Playing Biology's Name Game: Identifying Protein Names in Scientific Text

D. Hanisch, J. Fluck, HT. Mevissen, R. Zimmer

Pacific Symposium on Biocomputing 8:403-414(2003)

PLAYING BIOLOGY'S NAME GAME: IDENTIFYING PROTEIN NAMES IN SCIENTIFIC TEXT

DANIEL HANISCH [a], JULIANE FLUCK [a], HEINZ-THEODOR MEVISSEN [a]
Fraunhofer Institute for Algorithms and Scientific Computing (SCAI)
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
{Daniel.Hanisch, Juliane.Fluck, Heinz-Theodor.Mevissen}@scai.fhg.de

RALF ZIMMER
Institut für Informatik, Ludwig-Maximilians-Universität München
Theresienstraße 39, D-80333 München, Germany
Ralf.Zimmer@bio.informatik.uni-muenchen.de

A growing body of work is devoted to the extraction of protein or gene interaction information from the scientific literature. Yet, the basis for most extraction algorithms, i.e. the specific and sensitive recognition of protein and gene names and their numerous synonyms, has not been adequately addressed. Here we describe the construction of a comprehensive general purpose name dictionary and an accompanying automatic curation procedure based on a simple token model of protein names. We designed an efficient search algorithm to analyze all abstracts in MEDLINE in a reasonable amount of time on standard computers. The parameters of our method are optimized using machine learning techniques. Used in conjunction, these ingredients lead to good search performance. A supplementary web page is available at http://cartan.gmd.de/ProMiner/.

1 Introduction

Biological articles provide a wealth of information on genes and proteins and their interaction under different experimental conditions. To make this amount of data manageable to biological experts and to utilize these data in conjunction with bioinformatics methods (e.g. for contextual analysis of DNA microarray data) it is desirable to condense the free text information into a machine-readable, well-defined form. One example is the generation of biological interaction networks from scientific abstracts [1,2,3] . A requirement for all these approaches is an accurate, sensitive and efficient recognition of the entities under consideration, i.e. proteins and genes, in the free text. In our opinion, however, this building block of higher level information extraction did not receive sufficient attention.

A fundamental problem of gene name search in biological articles is the quite frequent deviation of authors from a recommended gene nomenclature or the absence of such a standard in some cases. Consequently, each gene might

[a] All authors contributed equally to this paper.

have several synonymous aliases and functionally unrelated genes might bear the same name [4,5]. For protein names the situation is even more complicated, as synonyms often consist of several words and permutations of these tokens may occur.

Methods for identifying gene and protein names in free text can be divided into methods utilizing a name dictionary and methods relying on other means. Methods which do not use a dictionary [6,7,8] can identify potential proteins in texts which are previously not contained in standard dictionaries and can thus be used to compose such dictionaries semi-automatically. In contrast, they face the problem to unify different aliases of found entities. This makes them hard to use as a building block for higher level analysis. A more straightforward method is to utilize a database of protein and gene names. Krauthammer et al. [9] treat the search problem as an alignment of a protein name against the database of scientific abstracts. Their approach is character-based and utilizes the BLAST algorithm. While the character-oriented approach has the advantage of finding slightly modified forms of words (e.g. plural forms), it faces the problem of how to detect semantically significant mismatches of characters (e.g. modifications in gene names). The success of all dictionary-based approaches obviously depends on the quality of the dictionary. Ono et al. [1] use a manually constructed dictionary which contains only a problem-specific subset of proteins. Jenssen et al. [3] employ a dictionary which only contains gene symbols and short gene names extracted from the HUGO database [10]. Thereby they circumvent the problems associated with longer synonyms, but face a sensitivity penalty.

In this paper we will show that simple text search of gene names leads to poor sensitivity, whereas naïve search of protein synonyms incurs a loss in specificity. Consequently, we build a large curated dictionary of protein and gene names and a corresponding token-based search algorithm to achieve our goal. In the following section we will discuss the underlying model for protein and gene names. Based on this, we describe the automated generation and curation of our synonym dictionary. This dictionary is used in the search algorithm which is presented in section 4. In section 5 we optimize the parameters of our method and validate our findings. The paper closes with a discussion of possible further enhancements of our method.

