Biomolecular Event Extraction using a Stacked Generalization based Classifier

Amit Majumder
Dept. of MCA
Academy Of Technology
West Bengal, India
cseamit48@yahoo.co.in

Asif Ekbal
Dept. of CSE
IIT Patna, Patna
India
asif@iitp.ac.in

Sudip Kumar Naskar
Dept. of CSE
Jadavpur University
West Bengal, India
sudip.naskar@gmail.com

Abstract

In this paper we propose a stacked generalization (or stacking) model for event extraction in bio-medical text. Event extraction deals with the process of extracting detailed biological phenomenon, which is more challenging compared to the traditional binary relation extraction such as protein-protein interaction. The overall process consists of mainly three steps: event trigger detection, argument extraction by edge detection and finding correct combination of arguments. In stacking, we use Linear Support Vector Classification (Linear SVC), Logistic Regression (LR) and Stochastic Gradient Descent (SGD) as base-level learning algorithms. As meta-level learner we use Linear SVC. In edge detection step, we find out the arguments of triggers detected in trigger detection step using a SVM classifier. To find correct combination of arguments, we use rules generated by studying the properties of bio-molecular event expressions, and form an event expression consisting of event trigger, its class and arguments. The output of trigger detection is fed to edge detection for argument extraction. Experiments on benchmark datasets of BioNLP-2011 show the recall, precision and F-score of 48.96%, 66.46% and 56.38%, respectively. Comparisons with the existing systems show that our proposed model attains state-of-the-art performance.

1 Introduction

Huge amount of electronic bio-medical documents, such as molecular biology reports, genomic papers or patient records are generated daily. These contents need to be organized in a more principled way so as to enable advanced search and efficient information retrieval and information extraction methods. This can be beneficial to the practitioners and researchers in biology, medicine and the other allied disciplines. Success of text mining (TM) is evident from the organization of different shared-task evaluation campaigns. The bulk of research in the field of biomedical natural language processing (BioNLP) have mainly focused on the extraction of simple binary relations. Some of the very popular bio-text mining evaluation challenges include TREC Genomics track (Voorhees, 2007), JNLPBA[1], LLL (Nedellec, 2005) and BioCreative (Lynette Hirschman, 2007). While the first two evaluation challenges were concerned with the issues of information retrieval and named-entity recognition (NER), the last two addressed the issues of information extraction and seeking relations between bio-molecules. Relations among biomedical entities (i.e. proteins and genes) must be extracted automatically from a large collection of biomedical datasets since they are very important in understanding biomedical phenomena. Simple binary relations are not itself sufficient for capturing the detailed phenomenon, and there is a growing demand for capturing more detailed and complex relations. Two large corpora, BioInfer (Pyysalo S, 2007) and GENIA (Tomoko Ohta and Tsujii, 2009), have been proposed for this purpose.

In recent times there has been a trend for fine-grained information extraction from text (Kim J-D, 2009). This was addressed in three consecutive text mining challenges, BioNLP-2009 (Hyoung-Gyu Lee, 2009), BioNLP-2011 (Jin-Dong Kim, 2011) and BioNLP-2013 (Lishuang Li, 2013). In this paper we propose an effective technique for

[1] http://www.geniaproject.org/shared-tasks/bionlp-jnlpba-shared-task-2004

D S Sharma, R Sangal and A K Singh. Proc. of the 13th Intl. Conference on Natural Language Processing, pages 55–64,
Varanasi, India. December 2016. ©2016 NLP Association of India (NLPAI)

information extraction (more specifically, event extraction) at more finer level. This is known as event extraction where the focus is to extract events and their different properties that denote detailed biological phenomenon. This can be thought of as a three-steps process, *viz.* event trigger detection, classification of triggers into predefined categories and argument extraction. The events are classified into 9 potential events, out of which 5 are simple which corresponds to *gene expression*, *transcription*, *protein catabolism*, *phosphorylation* and *localization.* Among the rest four events, one *binding* event and three regulatory or complex events namely *regulation*, *positive regulation* and *negative regulation.* For simple events we have a single primary theme, which is usually a protein. But a complex event can include a theme as well as a cause argument. These themes and causes can be either proteins or events. Moreover, number of themes could also vary. In order to explain this, we consider the following sentence as an example, where TRAF2 and CD40 denote proteins.

Sentence: *In this study we hypothesized that the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain.*

This sentence contains simple, binding and complex events. Each bio-molecular event expression contains a trigger word and one or more arguments. The identified events from the sentence are:

Simple event: The word *phosphorylation* is a trigger of type *Phosphorylation.* Argument of this trigger is *TRAF2* as theme.

