BIOINFORMATICS

Vol. 20 Suppl. 1 2004, pages i241–i247
DOI: 10.1093/bioinformatics/bth904

Protein names precisely peeled off free text

Sven Mika[1,2,3,*] and Burkhard Rost[1,2]

[1]CUBIC, Department of Biochemistry and Molecular Biophysics, Columbia University, 650 West 168th Street BB217, New York, NY 10032, USA, [2]Columbia University Center for Computational Biology and Bioinformatics (C2B2), Russ Berrie Pavilion, 1150 St. Nicholas Avenue, New York, NY 10032, USA and [3]Institute of Physical Biochemistry, University Witten/Herdecke, Stockumer Strasse 10, 58448 Witten, Germany

Received on January 15, 2004; accepted on March 1, 2004

ABSTRACT

Motivation: Automatically identifying protein names from the scientific literature is a pre-requisite for the increasing demand in data-mining this wealth of information. Existing approaches are based on dictionaries, rules and machine-learning. Here, we introduced a novel system that combines a pre-processing dictionary- and rule-based filtering step with several separately trained support vector machines (SVMs) to identify protein names in the MEDLINE abstracts.

Results: Our new tagging-system NLProt is capable of extracting protein names with a precision (accuracy) of 75% at a recall (coverage) of 76% after training on a corpus, which was used before by other groups and contains 200 annotated abstracts. For our estimate of sustained performance, we considered partially identified names as false positives. One important issue frequently ignored in the literature is the redundancy in evaluation sets. We suggested some guidelines for removing overly inadequate overlaps between training and testing sets. Applying these new guidelines, our program appeared to significantly out-perform other methods tagging protein names. NLProt was so successful due to the SVM-building blocks that succeeded in utilizing the local context of protein names in the scientific literature. We challenge that our system may constitute the most general and precise method for tagging protein names.

Availability: http://cubic.bioc.columbia.edu/services/nlprot/

Contact: mika@cubic.bioc.columbia.edu

1 INTRODUCTION

Problems in extracting protein names The automatic recognition of named entities is a key element for mining free text. Removing terms such as the names of people, countries or cities from a newspaper article may render the text useless. The same is true for protein names in the biomedical literature. Yet, compared with other types of named entities, the following reasons make it disproportionately difficult to precisely extract protein names from natural language text. Standards and nomenclature-rules as existing for organic compounds are missing for the vast majority of all proteins. To make things worse, protein names are often derived from descriptive terms (signal transducer and activator of transcription, STAT) and only later become accepted by the research community through repetition (STAT-4). Protein names also overlap with gene names (*myc-c* gene and myc-c protein), cell cultures (CD4$^+$-cells and CD4 protein), and may be rather similar to chemical compounds (Caeridin and Cantharidin). Previously developed methods have been reported to reach an harmonic average between accuracy and coverage [Equation (4)] (F-measure) (harmonic mean of precision and recall) of ∼70% (Collier *et al*., 2000; Hou and Chen, 2003; Krauthammer *et al*., 2000; Morgan *et al*., 2003; Tsuruoka and Tsujii, 2003). However, very few of these methods were tested on large, curated databases. Given the amount of scientific literature, any improvement is likely to impact the yield of automatic data-mining approaches considerably. A sufficiently accurate method extracting protein names could be exploited to automatically build databases of protein–protein interactions (Friedman *et al*., 2001), sub-cellular localizations (Stapley *et al*., 2002), gene expression patterns (Masys *et al*., 2001) and protein–disease associations, just to mention a few examples of the possible applications. Our particular project evolved in context of the need to generate large datasets with nuclear matrix associated proteins.

Definition of protein names Which words or tokens should be annotated by the nametag? For example, we have four ways to tag the name in the phrase 'yeast YSY6 protein': 'yeast YSY6 protein', 'yeast YSY6', 'YSY6 protein' or 'YSY6'. This ambiguity implies that annotators may include yeast today and may exclude it a year later, unless given some 'annotation rules'. Since we worked with the corpus used to develop Yapex (Franzen *et al*., 2002), we adopted their definition: 'A protein-name defines a single biological entity composed of one or more amino acid chains'. This definition excludes the names of genes, protein families, domains, fragments and organisms, as well as, unspecific references to

*To whom correspondence should be addressed.

single protein molecules (the 49-kDa protein or the previously mentioned protein). The reduction allows to directly lookup database identifiers once a name is identified. Applied to the example above, the 'correct' tagging is 'YSY6 protein' yielding the database of protein sequences (Bairoch and Apweiler, 2000) (SWISS-PROT, accession number P38374).

