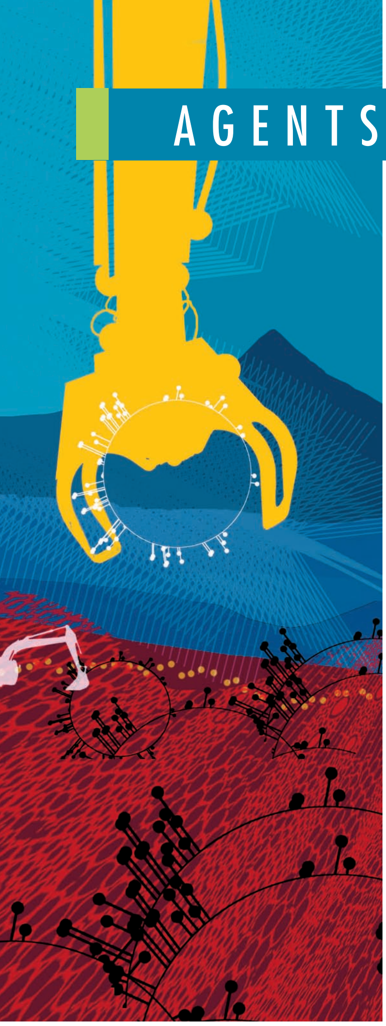


A Collaborative Multiagent System for Mining Transcriptional Regulatory Elements

Yun Xiong, Guangyong Zheng, Qing Yang, and Yangyong Zhu, *Fudan University*

A multiagent-based system mines transcriptional regulatory elements using novel algorithms combined with biological domain knowledge.

Biological sequences are one of the most important forms of biological data. There are two main types, comprising DNA and protein sequences. In bioinformatics, DNA sequences are seen as linear strings composed of nucleotides: A (adenine), G (guanine), C (cytosine), and T (thymine). Protein sequences are regarded as the linear order of amino acids. For example, the string "ACCTGTA" denotes a DNA sequence and the string "DLMDEFFIFHHI" denotes a protein sequence. One of the most important challenges molecular biologists face is deciphering the regulation mechanisms of gene expression on the transcription level. Identifying transcriptional regulatory elements offers a key means of insight into these mechanisms. Transcription factors (TFs) and cis-regulatory elements (that is, transcription factor binding sites; TFBSs) are primary transcriptional regulatory elements. TFs are special DNA-binding protein sequences. Under certain physiological conditions, TFs effectively regulate expression levels of downstream genes by binding to specific DNA fragments in the promoter regions. The specific DNA fragments are TFBSs. Such a process is closely related to important biological processes such as activating the cell cycle, regulating differentiation, maintaining immunologic tolerance, and so on. Investigation of TFs and TFBSs could provide new insight into such vital biological processes. However, existing tools suffer from low accuracy, and most of them are mutually interdependent. Moreover, the number of known elements is relatively scarce, and not all of them are publicly available.

Biologists need a system to help identify whether a new protein/DNA sequence is a TF/TFBS, find TF and TFBS relationships efficiently, and acquire the related information about the transcription-level sequences rather than having to select interdependent tools themselves. Therefore, biologists need a system that supports tool integration, exhibits its superior accuracy, automatically provides the appropriate service, and distributes the computation burden. Agents can contribute to these improvements. The adoption of multiagent systems constitutes an emerging area in bioinformatics,[1] but little consideration has

1541-1672/09/$25.00 © 2009 IEEE
Published by the IEEE Computer Society

Related Work in Transcriptional Regulatory Element Data Mining

As far as we know, transcription factors (TFs) and transcription factor binding sites (TFBSs) are primary transcriptional regulatory elements. With respect to TF prediction, some researchers previously adopted an unsupervised learning method that used sequence alignment to detect similarity among sequences and discover similar proteins with known TFs. Another approach is a supervised learning method, such as the methods based on the Support Vector Machine,[1] Hidden Markov Model, Nearest Neighbor Algorithm,[2] and so on. But these existing methods consider only sequence composition, while ignoring extra information about the TFs, such as protein domains.[3]

Prediction Methods

For predicting TFBSs, there are mainly two methods. One is motif discovery using the conservation feature. Numerous tools have become available, such as AlignACE and ANNSpec. The other is motif search—that is, to scan putative binding sites of a known TF. Several tools like Match[4] were developed to search for potential TFBSs by using position weight matrices. A recent review of TFBS prediction methods indicated that the average sensitivity from these methods is only about 5 percent at the nucleotide level and 10 percent at the site level.[5] These methods also suffer from a large number of false positives when applied on a genomic scale. Recently, researchers have adopted the classification method. However, it is rather problematic to determine if a negative set for only small sets of binding sites is experimentally verified.[6] Accordingly, we introduce novel mining algorithms to acquire superior accuracy. Our approach requires agent technology to improve the overall performance, because these algorithms demand flexible integration as well as relevant databases.

Agents in Bioinformatics

It's a new and important demand to ask a computer system to compose and carry out entire workflows across multiple available bioinformatics resources.[7] This means the software must perform ever higher cognitive processes, which in turn requires additional reasoning capabilities based on extensive domain-specific knowledge.[7] Moreover, most biological databases and computational resources are distributed. Accordingly, the requirement for intelligent and distributed software suggests a multiagent system approach as particularly suitable.[7,8]

In recent years, agent technology has found many applications in bioinformatics. Emanuela Merelli and her colleagues reviewed issues concerning applications of agent technology to genomics.[7] BioMAS is an early multiagent system applied to genomic annotation. The GeneWeaver is another work system that uses agents for genome analysis to automate bioinformatics processes and resource integration. BioAgent applies agents to integrate data and tools for polygenic disease analysis. With the increasing need to combine access to databases with computational steps, Giuliano Armano and his colleagues proposed a multiagent architecture that they applied to the protein secondary structure prediction.[9] Tzong-Yi Lee and his colleagues developed an agent-based system called AgentMultiProIdent that integrated protein identification tools to discover protein-protein interactions.[10]

Though agents have been widely applied and contribute to the improvement in performance of existing bioinformatics systems, researchers have paid little attention to regulatory mechanism research. Compared with the existing transcriptional regulatory element analysis tools without agent technology, TREMAgent is a more flexible system. It exhibits reactive, proactive, autonomous, and interactive features and provides more comprehensive regulatory element information, facilitates usage styles that satisfy various demands, and maintains continuous updates.

