Computational Intelligence

An International Journal

Computational Intelligence, Volume 27, Number 4, 2011

HIGH-PRECISION BIO-MOLECULAR EVENT EXTRACTION FROM TEXT USING PARALLEL BINARY CLASSIFIERS

Sofie Van Landeghem,[1,2] Bernard De Baets,[3] Yves Van de Peer,[1,2] and Yvan Saeys[1,2]

[1]*Department of Plant Systems Biology, VIB, Gent, Belgium*

[2]*Department of Plant Biotechnology and Genetics, Ghent University, Gent, Belgium*

[3]*Department of Applied Mathematics, Biometrics and Process Control, Ghent University, Gent, Belgium*

We have developed a machine learning framework to accurately extract complex genetic interactions from text. Employing type-specific classifiers, this framework processes research articles to extract various biological events. Subsequently, the algorithm identifies regulation events that take other events as arguments, allowing a nested structure of predictions. All predictions are merged into an integrated network, useful for visualization and for deduction of new biological knowledge.

In this paper, we discuss several design choices for an event-based extraction framework. These detailed studies help improving on existing systems, which is illustrated by the relative performance gain of 10% of our system compared to the official results in the recent BioNLP'09 Shared Task. Our framework now achieves state-of-the-art performance with 37.43 recall, 54.81 precision and 44.48 F-score.

We further present the first study of feature selection for bio-molecular event extraction from text. While producing more cost-effective models, feature selection can also lead to a better insight into the complexity of the challenge.

Finally, this paper tries to bridge the gap between theoretical relation extraction from text and experimental work on bio-molecular interactions by discussing interesting opportunities to employ event-based text mining tools for real-life tasks such as hypothesis generation, database curation and knowledge discovery.

Key words: BioNLP, machine learning, text mining.

1. INTRODUCTION

Text mining tools have become a necessity to keep up with the ever increasing pace of publications in the field of molecular biology. As they facilitate full integration of knowledge stored in both structured databases and research articles, text mining algorithms for life sciences have been widely studied during the last few decades. However, considering state-of-the-art performance of current tools, it becomes clear that they should still be significantly improved upon before being sufficiently reliable and applicable on a large scale.

This paper presents a machine learning (ML) framework that accurately extracts various complex bio-molecular events from text, ranging from binding events and protein catabolism to gene expression and regulation. Each event is characterized by a trigger, which is a continuous stream of tokens linked to a certain event type, e.g., "homodimerization" for a binding event. A trigger word is not restricted to a particular set of part of speech tags, though verbs and nouns are the most commonly used triggers. Furthermore, a trigger can consist of multiple consecutive words, e.g., "binding partner." To define a full bio-molecular event, a trigger is combined with one or several arguments, which are either simple named entities or recursively defined new events. Regulation events for example express regulatory pathways concerning various events, thus allowing for a nested structure of predictions.

Figure 1 shows a representative example of the complexity of this task. While the protein "HIV-TF1" (T1) is not grammatically dependant on the trigger "binding" (T4), it is semantically correct to link these words together to create a meaningful binding event. It is often necessary to scan the whole sentence or even a whole paragraph to find suitable

Address correspondence to Yves Van de Peer, Department of Plant Systems Biology, Technologiepark 927, B-9052 Gent, Belgium; e-mail: yves.vandepeer@psb.vib-ugent.be

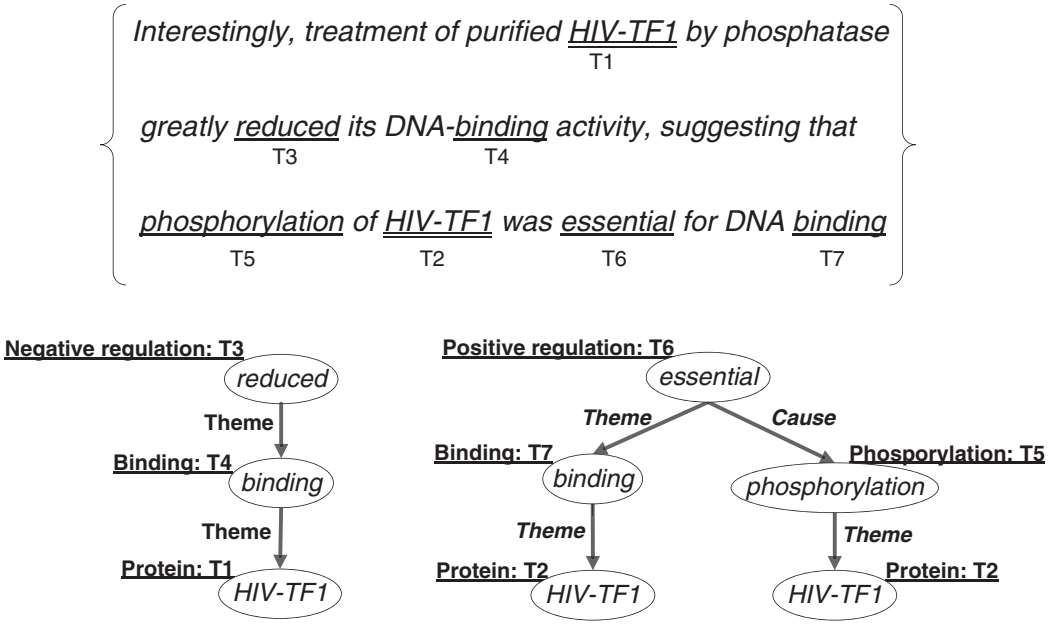

FIGURE 1. An example sentence from PubMed article 1653950 of the training corpus. It contains five events: two (unary) binding events, one phosphorylation, one negative (unary) regulation and one positive (binary) regulation event.

arguments for certain events. Another interesting example is shown for trigger T7 ("binding"), for which there is no explicitly expressed argument, resulting in a semantic copy of the protein "HIV-TF1" (T2).

Our ML framework is designed to process each type of event in parallel using binary support vector machines (SVMs). All predictions are assembled in an integrated graph, on which heuristic postprocessing techniques are applied to ensure global consistency (Section 2).

We have thoroughly investigated the influence of certain design choices for the ML framework, such as various syntactic parsers, instance definition, choice of kernel for the SVMs, parameter optimization and new postprocessing methods. Furthermore, this paper presents the first results of applying feature selection (FS) for this challenge. By gaining new insights from the training data, our initial system (Van Landeghem et al. 2009) has obtained a relative rise in performance of 10% on the test set (Section 3).

After presenting the final performance of our system, we explore how theoretical studies can be turned into practical applications such as hypothesis generation, database curation and knowledge discovery (Section 4). Finally, Section 5 presents the conclusions of this work.

2. DESIGN OF A SUPERVISED FRAMEWORK

2.1. Task Setup

The complex bio-molecular interactions extracted by our text mining tool can be classified into nine broad categories of events. Six event types directly influence proteins ("protein

events"), of which five always take exactly one argument: localization, gene expression, transcription, protein catabolism, and phosphorylation. A binding event can have one argument (e.g., protein–DNA binding), two arguments (e.g., protein–protein interaction) or more (e.g., complex formation). Furthermore, three regulation event types are defined: regulation, positive (up-) regulation and negative (down-) regulation, each consisting of a mandatory theme argument and an optional cause argument. Both argument slots can either consist of a protein or any other event (Figure 1).

The various event types were selected from the GENIA ontology (Kim, Ohta, and Tsujii 2008) and represent some of the most important events in protein biology, covering protein metabolism, protein modification, fundamental molecular events and causal relations (Kim et al. 2009).