Binding event: The word *binding* is a trigger word of *Binding* type. Arguments of this trigger are *TRAF2* and *CD40.*

Complex event: The word *inhibits* is a trigger word of *Negative_regulation* type. The previously mentioned two events(i.e. *Phosphorylation* and *Binding*) are *theme* type arguments of this trigger word.

Here, we propose a stack model for event extraction. In stacking, Linear Support Vector Classification (Linear SVC)[2], Logistic Regression (LR)[3] and Stochastic Gradient Descent (SGD)[4] from Scikit-learn (sklearn)[5] have been used as base-level learning algorithms. Linear SVC is an implementation of Support Vector Machine using liblinear library[6]. For meta learning we use Linear Support Vector Classification. The system is evaluated based on the framework of BioNLP 2011 shared task[7].

Event extraction systems find both triggers and their associated arguments. These could pose a number of challenges. Exact interpretation of triggers depend upon the context, e.g. expression of [gene] is an event of type Gene Expression, but expression of [mRNA] is of type Transcription. So there is ambiguity in sense of trigger words. It is a challenge to fix these ambiguities. Event arguments can be difficult to detect in case of binding and regulatory type of arguments, because in these cases, number of arguments is not fixed. Number of arguments in binding events can be one or more. Finding correct combination of arguments is very challenging. More precisely, we use the annotated data collected for these tasks and report the results returned by the evaluation servers on the test sets of the 2011 GE task. From the experiment it has been seen that evaluation on test data shows 2-3% less in performance than the performance on development data. So it is challenge to increase the performance on test data. Coreference resolution is required for the correct interpretation of certain event arguments. For example, in the sentence M-CSF treatment was also associated with a rapid induction of the jun-B gene, although expression of this gene was prolonged compared to that of c-jun. In this example, the word this gene refers jun-B and the word that refers expression. There are two gene expression events in the sentence. Arguments of these events can be identified correctly if coreference resolution method is applied on the dataset and it is a challenging issue.

2 Few Existing Methods for Combining Classifiers

In our day to day life, when crucial decisions are made in a meeting, a voting among the members present in the meeting is conducted when the opinions of the members conflict with each other.

[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[3] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[5] http://scikit-learn.org/stable/

[6] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

[7] http://2011.bionlp-st.org/

This principle of ”voting is very popular in data mining and machine learning. In voting, when classifiers are combined, the class assigned to a test instance will be the one suggested by most of the base level classifiers involved in the ensemble process. Bagging (Breiman, 1996) and boosting (SCHAPIRE, 1990) are the widely used variants of voting schemes. Bagging is a voting scheme in which n models, usually of same type, are constructed. For an unknown instance, each models predictions are recorded. Finally, that particular class is assigned which has the maximum votes among the predictions from models. Boosting is very similar to bagging in which only the model construction phase differs. Here the instances which are often misclassified are allowed to participate in training more number of times. There will be n classifiers which themselves will have individual weights for their accuracies. Finally, the class is assigned which is having the maximum weight.

The classifiers can be combined using two popular approaches, viz. majority voting and weighted voting. In majority voting, we select the class that receives maximum votes. In weighted voting, classifiers are combined based on the strengths and weaknesses of classifiers.

Stacked generalization or stacking is a method for combining multiple classifiers. The idea of stacked generalization (Wolpert, 1992; Georgios Sigletos, 2005) is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers, estimated as follows:
Define D as a training data set consisting of feature vectors, also referred to as level-0 data, and $L^1 ... L^N$ as a set of N different learning algorithms. During the K-fold cross-validation process, D is randomly split into K disjoint parts $D^1 ... D^K$ of almost equal sizes. At each j-th fold, j =1..K , the $L^1 ... L^N$ learning algorithms are applied to the training part D-D^j (i.e. part of training data excluding D^j, where D is the whole training data and D^j is the current test part for cross-validation) and the induced classifiers C^1(j)...C^N(j) are applied to the test part D^j. The concatenated predictions of the induced classifiers on each feature vector x_i in D^j, together with the original class value $y_i(x_i)$, form a new set MD^j of meta-level vectors.