References

1. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.
2. Z. Qian, Y.D. Cai, and Y.X. Li, "Automatic Transcription Factor Classifier Based on Functional Domain Composition," *Biochemical and Biophysical Research Comm.*, vol. 347, no. 1, 2006, pp. 141–144.
3. G. Zheng et al., "The Combination Approach of SVM and ECOC for Powerful Identification and Classification of Transcription Factor," *BMC Bioinformatics*, vol. 282, no. 9, 2008, pp. 1–8.
4. A.E. Kel et al., "Match: A Tool for Searching Transcription Factor Binding Sites in DNA Sequences," *Nucleic Acids Research*, vol. 31, no. 13, 2003, pp. 3576–3579.
5. M. Tompa et al., "Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites," *Nature Biotechnology*, vol. 23, no. 1, 2005, pp. 137–144.
6. B. Jang, M.Q. Zhang, and X.G. Zhang, "Oscar: One-Class SVM for Accurate Recognition of Cis-Elements," *Bioinformatics*, vol. 23, no. 21, 2007, pp. 2823–2828.
7. E. Merelli et al., "Agents in Bioinformatics, Computational and Systems Biology," *Bioinformatics*, vol. 1, no. 8, 2006, pp. 45–59.
8. L.B. Cao, C. Luo, and C.Q. Zhang, "Agent-Mining Interaction: An Emerging Area," *Autonomous Intelligent Systems: Agents and Data Mining*, LNAI 4476, Springer, 2007, pp. 60–73.
9. G. Armano et al., "PACMAS: A Personalized, Adaptive, and Cooperative Multi-Agent System Architecture," *Proc. 6th Workshop from Objects to Agents* (WOA 05), 2005, pp. 54–60.
10. T.Y. Lee et al., "An Agent-Based System to Discover Protein-Protein Interactions, Identify Protein Complexes and Proteins with Multiple Peptide Mass Fingerprints," *J. Computational Chemistry*, vol. 27, no. 9, 2006, pp. 1020–1032.

been given to issues of mining regulatory elements (see the sidebar "Related Work in Transcriptional Regulatory Element Data Mining"). A system based on agent technology can integrate various tools for element prediction and make them behave collaboratively. Utilizing agents' autonomous problem-solving capability, the system can provide the appropriate workflow and flexible usage. In addition, agents can detect and manage the global distribution of continually updated data for dynamic mining.

In this article, we propose the Transcriptional Regulatory Element Mining Agent (TREMAgent), a novel multiagent system that provides TF/TFBS prediction functions,

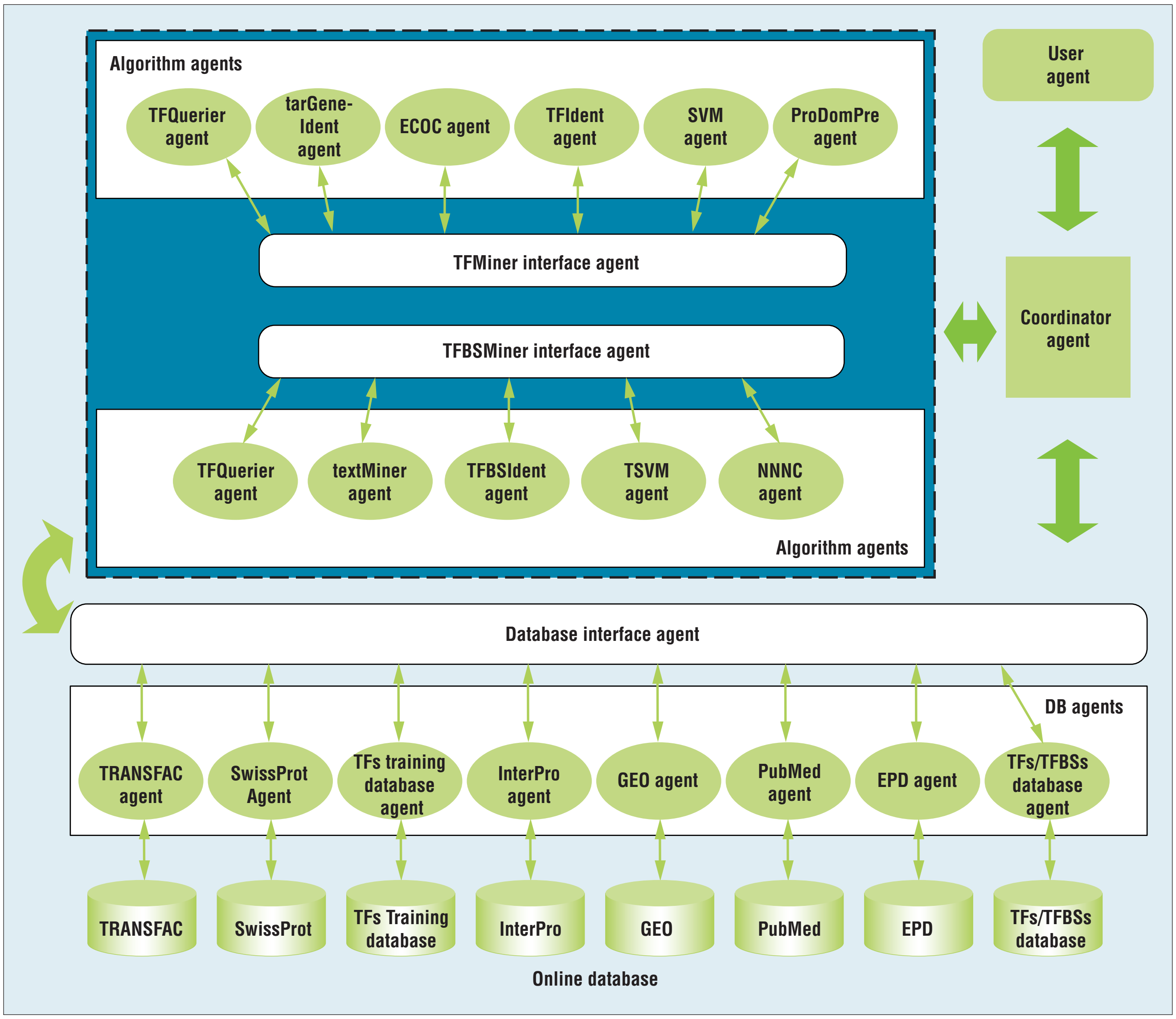


Figure 1. Overview of TREMAgent System. The TREMAgent system consists of four types of agents (algorithm agents, database agents, interface agents, and coordinator agent), each associated with a specific role.

supports a variety of retrieval and identification services, and allows the timely update of results. Experimental results show that TREMAgent provides superior accuracy and flexible service.

The TREMAgent System

TREMAgent aims at collecting TF/TFBS information containing both experimentally verified data and putative data through data mining, database construction, and identification and retrieval services for biologists. Accordingly, we developed a text-mining algorithm to extract verified data from the literature and two data mining algorithms to combine biological domain knowledge for TF and TFBS prediction. One of the data mining algorithms adopts the support vector machine (SVM) method. We call it PDI-SVM (protein domains incorporated SVM) because it integrates protein domains as feature vectors and then combines an error-correcting output coding (ECOC) algorithm to categorize TFs into four subclasses: basic TFs, zinc TFs, helix TFs, and beta TFs. The other data mining algorithm combines a nearest-neighbor algorithm (NNA) without negative samples and a transductive SVM (TSVM) method for limited known data. We employ agent technology to integrate these algorithms and databases and so to construct a multiagent regulatory elements mining system.

