SeeDev Binary Event Extraction using SVMs and a Rich Feature Set

Nagesh C. Panyam, Gitansh Khirbat,
Karin Verspoor, Trevor Cohn and Kotagiri Ramamohanarao
Department of Computing and Information Systems,
The University of Melbourne,
Australia
{npanyam, gkhirbat}@student.unimelb.edu.au
{karin.verspoor, t.cohn, kotagiri}@unimelb.edu.au

Abstract

This paper describes the system details and results of the participation of the team from the University of Melbourne in the SeeDev binary event extraction of BioNLP-Shared Task 2016. This task addresses the extraction of genetic and molecular mechanisms that regulate plant seed development from the natural language text of the published literature. In our submission, we developed a system[1] using a support vector machine classifier with linear kernel powered by a rich set of features. Our system achieved an F1-score of 36.4%.

1 Introduction

One of the biggest research challenges faced by the agricultural industry is to understand the molecular network underlying the regulation of seed development. Different tissues involving complex genetics and various environmental factors are responsible for the healthy development of a seed. A large body of research literature is available containing this knowledge. The SeeDev binary relation extraction subtask of the BioNLP Shared Task 2016 (Chaix et al., 2016) focuses on extracting relations or events that involve two biological entities as expressed in full-text publication articles. The task represents an important contribution to the broader problem of biomedical relation extraction.

Similar to previous BioNLP shared tasks in 2009 and 2011 (Kim et al., 2009; Kim et al., 2011), this task focuses on molecular information extraction. The task organisers provided paragraphs from manually selected full text publications on seed development of *Arabidopsis thaliana* annotated with mentions of biological entities like *proteins* and *genes*, and binary relations like *Exists_In_Genotype* and *Occurs_In_Genotype*. The participants are asked to extract binary relations between entities in a given paragraph.

Several approaches have been proposed to extract biological events from text (Ohta et al., 2011; Liu et al., 2013). Broadly, these approaches can be categorized into two main groups, namely rule-based and machine learning (ML) based approaches. Rule-based approaches consist of a set of rules that are manually defined or semi-automatically inferred from the training data (Abacha and Zweigenbaum, 2011). To extract events from text, first event triggers are detected using a dictionary, then the defined rules are applied over rich representations such as dependency parse trees, to extract the event arguments. On the other hand, ML-based approaches (Miwa et al., 2010) are characterized by learning algorithms such as classification to extract event arguments. Further, they employ various features computed from the textual or syntactic properties of the input text.

This article explains our SeeDev binary relation extraction system in detail. We describe the rich feature set and classifier setup employed by our system that helped achieve the second best F1-score of 36.4% in the shared task.

2 Approach

The seedev task involves extraction of 22 different binary events over 16 entity types. Entity mentions within a sentence and the events between them are provided in the gold standard annotations. In the rest of the article, we refer to an event with two entity arguments as simply a binary relation.

We treat relation identification as a supervised classification problem and created 22 separate

[1]Source: https://github.com/unimelbbionlp/BioNLPST2016/

classifiers denoted as $C_1, C_2, \ldots, C_{22}$, specific to relations $r_1, r_2, \ldots, r_{22}$, respectively. This design choice was motivated by two important aspects, namely vocabulary and relation type signature. We describe them below:

Vocabulary According to the annotation guidelines document, it is clear that different relations are expressed using different vocabulary. For example, "encode" is in the vocabulary of "Transcribes_Or_Translates_To" and "phosphorylate" is in the vocabulary of "Regulation_Of_Molecule_Activity". We hypothesize that treating the vocabulary as a set of trigger words for its corresponding relation would be beneficial. Therefore, we built 22 separate classifiers for each relation type, with vocabulary as a relation specific feature. Given an entity pair (e_a, e_b), we test it with the classifier C_i to detect if the relation r_i holds between e_a and e_b.