At the end of the entire cross-validation process, MD= $\bigcup_{j=1}^{K} MD^j$ constitutes the full meta-level data set, also referred to as level-1 data, which is used for applying a learning algorithm L^M and inducing the meta-level classifier C^M . The learning algorithm L^M that is employed at meta-level could be one of the L^1 ...L^N or a different one. Finally, the $L^1 ... L^N$ learning algorithms are applied to the entire data set D inducing the final base-level classifiers C^1 ...C^N to be used at runtime. In order to classify a new instance, the concatenated predictions of all base-level classifiers $C^1 ... C^N$ form a meta-level vector that is assigned a class value by the meta-level classifier C^M.

3 Proposed Approach

In this section we describe our proposed approach for event extraction. The steps to extract event expression are sentence splitting, tokenization, trigger detection, argument extraction by edge detection and finding correct combination of arguments as shown in figure 1. In sentence splitting and tokenization steps we use sentence split and tokenised data which was made available as the supportive resources in BioNLP-2011 shared task[8]. We use SVM to extract arguments and its correct combinations.

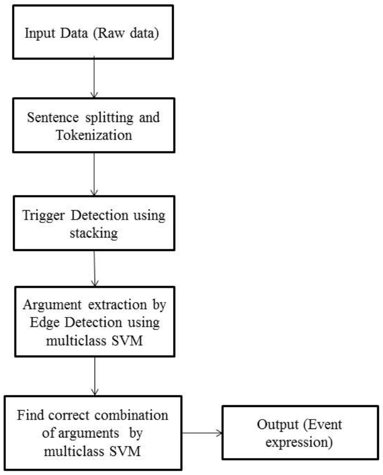
Input Data (Raw data)
Sentence splitting and Tokenization
Trigger Detection using stacking
Argument extraction by Edge Detection using multiclass SVM
Find correct combination of arguments by multiclass SVM
Output (Event expression)

Figure 1: Steps to extract event expression

Our method is based on the principle of stacking. We generate 10 classifiers for each of the algorithms (i.e. Linear SVC, SGD and LR) mentioned above. We describe the algorithms in the subsequent sections.

[8] http://weaver.nlplab.org/~bionlp-st/BioNLP-ST/downloads/support-downloads.html#parse-formats

3.1 Linear Support Vector Classification (Linear SVC)

Linear SVC is learning algorithm for classification. Using Linear SVC algorithm, we generate 10 classifiers by varying C parameter of the algorithm. The C parameter is a parameter for optimization that specifies the learning algorithm how much it wantS to avoid misclassifying each training example. Starting C value (here, 0.001) is chosen by running the algorithm using C values as 1000, 100, 1, 0.1, 0.001, 0.00001, and choose final value based on the highest accuracy obtained.

$$C_i^{SVC} = 0.001 + i \times 0.05 \tag{1}$$

where C_i^{SVC} represents C parameter for SVC_i classifier.

3.2 Stochastic Gradient Descent (SGD)

We generate 10 classifiers by varying the *alpha* parameters of SGD algorithm. The parameter *alpha* is Constant that multiplies the regularization term used in the implementation of the algorithm. These 10 classifiers are SGD_i, for i=0 to 9. The value of *alpha* is set to 0.000029 based on the different experiments executed.

$$alpha_i^{SGD} = 0.000029 + i \times 0.000015 \tag{2}$$

where $alpha_i^{SGD}$ represents *alpha* parameter for SGD_i classifier.

3.3 Logistic Regression (LR)

We generate 10 classifiers by varying the C parameter of LR algorithm. These classifiers are LR_i, for i=0 to 9. The value of C is set to 7.2101 which is chosen by running the algorithm with different C values.

$$C_i^{LR} = 7.2101 + i \times 0.05 \tag{3}$$

where C_i^{LR} represents C parameter for LR_i classifier.

The architecture of our proposed stacked method is shown in Fig 2 where SVC, SGD and LR have been used as base level classifiers. Here, SVC is used as meta-level classifier.

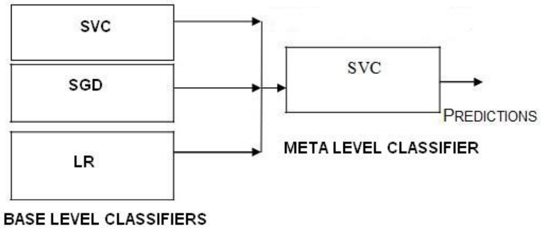

Figure 2: Model ensemble using stacking

4 Features

In this section we describe the features that we use for developing the models. To explain the features, let us consider the following example sentence: *"BMP-6 inhibits growth of mature human B cells; induction of Smad phosphorylation and upregulation of Id1"*. The sentence is tokenised and features are extracted for every token. We use token *"upregulation"* in the above example to explain the features.

