OMICS A Journal of Integrative Biology
Volume 8, Number 2, 2004
© Mary Ann Liebert, Inc.

Finding Kinetic Parameters Using Text Mining

JÖRG HAKENBERG,[1] SEBASTIAN SCHMEIER,[2] AXEL KOWALD,[2] EDDA KLIPP,[2] and ULF LESER[1]

ABSTRACT

The mathematical modeling and description of complex biological processes has become more and more important over the last years. Systems biology aims at the computational simulation of complex systems, up to whole cell simulations. An essential part focuses on solving a large number of parameterized differential equations. However, measuring those parameters is an expensive task, and finding them in the literature is very laborious. We developed a text mining system that supports researchers in their search for experimentally obtained parameters for kinetic models. Our system classifies full text documents regarding the question whether or not they contain appropriate data using a support vector machine. We evaluated our approach on a manually tagged corpus of 800 documents and found that it outperforms keyword searches in abstracts by a factor of five in terms of precision.

INTRODUCTION

OVER THE LAST FEW YEARS, there has been an explosion of information in biology. The sequencing of more than a hundred genomes, including the human genome, gave detail about thousands of genes. Efforts are under way to identify the proteins encoded by those genes. High-throughput technologies, especially microarray techniques deliver information about the expression of these genes under different conditions (Ideker et al., 2001). Yeast two-hybrid methods yield networks of protein interactions (Li et al., 2004; Grigoriev, 2003).

The next step forward toward a more comprehensive understanding of the underlying biological system is now to integrate these data. The goal is to no longer study individual genes or proteins, but to learn how these molecules interact to form a living cell. However, doing so requires a large amount of data as parameters for the appropriate models and methods. Many of these parameters have been experimentally measured, by finding them in the vast amount of available literature is extremely difficult.

In this paper, we describe text mining methods for the detection of scientific papers dealing with specific aspects of systems biology (i.e., parameters for kinetic models). To motivate our research project, we start the paper with an introduction into systems biology and into text mining.

Systems biology

The emerging field of systems biology aims at understanding and modeling biological systems as systems at the molecular and higher levels of organization (Kitano, 2002). These systems may be gene ex-

[1]Humboldt-Universität zu Berlin, Department of Computer Science, Berlin, Germany.
[2]Max-Planck-Institute for Molecular Genetics, Kinetic Modeling Group, Berlin, Germany.

pression networks, signal transduction pathways, metabolic networks, or combinations of them. In contrast to previous approaches, systems biology endeavors to quantitatively model and simulate complex biological processes and systems comprising thousands of chemical compounds and reactions. One wants to understand the structure, dynamics, control pattern, and design principles of the systems under investigation. Concerning size and fidelity of the models, there are two tendencies: First, the development of whole cell models, comprising virtually all compounds and processes in a eukaryotic cell. Second, very elaborated and detailed models are necessary for individual processes and events in organisms that are of general significance or show a very special behavior.

The kinetic modeling of biological reaction networks deals with the question of how the concentrations of substances change over time. The dynamics are determined by (a) the concentrations of substrates and products, (b) the structure of the whole reaction network, and (c) the kinetic parameters of the involved enzymes. To understand what kinetic parameters are and why they are important we have to look at enzyme kinetics.

Enzyme kinetics

For a biochemical reaction system, it is practice to use a set of ordinary differential equations (ODEs) to describe the changes in the concentration of a biochemical species. In a system of n species with the concentration c_i ($i = 1, \ldots, n$) and r biochemical reactions with the rates v_j ($j = 1, \ldots, r$) one may write

$$\begin{aligned}
\frac{dc_1}{dt} &= f_1(c_1, c_2, \ldots, c_m) = n_{11}v_1 + n_{12}v_2 + \cdots + n_{1r}v_r \\
\frac{dc_2}{dt} &= f_2(c_1, c_2, \ldots, c_m) = n_{21}v_1 + n_{22}v_2 + \cdots + n_{2r}v_r \\
&\vdots \\
\frac{dc_m}{dt} &= f_m(c_1, c_2, \ldots, c_m) = n_{m1}v_1 + n_{m2}v_2 + \cdots n_{mr}v_r
\end{aligned} \tag{1}$$

where the quantities n_{ij} denote the stoichiometric coefficients, i.e. the molecularity in which species i enters reaction j. A coefficient is zero if a reaction j does not consume or produce the species i; it is smaller or bigger than 0 (usually -1 or 1, depending on molecularity), if the species i is degraded or produced in reaction j, respectively. We present an example, a simplified glycolysis model, in Figure 1.

The rate of a reaction is a function of the concentrations of the substrates and products of the reaction and of parameters. These parameters may be the concentrations of effectors as well as kinetic constants with different physical units. The actual expression for a rate depends on the experimental knowledge about the kinetic characterization of a reaction and, partially, on the modeling purpose. Typical expressions for reaction rates are explained below. Variables c_S and c_P denote the concentrations of substrate and product, respectively.

All reaction kinetics are based on the mass action law, introduced by Guldberg and Waage in the 19th century. It states that the reaction rate is proportional to the probability of a collision of the reactants. This is in turn proportional to the concentration of the reactants to the power of the molecularity in which they enter the specific reaction. For a simple reaction like

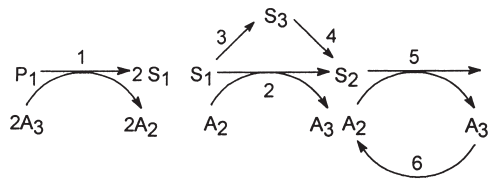
$$S_1 + S_2 \rightarrow 2P \tag{2}$$

this reads

$$v = v_+ - v_- = k_+ c_S 1 \cdot c_S 2 - k_- c_P^2 \tag{3}$$

where v_+ and v_- denote the forward and backward rates. k_+ and k_- are the forward and the backward rate constants, respectively (i.e., proportionality constants). In the general case, the rate expression for reversible linear kinetics reads (indices run over all substrates or products of this reaction, respectively).

$$v = k_+ \prod_i c_{S_i} - k_- \prod_l c_{P_l} \tag{4}$$



$$\frac{dS_1}{dt} = 2v_1 - v_2 - v_3$$
$$\frac{dS_2}{dt} = v_2 + v_4 - v_5$$
$$\frac{dS_3}{dt} = v_3 - v_4$$
$$\frac{dA_2}{dt} = 2v_1 - v_2 - v_5 + v_6$$
$$\frac{dA_3}{dt} = -2v_1 + v_2 + v_5 - v_6$$

FIG. 1. A simplified glycolysis model. P_1, glucose, considered as external; A_2, ADP; A_3, ATP; S_1, Pool of triosephosphates DHAP and GAP; S_2, 3-phosphoglycerat, 2.3-diphosphoglycerat; v_1, upper glycolysis; v_2, v_5, lower glycolysis; v_3,v_4, 1,3-biphosphoglycerat-bypass; v_6, ATP consumption.

Sometimes reactions are considered as irreversible, i.e. they are supposed to proceed only in one direction. The rate expression for irreversible linear kinetics is

$$v = k_+ \prod_i c_{S_i} \tag{5}$$

As opposed to simple chemical reactions, enzymatic reactions show saturation. The dependence of the rate on the substrate concentration gives a hyperbolic curve. For low substrate concentrations, the rate is proportionally to c_S. With increasing substrate concentrations, v increases slower and asymptotically approaches a maximal value. This can be explained by a two-step process: first, the reversible binding of the substrate to the enzyme (corresponding to the collision of molecules in the mass action law) and second, the irreversible release of the product from the enzyme. If the enzyme concentration is much lower than the substrate concentration, then the enzyme molecules are soon saturated with substrate. This performance is reflected in the Michaelis-Menten type of kinetic expression, which reads for irreversible transformation of one substrate:

$$v = \frac{V_{\max} \cdot c_S}{K_M + c_S} \tag{6}$$

$V_{\max}$ denotes the maximal rate for high substrate concentrations and K_M is the half-saturation concentration (Michaelis-Menten constant). $V_{\max}$ and K_M are the kinetic parameters that control this specific kinetic law. There are, however, several other types of kinetics for an enzymatic reaction. For a reversible uni-uni-reaction (one substrate, one product), we have

$$v = \frac{\frac{V^+_{\max}}{K_{M,S}} \cdot c_S - \frac{V^-_{\max}}{K_{M,P}} \cdot c_P}{1 + \frac{c_S}{K_{M,S}} + \frac{c_P}{K_{M,P}}} \tag{7}$$

where $V^+_{\max}$,$V^-_{\max}$ are the maximal rates of the forward or backward processes, respectively, and $K_{M,S}$, $K_{M,P}$ are the Michaelis-Menten constant of the substrate and product, respectively. To model this type of reaction properly, more kinetic parameters have to be known than in the irreversible case. $V_{\max}$ and K_M are so-called phenomenological parameters as opposed to the elementary rate constants k_+ and k_-. The former can usually be measured in standard biochemical experiments, while the latter are often hard to assess, at least for complex reactions. For reactions with more substrates or products the rate expression depends on the concentrations of all reactants and a large number of kinetic parameters.

