Extraction and Search of Chemical Formulae in Text Documents on the Web *

Bingjun Sun*, Qingzhao Tan*, Prasenjit Mitra*†, C. Lee Giles*†
*Department of Computer Science and Engineering †College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
{bsun,qtan}@cse.psu.edu, {pmitra,giles}@ist.psu.edu

ABSTRACT

Often scientists seek to search for articles on the Web related to a particular chemical. When a scientist searches for a chemical formula using a search engine today, she gets articles where the exact keyword string expressing the chemical formula is found. Searching for the exact occurrence of keywords during searching results in two problems for this domain: a) if the author searches for CH4 and the article has H4C, the article is not returned, and b) ambiguous searches like "He" return all documents where Helium is mentioned as well as documents where the pronoun "he" occurs. To remedy these deficiencies, we propose a chemical formula search engine. To build a chemical formula search engine, we must solve the following problems: 1) extract chemical formulae from text documents, 2) index chemical formulae, and 3) design ranking functions for the chemical formulae. Furthermore, query models are introduced for formula search, and for each a scoring scheme based on features of partial formulae is proposed to measure the relevance of chemical formulae and queries. We evaluate algorithms for identifying chemical formulae in documents using classification methods based on Support Vector Machines (SVM), and a probabilistic model based on conditional random fields (CRF). Different methods for SVM and CRF to tune the trade-off between *recall* and *precision* for imbalanced data are proposed to improve the overall performance. A feature selection method based on frequency and discrimination is used to remove uninformative and redundant features. Experiments show that our approaches to chemical formula extraction work well, especially after trade-off tuning. The results also demonstrate that feature selection can reduce the index size without changing ranked query results much.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering,Query formulation,Retrieval models,Search process*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*; J.2 [**Physical Sciences and Engineering**]: Chemistry

*This work was partially supported by NSF grant 0535656.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

General Terms

Algorithms, Design, Experimentation

Keywords

Chemical formula, entity extraction, support vector machines, conditional random fields, feature boosting, feature selection, query models, similarity search, ranking

1. INTRODUCTION

Increasingly, more scientific documents are being published on the World-Wide-Web. Scientists, especially chemists, often want to search for articles related to particular chemicals. One obvious method to express such searches is by using the chemical formulae of the chemical compounds. Current search engines do not support users searching for documents using chemical formulae. In this work, we show how one can construct a chemical formula search engine, which is one of the E-science application areas [20, 29].

When a scientist searches for a chemical formula using a search engine today, articles are usually returned where the exact keyword string expressing the chemical formula is found. Searching for the exact occurrence of keywords results in two problems for this domain: a) if the author searches for CH4 and the article has H4C, the article is not returned, and b) ambiguous searches like "He" return all documents where Helium is mentioned as well as documents where the pronoun "he" occurs. To remedy these deficiencies, we propose a chemical formula search engine. To build such a search engine, we must solve the following problems: (1) extract chemical formulae from text documents, (2) index chemical formulae, and (3) design ranking functions for the chemical formulae.

Extracting chemical formulae from text documents is a classification problem where the text is classified into two classes: a) chemical formulae and b) other text. Each chemical formula is then transformed into a canonical form (e.g. $ND4$ into ${}^2H4N$) and indexed to enable fast searches in response to queries posed by end-users. Furthermore, we also propose and provide the semantics of four types of queries to allow for fuzzy searches for chemical formulae. We have also devised a scoring scheme for each of these query types, based on features of partial formulae, to measure the relevance of chemical formulae and queries.

A simple rule-based algorithm can be designed to check if words are composed of the symbols of chemical elements and numbers. While this rule-based algorithm can give high

recall identifying all chemical formulae, it results in low precision because for terms like "He", "I", etc., the algorithm has to decide whether the term is a chemical formula or a pronoun or other non-formula terms. Because natural language understanding is a hard unsolved problem, we employ a classification algorithm based on the statistics of the context of the occurrence of a term to determine whether it is a chemical formula or not.

This problem can be addressed in two stages. The first stage is that of chemical formula extraction. Previous research on detecting names [5], biological entities [17], or even advertising keywords [28] uses a broad range of techniques from rule-based methods to machine-learning based ones. Among these approaches, the machine-learning-based approaches utilizing domain knowledge perform the best because they can mine implicit rules as well as utilize prior domain knowledge in statistical models to improve the overall performance. In the second stage, the chemical formulae are indexed and ranked against user queries. For the second stage of formula search, there are two categories of related research issues and previous work. One is how to represent and index patterns (graphs or formulae), including feature selection of substructures [26]. Indexing graphs or formulae is important to support various query models especially similarity searches, which compare the internal structures. The second set of issues involve data mining, such as mining frequent substructures [6, 11], and similarity structure search [25, 7, 19, 27], which use some specific methods to measure the similarity of two patterns. However, no previous research has addressed the issue of extracting and searching for chemical formulae in text documents.

The major contribution of this paper is to show how to build a chemical formula search engine. We evaluate algorithms for identifying chemical formulae in documents using classification methods based on SVM, and a probabilistic model based on CRF. Similar to decision threshold adjustment in the testing of SVM, a novel method of feature boosting for different classes is introduced to improve the performance of CRF, by tuning the trade-off of *recall* and *precision* of the true class for imbalanced data (i.e., where the number of occurrences of entities belonging to the two classes are substantially different). We propose a sequential feature selection algorithm, which first mines frequent substructures, and then selects features from shorter to longer partial formulae, based on criteria of feature frequency and discrimination with respect to the current set of selected features. A partial formula (e.g. *COOH*) is defined as a partial sequence of a formula (e.g. *CH3COOH*). We then propose four basic types of formula search queries: exact search, frequency search, substructure search, and similarity search. Finally, we describe relevance scoring functions corresponding to the types of queries. Our formula search engine is an integral part of $Chem^X Seer$, a digital library for chemistry and embeds the formula search into document search by query rewrite and expansion (Figure 1).

The rest of this paper is organized as follows: Section 2 reviews related works. In Section 3, we present entity-extraction approaches based on SVM and CRF, improve these methods based on decision-threshold tuning and feature boosting, and discuss the feature set used in our research. Section 4 introduces the sequential feature-selection algorithm for index construction of partial formulae, four categories of formula query models, and corresponding scoring functions. In Section 5, experiments about formula extraction, indexing, and search are described, and results are presented and discussed to show that our methods work well. Conclusions and some future directions of the research work are discussed in Section 6.

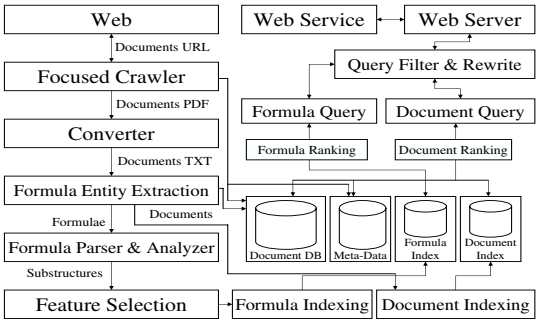


Figure 1: Architecture of $Chem^X Seer$ Formula Search and Document Search

2. RELATED PREVIOUS WORK

Related work involves two stages: 1) extraction of chemical structure information, and 2) indexing and search chemical molecules. The problems of mining chemical structural information from literature [2] include: 1) chemical structure information can be in text or image formats, and 2) there are many standards. Some partial solutions exist, especially for commercial applications [2].

