Fast Recursive Multi-class Classification of Pairs of Text Entities for Biomedical Event Extraction

Xiao Liu, Antoine Bordes, Yves Grandvalet
Université de Technologie de Compiègne & CNRS
Heudiasyc UMR 7253
60200 Compiègne Cedex, France
firstname.lastname@hds.utc.fr

Abstract

Biomedical event extraction from articles has become a popular research topic driven by important applications, such as the automatic update of dedicated knowledge base. Most existing approaches are either pipeline models of specific classifiers, usually subject to cascading errors, or joint structured models, more efficient but also more costly and more involved to train. We propose here a system based on a pairwise model that transforms event extraction into a simple multi-class problem of classifying pairs of text entities. Such pairs are recursively provided to the classifier, so as to extract events involving other events as arguments. Our model is more direct than the usual pipeline approaches, and speeds up inference compared to joint models. We report here the best results reported so far on the BioNLP 2011 and 2013 Genia tasks.

1 Introduction

Huge amounts of biomedical documents, such as molecular biology reports or genomic papers are generated daily. Automatically organizing their content in dedicated databases enables advanced search and eases information retrieval for researchers in biology, medicine or other related fields. Nowadays, these data sources are mostly in the form of unstructured free text, which is complex to incorporate into databases. Hence, many text-mining research initiatives are organized around the issue of automatically extracting information from biomedical text. Efforts specifically dedicated to biomedical text are necessary because standard Natural Language Processing tools cannot be readily applied to extract biomedical events since such texts, articles or reports involve highly domain-specific jargon, syntax and dependencies (Kim et al., 2011a).

This paper tackles the problem of event extraction from biomedical documents. Building on previous advances in named entity recognition (for detecting gene or protein mentions for instance), this task consists in associating to these entities the related events expressed in the text. Such events are of multiple types and involve at least one text entity as *argument* and another one as *trigger*; they can be quite complex since some events have several arguments, and recursive in the sense that arguments can themselves be events. An example of event is given in Figure 1.

Biomedical event extraction is attracting more and more attention, especially thanks to the organization of recurrent dedicated BioNLP challenges (Kim et al., 2009; Kim et al., 2011b; Kim et al., 2013). We propose here a new approach which relies on a single multi-class classifier for recursively detecting events from (*trigger*, *argument*) pairs. Compared to standard pipeline approaches based on sequences of classifiers (Björne and Salakoski, 2013; Hakala et al., 2013), we avoid the intermediate problem of associating isolated triggers to event types, relying on a tricky multi-label classification problem. Instead, we directly extract compounds of events in the form of (*trigger*, *argument*) pairs, simply relying on a multi-class problem, whereby (*trigger*, *argument*) pairs are associated to event types. Considering pairs of words also allows us to characterize examples by sophisticated joint features such as shortest path in the dependency parse tree, and hence to achieve much accurate trigger detection than pipeline models. Besides, compared to Markov random fields (Riedel and McCallum, 2011a), our discriminant model does not represent the full joint distribution of words and events. We thus have a simpler inference process, which results into drastically reduced training times (15

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 692–701,
Gothenburg, Sweden, April 26-30 2014.

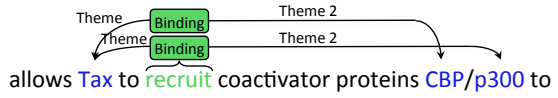

Figure 1: Part of a sentence and corresponding extracted events for the BioNLP 2013 Genia task.

times faster for processing about 800 training documents). In short, we propose in this work a *happy medium* between pipeline and joint models. Our approach builds on our previous proposal (Liu et al., 2013), where we detected triggers directly from (*trigger*, *argument*) pairs. Here, we upgrade our scheme by adding a recursive classification process that considerably improves the detection of complex events.

This paper is organized as follows: Section 2 introduces the problem of biomedical event extraction and discusses related works. Section 3 describes our recursive model and its training process. The post-processing procedures and the features used are detailed in Sections 4 and 5. Section 6 shows that our method achieves excellent empirical results, with the best performance reported so far on the BioNLP 2011 and 2013 Genia tasks, and a reduced training duration compared to the previously state-of-the-art models.

2 Context and Related Works

Biomedical event extraction aims at extracting *event formulas* from sentences, defined as sequences of *tokens* (words, numbers, or symbols).

2.1 Task Definition

Terminology regarding biomedical events, triggers, etc. varies from one task or data set to another; in the following, we use the definitions used by the Genia (GE) task 1 of the BioNLP challenges. An event is constituted of two kinds of elements: an event *trigger* and one or several *arguments*. The event trigger is an *entity*, that is, a sequence of consecutive tokens which indicates that an event is mentioned in the text. The arguments of an event are participants, which can be proteins, genes or other biomedical events.