Architecture and System Flow

The TREMAgent system consists of four types of agents, each associated with a specific role. Figure 1 shows the overall perspective of the architecture.

- *Algorithm agents* interact with other agents to handle various tasks (querying, identifying, and mining). For example, Figure 1 shows how the ECOC agent is used to categorize TFs into subclasses. Moreover, most agents must perform collaboratively (see the "Collaborations and Relationships among Agents" subsection).
- *Database agents* interact with external data sources. They also manage local copies of databases and present them in a format that allows other agents access. Meanwhile, the database agents keep the data up-to-date for dynamic mining.
- *Interface agents* provide an indirect communication service between the coordinator agent, algorithm agents, and database agents.
- A *coordinator agent* establishes communications among the interface agents and algorithm agents, assigns work to different agents, and coordinates processes and conflicts. In addition, it stores each agent's information in its agent repository to maintain the control of the whole system.

The multiagent system works in two cases.

First, it deals with user requests. For example, a biologist gets a protein sequence and wants to know whether it is a TF. The biologist can input the sequence into the multiagent system, which sends the request to the user agent. The system can also accept a query keyword as input for retrieval or a DNA sequence for TFBS identification. The user agent then passes the request to the coordinator agent. Subsequently, the coordinator agent decides what processing flow to adopt and transfers the input to the algorithm agents. The TFMiner/TFBSMiner interface agent passes the result from the algorithm agents back to the coordinator agent, which forwards the result to the user agent. In this case, the coordinator agent transfers the protein sequence to a TFIdent (TF identification) agent. The TFIdent agent decides whether the sequence is a TF and outputs the result to the coordinator agent, which forwards the result to the user agent. Finally, the biologist receives the result.

The second TREMAgent case constructs the classifier and predicts TF/TFBS sequences automatically. The database agent sends the training set to the corresponding algorithm agent to build a classifier. For example, for TFs, the SVM agent uses the SVM algorithm to build the SVM classifier and forwards the classifier to the TFIdent agent (for more detail, see the "Collaborations and Relationships among Agents" subsection); for TFBSs, the corresponding algorithm agent means the nearest-neighbor classification algorithm without negative samples (NNNC) agent. After these steps, the TFIdent/TFBSIdent agent uses the classifier interacting with the database interface agent to identify TFs/TFBSs for the sequences from the Swiss-Prot/EPD (Eukaryotic Promoter Database). Finally, the results are sent and saved in the TF/TFBS database. For experimentally verified TFBSs, the PubMed agent hands the literature to the TFBSMiner interface agent, and then invokes a textMiner agent. After that, it sends the results and saves them in the TFBS database.

This is the first system based on multiagent technology that supports transcriptional regulatory element mining and sustains personalized requirements.

In the bioinformatics and agent areas, to the best of our knowledge, this is the first system based on multiagent technology that supports transcriptional regulatory element mining, sustains personalized requirements, and offers flexible usage by making most of the analysis automatic.

Algorithm Agent Roles

The algorithm agents included in the TREMAgent system consist of two types of agents (TF mining algorithm agents and TFBS mining algorithm agents).

TF Mining Algorithm Agents. A TF regulates expression levels of downstream genes by binding to a TFBS through the TF functional domain. We can regard the functional domain as a symbol to decide whether a protein sequence is a TF. Therefore, we choose functional domains as a features vector of the sequences and adopt the SVM method to construct a classifier to distinguish TFs from non-TFs (for more detail, see the "TFBS Mining Algorithm" subsection). Hence, we design an SVM agent to generate the classifier and a TFIdent agent to identify TFs based on the classifier. Subsequently, we devise an ECOC agent to categorize TFs into subtypes. We use the Protein Domain Preparation (ProDomPre) agent to obtain the protein domain information.

TFBS Mining Algorithm Agents. Some classification approaches have been applied to TFBS prediction, but they require the generation of an artificial negative class. Constructing such instances can be problematic. Furthermore, if the instances aren't properly handled, they can dramatically affect the classifier's performance. Unfortunately, the limited number of known TFBSs makes it difficult to provide negative samples.

We present an NNNC for TFBS discovery. It utilizes only features of known TFBSs as positive instances to build a classifier. Hence, we construct an NNNC agent. We also adopt a TSVM method to prune the results, because it offers strong generalization ability on particular test sets with limited training samples. We use a TSVM agent to construct a classifier. Subsequently, we develop a TFBSIdent agent to identify TFBSs based on the classifier. Moreover, to obtain the experimentally verified TFBSs, we develop a text mining algorithm. Naturally, we also build a textMiner agent.

Transcriptional Regulatory Elements Mining Algorithms

The TREMAgent collects TF/TFBS information containing both experimentally verified data and putative data through data mining, including a text-mining algorithm to extract verified data from the literature and two data mining algorithms for TF and TFBS prediction.

We developed an algorithm for extracting transcription factor binding sites on the basis of a question-answering system.

TF Mining Algorithm. Given l sequences $s_1, s_2, ..., s_l$, we can develop an algorithm to predict whether these sequences are TFs and, if so, to identify which subclass they belong to. We use the PDI-SVM algorithm to construct a classifier for TF recognition, employing protein domains as feature vectors. An ECOC algorithm combines with SVM to categorize TFs into subclasses.

We base the SVM algorithm on the concept of the maximal margin hyperplane, which depicts the decision boundary of different categories. An SVM classifier finds a decision hyperplane that classifies training samples. The margin between this hyperplane and the nearest sample should be as large as possible. In the training phase, the SVM resolves the following optimization problem to find the decision hyperplane,

$$\min \frac{1}{2}\omega^T \omega + C\sum_i \xi_i$$

such that

$$y_i\left(\omega^T x_i + b\right) \geq 1 - \xi_i, \xi_i \geq 0,$$

where b is the bias, C is an influencing parameter for tradeoff, and ξ is a slack variable.

In the decision phase, the SVM labels the test samples according to the side of the hyperplane that they lie on. The decision formula is given by

$$f\left(x\right) = \omega^T \cdot \mathbf{X} + b.$$

If an unknown sample is represented in a feature vector of protein domains, we can predict category Y of the sample using

$$Y = \begin{cases} \textit{positive} \text{ if } f\left(x\right) \geq \delta, \\ \textit{negative} \text{ else.} \end{cases}$$

where ξ is the threshold of decision value for SVM and $f(x)$ is the decision function.

We obtained the basic SVM package from the SVMlight Web site (http://svmlight.joachims.org). The SVMlight package is free for academic research.