Another type of rate-substrate-dependence is the sigmoidal behavior. This is usually explained by the fact that the enzyme consists of several subunits. Each of the subunits can catalyze the reaction, but their

performance may be altered due to the binding of a substrate to another subunit. A typical example for sigmoidal kinetics is the Hill equation

$$v = \frac{V_{\max} \cdot (c_S)^h}{(K_B)^h + (c_S)^h} \tag{8}$$

where $V_{\max}$ denotes again the maximal rate, K_B the binding constant, and h the Hill coefficient (another kinetic parameter). The Hill coefficient was originally identified with the number of subunits, but may also be a lower rational number with typical values between about 2 and 6. Another type of sigmoidal kinetics with other types of kinetic parameters is the rate expression according to Monod, Wyman, and Changeux:

$$V = \frac{V_{\max} c_S}{(1 + K_R c_S)} \frac{K_R + K_T L \left(\frac{1 + K_T c_S}{1 + K_R S}\right)^{n-1}}{\left(1 + L \left(\frac{1 + K_T c_S}{1 + K_R c_S}\right)^n\right)} \tag{9}$$

Here, it is assumed that the enzyme has several subunits, the subunits may be present in an active or an inactive conformation, and there is an equilibrium between both conformations. K_R and K_T are the binding constants to the active and the inactive form, respectively, L is the equilibrium constant between both forms, and n is the number of subunits.

An important function of enzymes is the ability to be regulated by effectors. The effectors may bind to the enzyme, to the substrate or to complexes formed by both. In each case we obtain different rate equation. The rate equation for a Michaelis-Menten type of reaction inhibited by an effector binding to any form of the enzyme (so-called non-competitive inhibition) reads

$$v = \frac{V_{\max} \cdot c_S}{(K_M + c_S)\left(1 + \frac{c_I}{K_I}\right)} \tag{10}$$

(c_I, inhibitor concentration; K_I, inhibitor dissociation constant). The increase of the rate by an activator may in the simplest form be expressed by the following rate equation

$$v = \frac{k \cdot c_S + k' \cdot K_S}{(K_S + c_S)\left(1 + \frac{c_A}{K_A}\right)} \tag{11}$$

In this equation c_A is the concentration of the activator, K_S and K_A are the dissociation constants for substrate and activator, and k and k' denote elementary rate constants of product release.

In the model of Monod, Wyman, and Changeux activation and inhibition may be easily included by assuming that an effector pushes the equilibrium toward the active or the inactive state leading to an altered equilibrium constant:

$$L' = L \frac{(1 + K_I c_I)^n}{(1 + K_A c_A)^n} \tag{12}$$

The number of kinetic parameters increases with the number of substrates and/or products, with the number of effectors, and with the level of detail of the description of individual events in the considered reaction. In the rate equations listed above, the concentration of the catalyzing enzyme is already considered to be multiplicatively included in the values of the elementary kinetic constants or in the maximal rate. This is appropriate for the modeling of metabolic networks under the assumption of constant enzyme concentration. If the production or the degradation of the enzymes shall be considered, these quantities have to be replaced in an appropriate way (like $k \rightarrow c_E \cdot k'$).

For the quantitative modeling of biochemical reaction networks it is important to know the structure of the network (which will not be considered here), the kinetic type of each reaction (whether it exhibits linear, hyperbolic or sigmoidal kinetics, whether it is considered to be reversible or irreversible, which is the number of substrates), and it is especially essential to know the values of the various kinetic parameters.

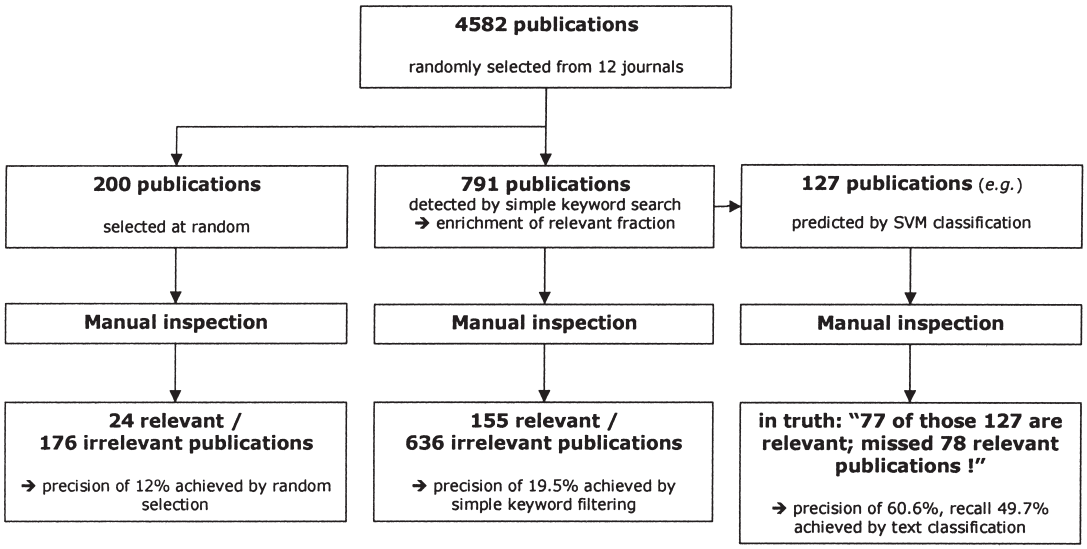


FIG. 2. Resource collections. Starting from 4582 publications loaded from 12 online journals, we found approximately 12% to contain relevant information, estimated by a manual inspection of 200 randomly selected publications. Applying a simple keyword filter reduced possible candidates to 791 and resulted in a precision of this method of 19.5%. As we do not know the percentage of relevant papers contained in the remaining $4582 - 791 = 3791$ publications, the recall is unknown. Classifying the 791 filtered publications with our SVM model yielded a precision of 60% at 50% recall.

Problems we aim to solve using text mining and machine learning

The kinetic modeling of biological systems depends on sets of different kinetic data and values measured in expensive experiments. Such data are published in thousands of scientific articles. It is infeasible for humans to read and analyze this number of papers within reasonable time constraints. However, to build a concrete model (e.g. for a certain metabolic pathway of a given species), one needs only a selected subset of kinetic parameters that is very likely to be found in only a few papers—but finding those is a challenge. To build up a model one has to gather, sort out, read, and understand a large number of publications.

We performed a study to quantify this claim; 4582 randomly chosen full text documents from 12 different journals were downloaded. Of those, 791 were selected by performing a search with 20 different keywords (Fig. 2). Reading those 791 documents, which meant about two person months of work, revealed that only 155 actually contained kinetic parameters, corresponding to a precision of 20% of the method. This task of reading all 800 articles only focused on deciding whether or not a paper contained any relevant parameters, while annotating each single parameter would definitely take a longer time.

Our project aims at developing methods for supporting systems biology researchers in their information retrieval needs. The problem was divided into the retrieval and the extraction of publications relevant to kinetic modeling. In this paper we explain our ideas and give results for the information retrieval step. Its goal is to identify appropriate documents obtainable from online journals by using text mining methods. To this end, we implemented and tested different methods for natural language processing, text processing, and text classification. A number of combinations were evaluated with respect to their individual strength and the overall performance of the classification process. Although there is still room for substantial improvements, our current system already resulted in a drastic reduction of the manual work necessary to filter out irrelevant documents.

Text mining

Text mining refers to computational methods for the automatic analysis of semi-structured text (i.e., text written in natural language). Text mining for molecular biology research has gained considerable attention

in recent years (Blaschke et al., 2002; de Bruijn and Martin, 2002). This happened mainly due to the increasing availability of full text papers and the development of high-throughput methods which generate thousands of data items per experiment, each item conveying a small piece of information, such as gene expression or proteome profiling. In such projects, it is vital to be able to enrich the primary experimental data with additional information about the objects being examined. Text mining aims at automatically finding such information by analyzing existing publications, a task which is otherwise performed manually. Typical applications of text mining in the life sciences are extraction of protein interactions (Koike et al., 2003; Proux et al., 2000), clustering of genes according to functional descriptions (Raychaudhuri et al., 2002b), or the prediction of sub-cellular locations (Stapley et al., 2002).

Text mining combines statistical methods from machine learning, exploiting differences in word distributions or word co-occurrence, with natural language processing (NLP), which tries to understand natural language by analyzing grammatical structures of sentences and paragraphs. Text mining algorithms require an intensive preprocessing phase in which documents are retrieved and converted to ASCII text, word and sentence boundaries are recognized, and words are reduced to their respective stems. Documents are eventually represented as a high dimensional document vector, where each dimension represents a word stem in the document collection, and the word coordinate corresponds to some measure for the frequency of this word in the document combined with a measure for the frequency of the word in the entire collection. Additional dimensions can be generated from the documents (e.g. to represent authorship, number and type of figures, length, year of publication).

We approached the problem of finding papers containing kinetic parameters by reformulating it as a classification problem, which we solved using support vector machines (SVM) (Vapnik, 1995). The SVM is trained using a set of manually labeled documents. Using this set, the SVM learns to discriminate documents which are relevant from those which are not relevant by finding discriminatory properties of the document vectors. For instance, papers on kinetic data frequently contain phrases such as:

- "(..) J_{rel} = (t/12.0 ms)exp(−t/12.0 ms)0.00487 mM/ms (..)"
- "(..) high affinity for calcium ($K_{pvca} = 10^7 - 10^9$ M^{-1}) (..)"
- "(..) Km values for methionine of 24 microM and approx. 1.8 mM (..)"

Given such sentences in positive examples, the SVM will correlate the appearance of words such as km, microM, or affinity with a positive prediction. Once the training is finished, the SVM has learned a model which can be applied to classify arbitrary new documents, that is documents that have not been processed within the training phase. Since the classification is never perfect, we carefully analyzed the results in terms of classification precision and accuracy.

Papers available online are mostly provided in PDF or PostScript format. For mere processing, the most adequate format is plain ASCII text, lacking markup or formatting instructions and containing every single word in clear text. It is far easier to implement computer routines for analyzing ASCII text than from PDF. A common and simple way to convert from PDF to ASCII formats is the usage of tools like PDFTOTEXT (Noonburg, 1996). Nevertheless, problems arise even during this basic step. For instance, some converters do not work properly for all versions of PDF. Others have problems with handling multi-column texts, which is used for most journals layouts. Hyphenation leads to additional difficulties and possible ambiguities. Important information often is presented in tables, and some tools do not solve the conversion properly (Table 1).

