Data and text mining

Combination of text-mining algorithms increases the performance

Rainer Malik[1,*], Lude Franke[2] and Arno Siebes[1]

[1]Universiteit Utrecht, Department of Information and Computing Sciences, Padualaan 14, 3584CH Utrecht,
The Netherlands and [2]Complex Genetics Section, Department of Medical Genetics, UMC Utrecht,
The Netherlands

ABSTRACT

Motivation: Recently, several information extraction systems have been developed to retrieve relevant information out of biomedical text. However, these methods represent individual efforts. In this paper, we show that by combining different algorithms and their outcome, the results improve significantly. For this reason, CONAN has been created, a system which combines different programs and their outcome. Its methods include tagging of gene/protein names, finding interaction and mutation data, tagging of biological concepts and linking to MeSH and Gene Ontology terms.

Results: In this paper, we will present data that show that combining different text-mining algorithms significantly improves the results. Not only is CONAN a full-scale approach that will ultimately cover all of PubMed/MEDLINE, we also show that this universality has no effect on quality: our system performs as well as or better than existing systems.

Availability: The LDD corpus presented is available by request to the author. The system will be available shortly. For information and updates on CONAN please visit http://www.cs.uu.nl/people/rainer/conan.html

Contact: rainer@cs.uu.nl

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Over the last decades, the use of large-scale experimental techniques has led to an increased pace at which scientific information is produced. Hence, also the biomedical literature presenting this information is growing at the same rate. This often quoted fact (Rebholz-Schuhmann *et al.*, 2005) poses a big problem for every experimentalist. Interesting and useful information, like interaction and mutation data, could appear in papers they have not read. Therefore, important facts might get overlooked and the scientific work might be affected.

Recently, several methods for extracting information out of biomedical text have been developed to solve this problem. These methods range from biological named entity recognition (NEK) (Chang *et al.*, 2004; Tanabe and Wilbur, 2002; Mika and Rost, 2004a) to natural language processing (NLP) (MacCallum

et al., 2000) to information extraction and text mining (Muller *et al.*, 2004) to finally protein interactions and relations (Donaldson *et al.*, 2003; Hoffmann and Valencia, 2005). An extensive overview over those system can be found in Krallinger and Valencia (2005).

Many of those systems, however, focus on a specific aspect of the literature, on a limited test set or on a single organism, like *Drosophila* or Yeast. Although those implementations achieve good results, it is not what experimentalists need in their work. They would need a complete solution, giving them cross-references over multiple species and providing them with complete datasets.

Another problem is that some of abovementioned algorithms and implementations have their strengths and weaknesses in specific areas of text mining, e.g. one algorithm yields a very high Precision, but delivers a very bad Recall and vice versa. This means that experimentalists do not know whether the data they are using is reliable or not and whether the data is complete or not. There might be many false positives included in the results or many false negatives might have been left out.

The first problem can be solved by including as much data as possible in the process. By implementing a program in a fast and reliable way, using high-quality input files and storing the results of the program in a way that they are easy and quickly accessible, it is ensured that all available PubMed/MEDLINE abstracts can be processed, which is a prerequisite to deliver an exhaustive system. The user should not only be provided with singular data (e.g. only protein names), but with as much information as possible, namely protein names, protein–protein interactions and mutations. However, it should not overload the user with information, but provide the exact information the user needs at a specific point of time.

It is well known that combinations of classifiers perform much better than the single classifiers themselves. The second problem can thus be solved by combining the best algorithms in the scientific world. Different outcomes of different systems should be compared, reducing the amount of false-positives and false-negatives as much as possible.

This paper is concerned with providing a solution for the second problem. We combine different algorithms and outcomes of different methods to produce high-quality data. We want to show that the combination of different algorithms improves the performance significantly.

For this reason we developed CONAN (Malik and Siebes, 2005), a system which offers a wide range of information. A big advantage

---

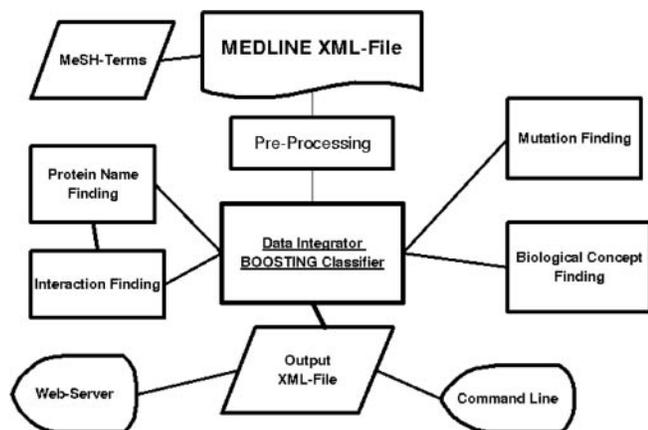*To whom correspondence should be addressed.