In the algorithm, we choose protein domains as feature vectors. Thus, a protein's features are represented by n dimension vectors. We obtain protein domains through the InterPro database, which contains 14,764 entries. To ensure vector independency, we remove the overlap according to InterPro database documents. Only 4,758 out of 14,764 entries are chosen as feature vectors. Thus, protein features are denoted with 4,758 dimension vectors. For example, if a protein contains the fifth and eighth elements in the feature list, then the fifth and eighth values are set to 1 and the rest are set to 0.

Text-Mining Algorithm for Extracting Experimentally Verified TFBSs. Traditional text-mining methods in bioinformatics compare text with a predefined rule base to find gene/protein sequences. However, the limited known TFBSs make it difficult to acquire a rule base. On the other hand, these traditional methods only consider the word frequencies rather than the context influences. We developed an algorithm for extracting TFBSs on the basis of a question-answering system. The algorithm transfers the sentence into a directed graph according to not only the weight of words but also their order. It's a two-phase algorithm, which first constructs a question net (Qnet) model using the training set, and then queries the test literature to find the sentences (answers) that are similar to the Qnet (questions), according to a specified similarity function—that is, the answers are the sentences describing TFBSs. Finally, we can extract the matched TFBS sequences from the sentences describing TFBSs by using the regular expressions because the TFBS is composed of the nucleotides A, G, C, T.

A stop word (SW) is a word deemed irrelevant for indicating the sentence feature even if it occurs frequently, such as "the." A tag word (TW) is formulated as the composition of a word and its part-of-speech (POS) tagging. Considering the semantic and grammar information, we construct the ordered pair of two TWs composing a tag word pair (TWP) because the order between the words needs to be considered. For example, let's consider "transcription factor" and "binding site." If they form a TWP, they are regarded as an ordered word; otherwise, they are two different words. A tag sentence (TS) is a directed diagraph in which the node is a TW and the edge is a TWP, denoted as (V, E). where V is the set of the nodes and E is the set of the edges. We can calculate the weight of a TS with the formula

$$TS.weight = \sum_{i}^{m} TW_i.weight + \sum_{j}^{n} TWP_j.weight$$

where *TW.weight* is the weight of a TW, and *TWP.weight* is the weight of a TWP. We can calculate the weight using term frequency/inverse document frequency (TF/IDF), which represents the importance of the words or sentences in the literature. We calculate TF/IDF with the formula

$$tf_i(d) * \log(N/n_i),$$

where $tf_i(d)$ is the frequency of the word TW_i in the document d, N is the number of texts, and n_i is the number of texts containing the word TW_i.

Accordingly, we regard the similarity between the sentences as the similarity between the tag sentences.

Given two tag words TW_a and TW_b, we calculate their similarity (*TWSim*) as

$$TwSim(TW_a, TW_b) = \begin{cases} \min(TW_a.weight, TW_b.weight) & \text{if} \begin{pmatrix} TW_a.norm = TW_b.norm \wedge \\ TW_a.pos = TW_b.pos \end{pmatrix}, \\ 0 & \text{else.} \end{cases}$$

Likewise, the similarity between two tag sentences TS_a and TS_b is defined as

$$TsSim(TS_a, TS_b) = \sum_{\substack{TW_a \in TS_a \\ TW_b \in TS_b}} TwSim(TW_a, TW_b) + \sum_{\substack{TWP_i \in TS_a \\ TWP_j \in TS_b \\ TWP_i = TWP_j}} TWP_i.weight.$$

The choice of a distance metric becomes crucial in determining the outcome of nearest-neighbor classification.

We define one set of the tag sentences describing TFBSs in the training set as a site sentence (SS) set, while we regard the other set of sentences describing protein/genes except TFBSs as a nonsite sentence (NSS) set. Correspondingly, a Qnet is a trained TS adopting our construction algorithm on the training set. Generally speaking, the contexts of the sentences describing the TFBSs are similar, but they are dissimilar for those not describing TFBSs. Therefore, we can base the construction of Qnet on the following hypothesis: The sentences describing the TFBSs are similar in the training set, while the ones not describing TFBSs are dissimilar (see the text mining algorithm in Figure 2).

TFBS Mining Algorithm. The NNNC algorithm is a modification of the nearest-neighbor classification method, which learns from positive examples only. At first, the algorithm operates by storing all the training samples as its model. Then it calculates the distance between each subsequence in a test sequence and the training samples according to a specific distance metric. The subsequence belongs to the target class (that is, it's putative TFBSs) when the distance satisfies a certain threshold—that is, the putative TFBSs are the nearest neighbors to the training samples.

The choice of a distance metric becomes crucial in determining the outcome of nearest-neighbor classification. For the sequences, we use the edit distance based on the nucleotide substitution replacement model to calculate the distance between the training example and the subsequence in a test sequence. Correspondingly, we use the Basic Local Alignment Search Tool (Blast) to find regions of local similarity between sequences and calculate the statistical significance of matches.

To further prune the prediction results, we choose TSVM because the number of positive training samples of known TFBSs is also relatively small. A TSVM is an SVM combined with a transductive learning procedure. It utilizes the information carried by the unlabeled samples and acquires better classification performance. It can label test samples with information carried by either the labeled or unlabeled samples in the training phase.

Let the given data set consist of n labeled samples x_i, $1 \le i \le n$, and m unlabeled samples x_i, $n + 1 \le i \le n + m$, with the labels denoted as y_i. According to the principle of transductive learning, the optimization problem of the TSVM is given by

Algorithm 1: buildQnet. // to construct the Qnet
Input: the site sentence set *SS*, the nonsite sentence set *NSS*
Output: the question net after trained Qnet

```
1)  for each $TS_i \in SS$ do
2)    $TS_i$ = normalizeTW($TS_i$)
3)    $TS_i$ = excludeSW($TS_i$)
4)    for each tag word $TW_k \in TS_i$, do
5)          if $TW_k$ in Qnet.V,
6)          $TW_k.weight$+=1
7)          else insert $TW_k$ into Qnet.V
8)  for each $TS_j \in NSS$ do
9)    $TS_j$ = normalizeTW($TS_j$)
// to normalize the weight of the tag words
10)   $TS_j$ = excludeSW($TS_j$)
// to delete the stop words
11)   for each tag word $TW_k \in TS_j$ do
12)     if $TW_k$ in Qnet.V
13)       $TW_k.weight$ -= 1
14)   Normalize the weight of $TW$ in Qnet.V
15)   for each $TS_i \in SS$ do
16)     for each tag word pair $TWP_j \in TS_i$ do
17)       if $TWP_j$ in Qnet.E
18)         $TWP_j.weight$ +=1
19)       else insert $TWP_j$ into Qnet.E
20)  Normalize the weight of $TWP$ in Qnet.E
```