Extraction of features

Despite a mere extraction of single term occurrences from texts, one can think of more ways to appropriately describe and represent documents by using the latter's properties. Terms are given explicitly in the texts and are therefore easy to identify. This subtask is called tokenization, resulting in a list of words (called tokens, in the same order as in the document), identified by defined word-boundaries. Words can be separated from each other by white spaces or a variety of punctuation marks, such as colons, periods, parenthesis, or quotation marks. Regrettably, not every such mark delimits single words. For instance, parentheses can be parts of compound words, or delimit words, as in

- "(..) these cells produced P2Y(11) mRNA (..)" and
- "(..) the neighbouring Cuneate nucleus (Cu) area (..)",

Table 1. The 16 Words with the Highest *t*-Values

Term	t-*value*
K_m	8.91
Half-life	8.66
V_{max}	7.95
Turnover	6.75
Enzyme	5.85
Activity	5.26
Radation	5.24
Di-	5.18
Taining	5.14
Michaelis-Menten	5.13
Lineweaver-Burk	5.09
Degradation	4.86
7-fold	4.86
V_{max}/K_m	4.79
Degrade	4.68
Enzymatic	4.65

The list coincides well with what a expert user would use as keywords, except for the terms *radation, taining,* and *di-*, which are probably artifacts of the PDF converter.

plus one might find quotation marks in

- "(..) using lithium-3′,5′-diiodosalicylate, 5′ SAR is (..)"

An additional problem comes with the separation of sentences. In general, each period delimits two sentences. But in some cases, there are ambiguities regarding the correct splitting, e.g.

- "(..) agar diffusion method for Y. enterocolitica, whereas (..)"
- "(..) covers 866 b.p. Immediately upstream, (..)"
- "(..) and another 10 b.p. T-rich stretch. In conclusion, (..)"

All these cases might not be regarded as problematic by a human reader. Even a novice within a particular field of research would be able to resolve most of the ambiguities of this kind. Defining rules and patterns for automatically analyzing texts on this basis, on the other hand, is a challenge. While this task might be much easier for prose of newspaper articles, it gets more complicated for scientific texts. Here, we encounter many standard and non-standard abbreviations, or large and complex compound names (e.g., protein names).

Feature generation

Having a text in its tokenized form is the first step toward feature extraction. Not only every single token should be regarded as a feature of the document, as some words or even phrases describe the same activity, entity, or concept. The simplest way to group words lies in finding their correct word stems. Terms like activates, activated or activation can then be counted as occurrences of the more abstract term activat.

Document representation

The representation most often used for the application of certain machine learning techniques is the vector space model (VSM) (Salton et al., 1975). This model describes each document as a set of properties, called features, and their respective relevancies to this document. This leads to a comparable representation of texts, regardless of their prior format, size, or structure (book, journal, article, paragraph). It becomes ir-

relevant whether the important and needed information is presented in the Results or the Methods section of a research article, or what the exact contained facts are (e.g., differences in nomenclature usage or spelling variants). Another advantage is the learnability of such vector formats, as machine learning techniques can easily gather hints on the importance and influence of a particular fact (a feature) or certain combinations of those.

Representing documents using the VSM, a fixed vector of features observed in the entire document collection (a feature vector) is calculated. Next, for each single document, an instance of this feature vector is filled with values describing the relevance of each feature for this particular document.

Some features or properties might be present (to some degree) in one document, but absent in others. A single document can contain a certain term, with a certain number of occurrences, or not. The corresponding coordinate in the document vector, an instance of the feature vector, is assigned a value depicting this occurrence, that is, the term frequency (tf).

After tokenization and stemming of the texts, a fixed feature vector can be extracted consisting of every word stem encountered and the number of documents containing this particular stem at least once (see below). We then fill instances of the feature vector with the corresponding occurrences of each term for this particular document, resulting in one document vector per publication. As an example, look at a simple collection of two short documents, containing only one single sentence each:

- doc_1: "*K+/H+ exchanging activity* can *explain the* lower *K+* contents of NHA1 cells grown without *K+* limitation."
- doc_2: "This behaviour *was* reported in *the* Sctrk1D mutant, and *was* ascribed to an *active K+/H+ exchanger*."

This example would result in the feature and document vectors presented in Table 2. The underlying approach is called a bag-of-words, as all words are represented by their frequency only, regardless of co-occurrences, collocations and context. Nevertheless, a huge variety of other document features can be extracted or generated, as we will present in the next section. Additionally, one might think of different weighting schemes to represent the significance of a term for describing a certain document. Most weighting schemes (Glenisson et al., 2003; Strasberg et al., 2000) comprise a combination of a term's local weight (i.e., within the document) and its global weight (i.e., in the document collection). In our approach, we used the *tf*idf* metric, combining the term frequency with the inverse document frequency, calculated as

$$tf*idf = (1 + \log tf_{t,d}) * (1 + \log (N/df_t)) \quad (14)$$

where $tf_{t,d}$ is the number of occurrences of term t in document d, N is the number of documents, and df_t is the number of documents containing t (called document frequency). The *tf* assigns high values to the most common terms in a document, the *idf* weight favors generally uncommon terms. Therefore, the more often a certain term appears in a single document, but the lesser it occurs throughout the document collection, the higher its weight in describing this specific document. Note that taking the logarithm of each frequency decreases the

Table 2. Sample Feature and Document Vectors for the Two Documents doc_1 and doc_2

Vector of word stems	*Number of documents containing the stem*	*Feature vector for doc_1 (simple occurrences only)*	*Feature vector for doc_2 (simple occurrences only)*
exchang	2	1	1
activ	2	1	1
K+	2	3	1
H+	2	1	1
report	1		1
explain	1	1	
was	2		2
the	2	1	1

Instead of *tf*idf* weights, we show the simple occurrence statistics for the word stems only.

influence of large N or $tf_{t,d}$. Adding one to each logarithm prevents the values from being zero in cases where a term occurs in every single document ($N = df_t$) or a term occurs exactly once in a document ($tf_{t,d} = 1$).

Dimensionality reduction by feature set transformation

Using a multitude of terms, each forming a coordinate, leads to very high dimensions of the vector space we describe our documents in. These extreme dimensionalities negatively affect the computing performance. On the other hand, the more information is used to describe the documents, the more precise is the representation for applying machine learning algorithms. The problem of overfitting is another important issue. It means that the learned classifier fits the selected training data too much, degrading the generalization performance. While the model predicts the labels of the training set very precisely, unseen data could be predicted badly. Especially for large dimensions, avoiding overfitting becomes very important—here, the number of training objects (i.e., documents) often is much smaller than the number of features describing these objects.

It is a major step to find an appropriate balance by selecting useful information (i.e., terms) only. To pick the most relevant features of a document (or the whole document collection), we followed different ideas. When describing documents using the contained words, it is unnecessary to list every single word (or stem). In every language, there are a lot of so-called stop-words, common terms which do not provide any information toward discriminating documents, as they tend to appear with the same frequency in every kind of text (e.g., and, are, it, with). These words can be filtered before further processing the remaining text, as long as they are not contained in basic parts of a concept. On the other hand, the least frequent words, appearing in only a few (or a single) document, provide just as little help. They might be simple misspellings, infrequent spellings, uncommon usages of out-dated identifiers, or author names. A pruning of such words helps to reduce the dimensionality of the vector space.

Different methods for the principal component analysis (PCA) have been proposed. In the following, we describe Student's t-statistic (Gosset, 1908; Ewens and Grant, 2001) to rank and select the most convenient terms as an example. For each term in a document collection, a t-value is calculated as:

$$t = \frac{|\bar{X} - \bar{Y}|}{\sqrt{\frac{S_X{}^2}{m} + \frac{S_Y{}^2}{n}}} \tag{13}$$

$\bar{X}$ and $\bar{Y}$ are the mean values for the occurrence of the term in the classes X and Y (positive and negative), S is the standard deviation regarding X and Y, and m and n are the number of elements in X and Y, respectively. The t-values for each term occurring in both the positive and the negative training set state their ability to discriminate both classes, with high values depicting the most appropriate terms.

The t-values calculated for each term allow for an easily adoptable feature vector size. As all terms are sorted by their corresponding t-value, only the n terms allowing for the best discrimination of the two classes can be considered. Each document is then represented by the occurrence statistics of these n terms in the document. This step reduces the size of the feature vectors and therefore the dimensionality of the vector space to n.

Statistical learning

Traditionally, machine learning theory evolved as an area of artificial intelligence. It deals with the development of computational learning techniques. Learning in this case always means deducing an appropriate model by automated analysis of data sets. The model is then applied to gather new knowledge from new and previously unseen data. Common techniques of machine learning can be summarized and subsumed into four categories:

- Supervised learning: finding a function which maps input data to a set of outputs
- Unsupervised learning: gaining insights in the structure of inputs
- Reinforcement learning: learning to react to observations
- 'Learning to learn': learning an inductive bias, based on previous experience

(for an introduction to machine learning, see Mitchell, 1997).

In text or data mining, supervised learning is often referred to as a classification of data or learning from labeled samples. Given a (manually) labeled set of training examples, a machine learning algorithm learns a mapping from input data (the documents) to a fixed set of output data (the labels, or categories). For the training set, these labels are known, and the result of the algorithm is a model which predicts the label of new data. In our case, we find a binary classification problem: deciding on the relevance or irrelevance of texts concerning their usefulness for kinetic modeling. In contrast, the so-called clustering techniques do not rely on labeled data, but depend on similarity measures to compare to data objects (e.g., texts). These algorithms result in clusters or groups of documents, each with a certain inner or pairwise similarity.