Coding textual features for machine-learning The two most commonly used features of a protein name are (1) the name itself and (2) its context meaning its surrounding words or tokens. For machine-learning purposes, these two types of features can be represented as vectors in which each component codes for certain words and their positions within the name or its environment. For example, given a pre-defined word list with five words (A B C D E) and the hypothetical protein name 'A D B B', we can represent the name through a position-unspecific vector by simply counting the occurrences of each word in the name (one A, two Bs and one D) {1,2,0,1,0}. We can include information of position and order by reserving the first five slots (because the word-list contains five words) for the first word in the name (A), and generally the n-th five slots for the n-th word. For our example, this position-specific vector is {1,0,0,0,0, 0,0,0,1,0, 0,1,0,0,0, 0,1,0,0,0}. The context, i.e. the words surrounding the name, can be coded similarly. A possible third feature of a protein name is the context of the entire document. This feature differs from the previous two in that it cannot be derived from the name and its local environment. For example, the phrase 'protein CD4 was detected in ...' will provide information for any token CD4 found anywhere in the document. Such global features improved the taggers Yapex (Franzen *et al.*, 2002) and KeX (Fukuda *et al.*, 1998) significantly (Hou and Chen, 2003).

Related work Three types of approaches—dictionary-rule-based and machine-learning—have previously been exploited to extract protein names from the literature. Fukuda *et al.* (1998) have suggested that even extremely simple rules can yield F ~96% when tailored to a certain subject, such as 30 SH3-domain-related articles. A more generally applicable rule-based system is Yapex (Franzen *et al.*, 2002) with an estimated $F = 67\%$. Krauthammer *et al.* (2000) developed a dictionary-based system that—after translating the input text into nucleotide sequences—uses fast sequence alignment method (BLAST) (Altschul *et al.*, 1997; Altschul and Gish, 1996) to find names in a database of protein names. The authors considered partial matches as correct (CD4 instead of CD4 kinase) and reported an $F = 75\%$ on a set of two papers. Hanisch *et al.* (2003) used a semi-automatically generated dictionary in combination with a linear algorithm and reported an $F = 93\%$ without specifying the dataset used for this estimate. Tsuruoka and Tsujii (2003) have implemented a similar idea using dynamic programming instead of BLAST. After additionally filtering high-scoring examples through a simple Bayesian classifier, they report $F = 70\%$. One problem with dictionary-based systems is the limitation to names that are present in the dictionary. Collier *et al.* (2000) and Morgan *et al.* (2003) reported levels of $F = 76\%$ and $F = 75\%$, respectively, by using hidden Markov models (HMMs).

In order to automatically extract protein sequences from MEDLINE abstracts, we needed a method that distinguishes between protein and gene names. We developed a novel method combining rules, dictionaries and support vector machines (SVMs) to specifically identify and tag protein names from scientific literature. Limited space prevented us from presenting the data explaining many of the design features of the final system that reaches an outstanding level of performance only through a particular combination of rules, dictionaries and SVMs.

2 SYSTEM AND METHODS

2.1 Resources

Dictionary of protein names To benefit from the advantages of a dictionary with protein names, we compiled a list of commonly used synonyms. SWISS-PROT and translation of the EMBL-nucleotide database coding DNA to protein sequences (TrEMBL) (Bairoch and Apweiler, 2000) collect over one million protein sequences along with the most commonly used names of these proteins (DE field), and the names of related genes (GN field). We generated a list of all SWISS-PROT + TrEMBL protein- and gene-names and linked each name to its associated database identifier. To increase robustness, we converted names into lower-case characters, converted non-letter/non-digit characters into spaces and inserted spaces between digits and letters ($2S \rightarrow 2 S$). Note that we applied the same conversions to the texts analyzed. We used this 'protein dictionary' to derive an input value for one of the machine-learning building blocks in our system (SVM4 in Fig. 2).

