ProtChew: Automatic Extraction of Protein Names from Biomedical Literature

Amund Tveit and Rune Sætre
Department of Computer and Information Science
NTNU, Norway
{amund.tveit,rune.satre}@idi.ntnu.no

Astrid Lægreid and Tonje Strømmen Steigedal
Department of Cancer Research and Molecular Medicine
NTNU, Norway
{astrid.laegreid,tonje.strommen}@ntnu.no

Abstract

With the increasing amount of biomedical literature, there is a need for automatic extraction of information to support biomedical researchers. Due to incomplete biomedical information databases, the extraction is not straightforward using dictionaries, and several approaches using contextual rules and machine learning have previously been proposed. Our work is inspired by the previous approaches, but is novel in the sense that it is fully automatic and doesn't rely on expert tagged corpora. The main ideas are 1) unigram tagging of corpora using known protein names for training examples for the protein name extraction classifier and 2) tight positive and negative examples by having protein-related words as negative examples and protein names/synonyms as positive examples. We present preliminary results on Medline abstracts about gastrin, further work will be on testing the approach on BioCreative benchmark data sets.

1. Introduction

With the increasing importance of accurate and up-to-date protein/gene information databases and ontologies for biomedical research, there is a need to extract protein information from biomedical research literature, e.g. those indexed in Medline [20].

Methodologically these approaches belong to the *information extraction field* [5], and in the biomedical domain they range from learning relationships between proteins/genes based on co-occurrences in Medline abstracts [9] to *manually* developed protein information extraction rules [21] and protein name classifiers trained on *manually* annotated training corpora [2].

Examples of protein names in a textual context

1. "duodenum, a **peptone** meal in the"
2. "subtilisin plus leucine **amino-peptidase** plus prolidase followed"
3. "predictable hydrolysis of **[3H]digoxin-12alpha** occurred in vitro"

1.1. Research Questions

Two of the main issues in information extraction in general are: 1) how to *automate* the generation of *annotated training data* needed to create extraction rules and classifiers and 2) how to select *appropriate negative examples* that are closely related but disjoined from the positive examples in order to ensure high accuracy for the information extraction of protein names. This leads to the following hypotheses:

1. Can existing *protein information databases* be used for *fully automatic generation of tagged training data* for protein name extraction classifiers?
2. Can existing *protein information databases* be used to create *appropriate negative examples* for information extraction of protein names?

The rest of this paper is organized as follows. Section 2 describes the materials used, section 3 presents our method, section 4 describes related work, section 5 presents empirical results, section 6 discusses our approach, and finally section 7 contains the conclusion and future work.

2. Materials

The materials used included biomedical (sample of Medline abstract) and general English (Brown) textual corpora, as well as protein databases, see below for a detailed overview.

As subject for the expert validation experiments we used the collection of 12.238 gastrin-related Medline abstracts that were available in September 2004. Gastrin was selected to fit the field of expertise of the researchers who evaluated our findings.

As a source for finding known protein names we use a web search system called Gsearch, developed at Department of Cancer Research and Molecular Medicine at NTNU. It integrates three common online protein databases, namely Swiss-Prot, LocusLink and UniGene.

The Brown repository (corpus) is an excellent resource for training a Part Of Speech (POS) tagger. It consists of 1,014,312 words of running text of edited English.

3. Our Approach

We have taken a modular approach where every submodule can easily be replaced by other similar modules in order to improve the general performance of the system. The main modules correlate with the main tasks that have to be solved in an information extraction setting. There are four modules connected to the data gathering phase, namely data selection, tokenization, POS-tagging and Stemming. Then three modules deal with classification, namely Gsearch, feature extraction and Classification. The last three modules are evaluation modules that handle cross-validation, expert evaluation and dataset statistics. See figure 2.

1. **Data Selection** The data selection module uses PubMed Entrez online system to return a set of PubMed IDs (PMIDs) and abstracts for a given protein, in our case "gastrin" (symbol GAS).

2. **Tokenization** The text is tokenized to split it into meaningful tokens, or "words". We use the WhiteSpaceTokenizer from NLTK. Words in parentheses were clustered together and tagged as a single token with the special tag *Paren*.