Reinforcement learning describes the learning of appropriate actions to control sequential processes. An agent explores its environment by perceiving its current state and proposing possible actions. These actions are graded and rewarded by the environment, and the agent tries to maximize the rewards accumulated during the process.

Support vector machines

The statistical learning technique used in our approach are the SVM (Vapnik, 1995). We encounter a binary classification problem (a document can either be relevant or irrelevant) and can easily transform each document into a sparse representation using the vector space model with the documents' features. A sparse representation of a document omits all vector coordinates with a value of zero (i.e., all terms non-existent in this particular document).

The representation of documents using their document vectors in a high-dimensional vector space would ideally lead to two easily distinguishable clouds of objects. These clouds would represent the groups of relevant and irrelevant documents. A separation hyperplane between both clouds could then be used for a classification decision: a new object can be found either on the relevant or the irrelevant side of the hyperplane (Fig. 3). Thus, the norm, a vector orthogonal to the hyperplane and passing through the new object, would

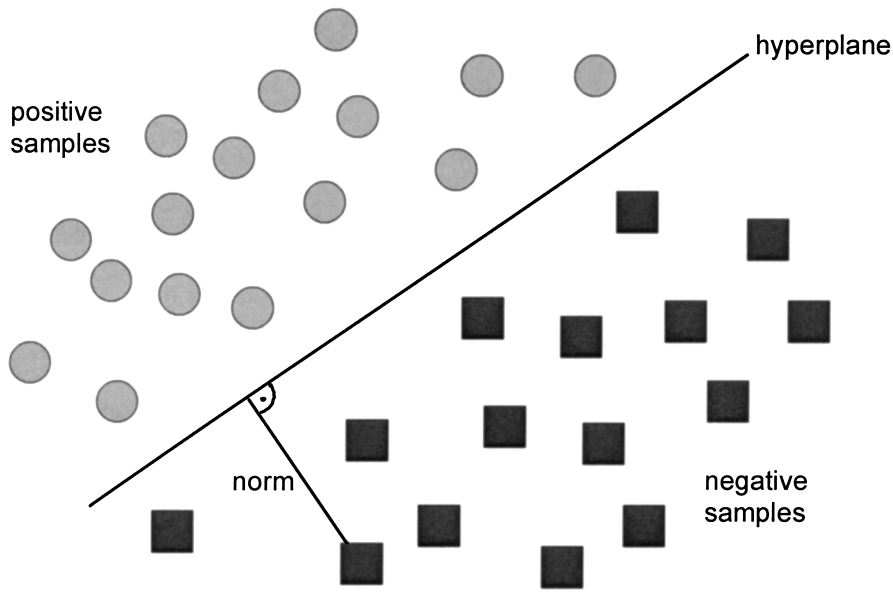


FIG. 3. Separation of the positive and negative class samples by a linear hyperplane. Circles, positive samples (relevant documents); squares, negative samples (irrelevant documents).

have a certain distance and direction from the hyperplane, where its sign alone can be used to predict the objects' position and class;

$$y' = sign(\langle w,x' \rangle + b) \tag{15}$$

with $y' \in \{-1,+1\}$ being the predicted class (-1: irrelevant, $+1$: relevant) of x' and (w,b) determining the hyperplane separating both classes.

The purpose of SVMs is to calculate a representation of the hyperplane which best separates both classes of objects in the vector space. Additionally, a maximal margin is introduced, in such a way that all samples lie outside a certain distance from the hyperplane. The margins on either side are called the margin hyperplanes H_1 and H_2:

$$H_1 : \langle w,x \rangle + b = -1$$
$$H_2 : \langle w,x \rangle + b = +1 \tag{16}$$

For this purpose, from both classes of objects sets of support vectors are identified, representing either class, and lying on the corresponding margin hyperplane (Fig. 4). Support vectors are the most critical members of the existing document vectors, and are selected due to their information content toward building a discriminating hyperplane. Extreme outliers do have a smaller impact than vectors in the *twilight zone*, where both classes might form almost intersecting parts.

Linear learning machines can be constructed on this basis using a dual description. Representing the normal vector w as a linear combination of training samples,

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{17}$$

(l = number of training samples x with label y; α_i = scalar coefficients), we can reformulate the decision function for classifying a new documents x' as

$$f(x') = sign\left(\sum_{i=1}^{l} \alpha_i y_i \langle x_i,x' \rangle + b\right) \tag{18}$$

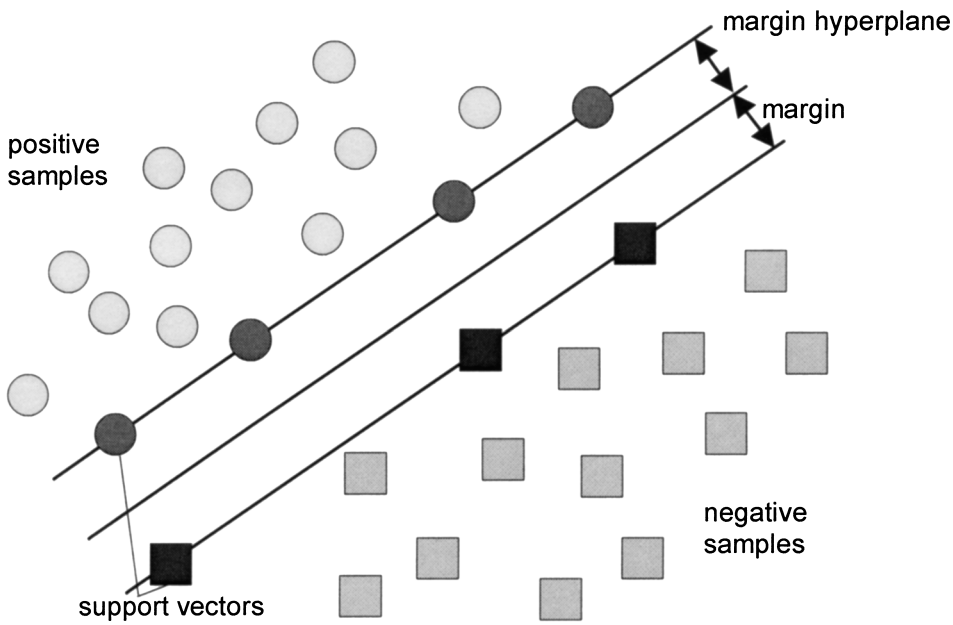


FIG. 4. Margin hyperplanes separating both classes. Objects representing support vectors (dark) lie on one of the two margin hyperplanes.

Training samples and new objects never act through their individual attributes any more, but appear as a pairwise inner product, with α_i being the so-called dual variables to be calculated.

In all cases mentioned above, the two datasets are separable by a simple linear classifier. This is generally not the case for most problems. When dealing with documents, there will always be some overlapping parts (i.e., terms occurring with similar frequencies in at least a certain amount of samples from both classes). The idea to overcome this problem is to map the input data to an even higher dimensional space (called a Hilbert feature space H), where a linear separating hyperplane exists. This approach is called the kernel technique and was first introduced in (Aizerman et al., 1964) as the method of potential functions. A kernel function can easily be calculated as a function in the original vector space, but behaves like a inner product in H (for further details, see Joachims, 2002; Cristianini et al., 2000; Markowetz, 2001). While tuning our system, we tested linear and polynomial kernels.

SVMs are perfectly suitable for classification tasks in many applications and have been proven to lead to produce acceptable results (Lewis, 1997). SVMs are very easy to use, and provide a good and robust performance in terms of accuracy and speed. The computational efficiency is due to the absence of local minima and the sparse represenation of feature vectors. In contrast to some other statistical learning methods (e.g., neural networks), the model (i.e., the support vectors) can give hints for interpreting the distribution of objects and their inherent structure. Another major advantage is the omission of strong hypothesis on the data. Naive Bayesian learning, for instance, relies on the assumption that (single word) terms occur completely independent from each other, which is definitely not the case. In contrast to rule-based methods, expert knowledge and heuristics are not required for training a SVM − besides manual annotation of the training sample, a process needed for all supervised learning approaches.

Evaluation

To evaluate how good different systems perform on similar tasks, there are various measures indicating the quality of their predictions. The precision of a method in our case states the percentage of documents correctly labeled as positive or relevant. It is influenced by the number of true positive predictions (TP) and the number of false negative predictions (FN; Table 3). The recall shows how many of the relevant documents the systems finds (using TP and the number of false positive predictions [FP]):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{19a,b}$$

For instance, a precision rate of 70% would indicate that from all documents predicted as relevant, 70% are relevant and 30% are irrelevant in truth. A recall of 80% describes the fact that of all relevant documents as labeled by a human expert, 80% have been found by the system, but 20% are missing. Ideally, one aims for a classification algorithm that gives 100% recall (all positive examples are found) and 100% precision (all examples that are predicted as positive, are positive). Unfortunately, there is a natural tradeoff between recall and precision. A high recall is often associated with a low precision and vice versa.

Table 3. Categories for the Four Possible Outcomes of a Prediction

	True label of the document (reality)	
	Relevant	*Irrelevant*
Predicted label (system)		
Relevant	TP = true positive	FP = false positive
Irrelevant	FN = false negative	TN = true negative

Depending on the label (i.e., class) of a document as provided by a gold standard (e.g., human expert) which reflects the reality, a prediction can be either true or false. For instance, a FN indicates that the document is relevant (i.e., positive) in reality, but the system falsely predicted it to be irrelevant. A document irrelevant in reality and predicted as such would be marked as a TN.