Any supervised learning approach heavily relies on high-quality annotated data. For the task of event extraction, three distinct data sets were provided by the organizers of the recent BioNLP'09 Shared Task on event extraction: training data (800 articles), development data (150 articles) and the final test data (260 articles). These data sets all consist of PubMed abstracts extracted from the GENIA corpus (Kim, Ohta, and Tsujii 2008). Stand-off annotation locating relevant bio-molecular entities such as proteins is provided for all three data sets. Both training data and development data further include gold standard annotations defining events as a specific trigger word in text, combined with one or several arguments.

To benchmark the performance of an event-based extraction system, event annotation on the test data of 260 articles can be produced and evaluated using an online submission system maintained by the Shared Task organizers. This ensures objective evaluation of the methods developed. However, in order not to overfit our system to the test set, all analyses were performed on the development data unless explicitly stated otherwise.

As the prediction of regulation events greatly depends on the ability to predict protein events, most experiments were only performed for the protein events. Conclusions drawn from this setup can easily be extended to the entire framework.

2.2. Text Preprocessing

To run an automated extraction algorithm, free text first has to be transformed into a machine readable format. To this end, data on sentence segmentation and tokenization has been made available for all articles in the data sets. Furthermore, syntactic analyses created by various parsers is also provided with the data set: both phrase structure and dependency representations are available. Phrase structures break a sentence down into constituents (phrases), which are then broken into even smaller constituents (part of speech tags). Dependency graphs on the other hand connect individual words by identifying their grammatical relations. Both formats have been widely used in the field of Natural Language Processing (NLP) and in BioNLP in particular.

A comparative study between various parsers providing both dependency graphs and phrase structure parses is shown in Table 1. This study included Dan Bikel's implementation of Collins' parsing model ("Bikel") and the Charniak–Johnson reranking parser using David McClosky's self-trained biomedical parsing model ("McClosky–Charniak"). Analyses of these parsers were provided for all articles in the data set. In addition to these, we have employed the freely available Stanford parser (De Marneffe, Maccartney, and Manning 2006). The Stanford parser performs best in our framework, yielding both higher precision and recall rates in comparison to the McClosky–Charniak parser and the Bikel parser.

Two additional NLP techniques have been used in this study: stemming and blinding of text fragments. Stemming algorithms map words to their stem, which is the basic concept of a word (e.g., "homodimerization" is mapped to "homodimer"). Blinding on the other hand

TABLE 1. Performance of Protein Events for Various Parsers.

Parser	Precision	Recall	F-score
Stanford	66.62	63.44	65.00
McClosky-Charniak	64.99	61.09	62.98
Bikel	60.94	57.87	59.37

completely transforms a word into another word and can be used to facilitate information retrieval. For example, all annotated proteins in the data set can be blinded with the string "proteinx," as the text mining module is not concerned with the exact identity of each protein. On the contrary, it is easier to learn and recognize a pattern such as "transcription of proteinx" instead of creating distinct patterns for each possible protein name.

2.3. Global System versus Parallel Processing

A crucial design choice for the extraction algorithm involves its modularity, which either has a global or a local nature. A global approach implies inferring all plausible events in a single step, which is highly computationally intensive. A local procedure on the other hand is characterized by a set of specific classifiers, creating predictions for distinct event types independently of each other.

We have chosen a local and parallel approach by designing a generic pipeline which can be employed for each event type. Such a pipeline consists of modules for dictionary creation, instance definition, feature generation, FS and classification. It can be run for each event type for which sufficient training material is available. Starting off with the nine broad categories of events (Section 2.1), we further make a distinction between unary and binary binding events. Binding events with more than two arguments are not accounted for as only five positive examples were found in the training set. Similar to binding events, we create a separate type for unary and binary regulation events, the latter defining events with a specified cause-argument. This results in a final set of 13 distinct classifiers.

The choice between a global and a local extraction method severely influences the size of the resulting data sets, limiting options for applicable classifiers. To illustrate, the global approach of Björne et al. (2009) yields a training set of 31,782 instances and 295,034 unique features. They state that the linear kernel of their multiclass SVM is the only practical choice to train the classifier with such large training sets. In contrast, the data sets obtained with our parallel design vary between 300 instances with 2,000 features (protein catabolism) and 15,000 instances with 50,000 features (unary positive regulation). This reduced complexity enables us to experiment with binary SVMs and more complex kernels such as the radial basis function. Performance results of various classifier setups are detailed in Section 2.7.

Another important advantage of the parallel design lies in the ability to easily add new event types to the framework without having to recalculate previous work. Any subset of classifiers can be run independently, resulting in a subset of predicted event types which can be extended whenever the need arises. This proves to be particularly beneficial when dealing with specific biological research questions. Life scientists are often focused on a certain set of research articles dealing with a predefined group of genes involved in only a few specific event types. Our parallel design greatly expedites the processing of the test set, offering personalized results to the user in real time.

Finally, in the parallel design all predictions of different event types can be merged after classification into an integrated network. Such a network is also compiled from the training data and then used as a model for the predicted network to locate false positives and prune the corresponding edges. This process ensures consistency of the predictions made by the parallel classifiers. More details are described in Section 2.9.

2.4. Trigger Detection

The first module of a pipeline that extracts bio-molecular events from text concerns the challenge of trigger detection. A trigger is defined by a continuous stream of tokens and is linked to a certain event type, e.g., "homodimerization" for a unary binding event. In the training data, triggers are annotated using their text offsets in a stand-off annotation format.

The challenge of trigger detection is tackled using carefully constructed dictionaries. First, all possible strings are collected by scanning the triggers in the training data and applying Porter's stemming algorithm (Porter 1980). In contrast to most other approaches, our algorithm does allow triggers to span multiple words, as this occurs frequently in the training data. This initial collection of all possible trigger strings results in entries of limited use, such as "through" for binding and "are" for localization. Such words lead to many negative and irrelevant instances as they are too general or too vague. To overcome this problem, a cleaning step is necessary.

In our original system, the dictionaries were cleaned *manually*, only keeping specific triggers for each event type (e.g., "interaction" for binding and "secretion" for localization). Inspired by the work of Buyko et al. (2009), we now use their proposed formula to calculate the importance of an event trigger t_i for a particular event type T: $Imp(t_i^T) = f(t_i^T) \backslash \sum_{p=0}^{n} f(t_p^T)$, where $f(t_i^T)$ is the frequency of the event trigger t_i of the selected event type T in a training corpus divided by the total number n of all event triggers of the selected event type T in that training corpus ($i = 0, \ldots, n$). By applying a cutoff value of 0.005, we only keep those words that are important enough for that specific event type. In contrast to the work of Buyko et al. (2009), this measure is not used for event trigger disambiguation as words are allowed to be included in trigger dictionaries of different event types. This choice was motivated by analysis of the training data, which has shown that the same word in text may actually trigger multiple events of different types. The word 'overexpression' is a frequently recurring example, as it is often linked to both a gene expression event as well as a regulation event.

Analyzing the nature of binary regulation events, it became clear that a vast majority of these events have a protein specified as its causal argument. The dictionaries of binary regulations were split up accordingly, differentiating between regulation events caused by proteins and those caused by other events. This automatically keeps the more general words (e.g., "causes") out of the dictionaries of events regulated by proteins (e.g., "response").

Table 2 shows the single most occurring trigger for each protein event type in the training data.

2.5. Instance Definition

The algorithm that defines instances in a ML framework has a severe influence on the balance of the data sets and ultimately on the performance of the framework. Careful design can limit the number of false positives obtained by the classifier, boosting precision of the predictions.