Algorithm 2: mineAnswer. // to mine the sentences (answers)
for the Qnet which are similar to Qnet (questions)
that is, the sentences describing TFBSs
Input: the set of tag sentences *D* and the Qnet
Output: the sentences that describe TFBSs

```
1)  for each $TS_i$ in $D$ do
2)    if $TS_i$ contents the strings of gene sequences
3)      $TS_i$ = normalizeTW($TS_i$)
4)      $TS_i$ = excludeSW($TS_i$)
5)    $S_{i,qnet}$ = TsSimilarity ($TS_i$, Qnet)
6)    if $S_{i,qnet}$ > $T$ // $T$ is a user specified threshold
7)      insert $TS_i$ and $S_{i,qnet}$ into $S$
8)  sort the $TS_s$ in the $S$ according to their similarity values
9)  return the top $N$ $TSs$ in $S$ as the answer for the Qnet
```

Figure 2. Text mining algorithm. The algorithm aims to extract transcription factor binding sites on the basis of a question-answering system. It's a two-phase algorithm, which first constructs a question net (Qnet) model using the training set, and then queries the test literature to find the sentences (answers) that are similar to the Qnet (questions), according to a specified similarity function.

$$\min \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\xi_i + C^*\sum_{i=n+1}^{n+m}\xi_i^*$$

such that

$$y_i\left(\omega^T x_i + b\right) \ge 1 - \xi_i, 1 \le i \le n;$$

$$y_i\left(\omega^T x_i + b\right) \ge 1 - \xi_i^*, n+1 \le i \le n+m;$$

where ξ and ξ^* are slack variables for the training and test samples respectively; C and C^* are the influencing parameters for the training and test samples.

Using the collaboration of the NNNC and TSVM, we can identify the putative TFBSs for the nucleotide sequences.

Collaborations and Relationships among Agents

In the TREMAgent system, agents aren't working alone and need to collaborate for further analysis. The following cases show the relationships among agents.

Once the coordinator agent detects the input request, it will decide the workflow to make the agents behave collaboratively. For identifying TFs, the SVM agent interacts with the TF training database agent to obtain the training set—that is, the SwissProt agent obtains the negative training set from Swiss-Prot and the TRANSFAC agent acquires the positive training set from TRANSFAC, an online biological database (www.gene-regulation.com/pub/databases.html). Both interact with the TFs' training database agent (Figure 3①i-ix). And then the SVM agent constructs a classifier (see Figure 3①i-ix) using these training data sets. After that, it forwards the classifier to the TFIdent agent (Figure 3①x).

Meanwhile, the Swiss-Prot agent collects the protein sequences from Swiss-prot and sends them to the TFMiner interface agent. Then the TFMiner interface agent invokes the TFIdent agent to identify these protein sequences (see Figure 3②i-iv). The TFIdent agent generates the putative TF and forwards them to the ECOC agent to categorize the subclasses (see Figure 3③i-v). After that, the TFs database agent transfers the results to the TFs database. Figure 3 shows the process.

Services in TREMAgent

TREMAgent can deal with various requests automatically to provide the appropriate service. The requests are divided into multiple types. The first type identifies whether the protein sequences are TFs. The second identifies whether the nucleotide sequences are TFBSs. The third performs queries on the basis of the input keywords (such as gene name). Therefore, the system automatically enables distinct requests rather than forcing biologists to plan a workflow by selecting the tools.

Agent technology supplies the specific workflow accord-

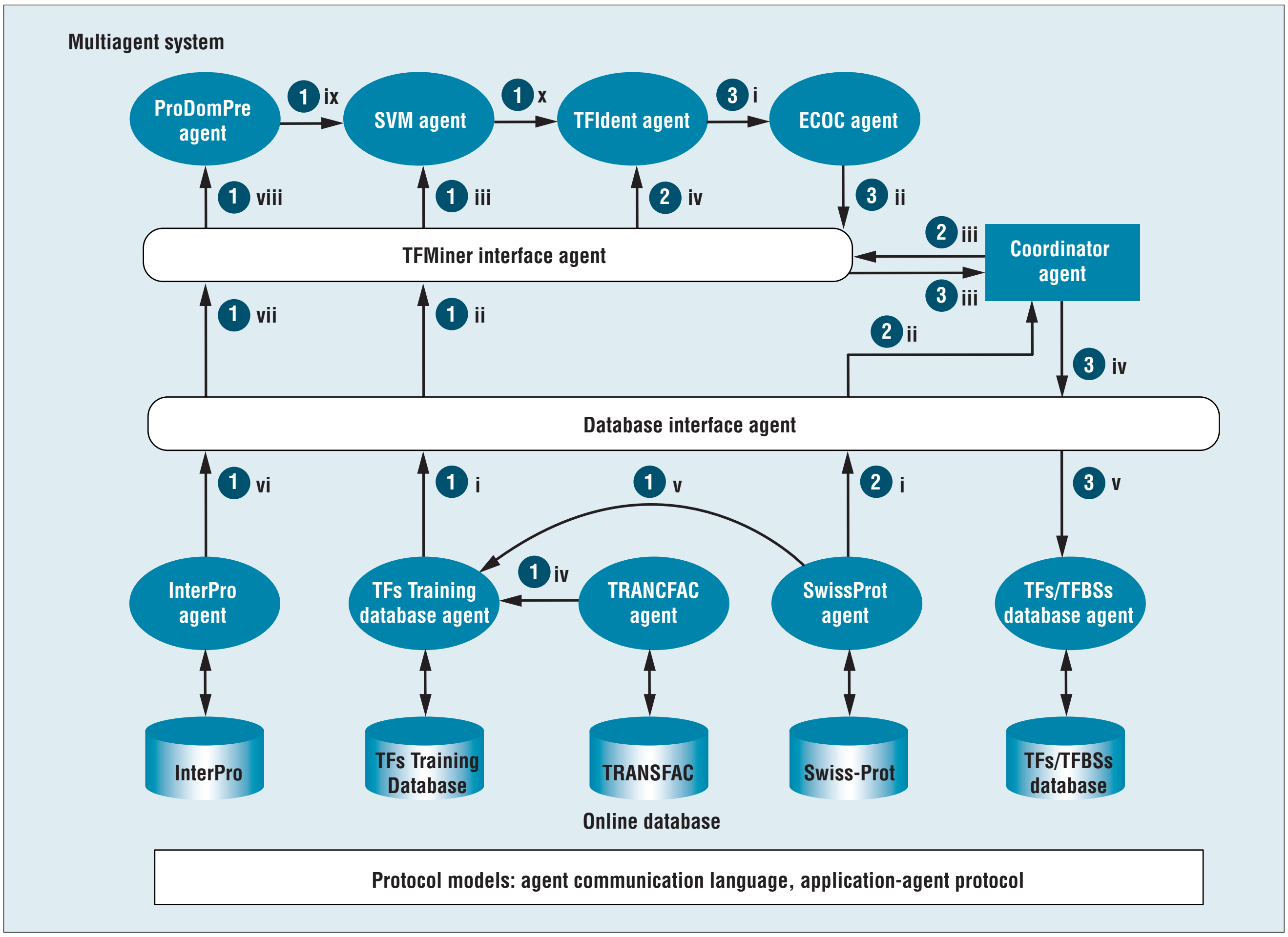


Figure 3. The collaborate process using agents for transcription factor identification. In the TREMAgent system, agents need to collaborate for further analysis. Once the coordinator agent detects the input request, it will decide the workflow to make the agents behave collaboratively.

ing to the user's interest. The user agent detects the requests. If an input is a sequence, it requires an identification process (for example, a user wants to know whether a nucleotide sequence is a TFBS); if an input is a keyword, it requires a query process (for example, a user wants to get the information about a TFBS whose gene name is known).