2.1 Entity Extraction

Labelling sequences is a task of assigning labels to sequences of observations, e.g. labelling *Part of Speech* (POS) tags and entity extraction. Labelling POS tags represents a sentence with a full tree structure and labels each term with a POS tag, while shallow parsers [22] are used to extract entities. Methods used for labelling sequences are different from those that are used for traditional classification, which only considers independent samples. Hidden Markov Models (*HMM*) [8] are one of the common methods used to label or segment sequences. HMM has strong independence assumptions and suffers the label-bias problem [12]. Another category of entity extraction methods is based on *Maximum Entropy* (ME) [5], which introduces an exponential probabilistic model based on binary features extracted from sequences and estimate parameters using maximum likelihood. *MEMM* [14] combines the ideas of *HMM* and *ME*, but also suffers the label-bias problem. Different from those directed graphic models of HMM and MEMM, CRF [12] uses the undirected graphic model, which can avoid the label-bias problem. It follows the maximum entropy principle [3] as ME and MEMM, using exponential probabilistic models and relaxing the independence assumption to involve multiple interaction and long-range dependencies. For labelling sequences, models based on linear-chain CRF have been applied to many applications, such as named-entity recognition [15], detecting biological entities, like protein [21] or gene names [17], etc. However, chemical formula tagging is different from them in the tokenizing process and the feature set used due to different domain knowledge.

Entity extraction can also be viewed as a classification problem where approaches such as SVM [10, 4] are applicable, since information about dependence in the text context of terms can be represented as overlapping features between adjacent terms. However, usually entity extraction is an asymmetric binary classification problem on imbalanced data, where there are many more *false* samples than *true* samples, but *precision* and *recall* of the *true* class is more important than the overall accuracy. In this case, the decision boundary may be dominated by false samples. Several methods such as cost-sensitive classification and decision-threshold tuning are studied for imbalanced data [23]. We have observed that CRFs suffer from this problem too, since in previous work based on CRF [17, 22], usually *recall* is lower than *precision*. To the best of our knowledge, no methods to tune the trade-off between them exist.

2.2 Graph Similarity Search

In Chemoinformatics and the field of graph databases, to search for a chemical molecule, the most common and simple method is the *substructure search* [25], which retrieves all molecules with the query substructure(s). However, sufficient knowledge to select substructures to characterize the desired molecules is required, so the *similarity search* is desired to bypass the substructure selection. Generally, a chemical similarity search is to search molecules with similar structures as the query molecule. Previous methods fall into two major categories based on different criteria to measure similarity. The first one is feature-based approaches using substructure fragments [25, 27], or paths [24]. The major challenge for this category is how to select a set of features, like substructure fragments or paths, to find a trade-off between efficiency and effectiveness and improve both of them. Previous work [26] focused on selection of good features for indexing. They selected *frequent* and *discriminative* substructure fragments sequentially. The basic idea is that the next feature selected from the candidate feature set should not appear only in a very small portion of data, and should not be redundant with respect to the current selected features. Thus, if a feature is too frequent in the data set, i.e., with a low entropy value, it is not a good candidate. However, in practice, no features are *too* frequent. After substructure analysis, feature extraction, and mapping to vector space, distance or kernel functions can be applied to measure the similarity. The second category of approaches uses the concept of Maximum Common Subgraph (MCS) [19] to measure the similarity by finding the size of the MCS of two graphs. The feature-based approaches are more efficient than the second one, since finding MCS is expensive and needs to be computed between the query graph and every graph in the collection. Thus, feature-based approaches based on substructure fragments are used to screen candidates before finding MCS [27]. Hierarchical screening filters are introduced in [19], where at each level, the screening method is more expensive but more accurate.

3. FORMULA EXTRACTION

A chemical formula like *CH4* is a kind of sequential string representation of a chemical molecule. Different from person names, locations, or biological entities, the number of chemical formulae is huge, but the string pattern is defined by some particular rules. A chemical formula can have only partial information of the corresponding molecule structure, and a molecule may have different formula representations.

Usually, a rule-based string pattern match approach can identify most of the chemical formulae, but two types of ambiguity exist. The first case is that even though the string pattern matches the formula pattern, and each letter seems like a chemical element, this chemical molecule may not exist at all. The matched term may be just an abbreviation, e.g. *NIH*. There are too many implicit relations which are infeasible for the rule-based approach. The second case is that even though the matched term is a chemical formula string, in fact it is an English word, or a person name, or an abbreviation in the semantic context. For example, ambiguity exists for *I*(Iodine) and *I*, *He*(Helium) and *He*, *In*(Indium) and *In*. Several text fragments are selected to show the two types of ambiguities.

Non-formula
*"... This work was funded under **NIH** grants ..."*
*"... YSI 5301, Yellow Springs, **OH**, USA ..."*
*"... action and disease. **He** has published over ..."*
Formula
*"... such as hydroxyl radical **OH**, superoxide O2- ..."*
*"... and the other **He** emissions scarcely changed ..."*

Thus, machine learning methods based on SVM and CRF are proposed for chemical formulae extraction. Advanced features from the rule-based string pattern match approach are utilized to improve the overall performance.

3.1 Support Vector Machines

SVM [4] is a binary classification method which finds an optimal separating hyperplane $\{x : w \cdot x + b = 0\}$ to maximize the margin between two classes of training samples, which is the distance between the plus-plane $\{x : w \cdot x + b = 1\}$ and the minus-plane $\{x : w \cdot x + b = -1\}$. Thus, for separable noiseless data, maximizing the margin equals minimizing the objective function $||w||^2$ subject to $\forall i, w \cdot y_i(x_i + b) \geq 1$. In the noiseless case, only the so-called support vectors, vectors closest to the optimal separating hyperplane, are useful to determine the optimal separating hyperplane. Unlike classification methods where minimizing loss functions on wrongly classified samples are affected seriously by imbalanced data, the decision hyperplane in SVM is not affected much. However, for inseparable noisy data, SVM minimizes the objective function: $||w||^2 + C\sum_{i=1}^{n} \varepsilon_i$ subject to $\forall i, w \cdot y_i(x_i + b) \geq 1 - \varepsilon_i$, and $\varepsilon_i \geq 0$, where ε_i is the slack variable, which measures the degree of misclassification of a sample x_i. This noisy objective function has included a loss function that is affected by imbalanced data.

3.2 Conditional Random Fields

Suppose we have a training set S of labelled sequences, where each sequence is an independently and identically distributed sample, but each sample has an internal dependent structure. For example, in a document, adjacent terms have strong dependence. Could we find a method for each sample to represent the conditional probability $p(\mathbf{y}|\mathbf{x}, \lambda)$, where $\mathbf{x}$ is the sequence of observations, $\mathbf{y}$ is the sequence of labels, and λ is the parameter vector of the model, and consider the dependent structure in each sample? Maximum Likelihood can be used to learn the parameter λ and find the best $\mathbf{y}$.