MATERIALS AND METHODS

In this section, we shall discuss the methods and techniques we combine to a work flow for the gathering and representation of publications, and the training, validation, and usage of a SVM to classify publications according to their relevance for kinetic modeling. An overview is shown in Figure 5.

Corpus generation

We developed a program that automatically retrieves publications in PDF format from a set of online journals. For this purpose we chose twelve journals that focus on biological areas that make use of kinetic data (*Journal of Biological Chemistry, Proceedings of the National Academy of Sciences, Yeast, Journal of Medical Microbiology, Biophysical Journal, Cell, FASEB Journal, Journal of Cell Biology, Journal of Cell Science, Molecular and Cell Biology, Molecular Biology of the Cell*, and *Molecular Cell*). We selected 4582 publications at random, covering the years from 1993 to 2003. To estimate the overall frequency of publications of those journals that do contain relevant kinetic data, a random sample of 200 publications were read and classified by an expert. It turned out that 12% of all papers were positive, meaning that they contained parameter information relevant to kinetic modeling (Fig. 2). With this sample size, the 95% confidence interval (CI) for this proportion extends from 8% to 17.5%. This means that there is a 5% chance that the entire collection would contain less than 8% or more than 17.5% of positive documents.

In a first preliminary step we tried to enrich the fraction of positive papers by selecting candidate articles based on a keyword search. The idea is that every article relevant to kinetic modeling makes use of certain common terms to present its subject. Irrelevant articles, on the other hand, will definitely not contain any of these terms. Simple tools like GREP provide the possibility to filter complete collections of papers in a serializable and performant process. After conversion to plain text, only those publications were chosen that contained at least one of the following keywords: Km, Menten, vmax, kcat, keq, hill kinetic, hill equation, enzyme kinetics, kinetic parameter, kinetic data, kinetic model, mathematical model, rate equation, half-life, and half life. After this selection step, 791 papers remained. Those were read carefully by an expert. 155 of these publications actually contained relevant kinetic data, leaving 636 which had to be read

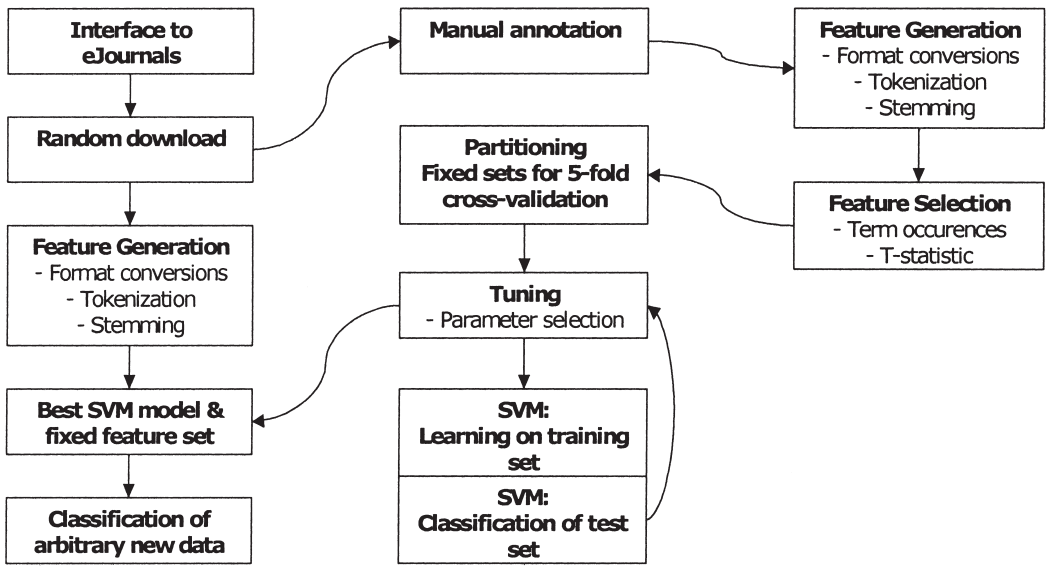


FIG. 5. Work flow for gathering and processing of publications, and training, validation and usage of a model learned by a SVM. After choosing the optimal model determined by the cross-validation, only the steps on the left side are needed to classify arbitrary new publications. Methods for feature generation are identical in training and application phases.

but revealed no useful information. Thus, the overall fraction of positive publications was 19.5% with a 95%-CI of 16.8% to 22.4% (Schmeier et al., 2003).

The work presented in this paper aims at improving this process with text mining methods.

Extraction and generation of features

We use the TreeTagger kit (Schmid, 1994) for feature generation. This kit provides a tokenizer, part-of-speech-tagger, and stemmer as a single command line application. It first splits a text into its sentences, and then performs a tokenization of each sentence using default word-boundaries, such as blanks, colons, commas etc. For the splitting of sentences, a list of common abbreviations (using periods) exists, such that a sentence remains intact. The part-of-speech tagger is a probabilistic tagger which estimates transition probabilities with decision trees. It uses the common Penn-Treebank tagset to label the tokens (Santorini, 1990). An exemplary output of the TreeTagger is given in Table 4. In this example, both the tokens protein and Proteins stem from *protein* and therefore should be counted as an occurrence of the same feature.

Dimensionality reduction

Stemming and exclusion of words appearing only once in the whole document collection led to a vector size of roughly 66.000 word stems (down from more than 100.000 terms). Table 1 contains the 16 terms with the highest *t*-value as examples, and Figure 6 plots the distribution of *t*-values within the 66.000 terms. There is a remarkably rapid drop of *t*-values in the first ≈2000 words, from where on the remaining 64.000 have a relatively constant, yet very small *t*-value. This range for instance contains common stop words such as *and* (rank 40.395, *t*-value 0.5) or *the* (rank 64.600, *t*-value 0.1). Those words are considered inappropriate for document discrimination.

Support Vector Machine (SVM^{light})

We used the SVM implementation SVM^{light} (Joachims, 1998). From a list of labeled examples, represented by their respective feature vectors, the SVM learns a model. This model consists of the predicted support vectors for each hypothesis (i.e., positive/relevant and negative/irrelevant classes). In a second step, the unlabeled (because new) documents are assigned a class based upon the learned model. The feature vector of these new documents have to be calculated from the same, fixed vector used before. A SVM classifier then computes the distance from each object to the separating hyperplane, and the output of SVM^{light} contains these distance vectors and their direction indicated by a positive or negative sign. Values greater or equal than zero in our case depict a documents' positive label and thus comprise all relevant publica-

Table 4. A Sample Output from the TreeTagger Kit

Token	*Part-of-speech-tag*	*Stem*
the	DT	the
protein	NN	*protein*
data	NNS	datum
extracted	VVN	extract
from	IN	From
various	JJ	various
sources	NNS	source
.	SENT	.
Proteins	NPS	*protein*
are	VBP	be

The sentences (parts shown in the first column, one line per token) are analyzed with lexical and contextual rules and the most likely Part-of-speech-tag is then assigned to each token. Additionally, the tool produces the word stem needed for our methods.

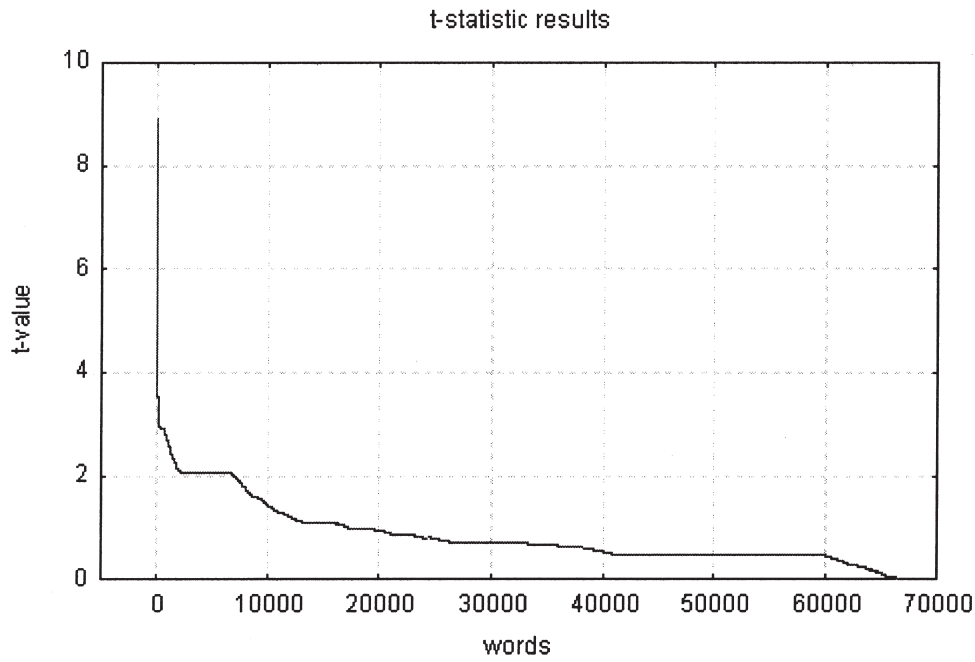


FIG. 6. Distribution of the *t*-values from all 66.616 single words.

tions found in the new collection. SVM^{light} is a command line tool and allows for large scale batch processing and adaptions in parameter settings. The latter will be described in the following section.

Tuning the system

We performed a systematic parameter optimisation to find the best set of parameters for the support vector machine. In our setup there are four parameters that can be optimized. First there is the feature vector length. With this parameter it is possible to control overfitting and reducing the computation time for the optimization step. In our case the first argument is of greater relevance, since the time needed for the calculation is not critical for the application.