Once the request information has been gathered, the user agent forwards it to the coordinator agent, which sends it to the corresponding algorithm agent. For the identification task, the TFIdent agent or TFBSIdent agent identifies the putative TFs or TFBSs for the input protein or nucleotide sequences according to the classifier from the algorithm agents. For the query process, a TF/TFBSQuerier agent interacts with the TF/TFBS database agent and returns the information relevant to the query keyword.

Experimental Study

In order to evaluate the performance of our system, we implemented the algorithms in the Java programming language and the system in the Java Agent Development Framework (Jade). We ran all the experiments on a server with a 2-GHz CPU and 6 Gbytes of main memory. We use the real-world data sets for our evaluation purpose. The details are described in the following subsections.

Data Sets for TFs

The data sets for the evaluation of the PDI-SVM algorithm include positive and negative samples. For positive factors, we extracted a primal data set including 6,464 items from the TRANSFAC database. For negative factors, we constructed the primal data set through searching the Swiss-Prot database using unambiguous non-TF terms such as "kinase," "ubiquitin," and so on. We collected a total of 23,057 entries. Subsequently, we refined the two primal data sets. We filterered out proteins that lacked Swiss-Prot accession numbers or annotation by any protein domains, and we eliminated redundancy in data sets against sequence similarity by programming CD-HIT (that is, an ultra fast protein sequence clustering program; http://cd-hit.org)

Table 1. PDI-SVM performance.

Number of positive items	Number of negative items	Sensitivity (%)	Specificity (%)	Total success rate (%)
450	450	88.44	88.00	88.22
450	675	88.67	89.19	89.07
450	900	88.44	90.67	89.93
450	1,125	87.56	90.93	89.97
450	1,350	86.67	91.41	90.22

Table 2. Blast, NNA, PDI-SVM Comparison.

	Blast		NNA		PDI-SVM	
Total items	Correct	Accuracy (%)	Correct	Accuracy (%)	Correct	Accuracy (%)
Positive TFs (450)	322	71.56	367	81.56	388	86.22
Negative TFs (1,727)	1,286	74.46	1,612	93.34	1,564	90.56

Note: "Correct" denotes the number of TFs that are identified correctly.

and Pisces (that is, a database server for producing lists of sequences from the protein data bank; http://dunbrack.fccc.edu/pisces) with a threshold of 25 percent. As a result, the final positive data set contained 450 items, among which 138 items had known class information. The final negative data set contained 1,727 entries.

Data sets for TFBSs

To survey the accuracy of the TFBS prediction algorithm, we selected the benchmark data sets provided by Martin Tompa and his colleagues.[2] They consist of 52 data sets, six from flies, 26 from humans, 12 from mice, and eight from yeast. To test the text mining algorithm, we use the other data set, which contains 1,048 publications from PubMed. There are 1,451 sentences describing TFBSs and 1,913 sentences describing gene/protein sequences, which exclude TFBSs.

Experimental Results

We implemented the algorithms in the Java programming language and the system in the Java Agent Development Framework (JADE). We ran all the experiments on a server with a 2-GHz CPU and 6 Gbytes of main memory.

PDI-SVM Performance for TF Prediction

We tested the PDI-SVM algorithm on the data set mentioned in the "Data Sets for TFs" subsection. We used the leave-one-out-cross-validation test to evaluate the capability of the classifier. We calculated four measures: the true positive (TP), the false positive (FP), the true negative (TN), the false negative (FN). TP and TN were correct predictions for TFs and non-TFs respectively. A false positive occurred when a non-TF was predicted as a TF, and a false negative occurred when a TF was predicted as a non-TF. Finally, the true positive rate (*sensitivity*), true negative rate (*specificity*), and total success rate were calculated by the following formulas: $sensitivity = TP / (TP + FN)$, $specificity = TN / (TN + FP)$, $total\ success\ rate = (TP + TN) / (TP + FP + TN + FN)$. Furthermore, we tested several conditions where positive and negative items were in different proportions (see Table 1).

When the numbers of positive and negative items were the same, the sensitivity reached 88.44 percent, and the specificity achieved 88.00 percent, which meant the classifier had good performance for both TF and non-TF identification. When the negative items increased, the accuracy of the classifier didn't change drastically. Tests with different mixture rates showed that the method was robust.

To further illustrate the performance, we made comparisons with Blast and NNA.[3] Blast identified the query protein as the same category as its best hit when searching similarity in the whole data set excluding the protein itself. We calculated the accuracy for positive and negative sets as follows (see results in Table 2):

$$\begin{cases} Accuracy\ for\ positive\ set = \dfrac{predicted\ number\ of\ positive\ items}{total\ number\ of\ positive\ items}, \\ Accuracy\ for\ negative\ set = \dfrac{predicted\ number\ of\ negative\ items}{total\ number\ of\ negative\ items} \end{cases}$$

As shown in Table 2, for the positive set, accuracy obtained by Blast and NNA was around 72 percent and 82 percent, which was lower than PDI-SVM by about 14 percent and 4 percent. For the negative set, the accuracy of Blast, NNA, and the PDI-SVM method was about 74 percent, 93 percent, and 91 percent. Essentially, Blast and NNA sort an unknown item through attributes of a local item (the nearest neighbor). Hence, classifiers based on these two algorithms are inclined to group an item into a category with a larger size.

Table 3. Performance of text mining for TFBSs.

Data	Positive	TP	Negative	FP	Recall (%)	Precision (%)	Time (s)
01	150	115	141	39	76.67	74.68	100
02	150	116	169	53	77.33	68.64	113
03	150	115	155	48	76.67	70.55	126
04	150	108	165	53	72.00	67.08	132
05	150	123	159	75	82.00	62.12	120
06	150	139	149	29	92.67	82.74	78
07	150	119	149	29	79.33	80.41	100
08	150	119	149	28	79.33	80.95	120
09	150	136	140	21	90.67	86.62	127
10	150	118	155	32	78.67	78.67	103
Average					80.53	74.79	112

In our TF identification scenario, the number of the negative set was much larger than that of the positive set, so items were more likely to be identified as negative by the NNA method. Therefore, the NNA method was slightly more accurate than the PDI-SVM algorithm in accuracy of identifying negative samples. We considered the balance issue of positive and negative items and carried out another comparison between these methods on a data set having an equal number of the two sample types. An integrated survey for both positive and negative samples indicated that PDI-SVM performed better than the Blast and NNA methods in a data set with balanced positive and negative items. Therefore, we believe the performance of PDI-SVM is superior to the Blast method and comparable to the NNA method.