A CRF model can be viewed as an undirected graphical model $G = (V, E)$. The conditional probability of the random vector $\mathbf{Y}$ given observation sequences $\mathbf{X}$ is estimated from the data set [12]. Each random variable Y_v in the random vector $\mathbf{Y}$ is represented by a node $v \in V$ in G. Each $e \in E$ represents the mutual dependence of a pair of labels

$Y_v, Y_{v'}$. A pair of vertices in G are conditionally independent given all other random variables. Even though the structure of G may be arbitrary, for sequential data, usually the simplest and most common structure is a first-ordered chain, where only neighbors in a sequence for $\mathbf{Y}$ are dependent.

To model the conditional probability $p(\mathbf{y}|\mathbf{x}, \lambda)$, we need to find a probability model that can consider not only the probability of each vertex, but also the joint probability of each pair of vertices. Then if the labels or observations of a pair of vertices have some changes, the probability of this model changes too. A probability model for each sequence based on feature functions is applied in CRF,

$$p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} exp(\sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x})), \quad (1)$$

where $F_j(\mathbf{y}, \mathbf{x})$ is a feature function which extracts a real-valued feature from the label sequence $\mathbf{y}$ and the observation sequence $\mathbf{x}$, and $Z(\mathbf{x})$ is a normalization factor for each different observation sequence $\mathbf{x}$. For chain-structured CRF models of sequential inputs, usually only binary functions with values of $\{0, 1\}$ are considered, and two types of feature are used, state feature $F_j(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{|\mathbf{y}|} s_j(y_i, \mathbf{x}, i)$ to model the probability of a vertex in G and transition feature $F_j(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{|\mathbf{y}|} t_j(y_{i-1}, y_i, \mathbf{x}, i)$ to consider mutual dependence of the vertex labels for each edge e in G, and each function has a weight λ_j. The feature weight λ_j specifies whether the corresponding feature is favored or not. The weight λ_j for the feature j should be highly positive if this feature tends to be on for the training data, and highly negative if it tends to be off.

Once we have $p(\mathbf{y}|\mathbf{x}, \lambda)$, the log-likelihood for the whole train set S is given by

$$L(\lambda) = \sum_{k=1}^{|S|} log(p(\mathbf{y}^{(k)}|\mathbf{x}^{(k)}, \lambda)) \quad (2)$$

The goal is to maximize this log-likelihood, which has been proved to be a smooth and concave function, and to estimate parameters λ. To avoid the over-fitting problem, *regularization* may be used; that is, a penalty $(-\sum_{j=1} \frac{\lambda_j^2}{2\sigma^2})$ is added to the log-likelihood function (3), where σ is a parameter which determines how much to penalize λ. Differentiating the log-likelihood function (3) with respect to λ_j, setting the derivative to zero and solving for λ does not have a closed form solution. Numerical optimization algorithms can be applied to solve this problem [12, 18, 22].

3.3 Trade-off Tuning for Imbalanced Data

As mentioned before, from the results of previous research, we see that usually for imbalanced data, *recall* is lower than *precision* for the true class. However, usually extraction of true samples is more important than the overall accuracy, and sometimes recall is more important than precision, especially in information retrieval (IR).

Usually some parameter tuning can improve recall with some loss of precision. Those classification approaches mainly fall into two categories, a) tuning in the training process and b) tuning in the testing process. Cross validation is used to estimate the best parameter [23]. The former approach oversamples the minority class, or undersamples the majority class, or gives different weights for two classes, or gives different penalties (costs of risk) to wrong classifications, every time during training. For example, if $Cost(predicted = true|real = false) < Cost(predicted = false|real = true)$ for each sample, recall is more important than precision. These asymmetric cost values affect the loss function, and finally change the decision boundary of the classes. The latter approach adjusts and finds the best cut-off classification threshold t instead of the symmetric value for the output, which actually only translates the decision boundary [23], but is more efficient because only one training process is required. For example, to increase the importance of recall in SVM, a cut-off classification threshold value $t < 0$ should be selected. In methods with outputs of class probability $[0, 1]$, then a threshold value $t < 0.5$ should be chosen. As noted before, for noiseless data, SVM is stable, but for noisy data, SVM is affected much by imbalanced support vectors. In our work, the latter approach is applied for SVM, i.e., when $t < 0$, *recall* is to be improved but *precision* decreases. When $t > 0$, a reverse change is expected.

In CRF we use a weight parameter θ to boost features corresponding to the *true* class during the testing process. Similar to the classification threshold t in SVM, θ can tune the trade-off between *recall* and *precision*, and may be able to improve the overall performance, since the probability of the *true* class increases. During the testing process, the sequence of labels $\mathbf{y}$ is determined by maximizing the probability model $p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} exp(\sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x}, \theta_\mathbf{y}))$, where $F_j(\mathbf{y}, \mathbf{x}, \theta_\mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \theta_{y_i} s_j(y_i, \mathbf{x}, i)$ or $\sum_{i=1}^{|\mathbf{x}|} \theta_{y_i} t_j(y_{i-1}, y_i, \mathbf{x}, i)$, $\theta_\mathbf{y}$ is a vector with $\theta_{y_i} = \theta$ when $y_i = true$, or $\theta_{y_i} = 1$ when $y_i = false$, and λ_j are parameters learned while training.

3.4 Feature Set and Induction

Generally, two categories of state features are extracted from sequences of terms: single-term features from a single term, and overlapping features from adjacent terms. There are two types of features: surficial features and advanced features. Surficial features are those that can be observed directly from the term, such as word or word prefix and suffix features, orthographic features, or lists of specific terms. Advanced features are those generated by complex domain knowledge or other data mining approaches in advance.

Usually for data mining, advanced features are more powerful than surficial features, but more expensive and sometimes infeasible. If an advanced feature has a very high correlation with the true class label, then a high accuracy is expected. Advanced features can be inferred using rule-based approaches or machine-learning approaches.

In our work, a rule-based approach using string pattern matching is applied to generate a set of features. Since we do not have a dictionary of all chemical molecules, and the formula of a molecule may have different string representations, we consider features of co-occurrence of two chemical elements in a formula to measure whether a matched string is a formula. For example, C and O co-occur frequently, but an element of the noble gases e.g. *He* and a metal element e.g. *Cu* are impossible to appear together in a formula.

As mentioned before, we need to distinguish formula terms from English words or personal names. Linguistic features like POS tags are used based on natural language processing (NLP), such as noun or proper noun. Those features are useful especially when combined with overlapping features in the context of a token. All the features that are used by our algorithms are summarized here. Note that all the features based on observations are combined with the state labels of tokens to construct transition features.