In the data settings of the GE task, protein mentions are already annotated in the text. Figure 1 illustrates biomedical event extraction in the GE task framework: given 3 proteins "*Tax*", "*CBP*" and "*p300*", one must detect "*recruit*" as an event trigger for two events of the *Binding* category , encoded by the formulas: ("*recruit*", theme:"*Tax*", theme_2:"*CBP*") and ("*recruit*", theme:"*Tax*", theme_2:"*p300*").

Class	*Type*	*Principal arg*	*Optional arg*
S V T	Gene_expression	theme (P)	
	Transcription	theme (P)	
	Protein_catabolism	theme (P)	
	Phosphorylation	theme (P)	
	Localization	theme (P)	
B I N	Binding	theme (P)	theme_2 (P)
R E G	Regulation	theme (P/E)	cause (P/E)
	Positive_regulation	theme (P/E)	cause (P/E)
	Negative_regulation	theme (P/E)	cause (P/E)

Table 1: Classes and types of events with their arguments (P stands for *Protein*, E for *Event*).

A key part of the task is to detect the trigger entities among the candidates sequences of tokens. The BioNLP GE task considers 9 types of events.[1] Table 1 lists these events. The 9 event types may be merged into three broader categories: the first 5 (termed SVT) have a single *theme* argument; the *Binding* event (or BIN) can accept up to two theme arguments; the last 3 types (termed REG) also accept up to two arguments, a theme and an optional *cause*. REG events can be recursive because their arguments can be proteins or events.

2.2 Related Works

Current approaches fall into two main categories: pipeline incremental models and global joint methods. Pipeline approaches (Sætre et al., 2009; Cohen et al., 2009; Quirk et al., 2011) are the simplest way to tackle the problem of event extraction. A sequence of specific classifiers are ran on the text to successively (P_1) detect event triggers, (P_2) assign them arguments, (P_3) detect event triggers whose arguments can be events, and (P_4) assign them arguments (steps (P_3) and (P_4) can be ran multiple times). Such systems are relatively easy to set up and experienced many successes: the TEES system (Björne et al., 2009; Bj[Pleaseinsertintopreamble]rne et al., 2012; Björne and Salakoski, 2013) won the BioNLP GE task in 2009 and ranked 2^{nd} in 2013, whereas the EVEX system won in 2013 (Van Landeghem et al., 2011; Hakala et al., 2013). However, all these methods suffer from error cascading. Besides, prediction must be formalized as

[1]The BioNLP 2013 challenge considered 13 types of events, but we only dealt with the 9 types defined in the previous challenges, because there was not enough data on the newly defined types for proper training or model selection.

a multi-label classification problem because some words can participate in the definition of several events of different types. Detecting triggers in isolation of their arguments in steps (P_1) and (P_3) are ill-posed intermediate problems, since the notion of trigger is intrinsically tied to its argument. The latter brings contextual information that is indisputably relevant for detection. Besides, rich features coding for (trigger, argument) pairs (Miwa et al., 2010) are only used by pipeline models for assigning arguments, whereas they could be useful for trigger detection as well.

Global joint approaches (Riedel et al., 2009; McClosky et al., 2011) aim at solving the event extraction task at once, so as to resolve the drawbacks of pipeline models. In (McClosky et al., 2011), event annotations are converted into pseudo-syntactic representations and the task is solved as a syntactic extraction problem by traditional parsing methods. In (Riedel et al., 2009; Riedel and McCallum, 2011a; Riedel et al., 2011; Riedel and McCallum, 2011b), some models are proposed based on the maximization of a global score taking into account the annotations of nodes and edges in a graph representing each sentence. This maximization problem is formalized as an integer linear program with consistency constraints, and solved via dual decomposition. Such joint models perform very well (winner of the BioNLP 2011 GE task), but suffer from consequential computing costs, as all possible combinations of words are considered as potential events. In the following, we show that our model is able to reach better accuracies than joint models while being computationally much cheaper.

A method based on the search-based structured prediction paradigm (Vlachos and Craven, 2012) has been proposed as an intermediate step between joint and pipeline approaches, by turning the structured prediction problem into a sequence of multi-class classification tasks. Our experiments demonstrate that, despite being conceptually simpler, our recursive pairwise model outperforms it.