4.1 Features for trigger detection and classification

Here, we describe the set of features that we use for event trigger detection and classification.

1. **Surface Word, Stem, PoS, Chunk and Named Entity:** We use surface forms, lemma, Part-of-Speech (PoS), chunk and named entity (NE) as features for trigger detection and classification. These information were extracted from the GENIA tagger[9]. For the token *"upregulation"*, the feature values extracted are *upregulation, upregulate, NN, I-NP and O* for the surface form, lemma, PoS, Chunk and NE features, respectively. All these information are very critical to identify the trigger and its class.

2. **Bag-of-Words:** The bag-of-word (BoW) feature plays a crucial role in many text mining tasks. This particular feature is defined in different ways. At first we extract BoWs within the context of sizes 3 and 5 (i.e., ± 1 and ± 2). We also extract NEs from this context, and use their counts as features. Entire sentence is then considered as a context and BoWs and NE features are extracted. For

[9] http://www.nactem.ac.uk/GENIA/tagger/

example, BoW feature for the token *“upregulation”* are *“and upregulation of”* and *“phosphorylation and upregulation of Id1”* for window sizes 3 and 5, respectively.

3. **Linear Features:** Linear features are generated by marking token with a tag that denotes their relative positions in the linear order. This feature is defined with respect to a context window. If *i* is a position (i.e. index) of the token under consideration, then the linear features are calculated from the words with indices *i-3* to *i+3*. In our experiment we use the word along with its PoS tag to generate linear features.

4. **upper_case_start, upper_case_middle, has_digits, has_hyphen:** These features are defined based on the orthographic constructions: whether the token starts with an uppercase character, or it has any uppercase character in the middle, or has any digit(s) or hyphen inside it.. These features are important from the observations that there are some trigger words in the dataset which start with uppercase character or hyphen inside it. For example, in the sentence *TGF-beta mediates RUNX induction and FOXP3 is efficiently up-regulated by RUNX1 and RUNX3 in human CD4+ T cells.*, the word *up-regulated* is *Positive_regulation* type rvent trigger which has hyphen inside it.

5. **Bi-gram and Tri-gram Features:** We use the character bi-gram and tri-gram sequences extracted from a token as features. For example, for the token *“upregulation”*, the bi-gram features will be *up pr re eg gu ul la at ti io on* and tri-gram features will be *upr pre reg egu gul ula lat ati tio ion .*

6. **Dependency Path Features:** There are are some trigger words which ca not be detected using context features or b-gram or tri-gram features. So we depend on dependency relations inside sentence. Dependency features are extracted from dependency graph generated by dependency parser(David McClosky and Manning, 2011; David McClosky and Johnson, 2006) . Figure 3 shows the dependency graph for the sentence *“BMP-6 inhibits growth of mature human B cells; induction of Smad phosphorylation and upregulation of Id1”*, generated by the Charniak-McCloskey parser (David McClosky and Johnson, 2006). In the graph, an edge label represents the dependency relation between two nodes. Each node in the graph is labelled by a number which represents a word appearing in that position (0-based index) of the sentence. For example, node labelled with number 0 indicates the word *BMP-6* and node labelled with number 1 indicates the word *inhibits*.

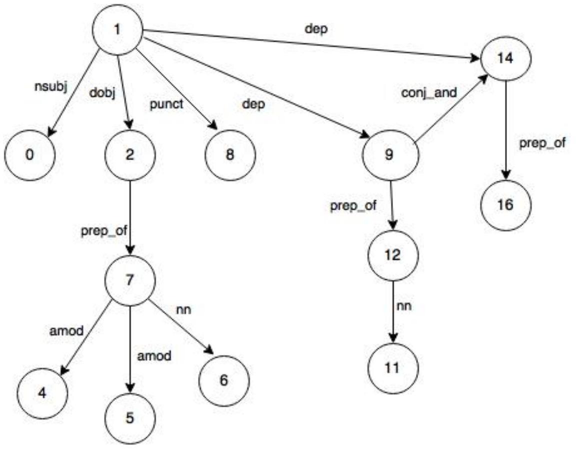

Figure 3: Dependency graph for the example sentence *“BMP-6 inhibits growth of mature human B cells; induction of Smad phosphorylation and upregulation of Id1”*

.

In the graph, node 0 and 16 represent proteins (i.e. NE) as specified in the training dataset. In the feature value NE is denoted by _NAMED_ENT.