Overfitting occurs when the dimensionality of the underlying feature vector is too high and so two classes are separable too easily. We varied this parameter from length 50 to length 1000 in steps of 50 words. Furthermore, we investigated linear and polynomial kernels of degree two and three. To keep the computational effort in reasonable bounds, we restricted ourselves to these kernels. Finally, different values for the earlier described kernel parameters c and j were investigated. We varied the c-values on a logarithmic scale from 10^{-5} to 10^{5}. The j-value can be used to apply different weights for the penalties for misclassifying positive and negative examples. This parameter was varied from 0.1 to 1 in steps of 0.1 and from 1 to 9 in steps of 2. To find the optimal values for the four parameters, we ran the SVM^{light} 9240 times with 791 articles and different settings.

Validation

The validation was done using a five fold cross-validation. For this purpose, we randomly split the overall corpus (791 papers) in five sets of equal size. Each of the five subsets is consecutively chosen as test set while the other four subsets were used as training sets. For a given set of parameters the SVMs were trained on the training set and evaluated on the test set (Table 5). Finally, we computed the average precision and recall values across all five trials. The advantage of this method is that in summary all data are used for training as well as for testing and thus best use is made of the available data resources. The disadvantage is that the training algorithm has to be executed five times (for a five fold cross validation), leading of course to a five fold increased computing time.

TABLE 5. PARTITIONING OF 791 ANNOTATED PAPERS FOR THE FIVE FOLD CROSS-VALIDATION

4 sets for training ≈ 632 publications		*1 set for testing ≈ 159 publications*	
≈123 relevant publications	≈508 irrelevant publications	≈32 relevant publications	≈128 irrelevant publications

Five groups of equal size were chosen as fixed sets. In strict rotation, each set acted as the test set, while the SVM trained its model on the other four combined sets. We chose the members for each set at random to maintain the relevant/irrelevant ratio of about 1 : 4 within each set.

To calculate the parameter optimization with all 9240 different parameter combinations and fivefold cross-validation took 625 min on an AMD Athlon™ MP 2200+ Linux machine with 2 GB of memory and two CPUs.

RESULTS

SVM parameter optimization

Figure 7 shows a scatter plot of the recall versus precision values of all 9240 tested parameter combinations. A few parameter combinations gave a very high precision (100%), but the maximal attainable recall was only 11% (point A). Other parameter combinations lead to 100% recall, but precision is down to 20% (point C). In this case, the SVM simply predicted all examples as being positive. All positive examples are now predicted to be positive (100% recall), but specificity is down to the lowest possible value, the fraction of positive examples in the total corpus (20%). At point B, predictions yield a precision of 60% at 50% recall. For low recall values, the points form a geometric pattern. This is a consequence of how recall and precision are calculated as fractions of natural numbers. For small numbers this results in discrete values, which form geometric patterns.

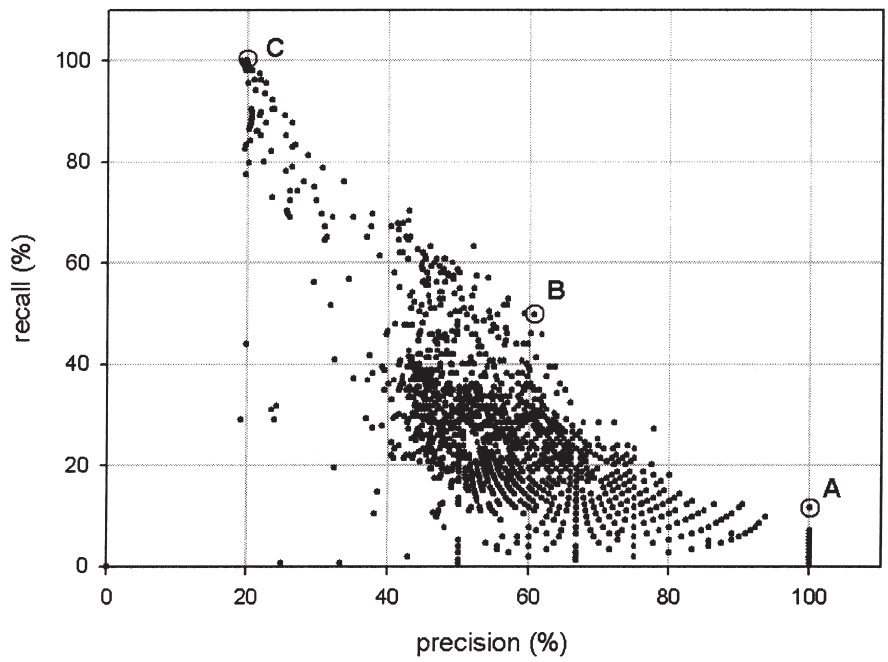


FIG. 7. Recall versus precision values of the 9240 different parameter combinations that were tested for the support vector machine (SVM). Each point is the result of a fivefold cross validation over the corpus of 793 publications. Points A, B, and C are discussed in the text.

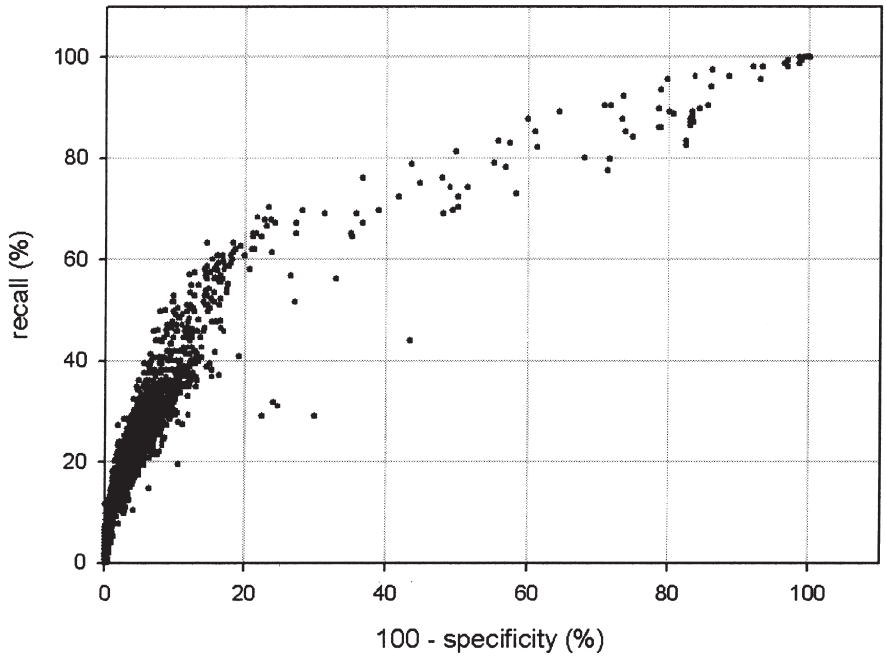


FIG. 8. ROC curve of the results of the SVM parameter optimization. Instead of recall versus precision, as in Figure 5, here recall versus the complement of specificity (100-specificity) is shown. Thus, the *y*-axis shows the percentage of true positive versus the percentage of false positives (*x*-axis).

Another commonly used way to display this type of data is the ROC (receiver operating characteristic) curve shown in Figure 8. Basically, the percentage of true positive versus the percentage of false positives is displayed. In this case the ideal shape is rectangular, with the optimal point having 100% true positives and 0% false positives. Figure 8 shows that roughly we can get 60% true positives (recall) with only 20% of the false positives.

Single parameter influence

For the SVM optimization shown in Figures 7 and 8 four different parameters were varied (kernel type, two kernel specific parameters and the feature vector length). To demonstrate how individual parameters influence the performance of the SVM we picked two points, indicated as "A" and "B" in Figure 7 and varied one parameter, while the others were kept constant. Point "A" has the highest recall of all parameter combinations that gave 100% precision and is therefore in some sense the best point. However, the recall value of 11% is still quite low. We therefore also wanted to test a point with a significantly higher recall and chose point "B" that displays similarly good recall and precision values. Table 6 shows the details of the parameters used for points "A" and "B."

TABLE 6. DETAILS OF THE KERNEL TYPE AND PARAMETER VALUES USED FOR POINT "A" AND "B" THAT ARE MARKED IN FIGURE 7

	Point A	*Point B*
Kernel	Polynomial	Polynomial
Degree	Cubic	Quadratic
c-value	0.0001	0.01
j-value	3	7
Vector length	250	100

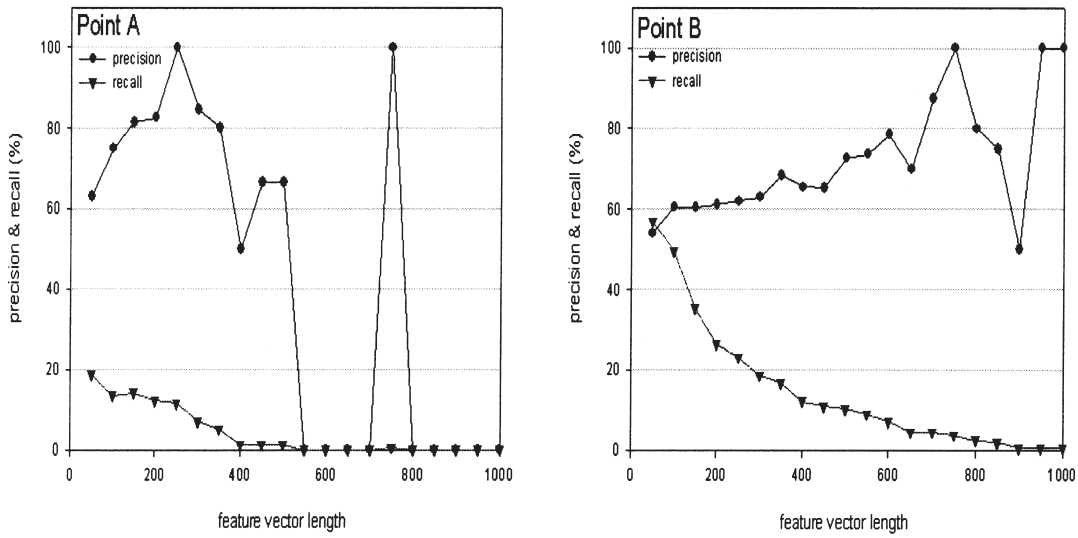


FIG. 9. Influence of the feature vector length on recall and precision of points A (left) and B (right) from Figure 5. For low recall values (large feature vector lengths) the calculation of the precision values, true positives/(true positives + false positives), becomes unreliable because of the small absolute numbers. The 100% precision at a vector length of 750, for example, was calculated from one true positive and no false positives.