3 Recursive Pairwise Model

In this section, we present our recursive pairwise model. It directly extracts pairwise interactions between entities, thereby contrasting with the usual pipeline approaches, which require detecting triggers as an intermediate problem. Our approach proceeds in two steps:

1. **Main (trigger, theme) pair extraction**: main event extraction step that detects the triggers with one of their arguments;

2. **Post-processing**: step that adds extra-arguments to BIN and REG events.

Step 1 is the main innovative part of our system, and is detailed in the remainder of this section. Step 2, which relies on more established techniques, is described in Section 4.

3.1 Direct Extraction of Simple Events

We process entities differently depending on whether they are marked as proteins in the annotation or not; the latter are termed *candidate* entities. We denote $\mathcal{C}_S = \{c_i\}_i$ the set of candidate entities, which is built from the sentence tokens (see Section 5 for details on its construction), $\mathcal{A}_S = \{a_j\}_j$ the set of candidate arguments (that is, the proteins identified by a named-entity recognizer beforehand) in a given sentence S, and the set of event types (augmented by *None*) is $\mathcal{Y}$.

The first steps of a pipeline model consist in predicting whether candidate entities $c_i \in \mathcal{C}_S$ are triggers or not and then, whether arguments $a_j \in \mathcal{A}_S$ can participate to a subset of events from $\mathcal{Y}$. Instead, our pairwise model directly addresses the problem of classifying the (candidate, argument) pairs $p_{ij} = (c_i, a_j)$ as events of type from $\mathcal{Y}$.

This classification is based on Support Vector Machines (SVMs), where the multi-class problem is broken down in a series of one-vs-rest binary problems, one for each event type. The final decision associated to each pair p_{ij} is simply taken as the event (including *None*) whose score is maximal. Classifying a pair p_{ij} as not-*None* jointly detects the event trigger c_i and its argument a_j.

3.2 Recursive Extraction of Complex Events

For simple SVT and BIN events, the set $\mathcal{A}_S$ of possible arguments is restricted to proteins, but the events of class REG may have other events as arguments, thus $\mathcal{A}_S$ has to be enriched. Considering all possible events would be intractable, so that the set of possible arguments is updated dynamically in the process of extracting events. As here possible arguments are exclusive to event types, in practice it is simpler to update the set of pairs that remain to be assessed.

Assume that an event has been actually predicted, that is, that $p_{\alpha\beta} = (c_\alpha, a_\beta)$ has been clas-

Algorithm 1 Recursive pairwise event extraction

input sentence S, candidate entities $\mathcal{C}_S = \{c_i\}_i$ and labeled proteins $\mathcal{A}_S = \{a_j\}_j$
initialize candidate pairs $\mathcal{P}_S = \{(c_i, a_j),\ c_i \in \mathcal{C}_S,\ a_j \in \mathcal{A}_S\}$
initialize extracted events $\mathcal{E}_S = \emptyset$
score and label the pairs in $\mathcal{P}_S$
while $\mathcal{P}_S \neq \emptyset$ **do**
  select the pair $p_{\alpha\beta} \in \mathcal{P}_S$ with highest score
  update $\mathcal{P}_S \leftarrow \mathcal{P}_S - \{p_{\alpha\beta}\}$
  if $\hat{y}_{\alpha\beta} \neq None$ **then**
    create event $\hat{e}_{\alpha\beta} = (c_\alpha, a_\beta, \hat{y}_{\alpha\beta})$
    update $\mathcal{E}_S \leftarrow \mathcal{E}_S \cup \{\hat{e}_{\alpha\beta}\}$
    update $\mathcal{P}_S \leftarrow \mathcal{P}_S \cup \{(c_i, \hat{e}_{\alpha\beta}) | c_i \in \mathcal{C}_S\}$
    censor pairs $\mathcal{P}_S$ to avoid cycles
    score and label the new $\{(c_i, \hat{e}_{\alpha\beta})\}$ pairs
  end if
end while
return extracted events $\mathcal{E}_S$

sified as $\hat{y}_{\alpha\beta} \neq None$; the predicted event is denoted $\hat{e}_{\alpha\beta} = (c_\alpha, a_\beta, \hat{y}_{\alpha\beta})$. We create all pairs with it as argument, $\{(c_i, \hat{e}_{\alpha\beta}) | c_i \in \mathcal{C}_S\}$, and add them to $\mathcal{P}_S$, so as to allow for the detection of recursive events. We assume that recursive events constitute a directed acyclic graph, where the ancestor of a candidate entity cannot be used as its argument. The dynamic updating process is thus constrained to prevent the creation of cycles.

Algorithm 1 summaries our event extraction algorithm. For all events with a single argument, predicting $\hat{y}$ variables directly responds to the event extraction problem. When appropriate, additional optional arguments are added after all pairwise events have been extracted, by the post-processing described in Section 4.