The Performance of TFBSs Prediction Algorithm. We performed tests on the data sets in the "Data Sets for TFBSs" subsection. We applied the leave-one-out cross-validation method to evaluate the performance. The evaluation metrics involve *sensitivity* (*Sn*) and *positively predicted value* (*PPV*) at both the site and the nucleotide level, and *average site performance* (*ASP*) at the site level. They are defined as: $xSn = xTP/(xTP + xFN)$, $xPPV = xTP / (xTP + xFP)$, $sASP = (sSn + sPPV) / 2$, where $x = n/s$ represents the assessment value at the nucleotide/site level.

To illustrate the advantage of collaboration using the agents, we compared the performance of the NNNC algorithm with the combination method of NNNC and TSVM. The *xSn* in the combination method is decreased a bit, but the *xPPV* is increased greatly, and the average site performance *sASP* is especially increased. Therefore, from the whole and average performance aspect, the combination method performs better than the method using only the NNNC algorithm (see Figure 4d).

Furthermore, we compared the accuracy of our method with the weight matrix approach Match (that is, a matrix search tool for TFBS; www.gene-regulation.com/cgi-bin/pub/programs/match/bin/match.cgi?) and the motif discovery method MEME (a Motif-based sequence analysis tool, http://meme.sdsc.edu/meme4_1/intro.html) on four different species (fly, human, mouse, and yeast) and the total data sets. Figure 4a–c shows the result from MEME, Match, and our method respectively. The results in Figure 4e show that Match performs slightly better than MEME. However, the accuracy is lower than our method both at the site and nucleotide level. In total, the average performance by our method is highest.

The Performance of TFBSs' Text-Mining Algorithm. We evaluate the algorithm using the measures $Recall = TP/(TP + FN)$ and $Precision = TP/(TP + FP)$, where *TP* is the number of sentences describing TFBSs, *FP* is the number of sentences not describing TFBSs, and *FN* is the number of sentences describing TFBSs but not predicted. We adopted the *k*-fold cross-validation to assess our algorithm (to evaluate whether the algorithm possesses stability with different data sets, let $k = 10, 5$ for Tables 3 and 4, respectively).

Compared with the test by 10-fold cross-valuation, the results generated from the test by 5-fold cross-valuation illustrates that the recall and precision changed slightly when the number of training samples were decreased and the search time was stable. The experimental results show that the algorithm has good performance. Recall and precision achieve 80.67 percent and 73.96 percent respectively and possess stability with the different size of data sets.

Case Study

We designed a unified interface in TREMAgent so that users could input various requests and provide cases to demonstrate how TREMAgent processes user requests. Figure 5 shows the system's identification process. The top of Figure 5 shows a protein sequence request. The sequence was sent to the TFIdent agent to identify whether it is a

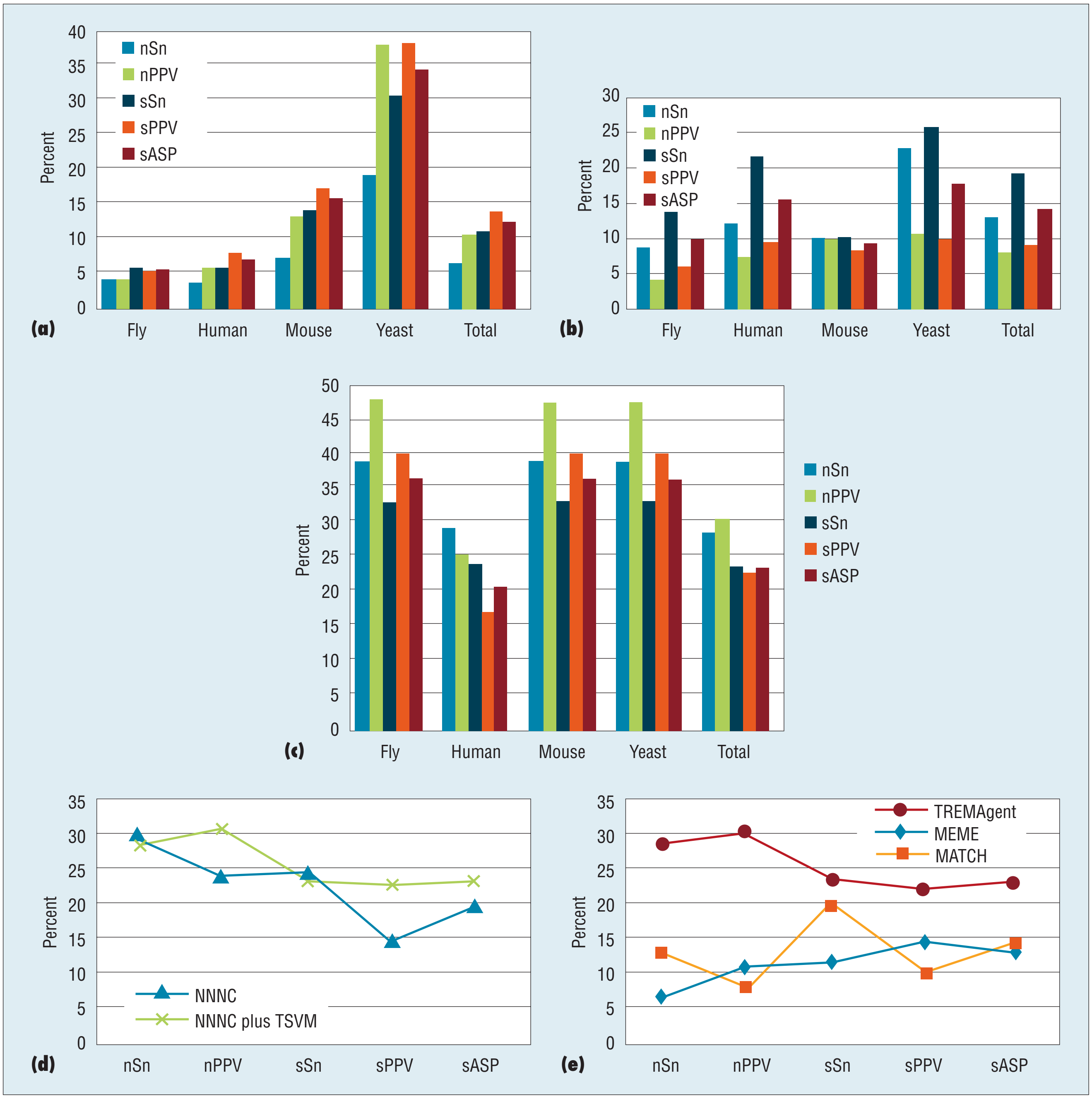


Figure 4. Average performance comparisons. The graphs show the accuracy of (a) the MEME method on four different species (fly, human, mouse, yeast) and the total data sets, (b) the Match method, (c) our methods, and (d) the performance comparison between the NNNC algorithm and the combination method of NNNC and TSVM. (e) This chart shows that Match performs slightly better than MEME. However, the accuracy is lower than our method both at the site and nucleotide level.