Summary of features
Surficial features: InitialCapital, AllCapitals, OneCapital, HasDigit, HasDash, HasPunctuation, HasDot, HasBrackets, HasSuperscripts, IsChemicalElementName, IsAmbiguousEnglishWord, IsAmbiguousPersonalName, IsAbbreviation, character-n-gram features. For features like IsChemicalElementName and IsAbbreviation, we have lists of names of chemical elements and common abbreviations, e.g. *NIH*.
Advance features: IsFormulaPattern, IsFormulaPatternWithCooccurrence, IsLongFormulaPattern, IsFormulaPatternWithSuperscript, IsFormulaPatternWithLowerCase, IsPOSTagNN, etc. String pattern matching and domain knowledge are used for features of formula pattern.
Overlapping features: Overlapping features of adjacent terms are extracted. We used -1, 0, 1 as the window of features, so that for each token, all Overlapping features about the last token and the next token are included in the feature set. For instance, for *He* in "*... . He is ...*", feature($term_{n-1}$ = "." $\wedge$ $term_n$ = $initialCapital$)=*true*, and feature($term_n$ = $initialCapital \wedge term_{n+1} = isPOSTagVBZ$)=*true*. This "*He*" is likely to be an English word instead of *Helium*.

Finally all features combine with labels. However, there are too many features and most occur rarely. We apply an approach to feature induction for CRF proposed in [13] to score candidate features using their log-likelihood gain: $\Delta L_G(f) = L(S)_{F\cup\{f\}} - L(S)_F$, where F is the current feature set, $L(S)_F$ is the log-likelihood of the training set using F, and $L(S)_{F\cup\{f\}}$ is the log-likelihood of the training set adding feature f. Thus, more useful features are selected.

4. FORMULA INDEXING AND SEARCH

We define chemical formulae formally as follows:

Definition 1. **Formula and Partial Formula:** Given a vocabulary of chemical elements, E, a chemical *formula* f is a sequence of pairs of a *partial formula* and the corresponding frequency $< s_i, freq_{s_i} >$, where each s_i is a chemical element $e \in E$ or another chemical formula f'. A *partial formula* is viewed as a *substructure* of f, denoted as $s \preceq f$, is a subsequence of f, or a subsequence of a partial formula s_i in $< s_i, freq_{s_i} >$ of f, or a subsequence of a partial formula s' of f, so that if $s' \preceq f \wedge s \preceq s'$, then $s \preceq f$. The length of a formula $L(f)$ or a partial formula $L(s)$ is defined as the number of pairs in the sequence.

A partial formula is also a formula by definition, but may not be a meaningful formula. For example, *CH3(CH2)2OH* is a chemical formula, and *C*, *CH2*, *(CH2)2*, and *CH3(CH2)2OH* all are partial formulae.

We discuss three issues in this section. First, we discuss how to analyze the structure of a chemical formula and select features for indexing, which is important for substructure search and similarity search. Since the full graphic structure information of a molecule is unavailable, we use partial formulae as substructures for indexing and search. The same chemical molecule may have different formula strings mentioned in text, e.g. acetic acid can be *CH3COOH* or *C2H4O2*. The same formula can represent different molecules, e.g. *C2H4O2* can be acetic acid (*CH3COOH*) or methyl formate (*CH3OCHO*). Second, different from keywords search of documents in IR, to search chemical formulae, we should consider structures of formulae instead of only frequencies of chemical elements, because in traditional IR, there are enough terms to distinguish documents, while in Chemoinformatics, using chemical elements and their frequencies is not enough to distinguish chemical formulae. Third, to score the relevance of the search results to the query formula, each feature is assigned a weight based on its length, frequency in formulae, and distribution among formulae.

Input: Candidate Feature Set C with frequency $Freq_s$ and support D_s for each substructure s, minimal threshold value of frequency $Freq_{min}$, minimal discriminative score α_{min}.
Output: Selected Feature Set F.
1. **Initialization**: $F = \{\emptyset\}$, $D_\emptyset = D$, length $l = 0$.
2. **while** C is not empty, do
3. $l = l + 1$;
4. **for each** $s \in C$
5. **if** $Freq_s > Freq_{min}$
6. **if** $L_s = l$
7. compute α_s using Eq (3) ($\alpha_s^{(0)} = \frac{|D|}{|D_s|}$, since no s satisfies $s' \preceq s \wedge s' \in F$)
8. **if** $\alpha_s > \alpha_{min}$
9. move s from C to F;
10. **else** remove s from C;
11. **else** remove s from C;
12. **return** F;

Figure 2: Algorithm: Sequential Feature Selection

4.1 Feature Selection for Index Construction

To support similarity search, partial formulae of each formula are useful as possible substructures for indexing. However, since partial formulae of a partial formula $s \preceq f$ with $L(s) > 1$ are also partial formulae of the formula f, the number of all partial formulae of the formula set is quite large. For instance, the candidate features of *CH3OH* are *C*, *H3*, *O*, *H*, *CH3*, *H3O*, *OH*, *CH3O*, *H3OH*, and *CH3OH*. We do not need to index every one due to redundant information. For example, two similar partial formulae may appear in the same set of formulae (e.g. *CH3CH2COO* and *CH3CH2CO*), because they generate from the same super sequence. In this case, it is enough to index only one of them. Moreover, it is not important to index infrequent fragments. For example, a complex partial formula appearing only once in the formula set is not necessary for indexing, if its selected fragments are enough to distinguish the formula having it from others. E.g. when querying formulae having partial formulae of *CH3*,*CH2*,*CO*,*OH*, and *COOH*, if only *CH3CH2COOH* is returned, then it is not necessary to index *CH3CH2COO*. Using a similar idea and notations about feature selection in [26], given a whole data set D, D_s is the *support* of substructure s, the set of all formulae containing s, and $|D_s|$ is the number of items in D_s. All substructures of a frequent substructure are frequent too. Based on these observations, two criteria may be used to sequentially select features of substructures into the set of selected features F. The feature selected should be 1) frequent, and, 2) its support should not overlap too much with the intersection of supports of its selected substructures in F.

For Criterion 1, mining frequent substructures is required in advance. After the algorithm extracts all chemical formulae from documents, it generates the set of all partial formulae and records their frequencies. Then, for Criterion 2, we define a discriminative score for each feature candidate with respect to F. Similar to the definitions in [26], a substructure s is *redundant* with respect to the selected feature set F, if $|D_s| \approx |\cap_{s'\in F \wedge s' \preceq s} D_{s'}|$. A substructure s is *discriminative* with respect to F, if $|D_s| << |\cap_{s'\in F \wedge s' \preceq s} D_f|$.

Thus, the discriminative score for each candidate s with respect to F is defined as:

$$\alpha_s = |\cap_{s' \in F \wedge s' \preceq s} D_{s'}|/|D_s|. \quad (3)$$

The sequential feature selection algorithm is described in Figure 2. The algorithm starts with an empty set F of selected features, scanning each substructure from the length $l = 1$ to $l = L(s)_{max}$. At each length of substructure, all frequent candidates with discriminative scores larger than the threshold are selected. This scanning sequence ensures that at each length of substructure, no scanned substructure is a substructure of another scanned one. Thus, only selected substructures at previous steps are considered to compute the discriminative scores. All substructures s with $L(s) > l$ but $Freq(s) <= Freq_{min}$ are removed directly from the candidate set C, because even when $L(s) = l$ after several scanning cycles to longer substructures, $Freq_s$ still has the same value, and α_s will decrease or remain the same. Consequently, the feature is not selected.