For the challenge of event extraction, an instance is defined as the combination of a trigger with one or more plausible arguments. To locate suitable triggers in text, we implemented a

TABLE 2. Most Frequently Occurring Trigger in the Training Data, for Each Event Type.

Event type	Highest ranked trigger	Percentage of occurrence
Phosphorylation	"phosphoryl"	96%
Protein catabolism	"degrad"	76%
Gene expression	"express"	68%
Unary binding	"bind"	47%
Transcription	"transcript"	45%
Localization	"secret"	31%
Binary binding	"bind"	30%

fast algorithm using Radix trees,[1] making use of the constructed dictionaries for each event type. Candidate arguments were subsequently selected from the same sentence as the trigger, as analysis on the training data reveals that 95% of all events are expressed within one single sentence.

By defining instances as any combination of a trigger co-occurring with its candidate arguments, too many instances end up being irrelevant, especially those originating from long sentences. This in turn resulted in imbalanced data sets with often less than 5% positives. For this reason, a negative-instances (NI) filter was implemented that applies some simple yet efficient heuristics to reduce the dimensionality of the data sets, relying on both dependency parsers and length of the subsentence spanning the candidate event.

As described in Section 2.2, the Stanford parser was selected to create dependency graphs for each input sentence. A minimal sub-graph is then extracted for each instance, spanning its trigger and all arguments. Figure 2 shows the dependency parse of a sentence containing several bio-molecular events. As an example, the subgraph of the phosphorylation event spans nodes 17 and 19, while the positive regulation event triggered by "essential" spans the subgraph consisting of nodes 17, 19, 21, 23, and 24.

The NI filter enforces a cutoff on the size of the dependency subgraph, as positive instances are known to be expressed in smaller sub-graphs than negative examples. Figure 3 shows that the subgraphs of positive binary binding instances are never larger than 10 edges, while negative instances may contain up to 18 edges. Only keeping instances with subgraphs smaller than 8 edges will successfully discard 35% irrelevant negatives, while keeping 92% of the positive instances.

Similarly, the length of the sub-sentence spanned by a candidate event is used as a second parameter of the NI filter. Setting the threshold at 175 characters for binary binding events includes 99% of the positive examples, while removing about 20% irrelevant negatives.

The NI filter thus reduces the size and skewness of the data sets, resulting in faster classification pipelines. Furthermore, by identifying a large portion of negative instances even before they are processed by the classifier, a gain of 2.1 points in F-score is obtained for protein events on the development set compared to a setup which does not use the NI filter.

For regulation events, the NI filter becomes a true necessity as considering all candidate argument sets from each sentence leads to data sets of tremendously high complexity. For example, the filter reduces the number of negative unary positive regulation events from 39.419 to 13.995, improving the percentage of positives from 4% to 10%, while reducing the feature set from 127.098 to 49.384.

[1] Java implementation by Tahseen Ur Rehman, http://code.google.com/p/radixtree/

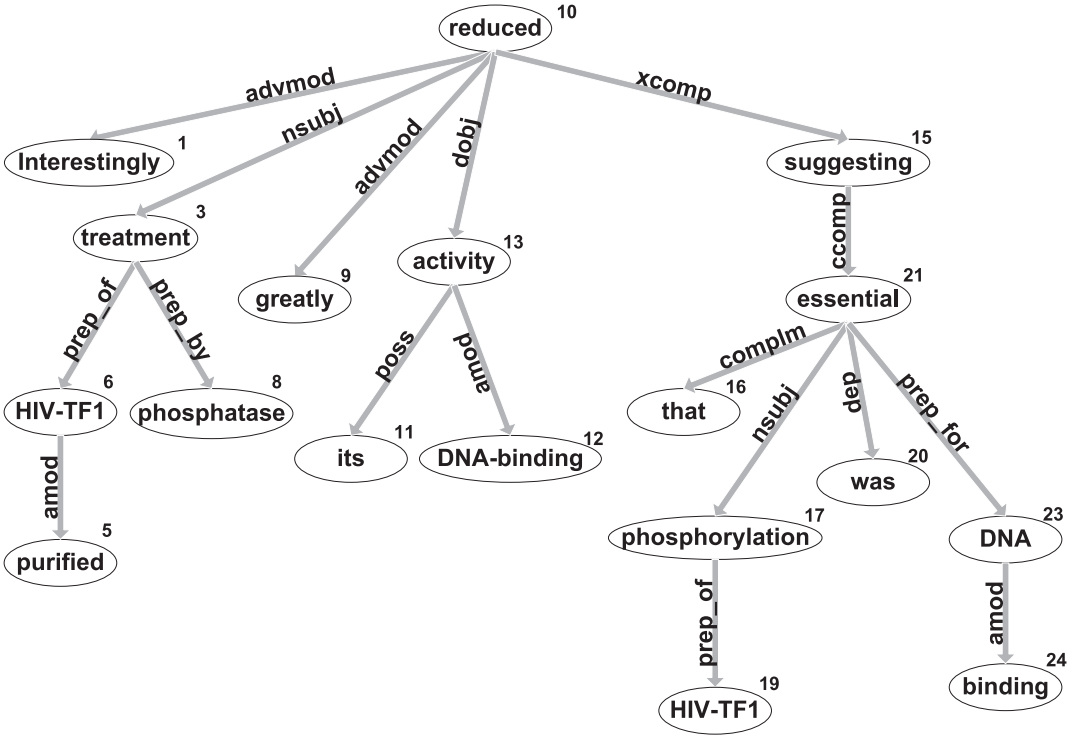

FIGURE 2. Example of a dependency graph for the sentence "*Interestingly, treatment of purified HIV-TF1 by phosphatase greatly reduced its DNA-binding activity, suggesting that phosphorylation of HIV-TF1 was essential for DNA binding*," retrieved from PubMed article 1653950. Each word and punctuation mark in the original sentence is numbered, and these numbers are used as unique labels for the nodes.

For training purposes, even positive instances exceeding the NI filter cutoff are taken into consideration as they add important information to the data set. To boost the number of positives even further, the system was extended with a module that processes "Equivalence" information. This type of annotation is included with the data sets and marks synonyms and acronyms referring to the same biological entity. For example, in the sentence "*The c-Rel homodimer has a high affinity for interleukin-6 (IL-6)*," *interleukin-6* and *IL-6* are annotated as equivalent entities. In the gold standard event annotation, only *c-Rel* and *IL-6* are annotated as binding partners. We thus implemented a function which recognizes the binding event of *c-Rel* and *interleukin-6* as a second event, and which recursively applies this methodology to create new regulation events when one of the two equivalent events occur as arguments for other events. The final distributions of positive and negative examples of protein events in the training data range between 7% and 51% (Table 3).

2.6. Feature Generation

Our feature generation module is based on previous work on extracting protein-protein interactions (PPIs) from text (Van Landeghem et al. 2008). PPIs were considered to be binary and there was no specification of trigger words. Only one path in the dependency graph was analyzed for each instance: the shortest path between the two candidate binding partners.