Relation type signature Relations are associated with entity type argument signatures, which specify the list of allowed entity types for each argument position. For example, the event "Protein Complex Composition" requires the first entity argument to be one of these four entity types {"Protein", "Protein Family", "Protein Complex", "Protein Domain"} and the second argument to be "Protein Complex". Alternately, relation argument signatures can be used as a filter that specifies the list of invalid relations between an entity pair. We can use this knowledge to prune the training sets of classifier C_i of invalid entity pairs. Relation type signatures overlap but are not identical. Therefore, training set of C_i is different from training set of $C_j, j \neq i$.

2.1 Training

The steps involved in training the aforementioned classifiers are described below.

1. Extract all pairs of candidates (e_a, e_b) that cooccur within a sentence from training documents to form a triple $t = (e_a, e_b, label)$. If e_a and e_b are known to be related by the type r_c, from the relation annotations, we set $label = r_c$. If they are not related, we set $label$ =*NR*. *NR* is a special label to denote no relation.
2. Add the triple $t = (e_a, e_b, label)$ to the training set of C_i, if (e_a, e_b) satisifies the type signature for relation $r_i, i \in [1, 22]$.

We now have classifier specific training sets, which are sets of triples $t = (e_a, e_b, label)$. To train the classifier, we regard these triples as training examples of class type $label$ and a feature vector constructed for the entity pair (e_a, e_b), as explained in section 2.4.

2.2 Testing

During the test phase, we generate candidate entity pairs from sentences in the test documents. We look up into the relation argument signatures to identify the list of possible relation types for this entity pair. For each such relation type r_i, we test the candidate with the classifier C_i. The entity pair (e_a, e_b) is considered to have the relation type r_i if the predicted label from the classifier C_i is r_i. A consequence of the above approach is that we may predict multiple relation types for a single entity pair in a sentence. This is a limitation of our system, as it is unlikely for a sentence to express multiple relationships between an entity pair.

2.3 Classifier details

The classifiers $C_i, i \in [1, 22]$ are trained as multiclass classifiers. Note that the training set of each classifier C_i may include examples of the form $(e_a, e_b, label), label = r_j$ and $j \neq i$, for the reason that (e_a, e_b) satisfies the type signature for r_i. Therefore, at test time a classifier C_i may classify an entity pair (e_a, e_b) as $r_j, j \neq i$. But we note that r_i is the dominant class for the classifier C_i and other relation types r_j are often under represented during its training. Therefore, we discard predictions r_j from C_i when $j \neq i$. For the entity pair (e_a, e_b) to be included in the final set of predicted relations with the type r_i, we require that the classifier C_i label it as r_i.

We experimented with classifiers from Scikit (Pedregosa et al., 2011). For each relation type, we selected a classifier type between linear kernel SVMs and Multinomial Naive Bayes. This choice was based on performance over development data. We combine the development dataset with training dataset and use it all for training. No parameter tuning was performed.

2.4 Feature Engineering

We developed a set of common lexical, syntactic and dependency parse based features. Relation specific features were also developed. For part of speech tagging and dependency parsing of the text, we used the toolset from Stanford CoreNLP (Manning et al., 2014). These features are described in detail below.

1. Stop word removal: For some relations ("Has Sequence Identical To", "Is Functionally Equivalent To","Regulates Accumulation" and "Regulates Expression") we found that it is beneficial to remove stop words from the sentence.
2. Bag of words: Include all words in the sentence as features, prefixed with "pre","mid" or "post" based on their location with reference to entity mentions in the sentence.
3. Part of Speech (POS): Concatenated sequence of POS tags were extracted separately for words before, after and in the middle of entity mentions in the sentence.
4. Entity features: Entity descriptions and entity types were extracted as features.
5. Dependency path features: We compute the shortest path between the entities in the dependency graph of the sentence and then find the neighboring nodes of the entity mentions along the shortest path. The text (lemma) and POS tags of these neighbors are included as features.
6. Trigger words: For each relation, we designate a few special terms as trigger words and flag their presence as a feature. Trigger words were mainly arrived at by examining the annotation guidelines of the task and a few representative examples.
7. Patterns: A common pattern in text documents is to specify equivalent representations using parenthesis. We find if the two entities are expressed in such a way and include it as a special feature for the relations "Is Functionally Equivalent To" and "Regulates Development Phase".