Fig. 1. Flow diagram of CONAN.

of our system is that this information is combined to construct new information, e.g. the output of a protein name tagging method is used as input for a method which finds Protein–Protein-Interaction Data and as input to find protein synonyms which is described in Section 2.3. The filtering of the outcome of the protein tagging methods (as described in Section 2.2) is a way to improve the precision significantly. By using more than one protein tagging method, the recall is improved. This results in a very high *F*-measure of our system. We find results which yield both a high recall and a high precision, resulting in accurate and precise data. In addition, the system is stable and quick, extracting as much information as possible out of abstracts. We find that by combining different methods, we can overcome difficulties experienced by other methods by integrating the best available algorithms and filtering the results.

We also want to show that our method can provide the scientific community with complete data, over all organisms and over all available literature data at the moment. This makes our system the most complete literature mining utility so far.

We noticed that there is a big need in the experimentalist community to get important literature information quickly and easily. This is also warranted by our method, which can be accessed by command-line or via a web-service.

The road-map to this paper is as follows: In the next section, we will briefly describe CONAN and its components. In Section 3, we show the evaluation of the system and we also show that by combining different methods, the performance is improved. In Section 4, we draw conclusions and give future directions.

2 SYSTEM AND METHODS

For an extensive description of CONAN, please refer to out previous publication (Malik and Siebes, 2005). In this section, we want to give a brief overview on how CONAN is implemented and the different building blocks that were used to construct CONAN (Fig. 1).

2.1 Input

The basis input for CONAN are the MEDLINE files released by the National Library of Medicine. Via a licensing system, users get the opportunity to download MEDLINE files or get the files wanted on tape. These files are provided in the MEDLINE XML format. For CONAN, only abstracts are processed which are in their final version to rule out multiple processing of the same file. Another necessity is that the abstracts are in English. Additional to the MEDLINE files, also other sources of input are used for different purposes.

2.1.1 Protein tagging For Protein tagging and getting more information about proteins mentioned in text, we have several sources of input.

First, the three different protein tagging methods (Section 2.2) are used to extract protein names out of the abstract text. They use the MEDLINE abstract as input.

Another source of information is the Gene Ontology (GO) Database (Ashburner *et al.*, 2000) and, more specifically, the current annotation of Uniprot terms by GO (GOA) (Camon *et al.*, 2004). The concept of GO is to develop three structured ontologies, namely Biological Processes, Cellular Components and Molecular Functions. Protein names found in text have a certain synonym in UniProt (Apweiler *et al.*, 2004). These synonyms again are annotated by the EBI, assigning GO-terms to Uniprot-terms. In our system, we use Gene Ontology Identifiers to give more information about proteins mentioned in the text and to give the user an overview to which processes, components and functions these proteins belong.

The final source of information is the ENSEMBL database. For every UniProt-term which is discovered while processing the files, we search for the corresponding ENSEMBL identifier. ENSEMBL (Birney *et al.*, 2004) is, as is UniProt, often used to link data together and this improves the system with additional function cross-references.

In filtering via the Boosting algorithm (Section 2.2), an abbreviation list compiled by iProLINK (Hu *et al.*, 2004) and a list of protein names and symbols from Swiss-Prot/TrEMBL (Boeckmann *et al.*, 2003) was used as an input for training the classifier. These two lists provide us with enough data to make the filtering procedure fast and efficient. The classifier itself takes a list of protein names as an input.

2.1.2 Interaction finding For interaction finding, the primary source of input is a list of protein names compiled by the three protein tagging methods (see below). In addition, a slightly changed set of regular expressions used in the PreBIND and BIND system are utilized by our system. This provides the user with complete and precise information about interactions mentioned in the text.

2.1.3 Mutation finding In the finding of mutations mentioned in text, we also use a set of regular expressions applied to the abstracts. These patterns usually start with one amino acid in one- or three-letter-code, followed by a number and another amino acid abbreviation. This extracts all available mutational information, which is further filtered in the data integration step.

2.1.4 Keywords For keyword searching, we use the UMLS Metathesaurus (Bodenreider, 2004) to retrieve lists of biologically relevant terms. These terms were split up into different databases, specified by their so-called semantic type. The BLAST-searching method is applied to find important keywords in text.