3.3 Fitting the Pairwise Model

The prediction process described above relies on a multi-class classifier. We stress again that, since pairs are assigned to a single class, there is no need to address the more difficult multi-label problem encountered in standard pipeline approaches. An entity may still be assigned to several events, possibly of different types, through the allocation of labels to several pairs comprising this entity.

Training SVMs For each event type, a series of binary linear SVMs is fitted to the available training data, using the implementation from `scikit-learn.org`. As events are rare, each binary classification problem is highly unbalanced. We thus use different losses for positive and negative examples (Morik et al., 1999; Veropoulos et al., 1999), resulting in two hyper-parameters that are set by cross-validation, so as to maximize the F-score of the corresponding event type taken in isolation.

For the SVT and BIN events, the training sets are all composed of the possible (candidate, argument) pairs $\mathcal{P}_S = \{p_{ij} = (c_i, a_j) | c_i \in \mathcal{C}_S,\ a_j \in \mathcal{A}_S\}$ readily extracted from all training sentences, and they only differ in the definition of the positive and negative class, according to the true label associated to each pair.

Creating the training sets for REG events is more complicated: since they can take events as arguments, new pairs are added to $\mathcal{P}_S$ by considering all the events already detected, as sketched in Algorithm 1. Hence, the sets of training examples are not deterministically known before training, but depend on predictions of all other classifiers. Training directly on them requires to use either online algorithms or complex search-based structured prediction procedures as in (Vlachos and Craven, 2012). In this paper, we prefer to use instead the true labels $y_{\alpha\beta}$ during the training phase of REG and *None* classifiers: the training sets are then the enriched sets of possible (candidate, argument) pairs $\mathcal{P}_S = \{p_{ij} = (c_i, a_j) | c_i \in \mathcal{C}_S,\ a_j \in \mathcal{A}_S\} \cup \{p_{i\alpha} = (c_i, e_{\alpha\beta}) | c_i \in \mathcal{C}_S, \exists \beta : y_{\alpha\beta} \neq None\}$. This allows to know all training examples beforehand and hence to use standard batch SVM algorithms. The drawback is that, since extracted events in test are imperfect, this creates a divergence between training and testing scenarios, which can lead to degraded performance. However, as our experiments show, this effect is marginal compared to the advantages of using fast reliable batch training algorithms.

Score Combination As said earlier, the decision rule simply consists in predicting the class corresponding to the highest SVM score. This simple scheme could be improved, either by using multi-class classifiers or by using more refined combinations optimizing a global criterion as proposed in (Liu et al., 2013). Though this route deserves to be thoroughly tested, we conjecture that only marginal gains should be expected since the vast majority of errors are due to the detection of an event when there is none or to the absence of detection of an existing event: when an event is de-

tected, its correct type is predominantly predicted.

3.4 Computational Considerations

The pairwise structure leads to a simple inference procedure, with a slight increase in computational complexity compared to pipeline models. We denote $m = \text{card}(\mathcal{C}_S)$, the number of candidate entities, $n = \text{card}(\mathcal{A}_S)$, the number of annotated proteins and m' the number of detected triggers. The complexity of a pipeline model is $\mathcal{O}(m'(n+m'))$, whereas that of our approach is $\mathcal{O}(m(n+m'))$. This implies more calls to the classifying mechanism, but this is not too penalizing, since SVM-based classification scales well with the number of examples. Besides, this is still cheaper than joint models such as (Riedel and McCallum, 2011a), whose complexity is $\mathcal{O}(m(n^2+m))$.

4 Post-Processing

This section describes the post-processing carried out once the (trigger, theme) pairs are detected and labeled as events. The goal is to look whether extra-arguments should be added to these extracted events.

4.1 Binding Theme Fusion

This step attempts to merge several pairs labeled as *Binding* to create multiple arguments events. We take the set of extracted *Binding* events $\{(c_\alpha, a_\beta)\}$ that share the same trigger c_α, and all combinations $\{(c_\alpha, a_\beta, a_\gamma)|\gamma \neq \beta\}$ are classified by a binary SVM. Once a combination $(c_\alpha, a_\beta, a_\gamma)$ is predicted as a correct merge, it is added to predicted events while both pairs (c_α, a_β) and (c_α, a_γ) are removed.

4.2 Regulation Cause Assignment

This step looks for optional *cause* arguments that may be added to the extracted REG events. Given an extracted event (c_α, a_β) and a candidate argument set $\mathcal{A}_S = \{a_\gamma\}$ containing all the proteins of the sentence S as well as all events extracted by the classifier, all combinations $\{(c_\alpha, a_\beta, a_\gamma)|\gamma \neq \beta\}$ are classified by a binary SVM. Since *cause* argument could be another event, we extract them incrementally in a dynamic process alike (trigger, theme) pair extraction, also with constraints avoiding the creation of cycles.