4.2 Query Models

We propose four types of queries for chemical formula search: *exact search*, *frequency search*, *substructure search*, and *similarity search*. Usually only frequency formula search is supported by current chemistry information systems. As mentioned before, substructure search and similarity search are common and important for structure search, but not for formula search, because formulae do not contain enough tructural information. Motivated by this, we propose heuristics for fuzzy formula search based on partial formulae.

Definition 2. **Formula Query and Frequency Range:** A *formula query* q is a sequence of pairs of a partial formula and the corresponding *frequency range* $< s_i, range_{s_i} >$, where token s_i is a chemical element $e \in E$ or another chemical formula f', and $range_{s_i} = \cup_k [low_k, upper_k], upper_k \geq low_k \geq 0$.

Exact search

The answer to an *exact search* query is formulae having the same sequence of partial formulae within the frequency ranges specified in the query. Exact search usually is used to search exact representation of a chemical molecule. Different formula representations for the same molecule cannot be retrieved. For instance, the query *C1-2H4-6* matches *CH4* and *C2H6*, but not *H4C* or *H6C2*.

Frequency searches

We say that a user runs a *frequency search*, when he specifies the elements and their frequencies. All documents with chemical formulae that have the specified elements within the specified frequency ranges are returned. As indicated above, most current chemistry databases support frequency searches as the only query models for formula search. There are two types of frequency searches: *full frequency searches* and *partial frequency search*. When a user specifies the query *C2H4-6*, the system returns documents with the chemical formulae with two *C* and four to six *H*, and no other atoms for *full frequency search*, e.g. *C2H4*, and returns formulae with two *C*, four to six *H* and any numbers of other atoms for *partial frequency search*, e.g. *C2H4* and *C2H4O*.

Substructure search

Substructure searches find formulae that may have a query substructure defined by users. In substructure searches, the query q has only one partial formula s_1 with $range_{s_1} = [1, 1]$, and retrieved formulae f have $freq_{s_1} \geq 1$. However, since the same substructure may have different appearances in formulae, three types of matches are considered with different ranking scores (Section 4.3). E.g. for the query *COOH*, *COOH* gets an exact match (high score), *HOOC* reverse match (medium score), and *CHO2* parsed match (low score).

Similarity search

Similarity searches return documents with chemical formulae with similar structures as the query formula, i.e., a sequence of partial formulae s_i with a specific $range_{s_i}$, e.g. *CH3COOH*. However, there are two reasons that traditional fuzzy search based on edit distance is not used for formula similarity search: 1) Formulae with more similar structures or substructures may have larger edit distance. E.g. H2CO3 can also be mentioned as HC(O)OOH, but the edit distance of them is larger than that of H2CO3 and HNO3 (6¿2). Using the partial formula based similarity search of H2CO3, HC(O)OOH has a higher ranking score than HNO3 based on Equation (6). 2) Compute edit distances of the query formula and all the formulae in the data set is computational expensive, so a method based on indexed features of partial formulae is much faster and feasible in practice. Our approach is feature-based similarity search, since full structure information is unavailable in formulae. The algorithm uses selected partial formulae as features. We design and present a scoring function in Section 4.3 based on all selected partial formulae that are selected and indexed in advance, so that the query processing and the ranking score computation is efficient. Formulae with top ranking scores are retrieved.

Conjunctive search

Conjunctive search of the four basic formula searches is supported for filtering search results, so that users can define various constraints to search desired formulae. For example, a user can search formulae that have two to four *C*, four to ten *H*, and may have a substructure of *CH2*, using a conjunctive search of a full frequency search *C2-4H4-10* and a substructure search of *CH2*.

Query rewriting

Since the ultimate goal of users is to search relevant documents, the users can search using formulae as well as other keywords. The search is performed in two stages. First, a query string is analyzed. All the embedded formula searches are taken out, and all possible desired formulae are retrieved. Then, after relevant formulae are returned, the original query string is rewritten by embedding those formulae into the corresponding positions of the original query string as subgroups with OR operators. To involve the relevance score for each retrieved formula, boosting factors with the values of relevance scores are added to each retrieved formula with a goal to rank corresponding documents. Second, the rewritten query is used to search relevant documents. For example, if a user searches documents with the term *oxygen* and the formula *CH4*, the formula search *CH4* is processed first and matches *CH4* and *H4C* with corresponding scores of 1 and 0.5. Then the query is written as "*oxygen* (*CH4*ˆ1 OR *H4C*ˆ0.5)", where 1 and 0.5 are the corresponding boosting factors. Then documents with *CH4* get higher scores.

4.3 Relevance Scoring

A scoring scheme based on the Vector Space Model in IR and features of partial formulae is used to rank retrieved formulae. We adapt the concepts of the term frequency tf and the inverse document frequency *idf* to formula search.

Definition 3. **SF.IFF and Atom Frequency:** Given the collection of formulae C, a query q and a formula $f \in C$, $SF(s, f)$ is the *substructure frequency* for each substructure $s \preceq f$, which is the total number of occurrences of s in f, $IFF(s)$ is the *inverse formula frequency* of s in C, and defined as

$$SF(s, f) = \frac{freq(s, f)}{|f|}, IFF(s) = log\frac{|C|}{|\{f|s \preceq f|\}|},$$

where $freq(s, f)$ is frequency of s in f, $|f| = \sum_k freq(s_k, f)$ is the total frequency of all selected substructures in f, $|C|$ refers to the total number of formulae in C, and $|\{f|s \preceq f|\}$ is the number of formulae that have substructure s. Since a chemical atom e is also a substructure of a formula f or a partial formula s, *atom frequency* refers to the substructure frequency of e in f or s.

Frequency searches

For a query formula q and a formula $f \in C$, the scoring function of frequency searches is given as

$$score(q, f) = \frac{\sum_{e \preceq q} W(e)SF(e, f)IFF(e)^2}{\sqrt{|f|} \times \sqrt{\sum_{e \preceq q}(W(e)IFF(e))^2}}, \quad (4)$$

where $|f| = \sum_k freq(e_k, f)$ is the total atom frequency of chemical elements in f, $1/\sqrt{|f|}$ is a normalizing factor to give a higher score to formulae with fewer atoms, $1/\sqrt{\sum_{e \preceq q}(W(e)IFF(e))^2}$ is a factor that makes scores comparable between different queries. It does not affect the rank of retrieved formulae for a specific formula query, but affects the rank of retrieved documents, if there are more than two formula searches embedded in the document search. Without this factor, documents containing the longer query formula get higher scores. Equation (4) considers f as a bag of atoms, where $e \preceq f$ is a chemical element. $W(e)$ is the weight of e that represents how much it contributes to the score. It can adjust the weight of each e together with $IFF(e)$. Without domain knowledge $W(e) = 1$.

Substructure search

The scoring function of substructure search is given as

$$score(q, f) = W_{mat(q,f)}SF(q, f)IFF(q)/\sqrt{|f|}, \quad (5)$$

where $W_{mat(q,f)}$ is the weight for different matching types, exact match (high weight, e.g. 1), reverse match (medium weight, e.g. 0.8), and parsed match (low weight, e.g. 0.25), which are defined by experiences.