2.1.5 MeSH In addition, MeSH (Medical Subject Headings)-terms are needed as an input. MeSH (http://www.nlm.nih.gov/mesh) is the National Library of Medicine's controlled vocabulary thesaurus. It consists of sets of terms naming descriptors in a hierarchical structure that permits searching at various levels of specificity. MeSH terms are part of NCBI's MEDLINE distribution, and so are part of the before-mentioned MEDLINE files. The MeSH terms are used (in combination with the biologically relevant keywords) to further annotate the abstract.

2.2 Algorithms for text mining

Different Algorithms have been implemented in CONAN using the input described in the previous section. For an extensive description of those

methods and how they are implemented, please refer to our previous publication (Malik and Siebes, 2005) and to the original publications of the methods.

BLAST-searching. This method uses the BLAST-algorithm (Altschul *et al.*, 1990) to discover relevant biological information in text. In the original method (Krauthammer *et al.*, 2000), only gene and protein names were used. We extended the program to identify biologically relevant terms and concepts as well as protein and gene names.

AbGene. Gene/Protein-tagging method called AbGene first published by Tanabe(Tanabe and Wilbur, 2002). It uses a combination of statistical and knowledge-based strategies.

NLProt. Another method (Mika and Rost, 20004b, a) to find protein names and information about these proteins in the text comes. It automatically extracts protein names from the literature and links those to associated entries in a sequence database.

MuText. Derived from a publication by Horn *et al.* (2004). Takes several regular expressions to detect mutations mentioned in an abstract.

Interaction finding. The basis of this method are regular expressions as used in the PreBIND and BIND system (Donaldson *et al.*, 2003; Bader *et al.*, 2003). Some regular expressions have been deleted by us from the set owing to redundant results

Boosting Classifier. As an extra classifier, Boosting was used to check the outcome of the protein tagging methods. Boosting is a machine learning meta-algorithm for performing supervised learning.

As all other methods have been earlier described in Malik and Siebes (2005), but the boosting classifier has not been described earlier, we will give an overview how boosting works and how the classifier was constructed.

Boosting occurs in stages, by incrementally adding to the current learned function. At every stage, a weak learner (i.e. one that has an accuracy greater than chance) is trained with the data. The output of the weak learner is then added to the learned function, with some strength (proportional to how accurate the weak learner is). Then, the data are re-weighted: examples that the current learned function get wrong are 'boosted' in importance, so that future weak learners will attempt to fix the errors.

A boosting implementation called BoosTexter (Schapire and Singer, 2000) was used that is designed to work with text. BoosTexter is an extension of the AdaBoost Algorithm (Schapire, 2002). It uses the so-called *n*-grams to classify a given document or set of words. This classification of words or phrases was used in our approach. In our case, we trained the classifier on a large dictionary containing protein names derived from Swiss-Prot/TrEMBL (Boeckmann *et al.*, 2003) as positive cases and a dictionary consisting of protein name abbreviations which is published by iProLINK (Hu *et al.*, 2004). The training data are weighted by checking how often a certain phrase is mentioned in the dictionary. For an overview of the classifier please refer to Figure 2.

In general, each instance is broken into multiple fields. For textual input fields, the test asks whether or not a particular sequence of words is present in the given text. The form of this sequence may be a simple sequence or 'ngram' if in ngram or fgram mode, or a 'sparse ngram' (sgram). An example of a sparse ngram is the pattern '∗ receptor' which matches any two word sequence beginning with any word and ending with the word 'receptor'. In our training, we trained the classifier with sgrams of the size one, two, three and five. This was done because the protein dictionary contains many phrases which are three or five words long (e.g. atrial natriuretic peptide). With this, it is ensured that short protein names like 'b-catenin' are recognized as well as normal protein names like 'beta catenin' and long protein names like 'beta catenin like protein 1'. The positive list of protein names is then passed on to the interaction finding method. An evaluation of this classifier can be found in Section 3.7.

It is unclear what a 'negative example' for protein detection should look like, given that Protein Names such as the *Drosophila* 'dachshund' gene exist. Therefore we only used positive examples and a minimal threshold on the weights.
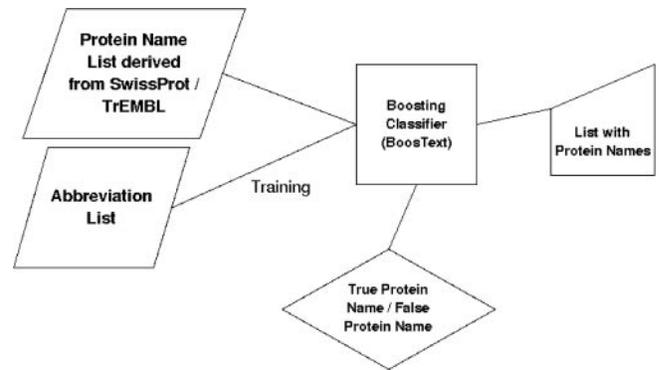

Fig. 2. Overview of the boosting classifier.