5 Features

This section details our features as well as the data preprocessing used by our system.

Pre-processing Tokenization and sentence splitting have an important impact on the quality of the dependency parse trees as well as the way we handle compound words that contain protein names. Data is split in sentences using both the `nltk` toolkit (`nltk.org`) and the sentence splitting provided for the BioNLP GE task. High quality dependency parse trees require a fine grained tokenization, whereas coarse tokenization conserves some biomedical jargon that could also provide essential information. Hence, two tokenizations are used for different features. *Tokenization1*, provided by the organizers of the BioNLP GE task, is a coarse tokenization that is used to characterize when a candidate entity and a protein are in the same token. *Tokenization2* is fined grained, based on the Stanford parser (McClosky et al., 2011) that is slightly modified for primary tokenization. It supplies the dependency parse, candidate entity match and most of our features. The dependency parse trees are finally obtained using a phrase structure parser (McClosky et al., 2010), using the post-processing of the Stanford `corenlp` package (De Marneffe et al., 2006). We used stems (obtained by Snowball stemmer provided in `nltk`) as base forms for the tokens.

Candidate set For each sentence S, the set $\mathcal{C}_S$ is built with a gazetteer: candidate entities are recursively added by searching first the longest token sequences (from Tokenization2) from the gazetteer. For entities with several tokens, a representative *head* token is selected by a heuristic based on the dependency parse.

Candidate entities Three types of tokens are considered: the head token, its parent and child nodes in the dependency tree, and the tokens belonging to a neighboring window of the entity. The size k of the word window is a hyper-parameter of our model. Table 2 lists all features which include stems, part-of-speech (POS) tags, etc. Special care was taken to design the feature for head token since it plays an extremely important role in candidate entities. We hence employed features and heuristics to deal with compound-words, hyphens and prefixes, inspired by such tools developed in the code of the UCLEED system (based on Tok-

Candidate entity features	Base form (stem) of the head token.
	Base form of the head token without '-' or '/' before of after.
	Sub-string after '-' in the head token.
	POS of the head token.
	First token of the entity is after '-' or '/'.
	Last token of the entity is before '-' or '/'.
	Head token has a special prefix: "over", "up", "down", "co"
	Concat. of base form and POS of parents of the head token in dependency parse.
	Concat. of base form and POS of children of the head token in dependency parse.
	Base forms of k neighboring tokens around the entity.
	POS of k neighboring tokens around the entity.
	Neighborhood of the entity has '-' or '/'.
	Sentence has "mRNA".
	Entity is connected with another string using Tokenization1.
Argument features	Argument is a protein.
	POS of the head token.
	Features extracted from IntAct when the argument is a protein.
	Base forms of k neighboring tokens around the argument.
	POS of k neighboring tokens around the argument.
	Concat. of base form and POS of children of the head token in dependency parse.
Pairwise features	Token sequence between candidate and argument has proteins.
	V-walk features between candidate and argument with base forms.
	E-walk features between candidate and argument with base forms.
	V-walk features between candidate and argument with POS.
	E-walk features between candidate and argument with POS.
	Candidate and the argument share a token using Tokenization1.

Table 2: Features used by our system. Most are based on Tokenization2 except when specified.

enization2).[2] Protein names and POS in tokens are substituted by the token PROT, e.g. transforming "LPS-activated" into "PROT-activated". There is total of a 35,365 candidate features.

Arguments Table 2 also lists the argument features, which are a subset of those for candidate entities. Most head word features are removed, but base forms and POS of the neighboring tokens and of the parent node in the dependency tree are still included. Assigning label from SVT or BIN event classes to a $(c_i, e_{\alpha\beta})$ pair should never occur, because only regulation events could have another event as argument. Therefore, we add a feature that indicates whether the argument is a protein or a trigger entity. Proteins are also described using features extracted from the Uniprot knowledge base (`uniprot.org`). There is total of 4,349 argument features.

[2]See `github.com/riedelcastro/ucleed`.

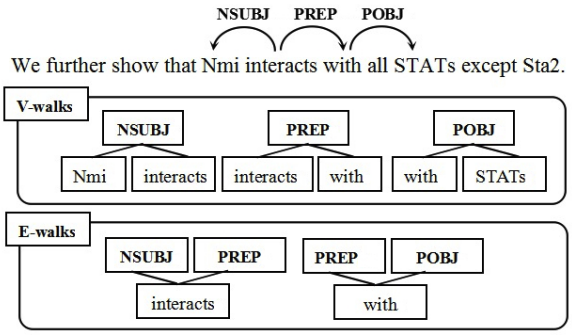

Figure 2: Example of E-walks and V-walks features for encoding a dependency parse tree.