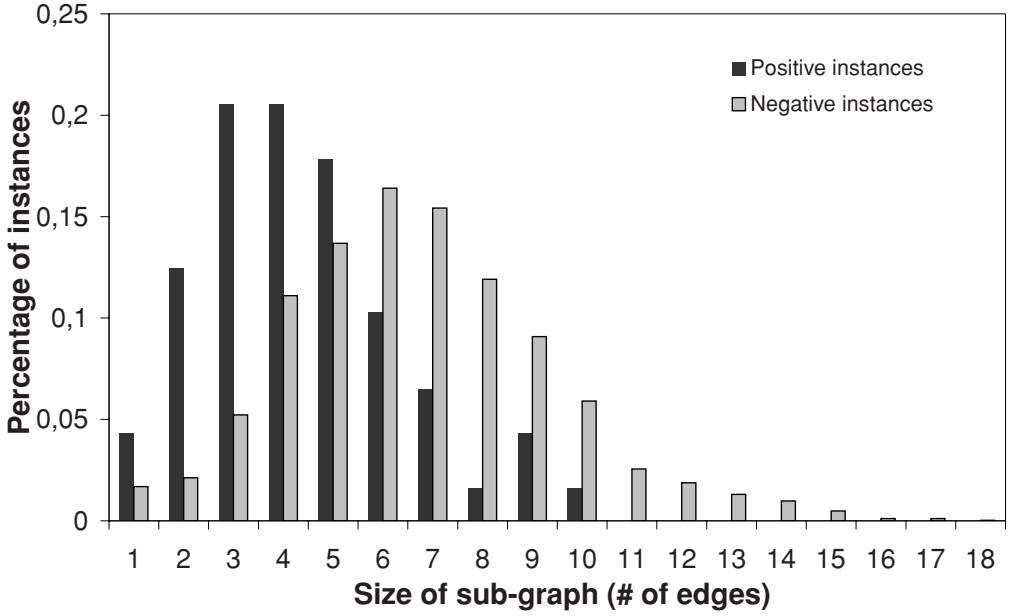

FIGURE 3. Distribution of binary binding instances in the training data.

TABLE 3. Final Distribution of Instances for Protein Events in the Training Data.

Event type	# neg. inst.	# pos. inst.	% pos. inst.
Phosphorylation	156	163	51.1
Protein catabolism	162	109	40.2
Gene expression	5,328	1,722	24.4
Unary binding	3,512	580	14.2
Binary binding	2,169	222	9.3
Transcription	7,196	562	7.2
Localization	3,561	269	7.0

However, this work deals with much larger and more complex sub-graphs, thus requiring a different set of features to fully capture the semantics of each instance. As we are now dealing with graphs instead of trees, we have to exclude "edge walks," i.e., patterns of two consecutive edges and their common vertex (e.g., "nsubj VBZ prep"). Vertex walks are now the main source of information derived from the dependency graph, consisting of two vertices and their connecting edge (e.g., "essential nsubj phosphorylation"). For these patterns, both lexical as well as syntactic information is considered. For the lexical variant, protein names and triggers were blinded to extract more general patterns. The same principle is applied with theme and cause arguments for regulation events. To illustrate using Figure 2, we blind the nodes 17 and 19 as "causex" and nodes 23 and 24 as "themex" when processing the positive regulation event triggered by the word "essential," blinded as "triggerx." Resulting features for the vertex walks would then include "triggerx nsubj causex" and "triggerx prep_for themex."

Blinding avoids overfitting of the classifier and simplifies the feature set. However, in order not to lose valuable information, we include a few other features that can be used to reconstruct the original instance when necessary. Lexical and syntactic information about the trigger are stored in separate features. Similarly, additional features are included for regulation events, storing whether the arguments are proteins or events, and specifying the exact event type.

In addition to these features, we augment each feature vector with lexical information. First of all, a bag-of-words (BOW) approach is applied to all vertices in the subgraph. This automatically excludes uninformative words such as prepositions, as they do not appear as individual nodes on the graph. Furthermore, we added trigrams derived from the whole sentence. These are three stemmed consecutive words from the subsentence spanning the event. As an example, the words "by inducing transcription" lead to the stemmed trigram "by induc transcript."

The size of the subgraph and the length of the subsentence are also included in the feature vector. Even though they are used in the previous step as parameters for the NI filter, these two parameters are still relevant for classification. Indeed, instances that only just pass the filter still have a higher chance of being negative.

2.7. Classification

Our framework needs a classifier able to deal with thousands of instances, thousands of features, and a class imbalance of up to 93% negative instances. To this end, we used the LibSVM implementation provided by WEKA,[2] which is a state-of-the-art binary classifier. Analyses were conducted experimenting with both the linear kernel and the radial basis function (RBF), and various strategies for parameter tuning were implemented.

The linear kernel requires tuning of the parameter c, which was implemented with an internal 5-fold cross-validation (CV) loop performing a grid search on the training portion of the data. Values between 2^{-5} and 2^{14} were tested, and the one producing the best F-score was automatically selected for each data set.

However, a more complex problem arises when both the parameters γ and c of the RBF kernel have to be tuned simultaneously. A combined search strategy has been tested extensively, using both a grid search and a more advanced pattern search. A pattern search is a parameter optimization algorithm that starts at the center of the search range and subsequently explores small steps in each direction for each parameter (Lewis and Torczon 1999). If the fit of the model improves, the search center moves to the new point and the algorithm is repeated. If no improvement is found, the step size is reduced and the search executed again. The pattern search stops when the search step size is reduced to a specified minimum value.

Instead of starting at the center however, the combined strategy first employs a grid search to determine rough values for γ and c. These values are then fine-tuned by the pattern search. This strategy should avoid ending up in a local optimum, while at the same time being less computational expensive than a true grid search.

Despite careful design of the tuning algorithm, performance drops significantly when using the RBF kernels tuned for both parameters (Table 4). However, comparing this classifier to one that only has been tuned for the c-parameter, about equal performance is reached within the internal CV loop. The most plausible explanation for the final drop in performance thus seems to be that the complex search strategy severely overfits the parameters on the training portion of the data. Eventually, we therefore decided to leave γ at its default value.

[2] Available at http://www.cs.waikato.ac.nz/ml/weka/

TABLE 4. Performance of Protein Events for Various Classification Settings.

Kernel	Parameters	Precision	Recall	F-score
RBF	tune c	66.62	63.44	65.00
RBF	tune c and γ	30.00	28.75	29.36
Linear	tune c	64.58	61.34	62.92

TABLE 5. Performance of Protein Events for Different Feature Sets, Tested with the RBF Kernel.

FS	Features	Precision	Recall	F-score
Manual	All	66.62	63.44	65.00
Manual	All except BOW	61.34	64.07	62.68
Manual	All except trigrams	62.21	65.15	63.64
Gain ratio	90%	69.71	61.71	65.46
Gain ratio	75%	69.08	61.59	65.12
Gain ratio	50%	68.58	60.97	64.55
Gain ratio	25%	67.13	60.10	63.42

Even though the linear kernel runs faster, it performs slightly worse than the RBF kernel, with a drop in performance of about 2 points F-score. Consequently, the RBF kernel was chosen for all other analyses.

Finally, we tested the influence of assigning higher weights to positive training instances, to try and correct for the imbalanced nature of the data, but this had almost no effect on overall classification performance.

2.8. Feature Selection

The rich feature representation of the framework results in high-dimensional feature sets and the question arises whether all these features are absolutely necessary for the classification task. In particular, one could worry about the amount of noise caused by lexical features from random words in the same sentence, i.e., the BOW and trigram features. To assess their influence on performance, new experiments have been conducted, each time excluding one specific type of feature (Table 5, first three rows). Clearly, all types of features in our rich feature set contribute to the global performance, as there is a drop in F-score of 2.32 and 1.34 when leaving out BOW or trigram features respectively.

However, there is still a considerable number of irrelevent features creating noise for the classifier. To account for this with a more systematic approach, we applied fully automated FS. FS techniques aim at identifying a subset of the most relevant features from a large initial set of features (Guyon and Elisseeff 2003). In contrast to other dimensionality reduction techniques such as methods based on projection, FS techniques preserve the semantics of the features and thus create the opportunity to gain a deeper insight into the specific properties of the most important ones.