2.3 Data integrator

The most important part of CONAN is the data integration step. There are three major stages in which CONAN joins data.

- First, combination of data is used storing it in the Output-XML-file. This is done in the interaction-finding method. The protein names found by the two protein-tagging methods are used as input for the interaction-finding method. The list of protein names is passed on to the regular expressions used to find interactions. This ensures a high precision of interaction finding, because only protein names which are found before are taken into account.

- Second, a case of this integration of data is the validation of those protein names found by NLProt. NLProt offers a reliability score (ranging from 0 to 1) for every protein name found in the text. By comparing those lists of protein names with the lists produced by the other methods, namely AbGene and the BLAST-searching, we ensure that as little as possible false positives are parsed on to the interaction-finding method. This is ensured by the fact that the score is increased by 0.5 if the protein name is found by the other methods as well or decreased by -0.5 if the protein name is not found by the other methods. Those lists are comparable, because they also contain the position of the protein name in the abstract.

- Third, data are combined is at the time when the Output-XML-file is queried. One example is the querying of the Mutation Data. Mutations are found in text by simple regular expressions, the pattern usually starting with one amino acid in one- or three-letter-code, followed by a number and another amino acid abbreviation. This method alone would result in a large number of false-positives, because many other biological concepts like Cell Lines share those abbreviations. To solve this problem, the BLAST-searching method is queried, in order to look for 'mutation-related' terms in text.

 Terms as 'Mutation', 'Mutational Analysis' or 'Amino Acid Substitution' are examples for those kind of keywords. If one if those keywords is found by the BLAST-searching method or is found in the MeSH-terms, then the chance of the mutation being a true positive is high. The BLAST-searching method also gives the possibility to compare the positions of the mutation and the mutational term in the abstract. More closer the two are found together, the higher is the chance of the mutation being a true positive. In fact, we compute the distance of the mutation and the mutational term. We assume, that an average sentence in an abstract is 10–20 words long. If the distance is not longer than 30 words, it is considered as a true positive.

2.4 Output

The data available consist of information per abstract. Gene and Protein Names are sought after in MEDLINE abstracts. Protein names

are linked to UniProt-synonyms and ENSEMBL synonyms. Via a BLAST-search-method, different biomedical concepts are extracted from abstracts. Protein names are used to find protein–protein-interaction data in those abstracts. protein-mutation-data is also extracted and stored via regular-expressions.

The basic output of CONAN is an XML file which holds all information about a certain abstract. XML was chosen because it offers several benefits in regard to storing data, a major advantage being its platform-independency.

There is also a web-server which displays the CONAN output in an user-friendly manner. This web-server will be released shortly and will feature many different search methods as well as graphical visualization of the results.

3 EVALUATION

In this section, we will present the evaluation of CONAN, tested on three different corpora and the evaluation of the boosting classifier (Section 2.2). We also show the experimental setup.

The corpora chosen for the evaluation represent collections which are commonly spread throughout the text mining community. The YAPEX corpus used in this evaluation is used by many protein name-tagging methods and thus provides a good opportunity to compare the results with other groups. The LLL-challenge corpus is the only protein–protein-interaction corpus available at the moment. For this reason, the LDD Corpus we present in this paper and we want to share with the scientific community, is available upon request to the author.

3.1 Partial matches

When analyzing abstracts and evaluating the results, one of the biggest problems are partial matches found. Protein tagging methods often retrieve partial matches like CD4 instead of CD4 kinase or protein kinase in place of the correct protein kinase c. This poses a big problem because people evaluating text-mining methods do not seem to have a common strategy. We evaluated our method with both partial and exact matches (Section 3.4).

3.2 Evaluation measures

Different measures are applied to show the efficiency of a method. The most important measures are precision, recall and the *F*-measure. Precision is defined as the percentage of retrieved documents that are relevant. In our case, when evaluating protein-name-tagging and interactions, it is defined as the percentage of protein names (interactions) retrieved that are relevant. Recall is defined as the percentage of relevant documents that are retrieved. Again, in our case, it is defined as the percentage of relevant protein names (interactions) that are retrieved.