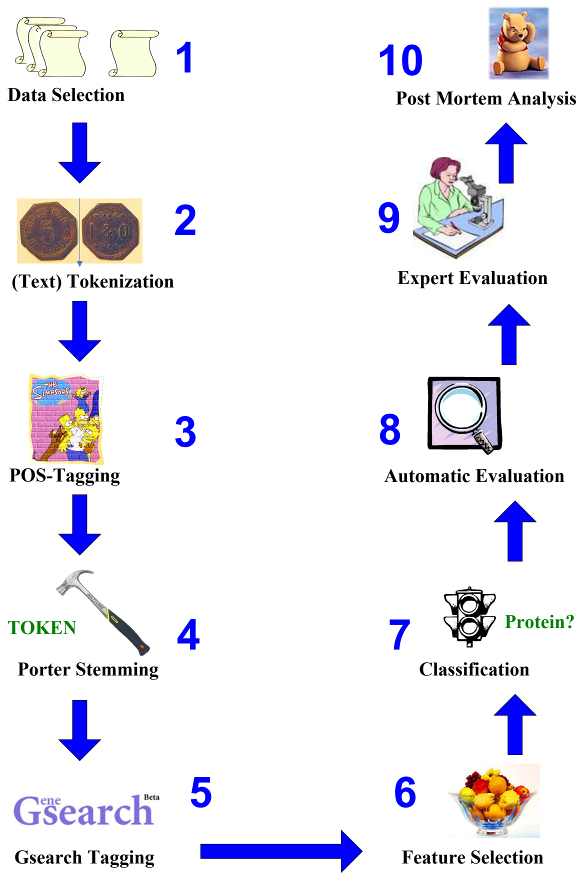

Figure 1. Overview of Our Approach

3. **POS tagging** using a Brill tagger trained on the Brown Corpus. This module acts as an advanced stop-word-list, excluding all the everyday common American words from our protein search. Later, the actual POS tags are also used as context features for their neighboring words.

4. **Porter-Stemming** If the stem of a word can be tagged by the Brill tagger, then the word itself is given the special tag "STEM", and thereby transferred to the common word list.

5. **Gsearch.org** tagging is our way of *automatically* creating positive and negative examples for the protein name extraction stage. Classifiers in general follow the rule “garbage in equals garbage out”. One way to improve this is to do careful feature selection (out of the scope of this paper). Another is in the selection of *positive* and *negative* training data - which is what we are focusing on. The idea is that if an information extraction clas-

sifier should be able to discern between protein names and other entities, it in particular needs to handle entities that are as close to protein names as possible, i.e. *protein-related* entities. We select negative examples (i.e. protein-related entities) by using words (not filtered out by past modules) describing proteins, and positive examples by using protein names and synonyms. The proteins, synonyms and corresponding descriptions are found using the Gsearch.org search engine. It enables simultaneous searches in the Swiss-Prot, UniGene and LocusLink protein databases.

The remaining words are the untagged words that need to be classified (with the classifier trained on the positive and negative data generated in this step).

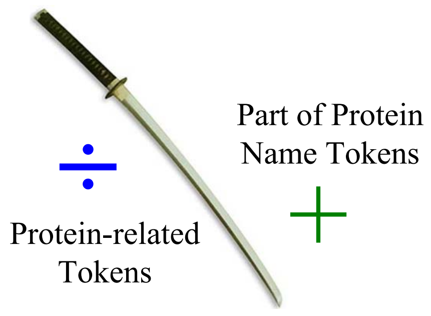

Figure 2. "Sharp edge" between positive and negative examples might improve classification accuracy

6. **Feature Selection** The features we use are the word itself (TEXT), the given tag (POS) from Brill or Gsearch (or None if the word is untagged), and other True/False features like HASBRACKET, HASFIRSTUPPER, HASNONALPHANUMPREFIX, ISLOWERCASE, ISNUMERIC, ISUPPERCASE. The features are collected for the word in question, and for the n nearest neighbors (we use $n = 3$ in our experiments).
7. **Classifier Performance** The positive and negative examples connected with the features described above are then used as training data for classification of *untagged* tokens as part of a protein name or not. Our selection of classifiers is quite pragmatic due to the *no free lunch theorem* [7], i.e. "there is no best classifier for all problems". We used the following classifiers: *Support Vector Machines* (with lin., pol., sig. and rbf kernels) in the SVM-Light tool [11], *Naive Bayes* in the Orange tool [6] and a *Proximal Support Vector Machine* (PSVM) in the Incridge tool [19] (PSVM is also known as Regularized Least Squares Classification)
8. **Automatic Evaluation** In order to efficiently test our extraction approach we first try to classify known data. If this gives extremely poor results there is no reason to pursue in classifying *untagged* tokens. The methods applied were "train and test" sets of 2500 examples each with various feature set combinations, as well as 10-fold cross-validation in order to test whether the "train and test"-set approach was ok.
9. **Expert Evaluation** The whole purpose of the extraction approach is to find proteins among *untagged* tokens. In order to do this we gave a sample of untagged tokens and their surrounding textual context to molecular biologists[1] so they could say if each token was a part of a protein name or not. We then used this as the golden standard to test our classifier performance and to measure true/false positives/negatives and to calculate F-Score and classification accuracy.
10. **Post Mortem Analysis** In order to characterize the size of the *untagged* protein names problem, we used the expert tagging from the molecular biologists in order to estimate a confidence interval for i) the probability of an untagged token being part of a protein name, and ii) the probability of a token being untagged, given our tagging sources.