Dictionary of common words and chemical compounds We based our 'common dictionary' on the online-version of the Merriam-Webster (MW) dictionary (http://www.m-w.com). The entries of this resource contain part-of-speech information such as 'adjective' or 'noun' that was useful for our filtering procedure (Table 1). To expand this list, we added medical terms [dictionary of medical terms (DMT), http://cancerweb.ncl.ac.uk/omd/], species names (http://us.expasy.org/cgi-bin/speclist), tissue types (http://us.expasy.org/cgi-bin/lists?tisslist), and minerals and formulas (un2sg4.unige.ch/Athena/mineral/min_lists.html). Our system used this common dictionary for filtering as one of the first steps in the procedure (Fig. 2). We downloaded a list of chemical names (http://www.speclab.com) and derived a list of 130 endings (last four letters). We used this list by removing all words from the input text (Table 1) that end in one of the 130 endings for chemical compounds (*hyde and *enyl).

Training corpus Currently, two datasets of text with expert annotated name tags are available, namely the GENIA-corpus (Kim *et al.*, 2003) and a corpus used for developing Yapex (Franzen *et al.*, 2002). The GENIA-corpus contains 2000

Table 1. Examples of filtering rules

Rule	Text examples
Set of regular expressions	16S, AGGTGGC, Ca2+, L214A, 26 kDa, mol/L, Asp-15
Name is followed by ‘cell(s)’ or ‘cyte(s)’	CD4+T lymphocytes, *Streptococcus mutans* cells
Name ending similar to chemical compound (list of 130 4-letter endings)	polypyrimidine, ether, ethanolamine
Name is in common-dictionary	interaction, factor, HIV-1, specific, leucocytes
Name seems to be an author	Miller *et al*., Smith (2002)
Name is in parentheses following a filtered-out word	CD4+T lymphocytes (CD4TL), Inositol-3-phosphate (IP3)
Name is number followed by noun in plural form	four proteins, three factors

abstracts related to transcription factors in human blood cells. These abstracts are tagged according to different categories for named entities (protein molecule, DNA-molecule). Since the GENIA-corpus is limited to a certain cell-type and protein family and also lacks the actual tag-class ‘protein-name’, we could not use it for developing our system (note for comparisons we did however use it for evaluation). The Yapex-corpus contains 200 abstracts, obtained by randomly picking 150 abstracts out of a PubMed [with query: ‘protein binding (Mesh term) AND interaction AND molecular’ with the parameters ‘human’ and ‘publication date 1996–2000’]. A total of 50 additional abstracts were taken at random from the GENIA-corpus. Protein names were tagged for all 200 abstracts. Finally, we also used the BioCreAtIvE Corpus (Critical Assessment of Information Extraction Systems in Biology, http://www.mitre.org/public/biocreative/) with 7500 sentences for training and 2500 for testing. These data are part of the evaluation of ‘Task 1A: entity extraction’ in the BioCreAtIvE experiment. According to the BioCreAtIvE criteria, an entity may also be a gene name.

2.2 Training

Generating samples We split each abstract into single tokens according to the following four rules: (1) convert into lowercase, (2) separate punctuation characters such as dots, commas and parentheses from the surrounding words by spaces, (3) spaces separate tokens and (4) dashes (/) and hyphens (-) do not separate. Since protein names are rarely longer than five tokens, we generated sample phrases by using 1–5 consecutive tokens from the text (A–E in Fig. 1). Additionally, we used the four tokens before and after this phrase (1–8 in Fig. 1). We refer to the tokens A–E as the ‘center’ and to tokens 1–8 as the ‘environment’. In order to account for words at the text borders, we added pseudo words to the beginning and the end of the input text. An input text of, e.g. 12 tokens/words, will generate exactly 60 samples of 9–13 tokens in length [$(12-5)\times 5+5+4+3+2+1=60$, where 5 is the maximum number of tokens in the center and 12 is the length of the text].

Filtering samples Since SVMs get easily confused when confronted with inconsistent data, every sample has to pass a simple filter. Simple filtering rules only apply to the sample ‘centers’ (e.g. regular expressions). Others also consider the sample ‘environment’ (e.g. ‘cell/cyte’-rule). We used the common-dictionary and the list of 130 chemical compound endings to pre-filter the text (examples in Table 1).