Similarity search

A scoring function like a sequence kernel [9] is designed to measure similarity between formulae for similarity search. It maps a query formula implicitly into a vector space where each dimension is a selected partial formula using the sequential feature selection algorithm. For instance, the query *CH3OH* is mapped into dimensions of *C*, *H3*, *O*, *H*, *CH3*, and *OH*, if only these six partial formulae are selected. Then formulae with those substructures (including reverse or parsed matched substructures) are retrieved, and scores are computed cumulatively. Larger substructures are given more weight for scoring, and scores of long formulae are normalized by their total frequency of substructures. The scoring function of similarity search is given as

$$score(q, f) = \frac{\sum_{s \preceq q} W_{mat(s,f)}W(s)SF(s, q)SF(s, f)IFF(s)}{\sqrt{|f|}}, \quad (6)$$

where $W(s)$ is the weight of the substructure s, which is defined as the total atom frequency of s.

Table 1: Average accuracy of formula extraction

Method	Recall	Precision	F-measure
String Pattern Match	98.38%	41.70%	58.57%
CRF,$\theta = 1.0$	86.05%	96.02%	90.76%
CRF,$\theta = 1.5$	**90.92%**	**93.79%**	**92.33%**
SVM linear,$t = 0.0$	86.95%	95.59%	91.06%
SVM linear,$t = -.2$	**88.25%**	**94.23%**	**91.14%**
SVM poly,$t = 0.0$	87.69%	96.32%	91.80%
SVM poly,$t = -.4$	**90.36%**	**94.64%**	**92.45%**
LASVM linear,$t = 0.0$	83.94%	90.65%	87.17%
LASVM linear,$t = -.2$	**85.42%**	**89.55%**	**87.44%**
LASVM poly,$t = 0.0$	75.87%	93.08%	83.60%
LASVM poly,$t = -.4$	**83.86%**	**88.51%**	**86.12%**

Table 2: P-values of 1-sided T-test on F-measure

Pairs of methods	F-measure
CRF,$\theta = 1.0$;CRF,$\theta = 1.5$	0.130
CRF,$\theta = 1.5$;SVM,linear,$t = -.2$	0.156
CRF,$\theta = 1.5$;SVM,poly,$t = -.4$	0.396
CRF,$\theta = 1.5$;LASVM,linear,$t = -.2$	0.002
CRF,$\theta = 1.5$;LASVM,poly,$t = -.4$	0.000
SVM,linear,$t = 0.0$;SVM,linear,$t = -.2$	0.472
SVM,poly,$t = 0.0$;SVM,poly,$t = -.4$	0.231
SVM,linear,$t = -.2$;SVM,poly,$t = -.4$	0.072
SVM,linear,$t = -.2$;LASVM,linear,$t = -.2$	0.009
SVM,poly,$t = -.4$;LASVM,poly,$t = -.4$	0.000

5. EXPERIMENTS

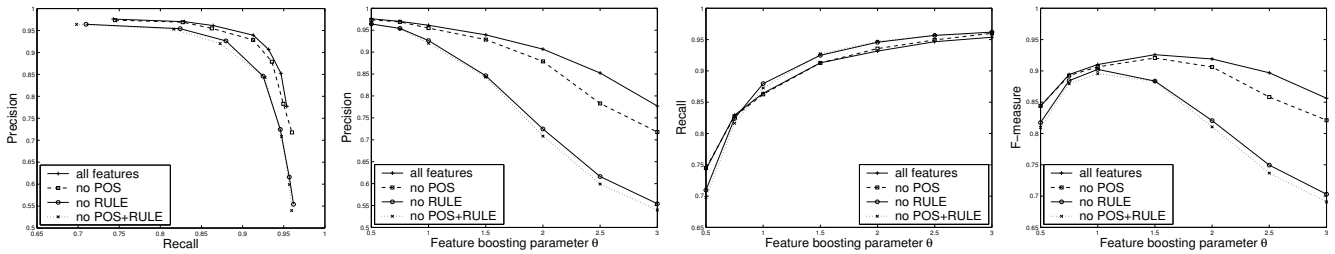
In this section, our proposed methods of formula extraction, feature selection, and formula search are tested.

5.1 Formula Extraction

The first data set we used for testing is randomly selected from chemistry publications crawled from the web-site of the Royal Society of Chemistry[1]. First, 200 documents are selected randomly from the publication set, and a random part of each document is chosen. Each token is labelled manually with a tag of formula or non-formula after tokenizing. This data set is very imbalanced, with only 1.59% true samples (5203 formulae vs. 321514 non-formula tokens). We use 10-fold cross-validation to evaluate the results of formula extraction. Thus, each time, we used a training set of samples obtained from 180 files and a testing set of samples obtained from the other 20 files. Several methods are evaluated for formula extraction, including String Pattern Match, SVM with the linear (SVM linear) and polynomial kernel (SVM poly), SVM active learning with the linear (LASVM linear) and polynomial kernel (LASVM poly), and CRF. *SVM light* [10] for batch learning and *LASVM* [4] for active learning are used. *MALLET* [16] is used for CRF. The same feature set is utilized for all the machine learning methods, and different feature subsets are tested for the CRF to evaluate the contribution of the subsets. We tested complex kernels of RBF and Gaussian, which are not shown here due to the worse performances and more expensive computational costs than the linear and polynomial kernel. For CRF, to avoid the overfitting problem, *regularization* is used, with $\sigma^2 = 5.0$.

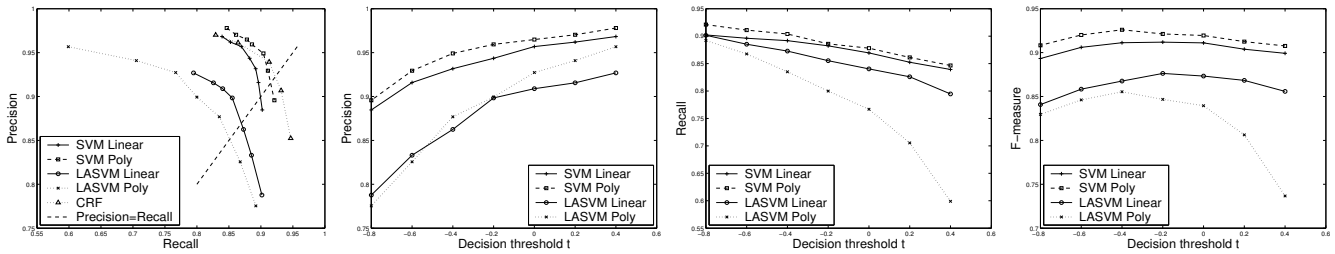
To measure the overall performance, we use *F-measure* [17], $F = 2PR/(P+R)$, where P is precision and R is recall, instead of using error rate, since it is always very small for imbalanced data. Results of average *recall*, *precision*, and

[1]http://www.rsc.org/



(a) (b) (c) (d)

Figure 3: CRF with different values of feature boosting parameter θ



(a) (b) (c) (d)

Figure 4: SVM and LASVM with different values of threshold t

F-measure are presented in Table 1, and Figures 3 and 4. P-values of T-test of significance are shown in Table 2. Note precision-recall curves here are different from the normal shape of precision-recall curves in IR. The shape in Figures 3(a) and 4(a) can generate a F-measure curve with a peak, so that we can optimize it by parameter tuning. Moreover, if a precision-recall curve is situated towards the upper-right corner, then a better F-measure is expected.