2 Model for protein and gene names

A crucial characteristic of protein names is that they are often composed of more than one word (or token). The order of these words is only semantically significant up to a certain limit, i.e. permutations of tokens may occur (cf. Ta-

Name	Description	Examples
Modifier	Semantic-modifying tokens	receptor, inhibitor
Non-descriptive	Annotating tokens	fragment, precursor
Specifier	Numbers and Greek letters	1,VI, alpha, gamma
Common	Common English words	and, was, killer
Delimiter	Separator tokens	( ) , . ;
Standard	Standard tokens	TNF, BMP, IL

Table 1: Definition of token classes with differing semantic significance.

ble 2 Examples 6 and 7). Moreover, we face the problem that general-purpose dictionaries of protein names must be automatically composed (e.g. from protein databases) as only the number of human proteins in the SWISSPROT and TREMBL databases is approximately 40.000. However, some tokens included in those databases are only rarely used in free text (cf. Table 2 Example 1, 2). An important observation to overcome these problems when identifying synonyms in free text is that words can be partitioned according to their semantic significance into *token classes*.

2.1 Definition of token classes

To assess the different significance of tokens, we extract all words from the dictionary with frequency of occurrence greater than one hundred. On the basis of these data, we manually define token classes which influence curation of the dictionary and the match procedure in various ways. The class of *non-descriptive tokens* contains words which often occur in databases but are rarely used in free text (cf. Table 2 Example 1, 'precursor') or have no influence on the significance of the match (Example 6, 'type'). In contrast, the class of *modifier tokens* contains words which are crucial for the correct recognition of the underlying entity. Names 3 and 5 of Table 2 clearly describe different proteins. The difference is expressed through the modifier token 'receptor'. Along the same lines, the class of *specifier tokens* which is comprised of Arabic and Roman numbers and Greek letters is usually used to discriminate among the members of a protein family (cf. examples 3 and 4). *Delimiter tokens* are used to gain specificity in the matching procedure. Name boundaries in the free text which are unknown *a-priori* can be detected more easily with the help of delimiters. Usually, the capitalization of protein names is insignificant. However, some gene identifiers are exceptions to this rule. For example, the gene names 'KILLER' or 'WAS' will be detected erroneously in free text when the search is case-insensitive. Tokens of this type can be assigned to the class of *common words* by comparison to a standard English dictionary. During

1.	Interleukin	-	1		beta	precursor
2.	INTERLEUKIN	1	-		beta	PROTEIN
3.	INTERLEUKIN	1			beta	
4.	Interleukin	2			beta	
5.	Interleukin	1	RECEPTOR		BETA	
6.	Collagen	type	XIII		alpha	1
7.	Alpha	1	type		XIII	collagen

Table 2: Examples of protein names tagged with token classes. Names 1-3 refer to the same entity, but differ in spelling and *non-descriptive tokens*. Examples 4 and 5 represent distinct entities differing in *modifier-* and *specifier tokens*. Examples 6 and 7 are synonyms demonstrating the presence of permutations of tokens.

the matching procedure, synonyms composed of only one common word are considered case-sensitive. All tokens not explicitly classified are termed *standard tokens*. This class also includes gene identifiers as they cannot be easily assigned to a separate class. The current definition of token classes is available through the supplementary web page (http://cartan.gmd.de/ProMiner).

3 The protein and gene name dictionary

The quality of the used dictionary is essential for the success of the matching procedure. Indeed, if a perfect general purpose dictionary of names in all occurring spelling variants were available, the matching procedure would be trivial. Unfortunately, it is highly unlikely that such a dictionary will be available in the near future. As a manual definition of a general purpose dictionary is infeasible, we focus on automatic generation of a dictionary and subsequent sensible curation and expansion steps. In this paper, we tested our approach on the basis of human genes and proteins as this field is especially relevant for clinical and pharmaceutical research.

3.1 Automatic generation of the dictionary

We extracted gene symbols, alias names, and full names for all human genes from the HUGO Nomenclature database [10] and created an entry in the dictionary (called an *object*) for each official gene symbol and added the corresponding names available in the OMIM database [11]. Furthermore, we extracted all synonyms of human proteins of the SWISSPROT and TREMBL databases [12] and matched these to HUGO entries.