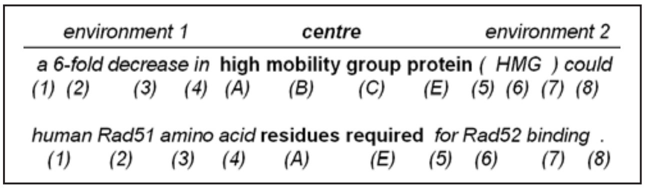
environment 1
centre
environment 2
a 6-fold decrease in high mobility group protein (HMG) could
(1) (2) (3) (4) (A) (B) (C) (E) (5) (6) (7) (8)
human Rad51 amino acid residues required for Rad52 binding .
(1) (2) (3) (4) (A) (E) (5) (6) (7) (8)

Fig. 1. Scheme for text parsing. Two examples of text from a MEDLINE abstract; we considered the one on the top as ‘true positive’, the other as ‘true negative’ (no protein name). Letters and numbers in brackets label words. A text sample will only count as ‘true positive’ if the center contains a complete protein name (no fragment). We divided samples (continuous words) into three parts: the environment 1 (before the name, words 1–4), the center (words A–E) and the environment 2 (after the name, words 5–8). Note that the environment always contains eight tokens (words), whereas we varied the size of the center from one to five. Regardless of the number of central words, we referred to the last token as E.

SVM-architecture and coding SVMs (Cortes and Vapnik, 1995) solve two-class classification problems by finding an optimally separating hyper-plane with a maximal distance to the closest data-points (vectors) of each class. The basic component is an arbitrary kernel function that maps all input data-points into a multi-dimensional feature space. Typically used kernel functions are linear, polynomial and radial basis functions. We used an off-shelf package to implement SVMs [‘svm-light’ (Joachims, 1999), http://www.joachims.org]. Our system combined four separate SVMs (Fig. 2): each of the first three SVMs specializes on a certain part of the samples (center, environment and overlap between the two). The fourth SVM combines the output values from the first three with a score from our protein-dictionary thus generating the final score for each sample. All four SVMs were trained by using the linear kernel function integrated in the svm-light package. Since the sample centers can overlap (NF kappa B and NF kappa), we used only the one with the highest score (Fig. 2).

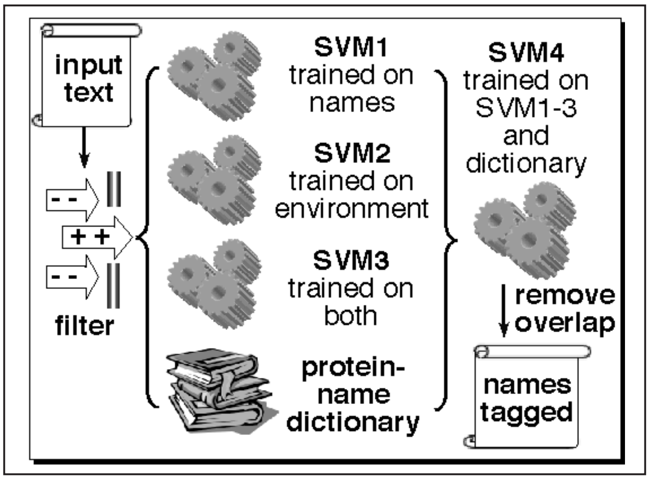

Fig. 2. Architecture of NLProt. We sliced the input text into single samples by a sliding window approach. These text-continuous samples of token-words were pre-filtered through dictionaries and then passed to three SVMs (SVM1–3). Each of these SVMs is specialized on a certain aspect of the problem: SVM1 focused on the centers (names), SVM2 on the environments and SVM3 on the overlap between the two. Next, we derived a dictionary score (both from a dictionary of protein names and a dictionary of non-protein names), and fed the output of SVM1–3 in combination with this dictionary score into SVM4. Finally, we removed all overlaps from the output of SVM4 (highest score wins).