For CRF, we test different feature sets. Features are categorized into three sets: surficial features, advanced features using rule-based string pattern match (*RULE*), and part-of-speech tags based on natural language processing (*POS*). Four combinations are tested: (1) all features, (2) no POS, (3) no RULE, and (4) no POS or RULE. We also test different values {0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0} for the feature-boosting parameter θ for the formula class. Note that when $\theta = 1.0$, it is the normal CRF, while when $\theta < 1.0$, the non-formula class gets more preference.

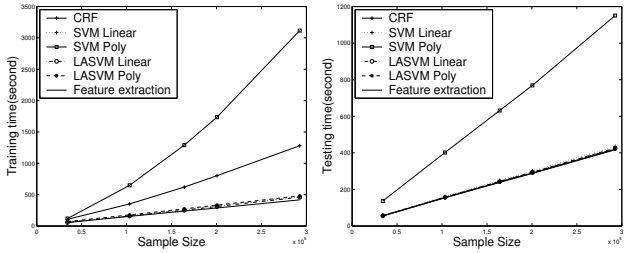
From Figure 3, we can see that the contribution of RULE features is much higher than that of POS features, since the difference between curves with or without POS features is quite smaller than that between curves with or without RULE features. Usually, the performance with more features is better than that with fewer features. We can observe that F-measure curves with fewer features are more peaky and sensitive to θ, because both recall and precision change faster. From the experiment results, we can see that when $\theta = 1.5$ using all features, we have the best overall performance based on *F-measure*, and for this case, recall and precision are much more balanced.

Based on experimental experiences of SVM, $C = 1/\delta^2$, where $\delta = 1/n \sum_{i=1}^{n} \sqrt{ker(\mathbf{x_i}, \mathbf{x_i}) - 2 \cdot ker(\mathbf{x_i}, \mathbf{0}) + ker(\mathbf{0}, \mathbf{0})}$ for SVM light, $C = 100$ for LASVM, and polynomial kernel $(\mathbf{x} \cdot \mathbf{x}' + 1)^3$ are applied in experiments. We test different decision threshold values {-0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8}. From Figure 4(a), we can see that CRF and SVM poly both have a better performance curve than does SVM linear, but the difference is not statistically significant at the level of 0.05 (Table 2). All of them are much better than LASVM, which is statistically significant. Moreover, we can see that CRF gives recall more preference instead of precision than does SVM poly. When $recall \geq precision$, CRF can reach a better F-measure. This is important for imbalanced data.

We show the results for all approaches using all features in Table 1 and compare them with the String Pattern Match approach, which has very high recall but quite low precision. Its error of recall is caused mainly by wrong characters recognized from image PDF files using optical character recognition. The only previous work we can find is the GATE Chemistry Tagger [1]. Since it cannot handle superscripts and can recognize names of chemical elements, e.g. oxygen, the GATE Chemistry Tagger is not fully comparable with our approach. Without counting these two cases, its *recall* is around 63.4%, *precision* 45.2%, and *F-measure* 52.8%.

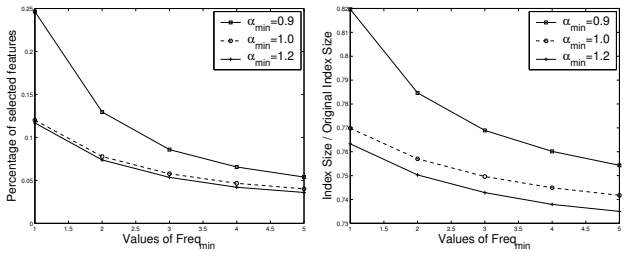
We also evaluate the time taken by these methods to run both for the training and testing process. Note that feature extraction and CRF are implemented in Java, while SVM and LASVM in C. Running time includes time of feature extraction and training (or testing) time, since in practice feature extraction must be counted. In Figure 5(a), we can see that CRF has a computational cost between SVM poly and other methods. We also observe that LASVM is much faster than SVM, especially for complex kernels.

Based on these observations from our experiment results, we can conclude that the boosting parameter for CRF and the threshold value for SVM can tune the relation of precision and recall to find a desired trade-off and are able to improve the overall F-measure, especially when recall is much lower than precision for imbalanced data. CRF is more desired than SVM for our work, since it not only has a high overall F-measure, but also a more balanced performance between recall and precision. Moreover, CRF has a reason-



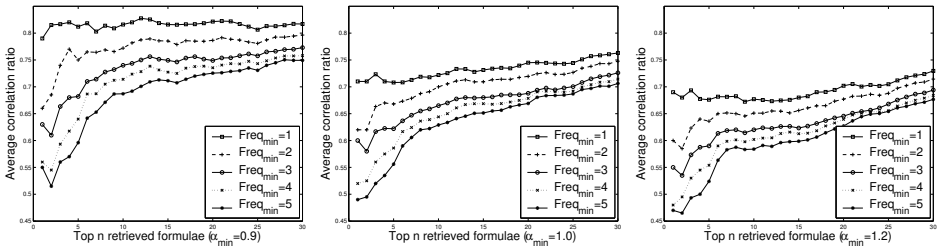
(a) Training time (b) Testing time

Figure 5: Running time of formula extraction including feature extraction



(a) Ratio of selected features (b) Ratio of index size

Figure 6: Features and index size ratio after feature selection



(a) $\alpha = 0.9$ (b) $\alpha = 1.0$ (c) $\alpha = 1.2$

Figure 7: Correlation of similarity search results after feature selection

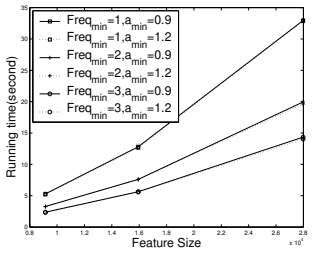


Figure 8: Running time of feature selection

able running time, lower than that of SVM with complex kernels. In addition, during the testing process, the testing cost of CRF is trivial compared with the cost of feature extraction.

5.2 Formula Indexing and Search

For formula indexing and search, we test the sequential feature selection algorithm for index construction and evaluate retrieved results. We select a set of 5036 documents and extract 15853 formulae with a total of 27978 partial formulae before feature selection. Different values for the frequency threshold $Freq_{min} = \{1, 2, 3, 4, 5\}$ and the discrimination threshold $\alpha_{min} = \{0.9, 1.0, 1.2\}$ are tested, and results are shown in Figures 6-9.

Note that when $\alpha_{min} = 0.9$, all frequent partial formulae are selected without considering the discriminative score α. When $\alpha_{min} = 1.0$, each partial formula whose support can be represented by the intersection of its selected substructures' supports is removed. We do not lose information in this case because all the information of a removed frequent structure has been represented by its selected partial formulae. When $\alpha_{min} > 1.0$, feature selection is lossy since some information is lost. After feature selection and index construction, we generate a list of 100 query formulae that are selected randomly from the set of extracted formulae and from a chemistry textbook and web pages. These formulae are used to perform similarity searches.