Pairwise relations Our pairwise approach is able to take advantage of features that code interactions between candidate triggers and arguments, such as those listed in Table 2. Hence, we have a feature indicating if both elements of a pair belong to the same token (based on Tokenization1).

But the most important pairwise features come from the shortest path linking two candidate and arguments in the dependency parse tree of the sentence. Incorporating such dependency information into the pairwise model relies on the process encoding the path into feature vectors. Many formatting methods have been proposed in previous works. Following (Miwa et al., 2010), our system use a combination of *E-walks*, that encode the path into triplets (*dep-tag*, *token*, *dep-tag*), and *V-walks* that encode it into triplets (*token*, *dep-tag*, *token*), where *tokens* are encoded via stem and POS tags, and *dep-tags* are the dependency labels. Figure 2 illustrates this formatting: from the dependency parse given on top, three *V-walk* and two *E-walk* features are defined. These are inserted in the feature vector using a bag-of-words process, thus losing any relative ordering information. These imperfect representations lose a lot of information and can even add noise, especially when the path is long. Therefore, we applied heuristics from the UCLEED system to remove some uninformative edges from the dependency parse. Moreover, dependency parse features are added only for pairs for which the (candidate, argument) path length is below a threshold whose value is a hyper-parameter. There is a total of 176,106 pairwise features.

Event Type or Class	TEES 2.1	EVEX	Pipeline counterpart	Our approach
Gene_expression	82.7	82.7	83.9	**85.1**
Transcription	55.0	55.0	61.7	**62.8**
Protein_catabol	56.3	56.3	66.7	**68.8**
Phosphorylation	72.6	71.5	**81.8**	**81.8**
Localization	**63.3**	60.7	56.9	57.7
SVT TOTAL	74.9	74.5	79.0	**79.6**
BIN TOTAL	**43.3**	42.9	41.6	42.4
Regulation	23.0	23.4	23.1	**31.8**
Positive_regul	38.7	39.2	36.5	**46.3**
Negative_regul	43.7	**43.9**	38.1	43.6
REG TOTAL	38.1	38.4	35.1	**43.2**
ALL TOTAL	50.7	51.0	50.8	**54.4**

Table 3: F-scores on the test set of the BioNLP 2013 GE task.

6 Experiments

In this section, we evaluate empirically our system in the framework (data, annotations and evaluation) of biomedical event extraction defined in the GE tasks of the BioNLP challenges. More precisely, we present results on the test sets of the fresh 2013 GE task, and of the 2011 edition to compare to joint methods.

In order to assess the efficiency of our modeling choices, we also implemented a pipeline counterpart system, following the structure of the TEES approach (Björne et al., 2009; Bj[Pleaseinsertintopreamble]rne et al., 2012; Björne and Salakoski, 2013) but using the same feature set, pre-processing and a similar post-processing as our system. This pipeline system comprises four steps: (1) *trigger classification*, which assigns event types from $\mathcal{Y}$ to candidate entities $c_i \in \mathcal{C}_S$ using a multi-class SVM classifier; (2) *edge detection*, which identifies the edges between extracted triggers and proteins and between REG triggers and all the triggers; labels from $\mathcal{Y}_{edge} = \{theme, cause, None\}$ are assigned to those pairs; (3) *binding theme fusion*, identical to as in Section 4.1; (4) *theme-cause fusion*, as in Section 4.2, given two predicted pairs $(c_i, theme : a_\beta)$, $(c_i, cause : a_\gamma)$, this step decides whether they should be merged into a single event $(c_i, theme : a_\beta, cause : a_\gamma)$.

6.1 Genia Shared Task 2013

For the BioNLP 2013 GE task, the hyper-parameters of our system have been optimized on the GE task development set (except for the regularization parameters of the SVMs, which are selected by cross-validation), after training on the corresponding training sets: token window size is 2 for candidate entities and 1 for arguments, the threshold for dependency path is 4. Using these hyper-parameter values, the final model submitted for test evaluation on the GE task server has been trained on all documents from training and development sets of BioNLP 2011 and 2013 GE tasks. Detailed descriptions of the BioNLP 2011 and 2013 GE data are respectively given in (Kim et al., 2011b) and (Kim et al., 2013).