Depending on the interaction with the model, three classes of FS techniques can be defined (Saeys, Inza, and Larranaga 2007). In this work, we focus on the class of *filter* methods, which perform FS by looking only at the intrinsic properties of the data, thus being independent of the classification model used afterwards. The filter method used in this work

is based on the information-theoretic concept of *gain ratio*. Previously, this method has been succesfully applied to produce faster and more cost-effective models for the extraction of PPIs from text (Van Landeghem et al. 2008).

Regarding a given set of training patterns S as a distribution over the class labels, its entropy can be calculated as

$$H(S) = -\sum_{i=1}^{s} p(c_i) \log_2 p(c_i)$$

with $p(c_i)$ denoting the proportion of patterns in S belonging to class c_i. The *information gain* $IG(S, D)$ then represents the expected reduction in entropy (uncertainty) when splitting on a feature D, and is calculated as

$$\begin{aligned} IG(S, D) &= H(S) - H(S|D) \\ &= H(S) - \sum_{j \in V(D)} \frac{|S_j|}{|S|} H(S_j) \end{aligned}$$

where $V(D)$ denotes the possible values for feature D and S_j is the subset of S for which feature D has value j.

Adjusting the bias towards features with a larger number of possible values, the information gain is scaled by the entropy of S with respect to the values of feature D. This finally results in the *gain ratio* $GR(S, D)$:

$$GR(S, D) = \frac{IG(S, D)}{-\sum_{j \in V(D)} \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}}$$

All features can now be ranked from most to least influential by sorting their gain ratios (Hall and Smith 1998). The top k features are then used to construct a simplified classifier. In this paper, this cutoff is defined as a percentage of the size of the original feature space. Analyses for the framework of event extraction confirm results of our previous PPI study: up to 50% of all features can be removed without losing more than 1 point of F-score, while at the same time creating faster classification models (Table 5, last four rows).

The explicit ranking of the gain ratio algorithm was statistically analysed to determine the type of features occurring more frequently at the top. Figure 4 shows the results for the top 30% of the gain ratio ranking. For each main type of feature, a curve represents the percentage of values occurring in the top. A random baseline method would resemble the identity function, shown with a thick black line. The chart, however, shows that syntactic features are overrepresented in the top ranking. Furthermore, lexical information about triggers and vertex walks are in general less significant than their syntactic counterparts, though they are usually bigger in number. This could explain the initial peak of lexical triggers and BOW features: only a small percentage of these features are truly relevant.

2.9. Postprocessing

LibSVM produces numeric values between 0 and 1 for each instance in the testing set, but a postprocessing step is necessary to obtain the final set of predictions. In our initial system, we determined cutoff values on the LibSVM scores by evaluating the classifiers on the development data. However, this methodology had a serious drawback: new cutoff values

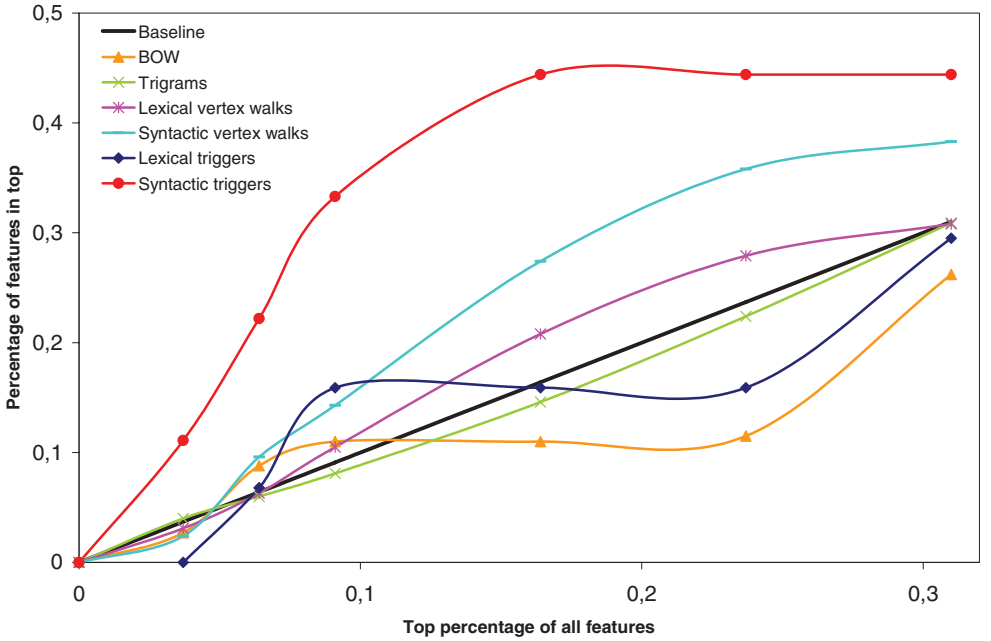

FIGURE 4. Analysis of top-ranked features.

had to be defined whenever a new classifier was trained or when other parameters had been changed in the system.

We now approach the problem of selecting the right predictions from a different angle, ensuring global consistency of the final set of predictions rather than making local decisions. As a first step, all instances from the testing set are collected and merged into an integrated network with weighted edges according to their LibSVM scores. Global consistency of the network is then imposed by using a model obtained from the training data. As instances are created in the same way for both training and testing data sets, the percentage of positives in the training set provides a reasonable estimate for the number of positives in the testing set. Furthermore, this measure is independent of the classifier. By keeping the top ranked predictions until a certain percentage of positives is reached, we gain 1.6 points F-score for protein events on the development data.

This method creates an additional advantage, allowing for the development set to be used as training data, as the cutoff values are now obtained from the training data only. The classifiers can thus be trained on the merged articles from both training and development data, extending the training set from 800 articles to 950. We have used this setup only to produce the final results on the final test set.

Our original system included a module to check overlapping triggers of different event types. Following our intuition, we assumed that one word should never lead to two triggers of distinct event types at the same time. However, after running a detailed analysis on the training data, we decided to reject this hypothesis. For example, the word "overexpression" can simultaneously lead to both a gene expression and a regulation event. Consequently, this postprocessing module has been discarded. While most other systems disambiguate the type of a trigger in an early stage of the pipeline, our parallel approach can easily avoid this issue and thus model the data more truthfully.

To fine-tune the model and ensure global consistency of the predictions even further, more detailed information can be extracted from the training data. For example, we notice that the training data does not contain any examples where two regulation events contain the same arguments, one being the cause in the first event, and the theme in the other. Consequently, when this does happen in the test data, we select the one with the highest SVM score.

3. RESULTS

3.1. System Improvement

Our official result for the Shared Task reports a global performance of 40.54 F-score on the test data, achieving 5th place out of 24 participants. In the previous sections, we have described extensions and fine-tuning of our ML system. To summarize, the following changes were made compared to the original system: first of all, the dictionaries are now compiled by calculating the importance of each trigger for a particular event, instead of using manual filtering. The system can now also process overlapping triggers of different event types, as this initial limitation was artificial and in contradiction with the training data. Regarding instance creation, the new system utilizes "Equivalence" information to create more positive examples. Furthermore, a module for FS was incorporated and is currently set to high-precision removal of 10% of irrelevant features. Finally, when all predictions are assembled, we now estimate the percentage of positives in the testing set by a model derived from the training set instead of using an arbitrary cutoff value. Our new and improved system achieves 37.43 recall, 54.81 precision, and 44.48 F-score.

During development, only a few experiments were run on the final test set, in order not to overfit the classifiers to the test data. All other analyses were performed on the development data set. Remarkably enough, a higher relative gain in performance was obtained on the final test data (10%) than on the development data (5%). This indicates that the original system could have been slightly overfitted to the development data, whereas the improved system is more general and can better cope with new data. It could also be due to using the development data as training data for the final predictions on the test set, resulting in a larger training corpus and more accurate classifiers.