#	Original syn.	Curated syn.	Status	Reason
1a	IL1	IL1	keep	—
1b		IL 1	add	separation
1c		Interleukin 1	add	acronym expansion
2	Transcription Factor	—	remove	expert curation list
3	Phosphohexokinase	—	ambiguous list	occurs in several objects
4	EPO	—	ambiguous list	occurs in several objects
5	BETA SUBUNIT	—	remove	regular expression match
6	fragment	—	remove	regular expression match

Table 3: Examples of curation of the dictionary. Synonyms may be modified, added or removed during the curation procedure for various reasons (cf. section 3.2).

3.2 Curation of the dictionary

To resolve ambiguities and to remove nonsensical names from the dictionary, a curation procedure consisting of an expansion and a pruning phase was implemented. Table 3 shows several curated synonyms which serve as examples in the following explanations of the curation procedure.

In the expansion phase, further synonyms for existing records were generated. This was achieved by separating alphanumeric tokens into numbers and words (cf. Example 1b), expanding known unambiguous acronyms within synonyms to broaden the scope of our dictionary (cf. Example 1c), and collecting a list of curation items which are maintained by biological experts to add or remove synonyms from the dictionary (cf. Example 2).

In the pruning phase, redundancies, ambiguities and irrelevant synonyms were removed from the dictionary. First, each token in the dictionary was tagged with its corresponding token class generating a string of token class identifiers for each synonym. Each string was then matched against a set of regular expressions representing patterns for unspecific synonyms (e.g. only *non-descriptive tokens*, only *specifier tokens* etc.). If a regular expression matched, the corresponding synonym was pruned (cf. Example 5,6). After an additional step of pruning manually defined superfluous synonyms, only one synonym of several alternatives differing only in capitalization was retained, as the matching procedure is basically case-insensitive (cf. section 4). Finally, a list of ambiguous names found in the curated dictionary was extracted and these names were stored in a separate dictionary with reference to their original records. Generally, ambiguities may arise for several reasons. One problem is the parallel invention of gene and protein names by several biologists (cf. Example 4). Moreover, reorganization of names for protein classes leads to inconsistent names. Besides these inherent name ambiguities, we encounter ambiguous names because databases do not only store name aliases but some-

times also protein class identifiers (cf. Example 3). Obviously, these identifiers apply to several entities described in our dictionary. In effect, the ambiguity list can be used to identify such entries and move them to our manual curation list based on their frequency of occurrence.

Using this curation procedure, we arrived at a name dictionary of reasonably high quality which can be used as the input for our matching procedure described in the next section. This dictionary consists of approximately 38.200 entries with 151.700 synonyms after curation.

4 Efficient detection of names in scientific abstracts

The MEDLINE database [13] contains approximately 11 million abstracts and is rapidly growing. As our ultimate goal is the detection of protein and gene names in all abstracts of the database, the running time of the detection algorithm is an important concern. Consequently, our search procedure should take time linear in the number of tokens of the parsed text. The basic idea is to sweep over the abstract, processing one token at a time and keep a set of candidate solutions and two associated scoring measures for the present position. One scoring measure, the *boundary score* s_β controls the end of the extension of a candidate match and is increased on a token mismatch (cf. Algorithm 1, line 13). If this score rises above a threshold, i.e. if a certain number of mismatches has occurred, the candidate is pruned from the candidate set and checked for reporting. Then, the second score measure, the *acceptance score* s_α, determines whether the candidate is reported as a match. The term s_α is a linear combination of token class specific match- and mismatch terms. A *match term* is defined as the *percentage of matched tokens* of the respective token class. A *mismatch term* counts for each token class the *number of tokens additionally found in the text* and, thus, mismatched in the candidate synonym. With appropriate weighting, the mismatch term allows to disregard false substring matches. To illustrate this, consider the following example:

synonym	Interleukin	1		precursor
candidate match	Interleukin	1	receptor	

As only the *non-descriptive token* "precursor" is unmatched in the candidate, a nearly maximal match score would be computed (if *non-descriptive tokens* receive a small weight). However, the presence of the semantically significant *modifier token* "receptor" leads to a substantial mismatch term for this token class (if weights are set appropriately).