SVM1 for center The first SVM was trained on the sample centers (putative protein names). After splitting all centers into tokens (A–E), we automatically compiled a list of the 3000 most frequent tokens. A simple function $f(t)$ returned unique values between 1 and 3000 for each token t in the list [$f1(t) = 0$ for non-frequent tokens]. Each input vector for SVM1 had 9000 components; the value of component $i\,(s_i)$ was

$$s_i = \begin{cases} 1 & \text{if } (f1(\text{A}) = i \text{ AND } i < 3001) \\ 1 & \text{if } (f1(\text{E}) = i - 3000 \text{ AND } 3000 < i < 6001) \\ 1 & \text{if } (f1(\text{B}|\text{C}|\text{D}) = i - 6000 \text{ AND } 6000 < i) \\ 0 & \text{else } \forall i \in \{1, \ldots, 9000\}, \end{cases} \tag{1}$$

where $f1$(A) was the value for the first center-token A, $f1$(E) that for the last center-token E and $f1$(B or C or D) that for the middle (Fig. 1). Note that for centers containing five tokens (we used all possibilities from 1 to 5 tokens) up to three components of s between s_{6001} and s_{9000} may be non-zero. For the example protein name A D B B (Introduction) and a list of five words (A B C D E) instead of 3000, the input for SVM1 becomes {1,0,0,0,0, 0,1,0,0,0, 0,1,0,1,0} where the first five components represent the word A in the name, the central five the word B and the last five the words D B. Note that we took the idea of stressing the first and last word of a name from Tsuruoka and Tsujii (2003). *Note*: we chose the number 3000 because that yielded a good performance for a single SVM. We did not fully optimize this number.

SVM2 for environment The second SVM was trained on the environment using a sliding window of eight tokens (tokens 1–8 in Fig. 1). As for SVM1, we compiled the 3000 most frequent tokens and defined a function $f2(t)$ mapping all tokens t to values 1–3000. The 12 000 input components for SVM2 were

$$s_i = \begin{cases} 0.25 & \text{if } (f2(1) = i \text{ AND } i < 3001) \\ 0.50 & \text{if } (f2(2) = i \text{ AND } i < 3001) \\ 0.75 & \text{if } (f2(3) = i \text{ AND } i < 3001) \\ 1.00 & \text{if } (f2(4) = i - 3000 \text{ AND } 3001 < i < 6001) \\ 1.00 & \text{if } (f2(5) = i - 6000 \text{ AND } 6000 < i < 9001) \\ 0.75 & \text{if } (f2(6) = i - 9000 \text{ AND } 9000 < i) \\ 0.50 & \text{if } (f2(7) = i - 9000 \text{ AND } 9000 < i) \\ 0.25 & \text{if } (f2(8) = i - 9000 \text{ AND } 9000 < i) \\ 0.00 & \text{else } \forall i \in \{1, \ldots, 12\,000\}, \end{cases}, \tag{2}$$

where $f1$(A) was the value for the first center-token A, $f1$(E) that for the last center-token E and $f1$(B or C or D) that for the middle (Fig. 1). Note that for centers containing five tokens (we used all possibilities from 1 to 5 tokens) up to three components of s between s_{6001} and s_{9000} may be non-zero. For the example protein name A D B B (Introduction) and a list of five words (A B C D E) instead of 3000, the input for SVM1 becomes {1,0,0,0,0, 0,1,0,0,0, 0,1,0,1,0} where the first five components represent the word A in the name, the central five the word B and the last five the words D B. Note that we took the idea of stressing the first and last word of a name from (Tsuruoka and Tsujii, 2003). *Note*: we chose the number 3000 because that yielded a good performance for a single SVM. We did not fully optimize this number.

SVM3 for overlap Both SVM1 and SVM2 have their 'blind spots' (SVM1 on the environment, SVM2 on the center). We accounted for this by training SVM3 on the overlap between the two. In particular, we considered tokens 4, A, E and 5 in Figure 1 as the overlap. As for SVM1 and SVM2, we used the 3000 most frequent tokens, defined a function $f3(t)$ that mapped all tokens t onto values 1–3000 and calculated the components $i\,(s_i)$ by:

$$s_i = \begin{cases} 1 & \text{if } (f3(4) = i \text{ AND } i < 3001) \\ 1 & \text{if } (f3(\text{A}) = i - 3000 \text{ AND } 3000 < i < 6001) \\ 1 & \text{if } (f3(\text{E}) = i - 6000 \text{ AND } 6000 < i < 9001) \\ 1 & \text{if } (f3(5) = i \text{ AND } 9000 < i) \\ 0 & \text{else } \forall i \in \{1, \ldots, 12\,000\} \end{cases}. \tag{3}$$

SVM4 for combination The last SVM (SVM4) was trained on the outputs of SVM1, SVM2 and SVM3. As a fourth input feature, we derived a score from the protein-dictionary (Fig. 2). This score was either the length of the center string if this string was in the protein-dictionary, and 0 if not. Longer names found in the protein-dictionary are weighted higher than shorter ones, because shorter names (oct or rna) are more likely non-specific for protein names.