4. Related Work

Our specific approach was on using existing databases to *automatically* annotate information extraction classifiers in biomedical corpora, and at the same time using these databases to create both positive and *negative examples*. We haven't been able to find other work that does this, but there are quite a few approaches on extracting protein names from biomedical literature. Below, a brief overview is given. See [17] for a more comprehensive overview.

Bunescu et al. present a method *similar to ours*, except that they train their classifiers on *manually* created corpora [2, 3, 4]. Ginter et al. describe a method weighting words by positions for resolving gene/protein name disambiguation, but they use a *manually* developed corpus for training [8]. Bickel et al. describe an

1 co-authors

Acronym	Description
F1	3 neighbors w/all
F2	3 neighbors w/text
F3	3 neighbors w/text & POS
F4	3 neighbors w/POS & word-has-bracket

Table 1. Feature approaches

approach using Support Vector Machine classifiers for gene name recognition, but it is also trained using a *manually* generated biomedical corpus [1].

Mukherjea et al. describe a method that combines *manually* generated rules with rules learned using UMLS to do biomedical information extraction [12]. Torii and Vijay-Shanker use an unsupervised bootstrapping technique from Word Sense Disambiguation [18]. This resembles our approach in the sense that it is *fully automatic*, but differs in the sense that they use an unsupervised bootstrapping technique on names found using the *manually* developed rules presented in [13]. Jiampojamarn et al. describe a supervised method using comprehensive domain knowledge and dictionaries together with classifiers for biological term extraction [10].

5. Empirical results

Since our motivation is to test the feasibility of 1) automatic creation of training data for protein name classifiers and 2) selection of appropriate negative examples in the training data, we didn't put much emphasis on the optimal selection of features for the information extraction classifiers. That is a natural next step, but outside the scope of this paper. The different feature sets we used are described in table 1, and more details about the features are given in *our approach*.

5.1. Automatic Evaluation

In order to get an overview of which classifier performance to expect, we first tested them on already tagged data, using protein names and symbols found in Gsearch as positive examples and other words from Gsearch (assumed to be protein-related) as negative examples (results in table 2). The data was first divided into a training and test set with 2500 examples each, and later we did a 10-fold cross-validation (XV) on all 5000 examples (train+test set) to verify the train and test approach.

Classifier	F1	F2	F3	F4
Majority	75.9	75.9	75.9	75.9
SVM Lin. t	75.9	75.9	75.9	75.9
SVM Pol.	**76.4**	75.9	75.9	75.9
SVM RBF	76.1	75.9	75.9	75.9
SVM Sig.	75.7	75.9	75.9	75.9
PSVM($\nu = 100$)	*68.0*	N/A	N/A	N/A
PSVM($\nu = 1$) XV	74.2	N/A	N/A	N/A

Table 2. Automatic Evaluation Results

Classifier	TP/TN	FP/FN	Prec/Rec/F	CA
Naive Bayes	**6**/*120*	*67*/**7**	8.2/**3.2**/4.6	*63.0*
Majority	*0*/**187**	**0**/*13*	*N/A/0.0/N/A*	**93.5**
SVM Lin.	*0*/**187**	**0**/*13*	*N/A/0.0/N/A*	**93.5**
SVM Pol.	**6**/159	28/**7**	17.7/**3.2**/**5.4**	82.5
SVM RBF	3/174	13/10	**18.8**/1.6/3.0	88.5
SVM Sig.	*0*/186	1/13	0.0/*0.0/N/A*	93.0

Table 3. Protein classification - untagged words

5.2. Expert Evaluation

The main purpose of our extraction approach is to detect which untagged words that are part of protein names. In order to do (and test) this, we first tagged using the Brown Corpus (regular English words) and Gsearch (protein names and protein related words) and then we selected a sample of 200 words that *hadn't been tagged*. These words and their corresponding textual contexts were classified using the classifier, and compared to manual annotations done by biologists (table 3).