Edges in a dependency graph are directed arcs. Each edge connects two nodes. Nodes represent words along with other information like PoS tags of the words. A node can be connected to two types of edges: one is in-type (or incoming) edges which are incident on the node and the other type is out-type (or outgoing) edges which emanate from the node.

Features for in-type edges:
For the in-type edges we consider the features as defined below. For illustration purpose, we consider node numbered 14 (*upregulation*) as the target node and we present below the feature values generated for the in-type edge emanating from node 1 (*inhibits*) and incident on

node numbered 14.

(a) Edge type (i.e. dependency relation) – *dep*
(b) PoS of source node – *VBZ*
(c) Edge type combined with PoS – *dep_VBZ*
(d) Text of the source node – *inhibits*
(e) Edge type merged with PoS and token of source node – *dep_VBZ_inhibits*
(f) Stem of the source node – *inhibit*
(g) Edge type combined with stem of the source node – *dep_inhibit*
(h) Stem of the current word combined with edge-type and stem of the source node – *upregul_dep_inhibit*

Features for out-type edges:
For out-type edges emanating from the target word, in addition to the target word feature we also take into consideration the features belonging to the edge (i.e. the dependency relation) and the destination node on which the edge is incident.

The list of features considered for out-type edges are listed below. For illustration, we consider node 14 (*upregulation*) as the target node and the corresponding feature values are shown next to the out-type edge features for the out-type edge emanating from node 14 (*upregulation*) and incident on node 16 (*Id1*).

(a) Edge type – prep_of
(b) PoS of destination node – NN
(c) Edge-type combined with PoS of destination node – prep_of_NN
(d) Text of the destination node – NAMED_ENT
(e) Edge-type combined with the token text of the destination node – prep_of_NAMED_ENT
(f) Stem of the current word combined with the edge-type and stem of the destination node – upregul_prep_of_NAMED_ENT

7. **Dependency chain:** These features are syntactic dependencies up to a certain depth limit, starting from a token of interest. In our case we consider the depth of limit three. They are used to define the immediate context of these words.

4.2 Features for argument extraction by edge detection

To find out features for argument extraction, for every sentence, we form a dependency graph consisting of the triggers detected in trigger detection step and proteins mentioned in the data set for that sentence. Following features are extracted for argument extraction.

1. **Token features:** These are features of two tokens connected by an edge. A token can potentially be a protein (i.e NE) or a trigger word detected during the trigger detection phase. The token features are surface word, stem, PoS, chunk and NE, BoW, prefix and suffix, linear features, bi-gram and tri-gram features. These have been discussed in details in Section 4.1.

2. **Dependency Features:** Dependency features play important role to extract arguments of triggers. The following dependency features are extracted from dependency graph.

   Single element feature: A path in dependency graph has a starting token, ending token and some intermediate tokens. Token can be considered as a node in the graph. A path consists of a sequence of edges starting from the initial node, followed by a set of intermediate nodes, and ending in a terminal node.

   (a) For each pair of adjacent tokens, all the dependency relations are considered.
   (b) Lexical features of all the internal tokens in the path are also considered. These are surface word, stems, PoS, chunk, NE, BoW, prefix, suffix, linear features, bi-gram and tri-gram features.
   (c) Dependency features that we use are mentioned in Section 4.1.

   N-gram feature: We compute the shortest path from starting to the end token in the dependency graph in figure-3. From this walk we compute bi-grams from all the combinations of two consecutive edges. In the same way, we compute n-grams (n=3,4) by considering three and four consecutive edges, respectively.

   Path edge feature: For each edge in the path, we use the edge features. Edge feature is defined as consisting of all the lexical-level features of the nodes connected by an edge.

5 Experimental Results and Analysis

We perform experiments on BioNLP-11 Genia event dataset[10]. Statistics of BioNLP-11 dataset for genia event extraction has been mentioned in table 1.

Attributes	Training	Development	Test
Abstracts+Full articles	908 (5)	259 (5)	347 (5)
Sentences	8,759	2,954	3,437
Proteins	11,625	4,690	5,301
Total events	10,287	3,243	4,457

Table 1: Statistics of BioNLP-ST 2011 Genia Event dataset (training, development and test). Value inside parentheses indicates the number of full articles