2.3 Testing

Evaluating performance Adequately estimating the performance of tagging methods on new data is not trivial; it is especially difficult for dictionary- and rule-based methods that may be limited to the data used for development and testing. Furthermore, tagging names is a time-demanding, expertise-required task; consequently, datasets are often small. The GENIA-corpus (Kim *et al*., 2003) addressed the number problem. Unfortunately, at the expense of resolution names are not tagged in the GENIA-corpus. Franzen *et al*. (2002) carefully marked 200 abstracts to develop Yapex. We used that corpus, and—in contrast to Franzen *et al*.—cross-validated our evaluation by using 180 abstracts to train the first three SVMs, 15 to train the fourth and 5 to test. Then we rotated through all abstracts, such that we used each abstract exactly once for testing (40-fold cross-validation). In other words, none of the abstracts for which we reported the results had ever been used for development. We used the harmonic mean (F-measure) of accuracy (precision) and coverage (recall) that are commonly used in the field to evaluate our results. With TP labeling true positives, FP the false positives and FN the false negatives, the measures were

$$\mathrm{ACC} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \left| \mathrm{COV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \right| F = \frac{2\cdot\mathrm{ACC}\cdot\mathrm{COV}}{\mathrm{ACC}+\mathrm{COV}}. \tag{4}$$

Strict and sloppy mode (Franzen *et al*., 2002) distinguish classes of correct identifications: 'strict' refers to names that are entirely correct (right: protein kinase c, wrong: protein kinase), 'sloppy' to partially identified names (right: kinase/c/protein, wrong: ismb). Evaluating the performance in the 'sloppy-mode' obviously inadequately favors methods that over-predict names and their length. In the extreme, identifying the entire abstract as 'the name' would yield $F = 100\%$. Therefore, we mainly used the strict mode in our evaluation, except for comparisons.

Reducing redundancy The recent explosion of biological data has revealed how important it is to reduce bias from datasets to analyze data and to develop prediction methods. Measures for sequence-similarity, such as percentage sequence-identity, BLAST E-values (Altschul *et al*., 1997) or the homology derived sequence structures of proteins (HSSP)-value (Mika and Rost, 2003; Sander and Schneider, 1991) capture some aspects of bias. Surprisingly, previous work on name tagging did not address the problem of bias. Here, we reduced bias through three rules. First, we ignored names identical between testing and training set (neither counted as true nor as false). Second, we ignored multiple occurrences of the same name if correctly identified (note: the same name may still contribute many times to our FN count). Third, in order to test the success of our machine-learning component, we removed all names in the test set from our protein-dictionary. For comparisons, we reported results with and without applying these rules (Table 2).

Table 2. Performance of NLProt[a]

Method	Testing corpus	Number of protein names	Bias?	Accuracy (%)	Coverage (%)	F (%)
NLProt	Yapex	1938	Yes	75	76	75
NLProt	GENIA	20 778	Yes	63	81	71
NLProt	Recent166	1349	Yes	70	85	77
NLProt	Yapex101	1938	Yes	76	78	77
Yapex	Yapex101	1938	Yes			67
NLProt	BioCreAtIvE	11 871	Yes	73	75	74
NLProt	Yapex	1109	No	61	59	60
NLProt	GENIA	4104	No	48	60	53

[a]All results are valid for cross-validated testing sets, except those with Yapex101. Datasets: 200/101 abstracts from Yapex (Franzen *et al*., 2002), 2000 abstracts from GENIA [Kim *et al*. (2003), marked in light gray since tagging was not explicitly carried out on protein names], 10 000 sentences from the BioCreAtIvE experiment (marked in dark gray since tagging explicitly includes also gene names as valid entries); the 166 abstracts 'recent' were annotated by us after completion of the method. Performance measures as defined in Equation (4) (values for 'strict' mode, e.g. wrong: protein kinase; right: protein kinase C).