Table 3 lists the detailed test F-scores, as returned by the official challenge test server (using the default *approximate span & recursive matching* evaluation setting). We compare our model to the winner of the challenge, EVEX (Hakala et al., 2013), and of the best runner-up, TEES 2.1 (Björne and Salakoski, 2013), which are both pipeline approaches.

Our approach is slightly below TEES 2.1 on BIN events, but overall, it outperforms all competitors significantly (by more than 3%), with a wide margin on REG events. Our pipeline counterpart has an overall performance similar to EVEX and TEES 2.1, while being better for SVT and worse for BIN and REG events. These disparities are due to the differences in features and in processing details. The benefits of the pairwise structure and the recursive process are demonstrated by the considerable improvement upon the pipeline counterpart (using the same features, pre- and post-processing). In particular, the recursive prediction process run on REG events brings about a very substantial improvement (more than 8%).

6.2 Genia Shared Task 2011

The best performing methods on the BioNLP 2013 GE task were pipeline approaches, but the joint models that were performing better in the previous challenge were not competing in 2013. As these joint models are quite tricky to train, we compare our system with joint models on the BioNLP 2011 GE task, where trustworthy performances have been publicly released. We train our model using the training and development sets available at the time of the challenge and we then get an evaluation on the same test data using the official test server maintained online by BioNLP organizers. Table 4 lists the results of our approach, its pipeline counterpart, and those of UCLEED (Riedel and McCallum, 2011a) and TEES (Bj[Pleaseinsertintopreamble]rne et al.,

Event Class	UCLEED	SEARN	TEES	Pipeline counterpart	Our approach
SVT	73.5	71.8	n/a	71.8	**74.0**
BIN	48.8	45.8	n/a	40.0	**50.5**
REG	43.8	43.0	n/a	35.7	**45.1**
ALL	55.2	53.5	53.3	50.0	**55.6**

Table 4: F-scores on the test set of the BioNLP 2011 GE task.

2012), which are respectively the best performing joint model and best pipeline on this task. We also added SEARN (Vlachos and Craven, 2012), which is a hybrid between them.[3]

As for 2013 data, our system achieves a higher F-score on all event classes compared to its pipeline counterpart. The benefits of the pairwise structure and the recursive process are larger here, thereby outperforming the overall F-score of TEES (no detailed results available), which itself performs better than our pipeline counterpart. Systematic improvements on all event classes are also observed compared to the joint model UCLEED and to the search-based structured prediction approach of SEARN. To our knowledge, our model thus reaches the best overall performance reported so far on this data set for a single model.[4]

By combining the use of the simple pair structure between triggers and arguments with a recursive prediction process, our approach is able to outperform pipeline models and to be at least at par with models relying on much more sophisticated structures. For this task, it is thus highly beneficial to consider pairwise interactions from beginning to end, but more complex dependencies seem not to be essential, especially since they come at a higher computational cost.

6.3 Training Durations

In this last section, we propose to illustrate the lower complexity of our approach compared to UCLEED by providing durations for training both systems on BioNLP 2011 GE. These timings do not involve preprocessing but only running cross-validation on the training set and evaluation on the development and test sets. For UCLEED, we used the code (in java & scala) provided by the authors[5] and we chose BioNLP 2011 GE because this code was primarily designed to run on it. Our code, in python, is publicly available from `github.com/XiaoLiuAI/RUPEE`. Experiments were conducted on the same computer, with a quad-core Intel Xeon CPU and 16GB of RAM. Both codes are multi-threaded and used all 4 threads simultaneously. Under these conditions, UCLEED requires around *8h30min* to run its 10 epochs,[6] while our code completes training in about *30min*. Some of these differences may be due to implementation choices, but we believe that the 15 fold speed increase (for around 800 training documents) is at least partially due to the lower complexity of our approach.

7 Conclusion

We introduced a recursive pairwise model designed for biomedical event extraction. This pairwise model improves on the best current approaches of the BioNLP 2013 and 2011 GE tasks. Our system breaks down the overall event extraction task into the classification of (trigger, theme) pairs, assigned to event types. These (trigger, theme) pairs enable to use joint features in off-the-shelf classifiers, without resorting to costly global inference models. We also implemented a recursive procedure that deals with regulation events, which may include other events in their definition. All operations are run in a unified framework, using a single event classifier.

Our system is fast and more accurate than the available pipeline models or joint models. Given its simplicity and scalability, we believe that our model is a strong basis for large-scale event extraction projects. Several refinements are possible, for example by exploring other types features, or by enabling the direct processing of triplets that may be encountered in binding or regulation events.