The overall algorithm comprises of three basic steps: trigger detection, edge detection and argument extraction. Trigger detection is performed using the stacked generalization method. Experimental results are shown in Table 2. From the experimental results of stacking mentioned in table 2, it is evident that performances of LR and SGD algorithms are better than Linear SVC. SGD classifier is an implementation of SVM with stochastic gradient descent (SGD) learning, whereas Linear SVC is an implementation of SVM using liblinear library. Linear SVC shows low performance, but it is very fast. We recorded the results for each class label (though, not shown in table) and we see that linear SVC generates best result for *Transcription* and *Regulation* type triggers. LR classifier shows best result, but it takes more time than the other two algorithms. This happens due to the fact that each of these classifiers have their own default parameter settings in sklearn tool. We have tuned *C* parameter in SGD and Linear SVC. In SGD classifier, we tune *alpha* parameter. Among all the base-level classifiers, classifier LR7 using Logistic Regression algorithm provides best result (F-score=71.62). Results show that stacked model achieves better performance compared to the best base-level classifier. Output of trigger detection is fed to the input of edge detection step. In this step we use multi-class SVM[11] as a classification algorithm to find out *theme* and/or *cause* relationships between triggers and proteins. After finding the relationships we generate the event expression. In Table 3 and Table 4, we show the results of experiments on BioNLP-11 shared task dataset on development and test set, respectively. We achieve satisfactory F-score of 80.04% on development dataset and 78.15% on test dataset for *Gene_expression* type event. System also performs well for *protein_catebolism* event with an F-score of 91.30% on the development set. Results of *phosphorylation* event is also satisfactory for the test and development datasets (around 84% F-score). The system suffers most for the relatively complex regulatory events where it shows around 42% to 51% F-score.

Base/Meta classifier	Classifiers	Recall	Precision	F-score
Base level classifiers	SVC0	48.22	84.81	61.48
	SVC1	74.42	67.58	70.84
	SVC2	65.82	76.61	70.81
	SVC3	71.46	70.42	70.94
	SVC4	66.07	75.49	70.46
	SVC5	70.25	71.20	70.72
	SVC6	66.13	74.92	70.25
	SVC7	69.57	71.81	70.67
	SVC8	66.04	74.54	70.03
	SVC9	68.89	72.08	70.45
	LR0	64.69	76.08	69.92
	LR1	64.95	76.28	70.16
	LR2	65.22	76.37	70.36
	LR3	65.91	76.46	70.80
	LR4	65.72	77.33	71.05
	LR5	65.91	77.43	71.21
	LR6	65.16	78.67	71.28
	LR7	66.13	78.10	71.62
	LR8	65.47	78.57	71.43
	LR9	64.97	79.66	71.57
	SGD0	71.43	71.35	71.39
	SGD1	71.38	71.34	71.36
	SGD2	71.43	71.33	71.38
	SGD3	71.34	71.36	71.35
	SGD4	71.39	71.38	71.38
	SGD5	71.42	71.39	71.41
	SGD6	71.38	71.38	71.38
	SGD7	71.41	71.36	71.38
	SGD8	71.38	71.42	71.40
	SGD9	71.38	71.37	71.37
Meta level classifier	SVC	69.13	74.96	71.93

Table 2: Stacked generalization result in trigger detection

Event Class	**gold (match)**	**answer (match)**	**recall**	**precision**	**fscore**
Gene_expression	749 (582)	704 (581)	77.70	82.53	80.04
Transcription	158 (79)	93 (79)	50.00	84.95	62.95
Protein_catabolism	23 (21)	23 (21)	91.30	91.30	91.30
Phosphorylation	111 (94)	111 (94)	84.68	84.68	84.68
Localization	67 (49)	58 (49)	73.13	84.48	78.40
=[SVT-TOTAL]=	1108 (825)	989 (824)	74.46	83.32	78.64
Binding	373 (171)	311 (171)	45.84	54.98	50.00
==[EVT-TOTAL]==	1481 (996)	1300 (995)	67.25	76.54	71.60
Regulation	292 (104)	182 (104)	35.62	57.14	43.88
Positive_regulation	999 (376)	667 (376)	37.64	56.37	45.14
Negative_regulation	471 (168)	264 (168)	35.67	63.64	45.71
==[REG-TOTAL]==	1762 (648)	1113 (648)	36.78	58.22	45.08
==[ALL-TOTAL]==	3243 (1644)	2413 (1643)	50.69	68.09	58.12

Table 3: Result on development data [Approximate Span/Approximate Recursive]