3 RESULTS AND DISCUSSION

NLProt with dictionary reached $F = 75\%$ Our system correctly identified 76% of the names tagged in the Yapex-corpus (coverage/recall), and 75% of the names that we identified were correct (accuracy/precision); the corresponding F was 75% [Equation (4), Table 2]. Removing SVM1 (trained on the centers, i.e. names) dropped F to 69%; removing SVM2 (context) dropped F to 63% (note that for both scenarios, we also removed SVM3 handling the overlap between SVM1 and SVM2). For comparison to the work from Franzen *et al*. (2002), we reproduced their evaluation by training on the first 99 and testing on the remaining 101 abstracts from the Yapex-corpus. The resulting—incorrect—estimate for F was 77%. For the 'sloppy-mode' (partial recognition of names), our system reached $F = 85\%$. Both these higher numbers clearly over-estimated performance, we only compiled them for comparisons. Using the GENIA-corpus for training and testing, i.e. considering tags for protein_molecule as the protein name, our system reached $F = 71\%$ with a rather high coverage, and a much lower accuracy (supposedly since protein_molecule tags entities other than the name). When explicitly trained on a corpus that includes gene names as valid entities (BioCreAtIvE corpus), NLProt appears to reach its average performance. Thus, our method could easily be generalized to slightly different tasks.

NLProt without dictionary and without bias reached $F = 60\%$ When reducing bias, i.e. testing how well the system handles novel names, the system reached $F = 60\%$; leaving out only the dictionary (rule 3), which yielded $F = 72\%$. This implied that the dictionary was not the most important component. Applying the same bias-reduction rules to the GENIA-corpus, F dropped to 53% (Table 2). Again, the coverage was considerably higher than the accuracy due to the inadequacy of the GENIA-tagging for our purposes. Note that by definition, all dictionary-based methods would yield $F = 0\%$ on this test!

NLProt rather competitive in comparison with other systems The only two methods that were reported to reach performance levels similar to our method used HMMs at $F = 75\%$ [Morgan *et al*. (2003) without cross-validation and based on automatically annotated abstracts as 'standard-of-truth'] and at $F = 73\%$ [Collier *et al*. (2000) cross-validated on 100 abstracts]. The only system that we could compare on identical datasets was Yapex (Franzen *et al*., 2002), when applying the same train/test conditions as the authors (Yapex101 in Table 2). Our system reached an F-measure 10 percentage points above that for Yapex. The best performance for a dictionary-based system was reported to be $F = 75\%$ (Krauthammer *et al*., 2000); however, this estimate originated from a very small dataset (two review papers). Furthermore, when applying our bias-reduction rules to the evaluation procedure NLProt still reached $F = 60\%$. This lower limit of $F = 60\%$ was striking given that it applied for situations in which all the names correctly identified were neither in the dictionary nor in the training set, i.e. the system had learned to correctly discover names it had never encountered. The only two papers that report results on data entirely unknown to the system are an HMM-based tagger reaching $F = 33\%$ (Morgan *et al*., 2003), and the system combining dictionaries with BLAST searches reaching $F = 4\%$ for unknown samples (Krauthammer *et al*., 2000).

Limitations and extensions Due to the nature of our training-corpora, our final system is specialized on mammalian proteins. It remains unclear, how accurately NLProt identifies names from organisms such as *Drosophila* for which the naming appears optimized to fool text-mining methods by choosing protein names similar to common words (white, bizarre, wing). In contrast, the conventions observed in organisms such as yeast appear to be ideal for our name-tagging method. While we anticipate that a larger training corpus may improve performance, this speculation also remains as a speculation.

Estimates valid in real life? After completion of our work, we applied NLProt to the 166 abstracts published in *Cell* and *EMBO Journal* over the last 60 days (November 2003–January 2004). Although this test did not reduce redundancy and benefited from the protein-name dictionary, it provided a snapshot of what to expect in 'real life'. NLProt reached 70% accuracy at 85% coverage ($F = 77\%$, Table 2). This sustained high-level performance on data that we had never used convinced us that NLProt is ready to roll.