Figure 9 shows how varying the length of the feature vector changes precision and recall for points "A" and "B." In both cases recall dropped with increasing vector length and approached zero. The behavior of the precision seems to differ for large vector length, going down for point "A," but increasing for point "B." However, if the recall gets very low the total number of positive signals from the SVM becomes very small, leading to large uncertainties in the calculation of the precision.

The c-value is a kernel parameter for the SVM that controls the trade-off between training error and margin of the SVM. As Figure 10 shows, there are some differences between point "A" and "B." Most importantly, for point "A" small c-values lead to decline of the recall, while for "B" the recall reaches 100%.

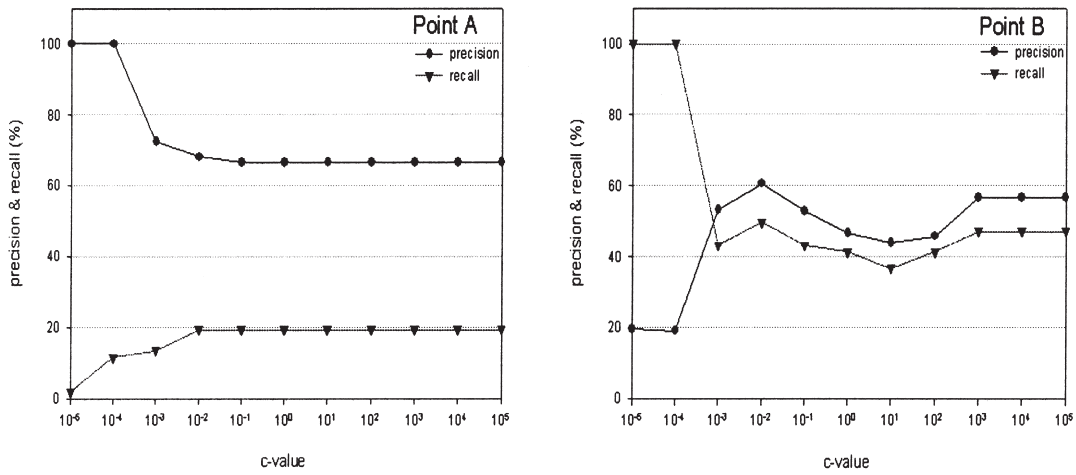


FIG. 10. Influence of the value of the kernel parameter *c* on recall and precision of points A (left) and B (right) from Figure 5. Above a *c*-value of approximately 0.01, precision and recall vary only moderately with *c* for both points investigated. For very small values, however, recall and precision develop differently for point A and B. While recall tends to zero for point A, it reaches 100% for point "B."

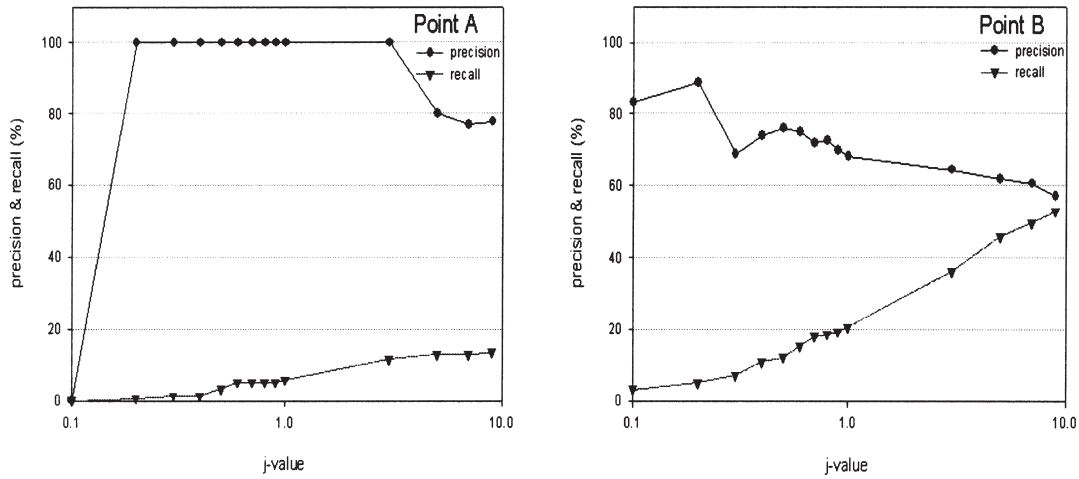


FIG. 11. Influence of the value of the kernel parameter *j* on recall and precision of points A (left) and B (right) from Figure 5. Points A and B behave qualitatively in a similar way, increasing recall and decreasing precision, but the strength of the response is stronger in B than in A. See the text for a description of the meaning of *j*.

This then causes the precision to drop to 20%, the frequency of positive examples in the whole data set. But both points also show a similar behavior in that the values of recall and precision are only weakly affected by c-values greater than 0.01.

The second kernel parameter that was optimized is the j-value. This is a cost-factor, by which training errors on positive examples outweigh errors on negative examples. Here the situation is quite similar for the two points (Fig. 11). A large j-value causes the recall to increase and the precision to decrease. For point "B," however, this trend is much stronger so that at a j-value of 10, recall and precision are of com-

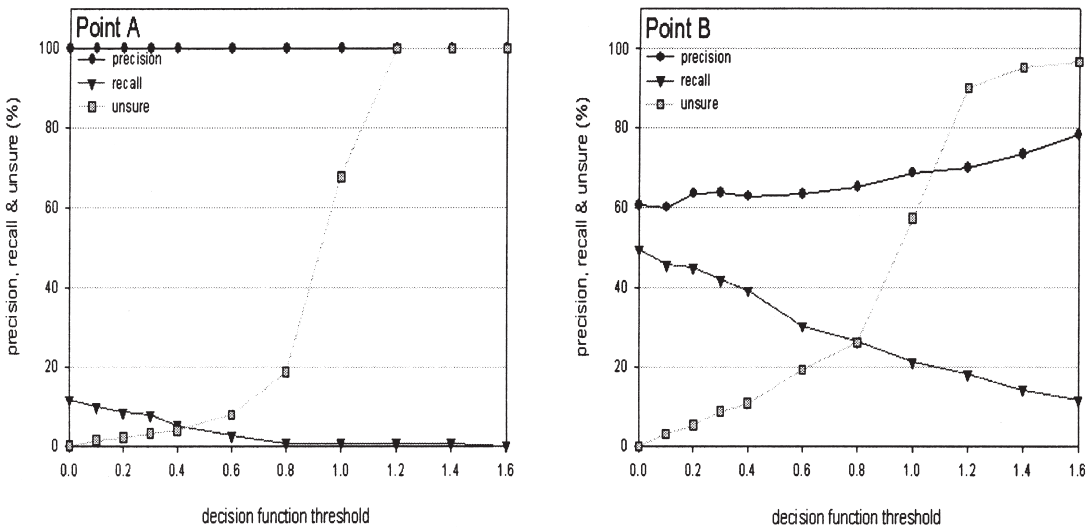


FIG. 12. Variation of the threshold of the SVM decision function. Support vector machines calculate a numerical value that decides how the example is classified. Normally, examples with positive values are predicted to be in one class and examples with negative values are predicted to be in the other class. In the calculation shown here, only examples with values higher than a certain threshold are classified. Examples with values smaller than the threshold belong to a new group termed *unsure*. See the text for more details

parable magnitude (approximately 55%), while for point "A" they are still far apart (14% recall, 78% precision).

Decision function threshold

Support vector machines calculate a numerical value that decides how the example is classified. This value is called decision function and can reach small negative or positive values. All examples for which the decision function is greater than zero are predicted to belong into class one and all examples for which the decision function is smaller than zero are predicted to belong into class two. We were interested to see if the results of the SVM can be improved by being more selective and introducing a threshold value, TV, for the decision function. To be classified, the decision function of an example had to be above +TV or below −TV. Examples whose decision function lies between −TV and +TV are not classified and are referred to as "unsure". The results of this calculation are shown in Figure 12.

As expected the number of examples that fall into the "unsure" group increases with increasing threshold, approaching 100% for a threshold of 1.2 (point A) and 1.6 (point B) respectively. Another trend that is common to both points is the monotonic decline of the recall value. The more examples belong to "unsure" and are thus omitted from classification, the less is the fraction of positive examples in the overall corpus (recall) that is recovered by the SVM. The only parameter that does benefit from the increased selectivity (threshold) is the precision. For point B the precision, the fraction of true positives among all examples that are classified as positive by the SVM, increases from 60% to 80%. For point A this effect cannot be seen, since the precision is already at 100% for a threshold of zero.