5.3. "Post Mortem" Analysis

In order to say something more general about the number of protein names that cannot be tagged with LocusLink, Swiss-Prot and UniGene, we used the results after stage 5 (Gsearch tagging) and the expert's classifications of untagged words. We created confidence intervals for the probability of a word being untagged after stage 5, and for the probability that an untagged word is a part of a protein name.

The total number of unique tokens in the 12000 abstracts covering *gastrin* is $N = 76359$, and 26885 of them were untagged. This gives an estimated probability of an untagged token $p_u = 26885/76359 = 35.21\%$ and $\sigma_u = \sqrt{(\frac{p_u(1-p_u)}{N})} \approx 0.0017$. The 95% confidence interval is $[0.3521 - 1.96 \times 0.0017,\ 0.3521 + 1.96 \times 0.0017] \approx [34.88\%, 35.54\%]$

The expert found 13 protein names among a random sample of $n = 200$ untagged tokens (random sample from 26885 unique untagged tokens in total), this gives an estimated probability that an untagged word is a part of protein name $p_p = 13/200 = 6.5\%$ and $\sigma_p = \sqrt{(\frac{p_p(1-p_p)}{n})} \approx 0.0173$. The 95% confidence interval of $[6.5 - 1.96 \times 1.73,\ 6.5 + 1.96 \times 1.73] = [3.11\%, 9.89\%]$

6. Discussion

In the following section we discuss our approach on a step-by-step level (steps as presented in figure 2).

1. **Data selection** Since the results were inspected by cancer researchers the focus was naturally on proteins with a role in cancer development, and more specifically cancer in the stomach. One such protein is gastrin, and even though a search in the online PubMed Database returned more than eighteen thousand abstract IDs, only twelve thousand of these were found in our local academic copy of Medline. Another important question is if the gastrin collection is representative for Medline in general or for the "molecular biology" part of Medline in particular?

2. **Tokenization into "words"** The tokenization algorithm is important in the sense that it dictates which "words" you have to deal with later in the pipeline. How to deal with parentheses is another question. Sometimes they are important parts of a protein name (often part of the formula describing the protein), and other times they are just used to state that the words within them aren't that important. We decided to keep the contents of parentheses as a single token, but this kind of parenthesis clustering is a hard problem, especially if the parentheses are not well balanced (e.g. smiley and "1), 2), 3)" style paragraph numbering). Parentheses in Medline are usually well balanced, though, so only very few tokens were missed because of erroneous clustering. Other tokens that require special attention are the multi-word-tokens. They can sometimes be composed using dash, bracket etc. as glue, and are at other times just normal single words separated with space, even though they should really be (grouped as) a single token. An example is protein names, such as "g-protein coupled receptor (GPCR)".

3. **a) Brown Corpus and tagging** We used the Brown Corpus, an American English corpus. It is rather old (1961) and maybe not completely representative of "Medline English". There is also the challenge of how quote symbol and apostrophes are used for protein names in Medline abstracts, e.g. as a marker for the five-prime or three-prime end of a DNA formula. Also, there are only one million words in the corpus, so not all lowercase and capital letter combinations of every word are present.

   b) POS tagging with Brill algorithm and the Brown Corpus The Brill tagger doesn't tag perfectly, so maybe classifier-based taggers such as SVM could perform better. The performance of the Brill tagger could be better if we used a higher-ordered tagger than the unigram tagger as input to Brill, but the memory need for n-gram taggers are $O(m^n)$, where m is the number of words in the dictionary. So with million word training- and test sets, even the use of just a bi-gram tagger gets quite expensive in terms of memory and time-use. Tagging itself may also introduce ambiguous tags (e.g. superman is a protein, but it may be tagged as a noun/name earlier in the pipeline, because that's the most common sense mentioned in the Brown Corpus).

4. **Porter-stemming** turns out to work poorly on protein and biological names, since they are often rooted in Latin or have acronyms as their name or symbol. E.g. the symbol for gastrin is GAS, and the porter stem of GAS becomes GA, which is wrong, and too ambiguous.