ACKNOWLEDGEMENTS

Thanks to Jinfeng Liu and Megan Restuccia (Columbia) for computer assistance; to Marco Punta and Rajesh Nair (Columbia) for valuable insights and fun discussions. This work was supported by the grants RO1-GM63029-01 from the National Institutes of Health (NIH), R01-LM07329-01 from the National Library of Medicine (NLM) and DBI-0131168 from the National Science Foundation (NSF). Last, not least, thanks to Amos Bairoch (SIB, Geneva), Rolf Apweiler (EBI, Hinxton), Phil Bourne (UCSD), Michael Ashburner (Cambridge) and their crews for maintaining excellent databases, and to all experimentalists who enabled this work by making their data publicly available.

REFERENCES

Altschul,S. and Gish,W. (1996) Local alignment statistics. *Methods Enzymol*., **266**, 460–480.

Altschul,S., Madden,T., Shaffer,A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D. (1997) Gapped Blast and PSI-Blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bairoch,A. and Apweiler,R. (2000) The Swiss-Prot protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, **28**, 45–48.

Collier,N., Nobata,C. and Tsujii,J. (2000) Extracting the names of genes and gene products with a Hidden Markov Model. *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)* Saarbrücken, Germany, 31 July–4 August. Morgan Kaufmann Publishers, pp. 201–207.

Cortes,C. and Vapnik,V. (1995) Support Vector Networks. *Machine Learning*, **20**, 273–297.

Franzen,K., Eriksson,G., Olsson,F., Asker,L., Liden,P. and Cöster,J. (2002) Protein names and how to find them. *Int. J. Med. Inf.*, **67**, 49–61.

Friedman,C., Kra,P., Yu,H., Krauthammer,M. and Rzhetsky,A. (2001) GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, **17**, S74–S82.

Fukuda,K., Tsunoda,T., Tamura,A. and Takagi,T. (1998) Toward information extraction: identifying protein names from biological papers. *Pac. Symp. Biocomput.*, 707–718.

Hanisch,D., Fluck,J., Mevissen,H. and Zimmer,R. (2003) Playing Biology's name game: identifying protein names in scientific text. *Pac. Symp. Biocomput.*, 403–414.

Hou,W. and Chen,H. (2003) Enhancing performance of protein name recognizers using collocation. *ACL-03 Workshop on Natural Language Processing in Biomedicine*, Sapporo Convention Center, Sapporo, Japan, 11 July. Association for Computational Linguistics (ACL), pp. 25–32.

Joachims,T. (1999) *Making large-Scale SVM Learning Practical. Advances in Kernel Methods—Support Vector Learning*. MIT-Press.

Kim,J.D., Ohta,T., Tateisi,Y. and Tsujii,J. (2003) GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, **19**(Suppl. 1), I180–I182.

Krauthammer,M., Rzhetsky,A., Morozov,P. and Friedman,C. (2000) Using BLAST for identifying gene and protein names in journal articles. *Gene*, **259**, 245–252.

Masys,D., Welsh,J., Lynn Fink,J.M.G., Klacansky,I. and Corbeil, J. (2001) Use of keyword hierarchies to interpret gene expression patterns. *Bioinformatics*, **17**, 319–326.

Mika,S. and Rost,B. (2003) UniqueProt: creating representative protein sequence sets. *Nucleic Acids Res.*, **31**, 3789–3791.

Morgan,A., Hirschman,L., Yeh,A. and Colosimo,M. (2003) Gene name extraction using FlyBase resources. *ACL-03 Workshop on Natural Language Processing in Biomedicine*, Sapporo Convention Center, Sapporo, Japan, 11 July. Association for Computational Linguistics (ACL), pp. 1–8.

Sander,C. and Schneider,R. (1991) Database of homology-derived structures and the structural meaning of sequence alignments. *Prot. Struct. Func. Genet.*, **9**, 56–68.

Stapley,B., Kelley,L. and Sternberg,M. (2002) Predicting the subcellular location of proteins from text using support vector machines. *Pac. Symp. Biocomput.*, 374–385.

Tsuruoka,Y. and Tsujii,J. (2003) *Boosting precision and recall of dictionary-based protein name recognition. ACL-03 Workshop on Natural Language Processing in Biomedicine*, Sapporo Convention Center, Sapporo, Japan, 11 July. Association for Computational Linguistics (ACL), pp. 41–48.