3.2. Learning Curve

To assess the influence of the size of the training data on the final performance, the training data was divided into 8 portions of 100 articles. The first classifier was trained using one portion only. By incrementally adding 100 articles to train the next classifier, prediction performance started rising (Figure 5).

Two versions of this experiment were conducted, experimenting with different dictionaries for the extraction of triggers. The first one was run with dictionaries constructed from all 800 training articles ("complete dictionaries"). For the second version however, the dictionaries were only based on the data available in the smaller portion of the training data ("smaller dictionaries"). The discrepancy between these two versions clearly shows the added value of having better curated dictionaries. As both curves do not indicate a level of saturation quite yet, we hypothesize that supervised learning systems might benefit from even more training data. This has motivated us to include the development data as training for the final system, which is benchmarked on the final test data. Consequently, the dictionaries were also based on this extended training set of 950 articles.

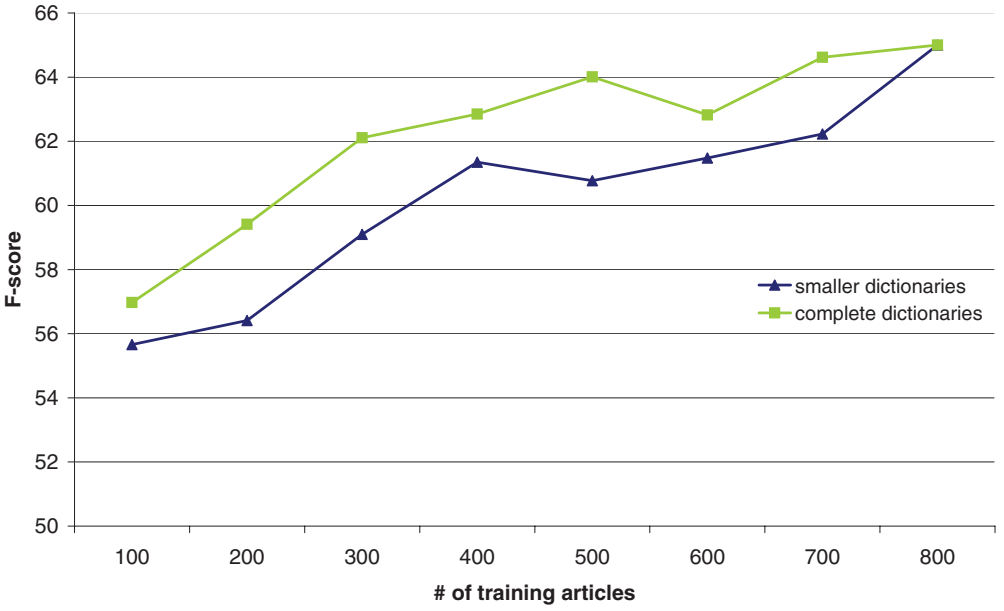

FIGURE 5. Learning curve of protein events, benchmarked on the development data.

TABLE 6. Maximal Recall for the Development Data.

Event type	Maximal recall
Protein catabolism	100.00%
Phosphorylation	95.74%
Gene expression	92.13%
Transcription	91.46%
Localization	88.68%
Binding	79.03%
Regulation	46.15%
Negative regulation	42.86%
Positive regulation	40.36%

Obviously, numbers reported in this paper for benchmarking on the development data were obtained by classifiers trained on the original 800 articles only.

3.3. Maximal Recall

To evaluate maximal recall of our instance extraction method, an evaluation using an all-true classifier was performed, which roughly corresponds to a naive baseline co-occurrence method. Table 6 clearly shows that maximal recall is quite high for almost all protein events; only binding and localization events achieve less than 90% recall. In general we

TABLE 7. Final Performance on the Test Data, Detailed for Each Event Type.

Event type	R	P	F
Phosphorylation	77.04	70.27	73.50
Gene expression	62.74	82.21	71.17
Protein catabolism	64.29	50.00	65.25
Localization	43.10	80.65	56.18
Transcription	57.66	53.02	55.24
Binding	33.43	42.03	37.24
Total protein events	54.68	67.69	60.49
Negative regulation	22.43	46.20	30.20
Positive regulation	22.99	37.79	28.59
Regulation	15.12	28.21	19.69
Total regulation events	21.48	37.85	27.40
Global total	37.43	54.81	44.48

can conclude that our instance definition module and the NI filter perform well for protein events.

In contrast, recall of the regulation events is never above 50%. This is partly due to missing protein events, but could also be caused by the restriction of only extracting events within one sentence. In future work, we hope to overcome this limitation by incorporating anaphora resolution.

3.4. Detailed Performance Analysis

Details of the final performance for each event type are listed in Table 7. Not suprisingly, the three types of protein events performing best in our ML framework, correspond to the data sets with the highest percentage of positive examples: phosphorylation, gene expression and protein catabolism. Furthermore, we found that each of these three types is linked to a possible trigger that accounts for more than 65% of the training examples (Table 2). In contrast, binding events are much more difficult to identify as they are expressed with a broad spectrum of possible triggers. This is also illustrated by the maximal recall study, as new triggers can not be identified in the test set if they have not occurred in the training data.

For regulation events, an F-score of 27.40 is obtained on the test data, which is significantly lower than the performance of protein events (60.49). This discrepancy is due to the inherent complexity of regulation events making them more difficult to extract.

4. DISCUSSIONS

4.1. Evolution of BioNLP

BioNLP emerged from the combined expertise of molecular biology and computational linguistics. At first, the community was mainly focused on named entity recognition (NER) and simple binary relation extraction, such as protein-protein interactions (Plake, Hakenberg, and Leser 2005; Giuliano, Lavelli, and Romano 2006; Fundel, Küffner, and Zimmer 2007; Saetre, Sagae, and Tsujii 2008). Two recent community-wide challenges, Biocreative (Hirschman et al. 2005; Krallinger et al. 2008) and the BioNLP'09 Shared Task,

have shown their merits by providing common benchmarking data and a framework for meaningful comparison between various systems.

The Biocreative tasks consist of mainly monolithic challenges and require implementation of a complete pipeline, including named entity recognition, relation extraction and gene normalization. In contrast, the BioNLP'09 Shared Task focuses on the subtask of relation extraction only, considering more complex interactions than ever before. The resulting predictions can model the true interaction graph more accurately.

Lessons learned, from both challenges attribute to the goal of bringing NLP research closer to its practical application in the biological sciences. In this section we describe a few opportunities to use an event-extraction framework in such practical applications.

4.2. Curation of Structured Databases

Event-based extraction algorithms can form a useful contribution to the development of detailed and structured databases. However, gene normalization is a crucial step to link text mining results to structured database entries. Mapping gene symbols to their unique identifiers allows for full integration between various resources, but it remains a nontrivial task considering the high degree of ambiguity in gene nomenclature. A recent study of Wermter, Tomanek, and Hahn (2009) obtains a competitive F-score of 86.4 on the Biocreative II test set.

Once the named entities are uniquely identified, extracted relations can be mapped to database facts using ontologies. All event types used in the Shared Task have been selected from the Genia ontology, and can be mapped onto the Gene Ontology (GO). GO is a commonly used resource consisting of three fine-grained ontologies, covering molecular functions, cellular components and biological processes (The Gene Ontology Consortium 2008). Mapping event types to GO terms enables integratation of text mining predictions with the database of Gene Ontology Annotation (GOA).