The *F*-measure combines precision and recall into one measure, being the harmonic mean. It is computed as

$$F = \frac{2*(\text{recall*precision})}{(\text{recall} + \text{precision})} \quad (1)$$

One important property of the *F*-measure is the fact that it goes up with precision and/or recall only when the other measure goes up as well, thus giving low scores to design options that trivially obtain high precision by sacrificing recall or vice versa.

3.3 LLL challenge corpus

The LLL-challenge-corpus, named after the LLL-challenge which took place in course of the Learning Language in Logic Workshop (LLL05), focuses on the evaluation of interaction-extraction from text. It consists of two parts: the training-dataset consisting of 57 sentences derived from MEDLINE-files and the test-dataset consisting of 87 sentences. Both sets contain only *Bacillus subtilis* protein names and interactions, focusing specifically on transcription in the bacterium. *B.subtilis* was chosen because it is a model bacterium and because transcription is both a central phenomenon in functional genomics involved in gene interaction and a popular IE problem. There are two types of sets, the so-called 'basic data set', including sentences, word segmentation and biological target information, and the 'enriched dataset' which also includes lemmas and syntactic dependencies. We focused on the 'basic data set'. We applied CONAN to the 86 sentences of the test-dataset sets, calculated the interactions and compared the results with the outcome of other groups which took part in the challenge. The results were automatically computed by a score-computation program provided by the workshop organizers. Hakenberg *et al.* (2005) report, that they were able to get an *F*-measure of 51% for the whole set. Other groups (Nedellec, 2005; Katrenko *et al.*, 2005) report an *F*-measure between 35 and 42%. Most other groups applied a learning strategy which learns patterns from the training data and applies these patterns to the test data.

With CONAN, we achieved a precision of 53% and a recall of 52%, resulting in an *F*-measure of 52.5%. So we show that even without learning patterns for interactions, our method can score at least as high as other groups, while being easier to use.

However, it has to be said that the results of CONAN might not be directly comparable with those of the other groups, because these other groups might have had a smaller training set.

3.4 YAPEX corpus

The YAPEX-corpus (Franzen *et al.*, 2002) consists of 101 MEDLINE abstracts annotated by domain experts connected to the YAPEX project. The corpus is divided into two parts, the first part being the result of a PubMed query, consisting of 48 abstracts, the second part being a randomly chosen subset of 53 abstracts of the GENIA corpus. The YAPEX corpus focuses on the extracting of protein names out of text. It contains 1890 annotated protein names.

We processed the YAPEX corpus with CONAN, tagging protein names with both NLProt and Abgene and compared the results with the annotated YAPEX-corpus. For the Sloppy Results (or partial matches, see Section 3.1), we achieved a precision of 88.91% and a recall of 85.33%. This results in an *F*-measure of 87.08%. In the original YAPEX-publication, an *F*-measure of 82.9% is reached. Mika *et al.* report in their original paper about NLProt (Mika and Rost, 2004a) that they achieved an *F*-measure of 85% with their method.

For the Strict Results (or full matches, see Section 3.1), we achieved a precision of 79.1% and a recall of 77.2%. This results in a *F*-measure of 78.1%. In the original YAPEX-publication, an *F*-measure of 67.1% is reached. Mika *et al.* report in their original paper about NLProt (Mika and Rost, 2004a) that they achieved an *F*-measure of 75% with their method.

For the evaluation with classifier, we compiled the output of CONAN for the YAPEX-corpus. This was the input for the Classifier (Section 2.2). The Classifier looked for false positives in the list. The Classifier identified 18 false positives correctly, but labeled also eight true positives as false positives (e.g. 'NOSIP'). The classifier improved the results for both the SLOPPY and the STRICT evaluation, resulting in *F*-measures of 87.4 and 78.2, respectively.

3.5 LDD Corpus

In order to further analyze interaction data as well as protein-name finding, the LDD corpus of 1768 abstracts has been created, all of these containing one or more interactions. A high percentage of MEDLINE abstracts do not contain any interactions at all. To bypass this problem, we selected these 1768 abstracts, a combination of available lists of PMIDs from BIND (Bader *et al*., 2003) and DIP (Xenarios *et al*., 2002). This gives us the guarantee that at least one interaction is in the abstract, which also implies that at least two different protein names are mentioned in the abstract.