[10] http://2011.bionlp-st.org/

[11] https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

Event Class	gold (match)	answer (match)	recall	precision	fscore
Gene_expression	1002 (728)	861 (728)	72.65	84.55	78.15
Transcription	174 (87)	123 (87)	50.00	70.73	58.59
Protein_catabolism	15 (7)	9 (7)	46.67	77.78	58.33
Phosphorylation	185 (154)	184 (154)	83.24	83.70	83.47
Localization	191 (99)	116 (99)	51.83	85.34	64.50
=[SVT-TOTAL]=	1567 (1075)	1293 (1075)	68.60	83.14	75.17
Binding	491 (234)	408 (234)	47.66	57.35	52.06
==[EVT-TOTAL]==	2058 (1309)	1701 (1309)	63.61	76.95	69.65
Regulation	385 (120)	221 (120)	31.17	54.30	39.60
Positive_regulation	1443 (549)	980 (549)	38.05	56.02	45.32
Negative_regulation	571 (204)	381 (204)	35.73	53.54	42.86
==[REG-TOTAL]==	2399 (873)	1582 (873)	36.39	55.18	43.86
==[ALL-TOTAL]==	4457 (2182)	3283 (2182)	48.96	66.46	56.38

Table 4: Results on test data [Approximate Span/Approximate Recursive]

5.1 Effectiveness of features

Finding importance of individual feature (e.g. root words, n-gram features, linear features etc.) used in our experiment is not an easy task. Transformation of textual features into numeric features generates a lot of features. For example, in trigger detection step we use 38 features, but when these are converted to numerical data for machine learning purpose, number of features increases to 5 lakhs. Finding the most relevant set of features from this collection is a complex problem. We keep a record of how the original 38 features (represented in text format) are mapped in higher dimensional space containing more than 5 lakh of features. For example, if we use 0-based index for feature, then in trigger detection 0-th feature (from the 38 original features) is mapped to feature indices in the range of 0 to 11,036 and 1-st feature (from 38 original features) is mapped to feature indices in the range of 11,037 to 20,080 in higher dimensional space and so on. Using Linear SVC classifier we select some of the top features which are mapped to the original features. We observe that the features for outgoing edges from the dependency graph are most important for *Gene_expression, Transcription, Localization, Phosphorylation and Binding* type event triggers. For *Positive_regulation,Negative_regulation and Regulation* type event triggers, the most important feature is the dependency chain features.

5.2 Comparison with existing systems

For bio-molecular event extraction, the state-of-the-art system is TEES (Björne, 2014), which ranked first place in BioNLP-ST-2009[12]. Along with this we also compare our proposed system with the other existing systems, which participated in BioNLP-2011 shared task (Jin-Dong Kim, 2011) . Our experimental results show recall, precision and F-score values of 50.69%, 68.09% and 58.12%, respectively on development dataset, whereas official result attained by TEES (Björne, 2014) is 52.45%, 60.05% and 55.99% respectively. As compared to the official scores of TEES on the test set (recall: 49.56%, precision: 57.65% and F-score: 53.30%), our system achieves recall, precision and F-score values of 48.96%, 66.46% and 56.38%, respectively. Hence, our system performs better with more than 3 points. While we compare our proposed model with the systems presented in BioNLP-2011 (Jin-Dong Kim, 2011), it shows that we achieve performance very close to the best performing system, FAUST (Sebastian Riedela and Manning, 2011)(recall:49.41%, precision: 64.75% and F-score:56.04%) and better than the second ranked system, UMass, (McCallum, 2011) (recall:48.49%, precision:64.08% and F-score:55.20%). A recently developed system named as EventMine (Makoto Miwa and Ananiadou, 2013), which made use of coreference resolution obtains significant improvement with recall, precision and F-score of 51.25%, 64.92% and 57.28%, respectively. The performance in our model is very close to system, EventMine (Makoto Miwa and Ananiadou, 2013).

5.3 Error Analysis

In order to gain more insights we analyse the outputs to find the errors and their possible causes. We perform quantitative analysis in terms of confusion matrix, and qualitative analysis by looking at the outputs produced by the systems. For trigger detection and classification we observe that the system performs satisfactorily well for gene_expression and phosphorylation types. However, the classifier does not show convincing results for regulation type events which are, in general, difficult to identify and classify. One of the reasons may be the less number of training instances of regulatory events. Classifier finds it difficult to disambiguate the cases when any particular instance belongs to more than one type. For example, token *transfection* originally belongs to both the types, Gene_expression and Positive_regulation, but our system is unable to detect it even as a trigger word. On the other hand, the word *Overexpression* is originally a non-trigger word, but our system detects *Overexpression* as

[12] `http://www.nactem.ac.uk/tsujii/GENIA/SharedTask/`

trigger word of types Gene_expression and Positive_regulation both.