Going one step further, negation information extracted from text might be used to locate inconsistencies between research articles and database facts, while speculation information could influence the confidence score of such database records. The official Shared Task included a subtask about the recognition of negation and speculation, for which promising results were obtained (Kilicoglu and Bergler 2009; Van Landeghem et al. 2009).

4.3. Precision vs. Recall

In most retrieval systems, an inverse relationship exists between recall and precision. An important advantage of a ML framework lies in its ability to be tuned to achieve either good precision or good recall. The nature of an application often prefers one of the two.

As observed by Cohen et al. (2009), high precision and low recall systems can be compensated by the amount of redundancy across the literature. Well-known and studied interactions will thus eventually be picked up by the method as more articles are being processed. It could be interesting to employ such a system to provide reliable textual evidence for claims made in structured databases, using an integrative approach as described in Section 4.2.

On the other hand, biological experimentalists might especially be interested in low-confidence predictions as these may represent interesting hypotheses for new studies. Humans are particularly good at selecting the right information from a collection of noisy predictions,

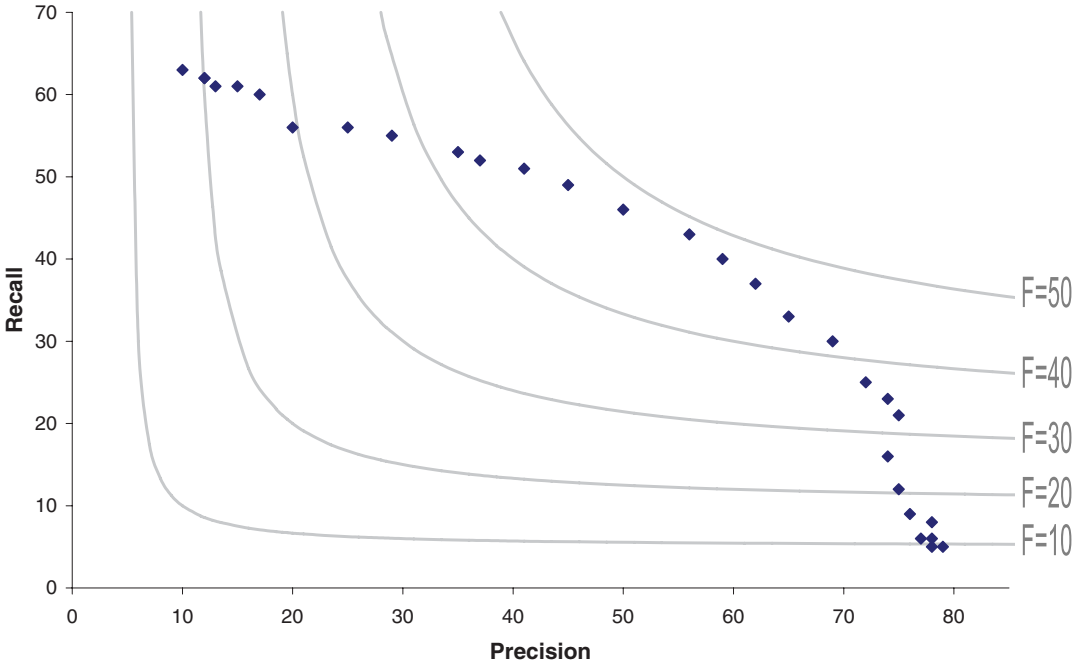

FIGURE 6. Precision-recall curve for predicting all events: dots indicate the total performance when varying the LibSVM thresholds. Plotted lines each mark a constant F-score level.

as illustrated daily by the success of Google,[3] a search engine that often produces thousands of informative hits of which only a few are truly relevant to the user.

To accommodate for specific needs regarding either high precision or high recall, the parameters of the postprocessing module can easily be tuned to create a new set of predictions accordingly. Instead of selecting about the same percentage of positives as found in the training data, this number can either be reduced or increased, trading off recall for precision. To illustrate, a precision-recall curve was compiled for the prediction of all events in the development data (Figure 6). The highest F-score (48.62) is achieved at 55.67 precision and 43.15 recall, and corresponds to an exact mapping of percentages between the training and testing set (Table 3). If we only select the top 40% of those original predictions, we achieve a precision of 74.90, recall of 21.24 and F-score of 33.10. The highest possible precision rate is 79.17, but performance then significantly drops to 9.42 F-score.

As benchmarking on the final test set is limited, a similar graph for the final system trained on 950 articles was not produced, but we expect the results to be similar. We did run one additional experiment on the final test data using the 40% factor and achieved 73.77 precision, 17.85 recall and 28.74 F-score. These numbers outperform the system of manually written rules that achieved the highest precision among all participants in the official Shared Task with 71.81 precision, 13.45 recall and 22.66 F-score (Cohen et al. 2009). These results show that a ML framework can compete with a rule-based method even when high precision is required.

[3] Google Inc, http://www.google.com/

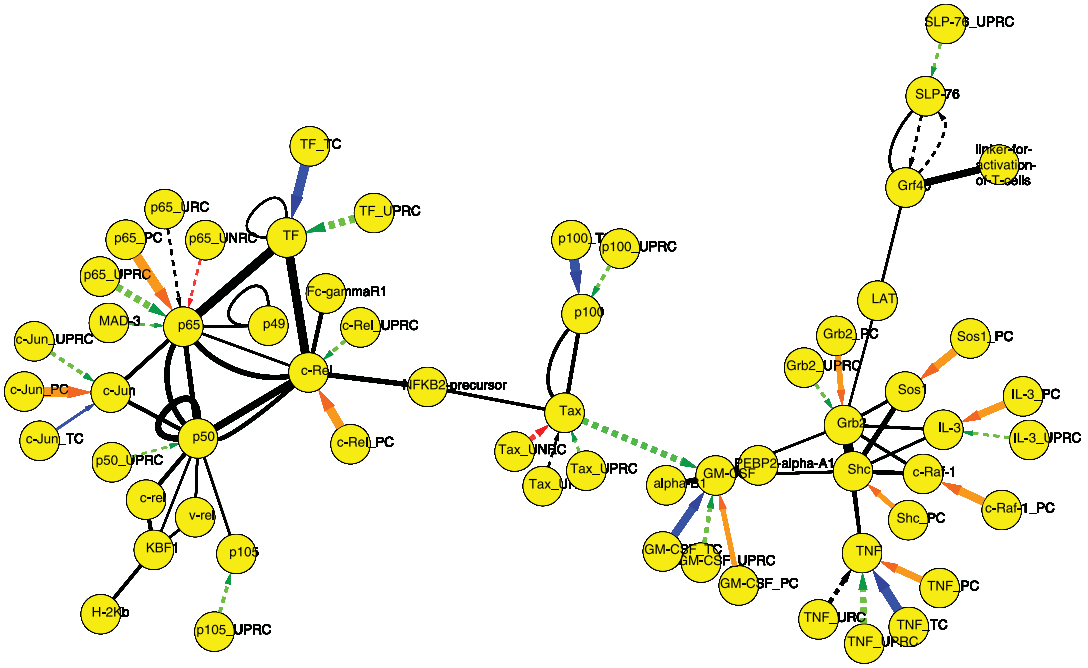

FIGURE 7. Visualization of a sub-graph of the integrated network. Colors display different types of interactions: black for binding and unspecified regulation events, orange for phosphorylation, blue for transcription, and green/red for positive/negative regulation events.