Because this set is not or only partially annotated, there was a need to annotate these abstracts by hand. Totally 100 of those 1768 abstracts have been selected completely at random by us to ensure that no organism or protein family is overrepresented and those interactions have been manually annotated. A total of 100 abstracts might seem quite a small number, but manually annotating 1000 abstracts would have been a too-big workload. Also, evaluating the results of our system on 1768 abstracts would have taken quite a substantial amount of time. The processing of those 100 abstracts by CONAN takes only 10 min, tagging protein names and extracting interactions for further evaluation.

This LDD corpus is available on request to the author. The annotation guideline and the inter-annotator agreement scores can be seen in the Supplementary Material. Two annotators with biological background knowledge were presented with the data. Their task was to identify protein/gene-names occurring in the LDD Corpus file. They achieve a *kappa* score of 0.8460 and an *F*-measure of 0.8862 for the inter-annotator agreement. For the tagging of interaction data, again two annotators were presented with the data. They agreed on 97.7% of the interactions. So, the LDD corpus is well-annotated and will become a valuable resource to the community. For an in-depth description, please refer to the Supplementary Material.

3.5.1 Protein tagging When analyzing the different protein-name-tagging methods, we see that the data integrator step boosts the performance of those methods. We used the LDD set of 100 manually annotated articles (see above). The original NLProt method shows a precision of 75% and a recall of 76%, the protein/gene-tagging method shows a correctness of 77%. When integrating the protein-tagging-data with the data found by the BLAST-search and the gene/protein-tagging method, we see an increase of precision to 80.9% and recall to 85%, when evaluating protein names. This results in an *F*-measure of 82.90%

This LDD dataset, however, has a small bias towards yeast-related articles. As also reported in Mika and Rost (2004b), text mining methods usually perform better on yeast-related articles than on other organisms, because protein-naming is much simpler in yeast than it is in other organisms.

This good result also shows in the evaluation of interactions (see next Section), because the interaction finding is highly dependent on good protein-name finding.

3.5.2 Interactions Finally, the protein–protein interactions were evaluated. It is important to say at this point that our method does discriminate between positive interaction and negative interactions (inhibitions). In this evaluation, we consider both positive and negative interactions as true positives.

In the 100 manually annotated abstracts, a total of 427 interactions are documented. Those 427 interactions were manually annotated. CONAN found 477 interactions in total, compared with the 427 interactions which were annotated manually in the abstracts, this yields a number of 50 or more false positives. Analyzing those abstracts achieved a precision of 81.55% (389/477) and a recall of 91.10% (389/427).

Here we see that, using our system, we get very good results, detecting almost all available interactions mentioned in the abstracts.

It also has to be said that the missed interactions were missed because of the performance of the protein-tagging methods. As stated before, the lists produced by these methods are passed on to the interaction-finding method. If these lists do not contain certain protein names, then of course also no interactions containing these proteins can be found.

3.6 Evaluation examples

In this section, we want to present some examples of the types of data extracted by our system.

Regarding interaction data, a good example is the abstract with PMID 10531067. The interactions found in this abstract are Nur77–MEF2, MEF2–Cabin1, MEF2–calmodulin and Cabin1–calmodulin.

The abstract itself confirms all those interactions. The biological interplay between Cabin1, MEF2 and calmodulin defines a distinct signaling pathway from the TCR to the Nur77 promoter during T-cell apoptosis.

One example for the mutation extraction is the abstract with PMID 10799524. The mutations found in this abstract are R235K and R238K, which impair the BNIP-2 GAP activity without affecting its binding to Cdc42. It is important to see that in this abstract, the sentence 'Site-directed mutagenesis confirmed that an R235K or R238K mutation....' plays an important role. Both 'keywords' mutagenesis and mutation signify clearly that those two terms refer to an actual point mutation. Because the term 'mutagenesis' is detected by the BLAST-searching method, we get a true-positive hit for a mutation in this case. If no such term would have been found by the BLAST-method, we would have considered this hit to be a false-positive.

3.7 Boosting evaluation

We tested the Boosting classifier on a test-set, containing 5113 protein names. This test-set was a MEDLINE-input file which was processed by CONAN. Those 5113 protein names were sent to the classifier which labeled 4917 protein names correctly (96.17%) and 196 wrongly (3.83%). The classifier was able to filter out 120 false-positives (e.g. motheaten, desmosterolosis).

It is important to note which percentage of protein names which were put into a different class by the Boosting classifier, are in the

dictionary. This gives a rough estimate, how good the classifier performs on 'unseen' examples. There are two classes of 'unseens', namely 'partial unseens' and 'full unseens'. A protein name is 'partial unseen' if parts of the protein name, but not the full name, appear in the dictionary. A 'full unseen' appears if no parts of the protein name are listed in the dictionary.