5. **Gsearch** The indexing algorithm of Gsearch also contains some stemming of search terms, leading to some "strange" results when creating the positive and negative training examples. The protein names found also cover "regular words" leading to other ambiguity problems, for example when "legal" protein names are removed earlier in the pipeline by the Brill tagger. Another weakness of Gsearch is that it isn't "complete enough" (yet). It should be extended with a larger selection of databases and dictionaries covering biological terms, so that protein names like "pentagastrin" could also be found in the database.

6. **Feature Selection** Features in Information Extraction are usually ad-hoc and fosters creativity. It seems that all the ones we created made some sense, and that all the features took part in optimizing the classification of "untagged" test protein names.

7. **Classifier performance** Our selection and tuning of classifiers was quite limited due to our pri-

mary focus on *automatic generation of training data* as well as *ensuring high quality of the negative examples.* An opportunity for further improvement is to try other classifiers, e.g. C5.0, the Maximum Entropy classifier or the Instance-Based classifier.

8. **Automatic Evaluation** The automatic evaluation used two approaches: train+test set (SVM, Majority and PSVM) and 10-fold cross validation (PSVM). They gave ok, though not incredibly accurate, results. Natural improvements are to incorporate more (available) domain knowledge and additional features. An interesting observation is that the Majority and SVM classifiers always gave the *same accuracy* when syntactic clues (e.g. HASFIRSTUPPER) were left out. This is probably because our naive features don't catch the essence of protein names and their context.

9. **Expert evaluation of untagged data** Even though our (part- of-) protein name classification accuracy is relatively high ($\approx 80-90\%$), the results for most of the classifiers provide low precision and recall. The most promising classifier, from a precision and recall (F-Score) perspective, is Support Vector Machines with a Polynomial Kernel. It gives relatively many true positives, that can later be used as input for more advances natural language parsing techniques, like the ones used in [15]. Precision and recall results also suffer from being very unbalanced (i.e. relatively few protein names compared to the number of untagged words). In order to improve precision and recall results (as well as accuracy) we probably also need to improve filtering of untagged words *before* they become candidates for being part of protein names.

   We also know for a fact that our selection of positive and negative examples using protein information databases leads to slightly biased training data, since many common biological words are part of protein names (not found in our English corpus). This may lead to ambiguities when processing the corpus since most of the occurrences of these biological words are *not* part of protein names. In further work we could potentially gain from adding the bootstrapping methods for handling disambiguation presented in [18]. Testing our approach on benchmark datasets, like GENIA, would be a natural next step.

10. **Post mortem Dataset characterization** *Why finding protein names at all? Aren't they all in the major protein information databases?* No, we found that approximately between 1 and 3 % of *all* words found in our gastrin abstract selection were protein names (by multiplying the two confidence interval boundaries presented). If these estimates are representative for other biomedical texts in Medline this means that this problem is rather large.

Acknowledgements

We would like to thank Waclaw Kusnierczyk and Tore Amble for continuous support. And finally a thanks to the Gsearch developers Jo Kristian Bergum, Hallgeir Bergum and Frode Jünge.

7. Conclusion and Future Work

This paper presents a novel method for automatically creating both positive and negative training data for protein name extraction classifiers. Since we focused on the automatization of creating training data and relevant negative examples, we only used relatively simple domain modeling and feature extraction/selection approaches. This leads to promising, though not yet highly accurate, empirical results. So in the next round we need additional work on i) feature extraction and selection, and ii) incorporating domain knowledge. The approaches presented in [10, 12] seems to be complementary to ours and might increase accuracy in future versions of ProtChew.

To sum up the contributions:

1. **fully automatic extraction of protein names**
2. **"tight" negative examples using existing protein information databases (served by Gsearch.org) in order to get a "sharp classifier"**

Opportunities for future work are:

- improved tokenization (splitting on space and punctuation characters is not good enough.)
- stemming (the Porter algorithm for English language gives mediocre results on biological terms.)
- improved detection of sentence boundaries might be used to get more accurate context boundaries for classification of terms (splitting on punctuation characters gives slightly erroneous results). One possibly approach could be to train classifiers for sentence boundary detection on the Brown Corpus, or better on the GENIA biomedical corpus.
- combine the presented approach with traditional search engines such as Google as an additional information source about protein names e.g. as a feature for the input classifier, [16].

