. Ding, D. Berleant, D. Nettleton, and E. Wurtele. Mining MEDLINE: abstracts, sentences, or phrases? *Proceedings of the Pacific Symposium on Biocomputing*, pages 326–337, 2002.
- [5] J. Hakenberg, C. Plake, U. Leser, H. Kirsch, and D. Rebholz-Schuhmann. LLL'05 Challenge: Genic Interaction Extraction - Identification of Language Patterns Based on Alignment and Finite State Automata. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 38–45, 2005.
- [6] L. Popelinsky, and J. Blatak. Learning Genic Interaction Without Expert Domain Knowledge: Comparison of Different ILP Algorithms. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 59–61, 2005.
- [7] M. Goadrich, L. Oliphant, and J. Shavlik. Learning to Extract Genic Interactions Using Gleaner. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 62–68, 2005.
- [8] M. Greenwood, M. Stevenson, Y. Guo, H. Harkema, and A. Roberts. Automatically Acquiring a Linguistically Motivated Genic Interaction Extraction System. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 46–52, 2005.
- [9] S. Katrenko, S. Marshall, M. Roos, and P. Adriaans. Learning Biological Interactions From Medline Abstracts. *Proceedings of the 4th Learning Language in Logic Workshop (LLL'05), Bonn, Germany*, pages 53–58, 2005.
- [10] S. Riedel, and E. Klein. Genic Interaction Extraction with Semantic and Syntactic Chains. *Proceedings of the 4th Learning Language in Logic Workshop, Bonn, Germany*, pages 69–74, 2005.