A second round of evaluation was performed on the YAPEX Corpus. In the YAPEX corpus, a total of 353 Protein names were 'unseen'. From those 353 protein names, 44 were classified as 'false' and 309 were classified as 'correct'. From the 44 unseen examples which were classified as false, 36 names are true negatives (e.g. 'chaperonins') and 8 are false negatives (e.g. 'SWUV39H1 HMTase', 'NOSIP').

From the 309 Protein names which were classified as 'correct', 244 were 'partial unseens' and 65 were 'full unseens'. From those 309 Protein names, 25 protein names are false positives (e.g. 'signal transducers and activators of transcription' or 'endothelial adhesion molecules'). From those 25 false positives, 8 were 'full unseens' and 17 were 'partial unseens'.

From the 44 unseen examples which were classified as false, only two are 'partial unseen', the other 42 are 'full unseen'.

So we show that when combining different methods for the tagging of protein names, the results significantly improve. For partial protein names, the score is better than for full protein names. The classifier not only works well on names which appear in the training set ("seen" names), but also on those which do not appear in the training dictionary ('unseen' names).

3.8 Comparison to other systems

iHOP (Hoffmann and Valencia, 2004, 2005) is one of the few systems which attempt to display information of biological interest extracted from abstracts. The emphasis of the system lies on protein–protein interactions, constructing networks of these interaction and gene synonym identification, all over multiple species. To compare two complete systems (CONAN and iHOP) with each other is not a simple task. Unfortunately, Hoffmann *et al*. state in their publication that owing to a lack of other 'complete systems', only a partial comparison is possible. Because of this lack of complete systems, only parts of their evaluation is presented in the Supplementary Material. They state that in their evaluation, the *F*-measure ranged between 70 and 91%, dependent on the organism. When comparing these results to our evaluation (see Tables 1 and Table 2), we see that the numbers are similar. In the Supplementary Material it becomes clear that the best results are obtained for *Saccharomyces cerevisiae* and the worst results are obtained for *Drosophila melanogaster*. This is attributed to the simple reason that the nomenclature of yeast genes/proteins is much simpler (e.g. SSK1, YPD1) than for *Drosophila* gene/proteins, which often resemble normal English words (e.g. dachshund, kruppel). This observation is also shared by other publications in this field (Mika and Rost, 2004b). Although an evaluation per species was not carried out, we also observed a similar behaviour in the LDD Corpus and in the YAPEX corpus. Unfortunately, most corpora available at the moment have a slight bias towards Yeast-related articles, which also slightly biases the evaluation. In conclusion, we see that our recall/precision/*F*-measure values are at least in the same range of those of iHOP, suggesting that the two systems yield a similar performance. As CONAN is quite different from iHOP

Table 1. Overview of results on YAPEX-Corpus-SLOPPY

Group	Precision (%)	Recall (%)	*F*-measure (%)
NLProt	—	—	85
GAPSCORE	81.5	83.3	82.5
YAPEX-method(SLOPPY)	83.3	82.1	82.9
KeX	82.1	83.5	82.8
CONAN without Boosting	88.92	85.33	87.08
CONAN with Boosting	89.65	85.26	87.40

Table 2. Overview of results on YAPEX-Corpus-STRICT

Group	Precision (%)	Recall (%)	*F*-measure (%)
NLProt	—	—	75
GAPSCORE	56.7	58.5	57.6
YAPEX-method(STRICT)	67.8	66.4	67.1
KeX	40.4	41.1	40.7
CONAN without Boosting	79.1	77.2	78.1
CONAN with Boosting	79.6	76.9	78.2

regarding the information displayed, a complete comparison seems almost impossible.

4 DISCUSSION

The main point of this paper is to show that combining different algorithms and applying the right filters can boost the performance of text mining. We did this by combining the output of three different protein-name tagging algorithms and filter the output by a boosting technique. We also show that our precision and recall percentages are better when compared to the methods themselves. For protein tagging, we obtain results which are 2–5% better in regard to precision, recall and *F*-measure than the results in the original publications or comparable methods.

For the interaction tagging, we perform 1% better than the best available method and 7% better in regard to the *F*-measure than all other groups, but without having to learn patterns from training data for each set.

Another goal is to address, as completely as possible, the problem of experimentalists to find certain information 'hidden' in the abstracts of biomedical literature. For this reason we include additional information, like mutation data in our results. So we see that our system not only performs better than other methods, it also is a more 'overall' approach to text mining and applications in the experimental field.