In argument extraction step, arguments of the triggers detected in trigger detection and classification step are identified. Relation between a trigger word and its argument is also found out in this step. Possible arguments are proteins and/or event trigger words. Possible relations are *theme* and *cause*. From a closer analysis we see that our system performs satisfactorily in detecting *theme* argument, but for detecting *cause* argument classifier is not very robust. This may be due to the fact that a cause expression could be both a protein (or, NE) or an event trigger expression. The system suffers most for the regulatory events as the errors might have propagated from the earlier step, i.e. trigger detection and classification step. For example trigger word *phosphorylation* in one example sentence is originally a *theme* argument of a regulatory event, but our system is unable to detect the trigger word *phosphorylation* as an argument of that regulatory event.

6 Conclusion and Future Works

In this paper we propose a stacking approach for event extraction. The idea of stacking is to perform cross-validation on the training data set using some learning algorithms in order to create a meta-level data set. Meta-level dataset is formed using predictions generated by the learning algorithms along with the actual output class. In edge detection step, we find out arguments of the triggers detected in trigger detection step using SVM algorithm. To find correct combination of arguments we use rules of bio-molecular events and form an event expression consisting of event trigger, its class and arguments. Experiments on BioNLP-2011 datasets show the efficacy of our proposed model with significant performance improvement over the state-of-the-art systems. This improvement is due to application of stacking aproach along with efficient features. In future we would like to study whether coreference resolution could improve the performance of the system.

References

Jari Björne. 2014. *Biomedical Event Extraction with Machine Learning*. Ph.D. thesis, University of Turku.

L Breiman. 1996. Bagging predictors. In *Machine Learning*, pages 123–140.

Eugene Charniak David McClosky and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 152–159. New York.

Mihai Surdeanu David McClosky and Christopher D. Manning. 2011. Event extraction as dependency parsing for bionlp 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 41–45. June.

Constantine D. Spyropoulos Georgios Sigletos. 2005. Combining information extraction systems using voting and stacked generalization. In *Journal of Machine Learning Research*, volume 6, pages 1751–1782.

Min-Jeong Kim Joo-Young Lee Gumwon Hong Hae-Chang Rim Hyoung-Gyu Lee, Han-Cheol Cho. 2009. A multi-phase approach to biomedical event extraction. in bionlp 09. In *Proceedings of the Workshop on BioNLP*, pages 107–110.

Tomoko Ohta-Robert Bossy Ngan Nguyen Junichi Tsujii Jin-Dong Kim, Sampo Pyysalo. 2011. Overview of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 1–6. June.

Pyysalo S-Kano Y Tsujii J Kim J-D, Ohta T. 2009. Overview of bionlp09 shared task on event extraction. In *BioNLP 09: Proceedings of the Workshop on BioNLP*, pages 1–9.

Degen Huang Lishuang Li, Yiwen Wang. 2013. Improving feature-based biomedical event extraction system by integrating argument information. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 109–115. August.

Alfonso Valencia Lynette Hirschman, Martin Krallinger. 2007. Proceedings of the second biocreative challenge evaluation workshop. In *CNIO Centro Nacional de Investigaciones Oncologicas*.

Tomoko Ohta Makoto Miwa, Sampo Pyysalo and Sophia Ananiadou. 2013. Wide coverage biomedical event extraction using multiple partially overlapping corpora. In *BMC Bioinformatics*.

Sebastian Riedel Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 46–50.

Claire Nedellec. 2005. Learning language in logic - genic interaction extraction challenge. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, pages 31–37.

Heimonen J Bjorne J Boberg J Jarvinen J Salakoski T Pyysalo S, Ginter F. 2007. A corpus for information extraction in the biomedical domain. In *BMC Bioinformatics*, pages 8–50.

ROBERT E. SCHAPIRE. 1990. The strength of weak learnability. In *Machine Learning*, pages 197–227.

Mihai Surdeanub Andrew McCalluma Sebastian Riedela, David McCloskyb and Christopher D. Manning. 2011. Model combination for event extraction in bionlp 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 51–55.

Sampo Pyysalo Tomoko Ohta, Jin-Dong Kim and Junichi Tsujii. 2009. Incorporating genetag-style annotation to genia corpus. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP)*, pages 1–9.

Ellen Voorhees. 2007. Overview of trec 2007. In *Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, pages 1–16. Gaithersburg,Maryland,USA.

D. Wolpert. 1992. Stacked generalization. In *Neural Networks*, pages 241–260.