Variable	Description
S	Set of all synonyms to search
T	Set of all tokens
$C = \{c_1, c_2, \ldots, c_n\} \subseteq S$	Current set of candidates
$\tau(c), c \in C$	Unmatched tokens of candidate c
$\sigma(t), t \in T$	Set of synonyms containing token t
$\#\text{token}(c), c \in C$	Number of tokens of candidate c

```
for each token t_j ∈ T read from the abstract do
2   for each synonym s ∈ σ(t_j) do
        if (s ∉ C) then
4           C = C ∪ s;
    end
6   for each candidate c ∈ C do
        if (t_j ∈ τ(c)) then do
8           Update match terms of s_α(c);
            τ(c) = τ(c)\t_j;
10      end
        else do
12          Update mismatch terms of s_α(c);
            s_β(c) += 1 / #token(c);
14      end
        if (s_β(c) > boundaryThreshold or
16              mismatchedDelimiter found) then do
            C = C\c;
18          if (s_α(c) > acceptanceThreshold) then
                report c;
20      end
    end
22 end
```

Algorithm 1: Linear algorithm for detection of protein names in free text.

Delimiter tokens play a special role in the algorithm. If the current token t_j is a delimiter but is unmatched in the current candidate (cf. line 16 in Algorithm 1), the candidate match will not be extended further but checked for reporting immediately (cf. line 18). The rationale is that a delimiter signifies a break in the text possibly followed by another protein name which should not be mixed up with previous candidates. To further extend the sensitivity of the search, we maintain a list of synonymous token. Most importantly, Arabic and Roman numbers are treated equivalently as both variants frequently occur in abstracts. In general, the algorithm ignores the order of the synonym tokens. However, permuted matches of some synonyms (e.g. EC numbers) are

meaningless. Therefore, we extended the search algorithm to respect the order of tokens when this is necessary. This complicates the operations on τ (lines 7 and 9), but the concept is applicable analogously.

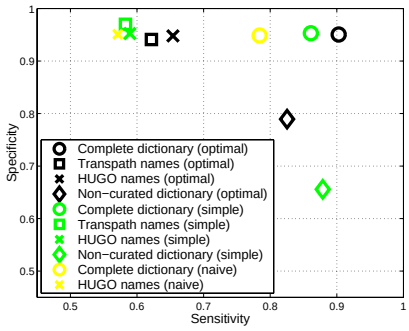
The parameters of our search procedure, namely the match- and mismatch-weights, can be used to control the fuzziness of the search as demonstrated in the above example. A method to optimize these parameters using a set of training instances is discussed in the next section.

5 Parameter optimization and validation

To examine the performance of our name identification procedure, we need a standard-of-truth of annotated abstracts. As creation of a substantial data-set is extremely work-intensive, we based our benchmark set on the TRANSPATH database [14] on regulatory interactions (Version 2.3). We extracted all human proteins with SWISSPROT annotations and all corresponding MEDLINE identifiers and re-examined all abstracts for further occurrences. Additionally, we discarded abstracts if no text was available or a protein was described for the first time and, thus, was not associated with any name, at all. Our resulting benchmark set consists of 611 associations (141 objects in 470 abstracts). The subset of the curated dictionary relevant to this benchmark is available through the supplementary web page (http://cartan.gmd.de/ProMiner).

As the acceptance score s_α is a linear function of the token class specific scoring terms, we can use *robust linear programming (RLP)* [15] to compute a set of sensible weights. This supervised machine learning technique uses a set of positive samples, i.e. correctly identified protein names, and a set of negative examples, i.e. erroneous matches. The RLP procedure computes a separating hyperplane in the vector space of scoring contributions, thus determining an optimized scoring function. We calibrated the match and mismatch weighting parameters for *delimiter-*, *specifier-*, *modifier-* and *standard-tokens*. To this end, we generated a number of training instances for the RLP algorithm by setting weighting parameters to unity and acceptance and mismatch thresholds to lenient values and classified the resulting matches according to our standard-of-truth. The RLP parameter optimization consistently led to plausible weightings which penalize mismatch terms of *modifier* and *number tokens* and reward matching terms of the other token classes to various extents.

To assess the performance of our method, we use a 5-fold cross-validation procedure, i.e. the set of abstracts is divided into five parts. Four parts are used for optimization of parameters and the remaining one determines matching performance. Figure 1 depicts the averaged results of our matching procedure individually optimized for different dictionaries. Additionally, results for a



Dictionary	Method	Spec	Sens
Compl. Dict.	opt.	0.95	0.90
Transpath	opt.	0.94	0.62
HUGO	opt.	0.94	0.65
Non-cur. Dict.	opt.	0.78	0.82
Compl. Dict.	simple	0.95	0.86
Transpath	simple	0.97	0.58
HUGO	simple	0.95	0.58
Non-cur. Dict.	simple	0.65	0.87
Compl. Dict.	naïve	0.94	0.78
HUGO	naïve	0.95	0.57