4.4. Knowledge Discovery

As explained in Section 2.9, predictions from the parallel pipelines are merged into an integrated graph. Subsequently, postprocessing techniques are applied to prune the graph and ensure consistency of the predictions. Figure 7 shows a subgraph of the final network. Edge thickness corresponds to the prediction confidence of the interaction as provided by the LibSVM classifiers, and colors display different types of interactions. This graph-based visualization enables intuitive inference of new biological knowledge or hypotheses. The network depicted in Figure 7 shows an example: the positive regulation of *GM-CSF* by *Tax*, which is in turn negatively regulated by *Tax UNRC*, suggests an indirect negative regulation of *GM-CSF* by *Tax UNRC*.

Instead of inferring such hypotheses manually, the data can be converted to a machine-readable data format such as RDF or OWL.[4] Automated reasoning systems could then provide valuable new predictions. These could be either false negatives which were missed by the classifier, or negatives that were not present in the literature and thus represent new hypotheses.

[4] Details of W3C standards such as RDF or OWL can be found at http://www.w3.org/

5. CONCLUSIONS

We have developed a ML system that extracts bio-molecular events from research articles with high precision. In this study we have discussed and benchmarked several important design choices for such a framework, ranging from various text pre-processing methods and parameter optimization to the modularity of the system and size of the training data. Both learning curves as well as precision-recall curves have shown interesting characteristics related to the challenge of event extraction.

Furthermore, we have presented the first application of FS to bio-molecular event extraction from text, obtaining faster and more cost-effective models. Analysis of the top ranked features has shown that while lexical features are important, a huge percentage of them are irrelevant and mainly add noise to the classifier. Syntactic features on the other hand are in general highly useful.

We are confident that insights learned during our experiments can lead to substantial improvements in this field. For example, we have been able to obtain a relative performance gain of 10% compared to the first version of our framework, benchmarked on the BioNLP'09 Shared Task on event extraction. Our system now achieves 37.43% recall, 54.81% precision, and 44.48% F-score.

Twenty-four international teams participated in the official BioNLP'09 Shared Task, resulting in 24 distinct extraction systems. Our analysis of various design choices is not only highly relevant for various ML approaches, but can also offer a meaningful contribution to the development of rule-based systems. The results of our FS experiments are particularly useful for this purpose. For future work, we plan on employing a variety of FS algorithms and analyse the resulting feature ranking lists.

Finally, this paper has attempted to bridge the gap between theoretical relation extraction and real-life applications such as database curation and knowledge discovery. We illustrated the possibility of merging all predictions into an integrated network. This is interesting not only for visualization, but can also be used to infer new biological hypotheses to be tested in the wet labs. This creates the opportunity to discover meaningful relations through the cooperation of fully automated, supervised learning techniques on one hand, and an expert user able to interpret its results on the other hand. We plan on investigating these opportunities in more detail in the future.

ACKNOWLEDGMENTS

SVL and YS would like to thank the Research Foundation Flanders (FWO) for funding their research. This work is supported by IUAP P6/25 (BioMaGNet).

REFERENCES

BJÖRNE, J., J. HEIMONEN, F. GINTER, A. AIROLA, T. PAHIKKALA, and T. SALAKOSKI. 2009. Extracting complex biological events with rich graph-based feature sets. *In* BioNLP '09: Proceedings of the Workshop on BioNLP, Boulder, CO, pp. 10–18.

BUYKO, E., E. FAESSLER, J. WERMTER, and U. HAHN. 2009. Event extraction from trimmed dependency graphs. *In* BioNLP '09: Proceedings of the Workshop on BioNLP, Boulder, CO, pp. 19–27.

COHEN, K. B., K. VERSPOOR, H. L. JOHNSON, C. ROEDER, P. V. OGREN, W. A. BAUMGARTNER, Jr., E. WHITE, H. TIPNEY, and L. HUNTER. 2009. High-precision biological event extraction with a concept recognizer. *In* BioNLP '09: Proceedings of the Workshop on BioNLP, Boulder, CO, pp. 50–58.

DE MARNEFFE, M., B. MACCARTNEY, and C. MANNING. 2006. Generating typed dependency parses from phrase structure parses. *In* Proceedings of LREC-06, Genoa, Italy, pp. 449–454.

FUNDEL, K., R. KÜFFNER, and R. ZIMMER. 2007. Relex–relation extraction using dependency parse trees. Bioinformatics, **23**(3): 365–371.

GIULIANO, C., A. LAVELLI, and L. ROMANO. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. *In* Proceedings of EACL 2006, Trento, Italy, pp. 401–408.

GUYON, I., and A. ELISSEEFF. 2003. An introduction to variable and feature selection. Journal of Machine Learning Research, **3**: 1157–1182.

HALL, M., and L. SMITH. 1998. Practical feature subset selection for machine learning. *In* Proceedings of the 21st Australian Computer Science Conference, Auckland, New Zealand, pp. 181–191.

HIRSCHMAN, L., A. YEH, C. BLASCHKE, and A. VALENCIA. 2005. Overview of biocreative: Critical assessment of information extraction for biology. BMC Bioinformatics, **6**(Suppl 1): S1.

KILICOGLU, H., and S. BERGLER. 2009. Syntactic dependency based heuristics for biological event extraction. *In* BioNLP '09: Proceedings of the Workshop on BioNLP, Boulder, CO, pp. 119–127.

KIM, J.-D., T. OHTA, and J. TSUJII. 2008. Corpus annotation for mining biomedical events from literature. BMC Bioinformatics, **9**(1): 10.

KIM, J.-D., T. OHTA, S. PYYSALO, Y. KANO, and J. TSUJII. 2009. Overview of BioNLP'09 shared task on event extraction. *In* Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task, Boulder, CO, pp. 1–9.

KRALLINGER, M., A. MORGAN, L. SMITH, F. LEITNER, L. TANABE, J. WILBUR, L. HIRSCHMAN, and A. VALENCIA. 2008. Evaluation of text-mining systems for biology: Overview of the second biocreative community challenge. Genome Biology, **9**(Suppl 2): S1.

LEWIS, R. M., and V. TORCZON. 1999. Pattern search algorithms for bound constrained minimization. SIAM J. on Optimization, **9**(4): 1082–1099.

PLAKE, C., J. HAKENBERG, and U. LESER. 2005. Optimizing syntax patterns for discovering protein-protein interactions. *In* SAC '05: Proceedings of the 2005 ACM symposium on Applied Computing, New York, pp. 195–201.

PORTER, M. F.. 1980. An algorithm for suffix stripping. Program, **14**(3): 130–137.

SAETRE, R., K. SAGAE, and J. TSUJII. 2008. Syntactic features for protein-protein interaction extraction. *In* Proceedings of the 2nd International Symposium on Languages in Biology and Medicine (LBM), Singapore.

SAEYS, Y., I. INZA, and P. LARRANAGA. 2007. A review of feature selection techniques in bioinformatics. Bioinformatics, **23**(19): 2507–2517.

THE GENE ONTOLOGY CONSORTIUM. 2008. The gene ontology project in 2008. Nucleic Acids Research, **36**(Database issue): D440–D444.

VAN LANDEGHEM, S., Y. SAEYS, B. DE BAETS, and Y. VAN DE PEER. 2008. Extracting protein-protein interactions from text using rich feature vectors and feature selection. *In* Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM), Turku Centre for Computer Sciences (TUCS), Turku, Finland, pp. 77–84.

VAN LANDEGHEM, S., Y. SAEYS, B. DE BAETS, and Y. VAN DE PEER. 2009. Analyzing text in search of bio-molecular events: a high-precision machine learning framework. *In* BioNLP '09: Proceedings of the Workshop on BioNLP, Boulder, CO, pp. 128–136.

WERMTER, J., K. TOMANEK, and U. HAHN. 2009. High-performance gene name normalization with GENO. Bioinformatics, **25**(6): 815–821.