DISCUSSION

We are not aware of any other attempts to apply text mining for the detection and extraction of parameters for kinetic modeling. However, document classification is currently studied intensively for supporting database curators in there selection of documents to read (Yeh et al., 2003) or for the assignment of genes to function and vice versa (Rindflesch et al., 1999; Blaschke et al., 1999; Raychaudhuri et al., 2002a). Donaldson et al. (2003) describe a system for the classification of abstracts for consideration in the DIP database. Using support vector machines, they reach precision / recall of over 90%. The approach in Faulstich, 2003 automatically decides whether papers describe conserved RNA structures in certain virus strains. The system reaches 80% recall at 30% precision.

Our own system currently reaches 60% precision at 49% recall and is therefore far from the 92% reported by Donaldson et al. However, comparing classification procedures developed for different purposes and used on different corpora is usually not possible, since every corpus has its specific bias and every problem has its specific characteristics. For instance, the Donaldson corpus consists of about 60% positive and 40% randomly selected negative examples; in our case, even the negative examples are biased since they contain at least one of the 'magic' keywords. Our corpus in this sense consists out of relatively favorable candidates with certain inherent similarities. We think that distinguishing the true positives from the true negatives in our corpus is somewhat harder than for arbitrary data. Future experiments will reveal the precision / recall values concerning the classification of randomly selected and unbiased publications. We expect at least slight improvements of the prediction quality.

Nevertheless, we are anyway confident that the performance of our system can be further improved. This can be based on several ideas. (a) Kinetic data is not only presented in continuous text, but very often in tables and figures. This data is currently inaccessible to our system, since most tables and figures are included into PDF documents as images and are therefore lost during the translation from PDF to ASCII. (b) We are not interested in papers dealing with theoretical aspects of kinetic modeling, where all names of constants and differential equations also appear, but without any real experimental data. Our current system is not capable of separating these two categories. (c) We encountered several problems in the tools we use. For instance, TreeTagger has been developed for newspaper text and sometimes fails to properly separate words and sentences containing special characters. We also found the PdfToText tool we use to have difficulties in correctly concatenating multi-column text. (d) Our system currently cannot recognize equal

entities described by different names, different spellings, or abbreviations. Detecting such synonyms, which is possible using gene and protein name lists as available in database such as FlyBase (FlyBase Consortium, 2003) or SGD™ (Issel-Tarver et al., 2002), would enhance the information content of the model and hence of the classifier.

A necessary and essential task is the storage of documents. This can be done using the standard file system structure, though some database systems provide dedicated facilities to store textual content or whole document collections (for instance, Oracle® Text). Database systems use standard SQL queries to index, search, and analyze text. Different forms of linguistic analysis may be performed as well. The latest improvements (see, e.g. Oracle® version 10^g) even support classification by techniques such as SVM or clustering using k-Means or hierarchical algorithms. Future enhancements of our system are sought to be based upon such data base options.

The ultimate goal of our project is the creation of a database holding kinetic data for various species. One comparable database available is BRENDA (Schomburg et al., 2002), which is manually curated and hence provides few data on only a small set of species. Especially, data on yeast enzymes is included only to a very low extent. Considering our ultimate goal, we view the results of our current system as very encouraging. Compared to the keyword-list approach, we can reach a four times greater precision at the cost of losing approximately 50% of the documents. High precision is the most important feature for this purpose, since the limiting factor are huamsn who have to read the documents to actually extract kinetic parameters. Saving their time is important. In contrast, a low recall is tolerable since the number of documents to be considered can be increased almost to infinity, thus automatically increasing the number of positives documents.

ACKNOWLEDGMENTS

This work is supported by the German Federal Ministry of Education and Research (BMBF) under grant contracts 0312705B and 031U109C.

REFERENCES

AIZERMAN, M.A., BRAVERMAN, E.M., and ROZONOER, L.I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Autom Remote Control **25,** 821–837.

BLASCHKE, C., HIRSCHMAN, L., and VALENCIA, A. (2002). Information extraction in molecular biology. Brief. Bioinform. **3,** 154–165.

CRISTIANINI, N., and SHAWE-TAYLOR, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press), New York.

DE BRUIJN, B., and MARTIN, J. (2002). Literature mining in molecular biology. Presented at the EFMI Workshop on Natural Language: Processing in Biomedical Applications, Nicosia, Cyprus.

DONALDSON, I., MARTIN, J., DE BRUIJN, B., et al. (2003). PreBIND and Textomy—mining the biomedical literature for protein-protein interactions using a support vector machine. BMC Bioinform **4,** 11.

EWENS, W.J., and GRANT, G.R. (2001). *Statistical Methods in Bioinformatics* (Springer-Verlag, New York).

FAULSTICH, L. STADLER, P., THURNER, C., et al. (2003). litsift. Automated text categorization in bibliographic search. Presented at the Workshop on Data Mining and Text Mining in Bioinformatics, ECML/PKDD 2003, Cavtat, Croatia.

FLYBASE CONSORTIUM. (2003). The FlyBase database of the Drosophila genome projects and community literature. Nucleic Acids Res **31,** 172–175.

GLENISSON, P., ANTAL, P., MATHYS, J., et al. (2003). Evaluation of the vector space representation in text-based gene clustering. Proc Pac Symp Biocomp 391–402.

GOSSET, W.S. (1908). The probable error of a mean. Biometrika **6,** 1–25.

GRIGORIEV, A. (2003). On the number of protein-protein interactions in the yeast proteome. Nucleic Acids Res **14,** 4157–4161.

HUCKA, M., FINNEY, A., SAURO, H.M., et al. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics **19,** 524–531.

IDEKER, T., THORSSON, V., RANISH, J.A., et al. (2001). Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. Science **292,** 929–934.

ISSEL-TARVER L., CHRISTIE, K.R., DOLINSKI, K., et al. (2002). Saccharomyces genome database. Methods Enzymol **350,** 329–346.

JOACHIMS, T. (1998). Text categorization with support vector machines: learning with many relevant features. Presented at the European Conference of Machine Learning.

JOACHIMS, T. (2002). *Learning to Classify Text Using Support Vector Machines*. (Kluwer, Amsterdam).

KITANO, H. (2002). Systems biology: a brief overview. Science **295,** 1662–1664.

KOIKE, A., KOBAYASHI, Y., and TAKAGI, T. (2003) Kinase pathway database: an integrated protein-kinase and NLP-based protein-interaction resource. Genome Res **13,** 1231–1243.

LEWIS, D. (1997). The TREC-5 filtering track. Presented at the Fifth Text Retrieval Conference.

LI, S., ARMSTRONG, C. M., BERTIN, N., et al. (2004). A map of the interactome network of the metazoan *C. elegans*. Science **303,** 540–543.

MARKOWETZ, F. (2001). Support vector machines in bioinformatics [Master's thesis]. Heidelberg: University of Heidelberg.

MITCHELL, T. (1997). *Machine Learning* (McGraw Hill, New York).

NATIONAL LIBRARY OF MEDICINE. (2003). MEDLINE® [On-line]. Available: http://www.ncbi.nih.gov/PubMed/.

NOONBURG, D.B. (1996). PDFTOTEXT [On-line]. Available: http://www.aimnet.com/~derekn/xpdf/.

ORACLE. (2003). Oracle text [On-line]. Available: http://otn.oracle.com/products/text/.

PROUX, D., RECHENMAN, F., and JULLIARD, L. (2000). A pragmatic information extraction strategy for gathering data on genetic interactions. Proc Int Conf Intell Syst Mol Biol **8,** 279–285.

RAYCHAUDHURI, S., CHANG, J.T., SUTPHIN, P.D., et al. (2002a). Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. Genome Res **12,** 203–214.

RAYCHAUDHURI, S., SCHUETZE, H., and ALTMAN, R.B. (2002b). Using text analysis to identify functionally coherent gene groups. Genome Res. **12,** 1582–1590.

SALTON, G. WONG, A., and YANG, C.S. (1975). A vector space model for automatic indexing. Commun ACM **18,** 613–620.

SANTORINI, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project [Technical Report MS-CIS-90-47]. Philadelphia: University of Pennsylvania.

SCHMEIER, S., HAKENBERG, J., KOWALD, A., et al. (2003). Text mining for systems biology using statistical learning methods. Presented at LLWA, Arbeitskreis Knowledge Discovery (AKKD), Karlsruhe, Germany.

SCHMID, H. (1994). Probabilistic part-of-speech tagging using decision trees. TREETAGGER, version 3.1 [On-line]. Available: http://www.ims.uni-suttgart.de/projekte/corplex/TreeTagger/.

SCHOMBURG, I., CHANG, A., and SCHOMBURG, D. (2002). BRENDA, enzyme data and metabolic information. Nucleic Acids Res **30,** 47–49.

STAPLEY, B.J., KELLEY, L.A., and STERNBERG, M.J. (2002). Predicting the sub-cellular location of proteins from text using support vector machines. Proc Pac Symp Biocomput 374–385.

STRASBERG, H.R., MANNING, C.D., RINDFLESCH, T.C., et al. (2000). What's related? Generalizing approaches to related articles in medicine. Presented at the 2000 AMIA Annual Symposium.

VAPNIK, V.N. (1995). *The Nature of Statistical Learning Theory* (Springer-Verlag, New York).

WHEELER, D.L., CHURCH, D.M., FEDERHEN, S., et al. (2003). Database resources of the National Center for Biotechnology. Nucleic Acids Res **31,** 28–33.

YEH, A.S., HIRSCHMAN, L., and MORGAN, A.A. (2003). Evaluation of text data mining for database curation: lessons learned from the KDD Challenge Cup. Bioinformatics **19,** I331–I339.

Address reprint requests to:
Jörg Hakenberg
Humboldt-Universität zu Berlin
Department of Computer Science
Knowledge Management in Bioinformatics
Unter den Linden 6
D-10099 Berlin, Germany

E-mail: hakenberg@informatik.hu-berlin.de