Figure 1: Specificity (true positives/ (true positives + false negatives)) and sensitivity (true positives/ (true positives + false positives)) of matching procedure using different dictionaries of synonyms and parameters for the matching procedure. These concepts are closely related to the alternative measures of *precision* and *recall*. *Complete dictionary* and *non-curated dictionary* refer to the comprehensive dictionary in its fully curated and non-curated form, respectively. For *Transpath names* and *HUGO names* only the names defined in the respective databases are considered. In addition, three different modes of matching are employed. *Optimal* denotes cross-validated parameter optimization, *simple* refers to a simpler choice of parameters (non-descriptive tokens ignored, permutations allowed, no additional mismatches) and *naïve* to performance of naïve string matching.

simpler choice of weight parameters are shown.

Clearly, the optimized version in conjunction with the curated dictionary performs best. The reason is twofold. On one hand, there is a significant gain in sensitivity when switching from either HUGO or TRANSPATH names to the complete dictionary. On the other hand, our matching procedure adequately addresses the characteristics of protein names, again resulting in increased sensitivity and near constant specificity. In contrast, the use of the non-curated version of the dictionary leads to a large specificity penalty.

Table 4 summarizes major sources of errors in our protein name identification algorithm. One source of error for false positive matches are unspecific dictionary entries (e.g. glutamate receptor). Another source are semantically ambiguous names (e.g. HEK protein and HEK cells) which could not be discovered in the dictionary curation process. Imperfect parameter settings are a cause of further false positive matches. This shortcoming could possibly be remedied by a larger training set or an extension of token classes. Main reasons for failures in name identification are missing synonyms. This includes renaming of synonyms not reflected in our constituent databases (e.g. lymphotoxin to lymphotoxin 1), use of delimiters within synonyms (FGF.6 for FGF 6), or spelling variants of individual authors (e.g humEAA1 for EAA1 or pRB2 for RB2). In some cases, synonyms were written as one token (e.g. 'IL1beta' in-

Description of error	No. abstracts	No. objects	Type of error
ambiguous synonyms	7	3	false positive
unspecific synonyms	14	4	false positive
not in dictionary	25	15	false negative
plural form	2	2	false negative
tokenization problem	4	4	false negative
linguistics required	10	10	false negative

Table 4: Major sources of detection errors. Upper part contains major sources of false positive matches, lower part summarizes reasons for false negative matches.

stead of 'IL 1 beta') or plural forms were not recognized. In other cases, more linguistic information is required to recognize the protein name.

6 Discussion and future work

Analysis of today's large-scale experiments requires knowledge of relationships among a large number of genes and proteins. The successful identification of these objects including all known synonyms requires a comprehensive, curated, general-purpose dictionary. As the manual generation of such a protein name dictionary is infeasible, we presented a semi-automated method to arrive at a dictionary of reasonable high quality. The procedure is based on the definition of token classes for name components of different semantic significance and the specification of acronym and manual curation lists. Through improvement and adaptation of these lists to user needs, the quality of the resulting dictionary can be enhanced. An important point is that the invested knowledge is reused on integration or update of the underlying synonym databases. For example, the identification of further *modifier tokens* could enhance specificity especially when considering members of protein families. Manual addition or subtraction of correct synonyms is also essential, e.g. to increase specificity for certain objects (cf. Table 4) or to take into account lists of unspecific annotations (e.g. in the automatically annotated TREMBL database). To this end, ambiguities identified in the curation process can be inspected and selectively added to the manual curation lists. As our name dictionary is based on the most relevant reference databases, it is easily possible to link other datatypes to the generated results. For example, co-occurrence networks can be visualized in conjunction with data from gene expression experiments.

The inclusion of protein names to enhance sensitivity complicates the design of the search algorithm as longer phrases, permutations of tokens, and spelling variants must be taken into account (cf. Figure 1, "Non-curated dictionary" and "Complete dictionary (naïve)"). The presented algorithm was designed for high efficiency while respecting the defined characteristics of pro-

tein names. To this end, we approximated protein phrase boundaries by a mismatch threshold and the use of delimiter tokens. As demonstrated in Figure 1, the method is both specific and sensitive. The biological use of the presented work is currently evaluated in the context of the "Leitprojekt Osteoarthrose" [16], a research project dedicated to elucidate the pathomechanisms of the degenerative joint disease osteoarthritis. In that context, name search is employed to generate co-occurrence networks of proteins associated with the disease under consideration. Examples of networks can be found on the supplementary web page (http://cartan.gmd.de/ProMiner/).