- part-of-speech tagging of Protein names (the complete protein names, frequently combined of many words) and then use inductive algorithms in order to find common grammars of protein names can potentially increase accuracy on detecting complete protein names, and not only part of protein names as we have focused on this work. (We have found that protein names often look like small sentences themselves, [14]).
- strongly improve the evaluation of our approach by applying it on the BioCreative datasets (http://www.mitre.org/public/biocreative/).

References

[1] S. Bickel, U. Brefeld, L. Faulstich, J. Hakenberg, U. Leser, C. Plake, , and T. Scheffer. A Support Vector Machine classifier for gene name recognition. In *Proceedings of the EMBO Workshop: A Critical Assessment of Text Mining Methods in Molecular Biology*, March 2004.

[2] R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong. Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. *Journal Artificial Intelligence in Medicine: Special Issue on Summarization and Information Extraction from Medical Documents (Forthcoming)*, 2004.

[3] R. Bunescu, R. Ge, R. J. Kate, R. J. Mooney, Y. W. Wong, E. M. Marcotte, and A. K. Ramani. Learning to Extract Proteins and their Interactions from Medline Abstracts. In *Proceedings of the ICML-2003 Workshop on Machine Learning in Bioinformatics*, pages 46–53, August 2003.

[4] R. Bunescu, R. Ge, R. J. Mooney, E. Marcotte, and A. K. Ramani. Extracting Gene and Protein Names from Biomedical Abstracts. Unpublished Technical Note, Machine Learning Research Group, University of Texas at Austin, USA, March 2002.

[5] J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

[6] J. Demsar and B. Zupan. Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper, Faculty of Computer and Information Science, University of Ljubljana, 2004.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2nd edition, 2001.

[8] F. Ginter, J. Boberg, J. Jarvinen, and T. Salakoski. New Techniques for Disambiguation in Natural Language and Their Application to Biological Texts. *Journal of Machine Learning Research*, 5:605–621, June 2004.

[9] T.-K. Jenssen, A. Lægreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28(1):21–28, May 2001.

[10] S. Jiampojamarn. Biological term extraction using classification methods. Presentation at Dalhousie Natural Language Processing Meeting, June 2004.

[11] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter 11 - Making Large-scale SVM Learning Practical. MIT Press, 1999.

[12] S. Mukherjea, L. V. Subramaniam, G. Chanda, S. Sankararaman, R. Kothari, V. Batra, D. Bhardwaj, and B. Srivastava. Enhancing a biomedical information extraction system with dictionary mining and context disambiguation. *IBM Journal of Research and Development*, 48(5/6):693–701, September/November 2004.

[13] M. Narayanaswamy, K. Ravikumar, and K. Vijay-Shanker. A biological named entity recognizer. In *Proceedings of the Pacific Symposium on Biocomputing 2003*, pages 427–438, 2003.

[14] R. Sætre. GeneTUC, A Biolinguistic Project. Master's thesis, Norwegian University of Science and Technology, Norway, June 2002.

[15] R. Sætre. Natural Language Processing of Gene Information. Master's thesis, Norwegian University of Science and Technology, Norway and CIS/LMU Munchen, Germany, April 2003.

[16] R. Sætre, A. Tveit, T. S. Steigedal, and A. Lægreid. Semantic Annotation of Biomedical Literature using Google. In *Proceedings of the International Workshop on Data Mining and Bioinformatics (DMBIO2005)*, Lecture Notes in Computer Science (LNCS) (Forthcoming), Singapore, May 2005. Springer-Verlag Heidelberg.

[17] H. Shatkay and R. Feldman. Mining the Biomedical Literature in the Genomic Era: An Overview. *Journal of Computational Biology*, 10(6):821–855, 2003.

[18] M. Torii and K. Vijay-Shanker. Using Unlabeled MEDLINE Abstracts for Biological Named Entity Classification. In *Proceedings of the 13th Conference on Genome Informatics*, pages 567–568, 2002.

[19] A. Tveit, M. L. Hetland, and H. Engum. Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients. In *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2003)*, volume 2737 of *Lecture Notes in Computer Science (LNCS)*, pages 422–429. Springer-Verlag, 2003.

[20] L. Wong. Gaps in Text-based Knowledge Discovery for Biology. *Drug Discovery Today*, 7(17):897–898, September 2002.

[21] H. Yu, V. Hatzivassiloglou, C. Friedman, A. Rzhetsky, and W. J. Wilbur. Automatic Extraction of Gene and Protein Synonyms from MEDLINE and Journal Articles. In *Proceedings of the AMIA Symposium 2002*, pages 919–923, 2002.