We show that applying multiple methods does not compromise the speed of the system. By providing accurate, up-to-date data for experimentalists, we show that a extensive approach towards the literature problem can help biologists to get the information they want in a fast an easy-to-use way.

Future directions include the processing of entire MEDLINE, the further development of the web-server to fit the biologists needs and better graphical visualization of the results.

ACKNOWLEDGEMENTS

Conflict of Interest: none declared.

REFERENCES

Altschul,S. *et al.* (1990) Basic local alignment search tool. *J Mol. Biol.*, **215**, 403–410.

Apweiler,R. *et al.* (2004) UniProt: the Universal Protein knowledgebase. *Nucleic Acids Res.*, **32**, D115–D119.

Ashburner,M. *et al.* (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, **25**, 25–29.

Bader,G. *et al.* (2003) BIND: the biomolecular interaction network database. *Nucleic Acids Res.*, **31**, 248–250.

Birney,E. *et al.* (2004) An Overview of Ensembl. *Genome Res.*, **14**, 925–928.

Bodenreider,O. (2004) The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, **32**, 267–270.

Boeckmann,B. *et al.* (2003) The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, **31**, 365–370.

Camon,E. *et al.* (2004) The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.*, **32**, 262–266.

Chang,J. *et al.* (2004) GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics*, **20**, 216–225.

Donaldson,I. *et al.* (2003) PreBIND and Textomy–mining the biomedical literature for protein–protein interactions using a support vector machine. *BMC Bioinformatics*, **4**, 11.

Franzen,K. *et al.* (2002) Protein names and how to find them. *Int. J. Med. Inf.*, **67**.

Hakenberg,J., Plake,C., Leser,U., Kirsch,H. and Rebholz-Schuhmann,D. (2005) Lll'05 challenge: genic interaction extraction with alignments and finite state automata. In *Learning Language in Logic Workshop (LLL'05) at ICML 2005*.

Hoffmann,R. and Valencia,A. (2004) A gene network for navigating the literature. *Nat. Genet.*, **36**, 664.

Hoffmann,R. and Valencia,A. (2005) Implementing the iHOP concept for navigation of biomedical literature. *Bioinformatics*, **21** (Suppl. 2), ii252–ii258.

Horn,F. *et al.* (2004) Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics*, **20**, 557–568.

Hu,Z. *et al.* (2004) iProLINK: an integrated protein resource for literature mining. *Comput. Biol. Chem.*, **28** (5-6), 409–416.

Katrenko,S., Marshall,M., Roos,M. and Adriaans,P. (2005) Learning biological interactions from medline abstracts. In *Learning Language in Logic Workshop (LLL'05) at ICML 2005*.

Krallinger,M. and Valencia,A. (2005) Text-mining and information-retrieval services for molecular biology. *Genome Biol.*, **6**, 224.

Krauthammer,M. *et al.* (2000) Using BLAST for identifying gene and protein names in journal articles. *Gene*, **259**.

MacCallum,R. *et al.* (2000) SAWTED: structure assignment with text description–enhanced detection of remote homologues with automated Swiss-Prot annotation comparisons. *Bioinformatics*, **16**, 125–129.

Malik,R. and Siebes,A. (2005) Conan: An integrative system for biomedical literature mining. In Bento,C., Cardoso,A. and Dias,G. (eds), *LNAI 3808, EPIA05*, pp. 248–259.

Mika,S. and Rost,B. (2004a) NLProt: extracting protein names and sequences from papers. *Nucleic Acids Res.*, **32**.

Mika,S. and Rost,B. (2004b) Protein names precisely peeled off free text. *Bioinformatics*, **20** (Suppl 1), I241–I247.

Muller,H. *et al.* (2004) Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biol.*, **2**, e309.

Nedellec,C. (2005) Learning language in logic—genic interaction extraction challenge. In *Learning Language in Logic Workshop (LLL'05) at ICML 2005*.

Rebholz-Schuhmann,D. *et al.* (2005) Facts from text—is text mining ready to deliver? *PLoS Biol.*, **3**, e65.

Schapire,R. (2002) The boosting approach to machine learning: an overview. In *MSRI Workshop on Nonlinear Estimation and Classification*.

Schapire,R. and Singer,Y. (2000) BoosTexter: a boosting-based system for text categorization. *Mach. Learn.*, **39**, 153–168.

Tanabe,L. and Wilbur,W. (2002) Tagging gene and protein names in biomedical text. *Bioinformatics*, **18**, 1124–1132.

Xenarios,I. *et al.* (2002) DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, **30**, 303–305.