As future work, we would like to enhance the performance of our method further. The incorporation of lexical rules (e.g. plural forms) during token matching could improve sensitivity. Moreover, recent work on disambiguating the semantic context of found names [17] might be used to decrease the number of false positive matches. In conjunction with methods that work independent of predefined dictionaries, the name dictionary could be extended further in a semi-automated fashion.

In addition, we plan to utilize ontological information to control ambiguous synonyms and abstract our search results in a hierarchical manner. A good candidate for this enhancement is GO, the gene ontology database [18], as it provides a direct mapping to SWISSPROT which is a constituent database of our name dictionary. In general, we feel that there is an urgent need for publicly available, annotated, domain-specific corpora for more accurate estimation of matching parameters and, even more important, for standardized comparison of different algorithms for name identification on a large scale.

We plan to develop a publicly available web-interface to the matching procedure itself, when further work on information extraction algorithms has been incorporated. Used in conjunction, these methods promise to make the huge body of biological literature accessible to bioinformatics methods and biological experts alike.

Acknowledgments This work was in part supported by Aventis Pharma, Frankfurt. We thank the members of the "Leitprojekt Osteoarthrose" for discussion on co-occurrence matrices and networks. We are indebted to Alexander Zien for his implementation of the RLP algorithm. We thank Biobase Inc. for providing us with a TRANSPATH license.

1. T. Ono, H. Hishigaki, A. Tanigami, and T. Takagi. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17:155, 2001.
2. J. Thomas, D. Milward, C. Ouzounis, S. Pulmann, and M. Carroll. Automatic extraction of protein interactions from scientific abstracs. *Pacific*

Symposium on Biocomputing, 5:514, 2000.
3. T.K. Jenssen, A. Lagreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28:21, 2001.
4. J. Pustejovsky, J. Castao, B. Cochran, M. Kotecki, M. Morrell, and A. Rumshisky. Extraction and disambiguation of acronym-meaning pairs in medline. *Medinfo*, 2001.
5. H. Pearson. Biology's name game. *Nature, Features*, 2001.
6. K. Fukada, A. Tamura, T. Tsunoda, and T. Takagi. Toward information extraction: identifying protein names from biological papers. *Pacific Symposium on Biocomputing*, page 701, 1998.
7. D. Proux, F. Rechenmann, L. Julliard, V. Pillet, and B. Jacq. Detecting gene symbols and names in biological texts: a first step toward pertinent information extraction. *Genome Informatics Workshop*, pages 72–80, 1998.
8. N. Collier, C. No, and J. Tsujii. Extracting the names of genes and gene products with a hidden markov model. In *Proc. COLING 2000*, pages 201–207, 2000.
9. M. Krauthammer, A. Rzhetsky, P. Morozov, and C. Friedmann. Using blast for identifying gene and protein names in journal articles. *Gene*, 259:245, 2000.
10. H.M Wain, M. Lush, F. Ducluzeau, and S. Povey. Genew: the human nomenclature database. *Nucleic Acids Research*, 30:169, 2002.
11. Online Mendelian Inheritance in Man, OMIM (TM), 2000.
12. A. Bairoch and R. Apweiler. The swiss-prot protein sequence database and its supplement trembl in 2000. *Nucleic Acids Research*, 28:45, 2000.
13. Pubmed - national library of medicine. http://www.ncbi.nlm.nih.gov/entrez/, 2000.
14. F. Schacherer, C. Choi, U. Gtze, M. Krull, S. Pistor, and E. Wingender. The transpath signal transduction database: a knowledge base on signal transduction networks. *Bioinformatics*, 17:1, 2001.
15. K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly separable sets. *Optimization Methods and Software*, 1:23–34, 1992.
16. Leitprojekt Osteoarthrose. http://www.leitprojekt-oa.de, 2000.
17. Vasileios Hatzivassiloglou, Pablo A. Duboue, and Andrey Rzhetsky. Disambiguating proteins, genes, and RNA in text: a machine learning approach. *Bioinformatics*, 17(90001):97S–106, 2001.
18. The Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Research*, 11:1425, 2001.