The experiment results (Figure 6) show that depending on different threshold values, most of the features are removed after feature selection, so that the index size decreases correspondingly. Even for the case of $Freq_{min} = 1$ and $\alpha_{min} = 0.9$, 75% of the features are removed, since they appear only once. We can also observe that from $\alpha_{min} = 0.9$ to $\alpha_{min} = 1.0$, many features are also removed, since those features have selected partial formulae with the same support D. When $\alpha_{min} \geq 1.0$, the selective ratio change a little. We also evaluated the runtime of the feature selection algorithm, illustrated in Figure 8. We can see that a larger $Freq_{min}$ can filter infrequent features directly without computing discriminative scores, which speeds up the algorithm, while the value of α_{min} affect the runtime little.

The most important result from our experiments is that for the same similarity search query, the search results with feature selection are similar to those without feature selection when the threshold values are reasonable. To compare the correlation between them, we use the average of the percentage of overlapping results for the top $n \in [1, 30]$ retrieved formulae, which is defined as $Corr_n = |R_n \cap R'_n|/n, n = 1, 2, 3, ...$, where R_n and R'_n are the search results of applying feature selection or not, correspondingly. Results are presented in Figure 7. As expected, when the threshold values of $Freq_{min}$ and α_{min} increases, the correlation curves decrease. In addition, the correlation ratio increases for more retrieved results (n increases). From the retrieved results, we also find that if there is an extract matched formula, usually it is returned as the first result. This is why the correlation ratio of the top retrieved formula is not much lower than that of the top two retrieved formulae. We also can see from those curves that a low threshold value of $Freq_{min}$ can keep the curve flat and have a high correlation for smaller n, while a low threshold value of α_{min} can improve the correlation for the whole curve. For the case of $Freq_{min} = 1$ and $\alpha_{min} = 0.9$, more than 80% of the retrieved results are the same for all cases, and 75% of the features are removed, which is both efficient and effective enough.

For exact search and frequency search, the quality of retrieved results depends on formula extraction. For similarity search and substructure search, to evaluate the search results ranked by the scoring function, enough domain knowledge is required. Thus, we only show an example with top

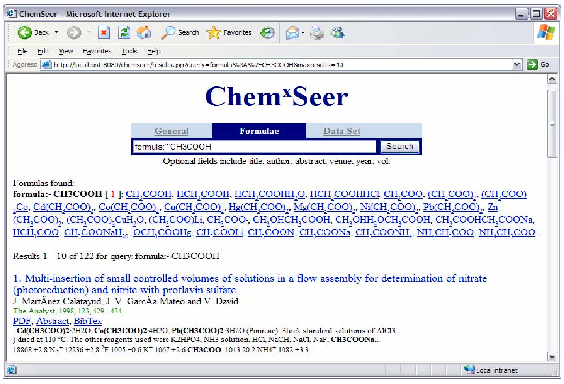


Figure 9: Similarity search results of $Chem^X Seer$

retrieved results for the feature selection case of $Freq_{min} = 1$ and $\alpha_{min} = 0.9$ in Figure 9, a snapshot of $Chem^X Seer$.

6. CONCLUSIONS AND FUTURE WORK

We evaluated several methods for chemical formula extraction based on SVM and CRF. We also proposed different methods for them to tune the trade-off between recall and precision for imbalanced data. Experiments illustrated that CRF is the best regarding effectiveness and efficiency, and SVM linear also has a good performance. Our trade-off tuning methods can improve the overall F-measure scores and increase recall of formulae as expected. Studying different subsets of features shows that combining prior knowledge as advanced features is important to improve the accuracy. A sequential feature selection algorithm is designed to select frequent and discriminative substructures. Results show that it can reduce the feature set and the index size tremendously. Retrieved results of similarity search with and without feature selection are highly correlated. We also introduced several query models for chemical formula search, which are different from keywords searches in IR. Corresponding scoring schemes are designed, which extended the idea of *tf-idf*, and considered sequential information in formulae. Experiment results show that the new scoring schemes work well.

Several future directions are discussed here. First, user study is required to evaluate precision and recall of additional search results. Second, the scoring schemes can be optimized by global knowledge learned from users' clickthrough information. We can expect non-formula strings among extracted formulae can get a lower rank and can be identified easily. Third, when we analyze the structures of chemical formulae, which occurrence of a string pattern is a meaningful substructure? This is a challenging issue since formulae may not have enough information and how to define a substructure is a problem. Furthermore, which substructures are useful to measure the similarity? Finally, the identity ambiguity exists for formulae, since even though we have a unique *id* for each object in a database, *id* matching of each appearance is a challenging issue.

7. ACKNOWLEDGMENTS

Seyda Ertekin is acknowledged for the LASVM code.

8. REFERENCES

[1] Gate. http://gate.ac.uk/.
[2] D. L. Banville. Mining chemical structural information from the drug literature. *Drug Discovery Today*, 11(1-2):35–42, 2006.
[3] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.
[4] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005.
[5] A. Borthwick. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University, 1999.
[6] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *Proceedings of SIGKDD*, 1998.
[7] S. J. Edgar, J. D. Holliday, and P. Willet. Effectiveness of retrieval in similarity searches of chemical databases: a review of performance measures. *Journal of Molecular Graphics and Modelling*, 18(4-5):343–357, 2000.
[8] D. Freitag and A. McCallum. Information extraction using hmms and shrinkage. In *AAAI Workshop on Machine Learning for Information Extraction*, 1999.
[9] D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, 1999.
[10] T. Joachims. Svm light. http://svmlight.joachims.org/.
[11] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of ICDM*, 2001.
[12] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, 2001.
[13] A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of Conference on UAI*, 2003.
[14] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML*, 2000.
[15] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL*, 2003.
[16] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.
[17] R. McDonald and F. Pereira. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, 6(Suppl 1):S6, 2005.
[18] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.
[19] J. W. Raymond, E. J. Gardiner, and P. Willet. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.
[20] S. S. Sahoo, C. Thomas, A. Sheth, W. S. York, and S. Tartir. Knowledge modeling and its application in life sciences: A tale of two ontologies. In *Proceedings of WWW*, 2006.
[21] B. Settles. Abner: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.
[22] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, 2003.
[23] J. G. Shanahan and N. Roma. Boosting support vector machines for text classification through parameter-free threshold relaxation. In *Proceedings of CIKM*, 2003.
[24] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of PODS*, 2002.
[25] P. Willet, J. M. Barnard, and G. M. Downs. Chemical similarity searching. *J. Chem. Inf. Comput. Sci.*, 38(6):983–996, 1998.
[26] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proceedings of SIGMOD*, 2004.
[27] X. Yan, F. Zhu, P. S. Yu, and J. Han. Feature-based substructure similarity search. *ACM Transactions on Database Systems*, 2006.
[28] W. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of WWW*, 2006.
[29] J. Zhao, C. Goble, and R. Stevens. Semantic web applications to e-science in silico experiments. In *Proceedings of WWW*, 2004.