Acknowledgments

This work was carried out in the framework of the Labex MS2T funded by the French National Agency for Research through the program "Investments for the future" (ANR-11-IDEX-0004-02), and supported by the "young researchers" program (EVEREST-12-JS02-005-01).

[3]The results for UCLEED, TEES and SEARN models are reproduced from (Riedel and McCallum, 2011a; Bj[Pleaseinsertintopreamble]rne et al., 2012; Vlachos and Craven, 2012) respectively.

[4]We do not compare with the results of FAUST (Riedel et al., 2011), which achieved the best F-score on this task (56.0) because this is an ensemble of various models of UCLEED and of the Stanford system (McClosky et al., 2011), which makes it an unfair comparison.

[5]See `github.com/riedelcastro/ucleed`.

[6]UCLEED might be faster by using feature caching, but we had to disable it because it was taking up too much RAM.

References

J. Björne and T. Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 shared task. In *Proceedings of BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, August. Association for Computational Linguistics.

J. Björne, J. Heimonen, F. Ginter, A. Airola, T. Pahikkala, and T. Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 10–18, Boulder, Colorado, June. Association for Computational Linguistics.

J. Björne, F. Ginter, and T. Salakoski. 2012. University of turku in the bionlp'11 shared task. *BMC Bioinformatics*, 13(Suppl 11):S4.

K. B. Cohen, K. Verspoor, H. Johnson, C. Roeder, P. Ogren, W. Baumgartner, E. White, and L. Hunter. 2009. High-precision biological event extraction with a concept recognizer. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 50–58, Boulder, Colorado, June. Association for Computational Linguistics.

M.-C. De Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

K. Hakala, S. Van Landeghem, T. Salakoski, Y. Van de Peer, and F. Ginter. 2013. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, August. Association for Computational Linguistics.

J.-D. Kim, 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*, pages 1–9, Boulder, Colorado, June. Association for Computational Linguistics.

J.-D. Kim, T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii. 2011a. Extracting bio-molecular events from literature. *Computational Intelligence*, 27(4):513–540.

J.-D. Kim, Y. Wang, T. Takagi, and A. Yonezawa. 2011b. Overview of genia event task in bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 7–15, Portland, Oregon, USA, June. Association for Computational Linguistics.

J.-D. Kim, Y. Wang, and Y. Yasunori. 2013. The genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria, August. Association for Computational Linguistics.

X. Liu, A. Bordes, and Y. Grandvalet. 2013. Biomedical event extraction by multi-class classification of pairs of text entities. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 45–49, Sofia, Bulgaria, August. Association for Computational Linguistics.

D. McClosky, E. Charniak, and M. Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 28–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. McClosky, M. Surdeanu, and C. D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1626–1635, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Miwa, R. Sætre, J.-D. Kim, and J. Tsujii. 2010. Event extraction with complex event classification using rich features. *J. Bioinformatics and Computational Biology*, 8(1):131–146.

K. Morik, P. Brockhausen, and T. Joachims. 1999. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*.

C. Quirk, P. Choudhury, M. Gamon, and L. Vanderwende. 2011. MSR-NLP entry in BioNLP shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 155–163, Portland, Oregon, USA, June. Association for Computational Linguistics.

S. Riedel and A. McCallum. 2011a. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

S. Riedel and A. McCallum. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 46–50, Portland, Oregon, USA, June. Association for Computational Linguistics.

S. Riedel, H.-W. Chun, T. Takagi, and J. Tsujii. 2009. A Markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49, Boulder, Colorado, June. Association for Computational Linguistics.

S. Riedel, D. McClosky, M. Surdeanu, A. McCallum, and C. D. Manning. 2011. Model combination for event extraction in BioNLP 2011. In *Proceedings*

of BioNLP Shared Task 2011 Workshop, pages 51–55, Portland, Oregon, USA, June. Association for Computational Linguistics.

R. Sætre, M. Miwa, K. Yoshida, and J. Tsujii. 2009. From protein-protein interaction to molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 103–106, Boulder, Colorado, June. Association for Computational Linguistics.

S. Van Landeghem, F. Ginter, Y. Van de Peer, and T. Salakoski. 2011. Evex: A pubmed-scale resource for homology-based generalization of text mining predictions. In *Proceedings of BioNLP 2011 Workshop*, pages 28–37, Portland, Oregon, USA, June. Association for Computational Linguistics.

K. Veropoulos, C. Campbell, and N. Cristianini. 1999. Controlling the sensitivity of support vector machines. In T. Dean, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 55–60.

A. Vlachos and M. Craven. 2012. Biomedical event extraction from abstracts and full papers using search-based structured prediction. *BMC bioinformatics*, 13(Suppl 11):S5.