TF. The result is sent to the user agent as output to the users. The bottom of Figure 5 shows the result. The system identifies whether an input sequence would be a TF/TFBS when the user provides a request sequence.

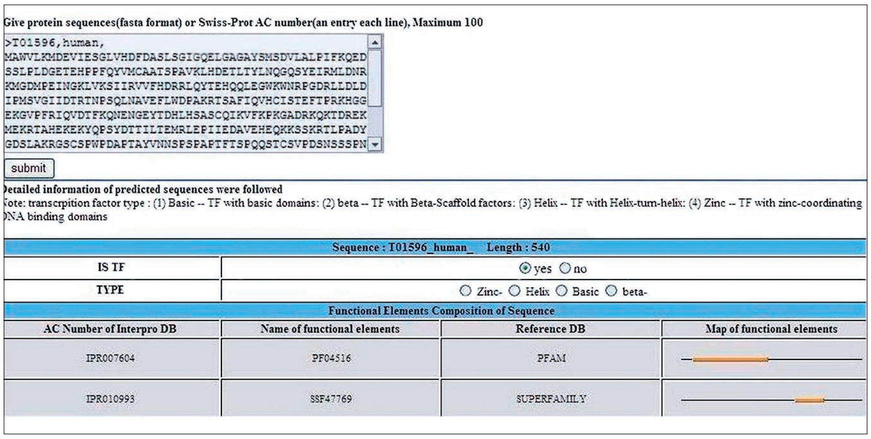
In another case, a user inputs a query keyword and the user agent detects the request. Then the coordinator agent decides the workflow and finally the system outputs the result. This shows that the system can execute the query task if the users input the keywords. The system can also provide results from automatic identification for online databases when they update.

These cases show that our system can distinguish and deal with various requests automatically, using agent technology to supply the specific workflow according to user interests

Table 4. Performance of text mining for TFBSs.

Data	Positive	TP	Negative	FP	Recall (%)	Precision (%)	Time (s)
01	300	228	305	104	76.00	68.67	122
02	300	223	321	103	74.33	68.40	120
03	300	263	330	106	87.67	71.27	106
04	300	242	296	60	80.67	80.13	97
05	300	254	292	53	84.67	82.74	101
Average					80.67	73.96	109

rather than requiring biologists to plan a workflow by themselves.



Give protein sequences(fasta format) or Swiss-Prot AC number(an entry each line), Maximum 100

```
>T01596,human,
MAWVLKMDEVIESGLVHDFDASLSGIGQELGAGAYSMSDVLALPIFKQED
SSLPLDGETEHPPFQYVMCAATSPAVKLHDETLTYLNQGQSYEIRMLDNR
KMGDMPEINGKLVKSIIRVVFHDRRLQYTEHQQLEGWKWNRPGDRLLDLD
IPMSVGIIDTRTNPSQLNAVEFLWDPAKRTSAFIQVHCISTEFTPRKHGG
EKGVPFRIQVDTFKQNENGEYTDHLHSASCQIKVFKPKGADRKQKTDREK
MEKRTAHEKEKYQPSYDTTILTEMRLEPIIEDAVEHEQKKSSKRTLPADY
GDSLAKRGSCSPWPDAPTAYVNNSPSPAPTFTSPQQSTCSVPDSNSSSPN
```

submit

Detailed information of predicted sequences were followed
Note: transcrpition factor type : (1) Basic -- TF with basic domains: (2) beta -- TF with Beta-Scaffold factors: (3) Helix -- TF with Helix-turn-helix: (4) Zinc -- TF with zinc-coordinating DNA binding domains

Sequence : T01596_human_ Length : 540

IS TF	[✓ yes] ○ no
TYPE	○ Zinc- ○ Helix ○ Basic ○ beta-

Functional Elements Composition of Sequence

AC Number of Interpro DB	Name of functional elements	Reference DB	Map of functional elements
IPR007604	PF04516	PFAM	
IPR010993	SSF47769	SUPERFAMILY	

Figure 5. The system's identification process. The TREMAgent system can distinguish and deal with various requests automatically. The top shows a protein sequence request and the bottom shows the result.

TREMAgent is the first flexible and collaborative multiagent system for transcriptional regulatory elements mining. The experimental results tested on real data sets show that the TREMAgent system improves the overall performance and provides feasible services thanks to the collaboration supported by agents. It could serve as an important resource for transcriptional regulatory research. In future work, we will complement TREMAgent and extend it to investigate the construction of regulatory networks involving MicroRNAs (miRNAs).

Acknowledgments

We acknowledge Boren Qiu, Ming Chen, Kun Zhou, and Yue Chen for the system implementation. This work was supported in part by grants from the High-Tech Research and Development Program of China (2006AA02Z329) and the National Natural Science Foundation of China (60573093).

References

1. E. Merelli et al., "Agents in Bioinformatics, Computational and Systems Biology," *Bioinformatics*, vol. 1, no. 8, 2006, pp. 45–59.
2. M. Tompa et al., "Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites," *Nature Biotechnology*, vol. 23, no. 1, 2005, pp. 137–144.
3. Z. Qian, Y.D. Cai, and Y.X. Li, "Automatic Transcription Factor Classifier Based on Functional Domain Composition," *Biochemical and Biophysical Research Comm.*, vol. 347, no. 1, 2006, pp. 141–144.

THE AUTHORS

Yun Xiong is a lecturer in computer science at Fudan University. Her research interests include data science, data mining, bioinformatics, and databases. Xiong has a PhD in computer science from Fudan University. Contact her at yunx@fudan.edu.cn.

Guangyong Zheng is a PhD candidate in life science at Fudan University. His research interests include bioinformatics and data mining. Zheng has a MS in life science from the Chinese Academy of Sciences. Contact him at zhenggy@sibs.ac.cn.

Qing Yang is a master's candidate in computer science at Fudan University. His research interests include databases, data mining, and bioinformatics. Yang has a BS in Computer science from Tongji University. Contact him at 062021131@fudan.edu.cn.

Yangyong Zhu is a professor in computer science at Fudan University. His research interests include data science, data mining, and bioinformatics. Zhu has a PhD in computer science from Fudan University. Contact him at yyzhu@fudan.edu.cn.