3 Evaluation

The SeeDev-Binary task objective is to extract all related entity pairs at the document level. The metrics are the standard Precision (P), Recall(R) and F1-score ($\frac{2PR}{P+R}$).

3.1 Dataset

The SeeDev-Binary (Chaix et al., 2016) task provides a corpus of 20 full articles on seed development of *Arabidopsis thaliana*, that have been manually selected by domain experts. This corpus consists of a total of $7,082$ entities and $3,575$ binary relations and is partitioned into training, development and test datasets. Gold standard entity and relation annotations are provided for training and development data and for test data only entity annotations have been released. The given set of 16 entity types are categorized into 7 different entity groups and 22 different relation types are defined. Pre-defined event signatures constrain the types of entity arguments for each relation.

3.2 Results

In the development mode, we used the training dataset for training the relation specific classifiers and predicted the relations over the development dataset. Finally, we trained our classifiers with the full training and development data together. With this system, the predicted relations over the test dataset was submitted to the task. Performance results over the test dataset was made available by the task organizers at the conclusion of the event. These results are detailed in Table 3.2.

4 Discussion

We note that the final relation extraction performance is quite low (36.4%), suggesting that SeeDev-Binary event extraction is a challenging problem. Further, for many event types our system was unable to identify any relation mentions. It is not clear as to why our methods are not effective for these relation types, but it is likely that scarcity of training data is the problem. We observed that our system performed poorly on relation types that have < 100 training samples and has generally succeeded on the rest. It is likely that for these sparsely represented relation types, alternate techniques such as rule based methods might be more successful.

We attempted a few alternate techniques and describe the findings from these approaches below.

4.1 Alternate approaches

1. Two stage approach: We attempted building a first stage general filter that identifies event pairs as "related" or "not related". For this, we grouped all candidate pairs with any of the 22 given relation types into the "positive" class and the rest into the "negative" class in a SVM classifier. In the second stage, we built a multiclass classifier that was to further tune the label of an entity pair from "related" to

Event type	Clasifier used	Metrics on Development data			Metrics on Test data		
		F1	Recall	Prec.	F1	Recall	Prec.
Binds To	SVM	0.269	0.291	0.250	0.262	0.250	0.276
Composes Primary Structure	NB	0.482	0.466	0.500	NA	0	0
Composes Protein Complex	NB	NA	0	0	0.500	0.667	0.400
Exists At Stage	NB	NA	0	0	NA	0	0
Exists In Genotype	SVM	0.248	0.222	0.281	0.354	0.315	0.404
Has Sequence Identical To	SVM	0.336	0.800	0.213	NA	0	0
Interacts With	SVM	0.245	0.218	0.280	0.286	0.241	0.351
Is Functionally Equivalent To	SVM	0.238	0.256	0.222	NA	0	0
Is Involved In Process	SVM	NA	0	0	NA	0	0
Is Localized In	SVM	0.431	0.468	0.400	0.388	0.435	0.351
Is Member Of Family	SVM	0.389	0.545	0.303	0.417	0.523	0.346
Is Protein Domain Of	SVM	0.111	0.068	0.285	0.295	0.419	0.228
Occurs During	NB	NA	0	0	NA	0	0
Occurs In Genotype	SVM	NA	0	0	NA	0	0
Regulates Accumulation	SVM	0.444	0.344	0.625	0.316	0.188	1
Regulates Development Phase	SVM	0.380	0.338	0.434	0.376	0.442	0.327
Regulates Expression	SVM	0.486	0.477	0.495	0.386	0.471	0.327
Regulates Process	NB	0.420	0.513	0.355	0.400	0.394	0.406
Regulates Tissue Development	NB	NA	0	0	NA	0	0
Regulates Molecule Activity	NB	NA	0	0	NA	0	0
Transcribes Or Translates To	NB	0.100	0.076	0.142	NA	0	0
Is Linked To	SVM	NA	0	0	NA	0	0
All Relations	-	0.354	0.360	0.348	0.364	0.386	0.34

Table 1: Results for relation extraction. NB is Multinomial Naive Bayes. Prec is Precision.

one of the 22 relation types. We observed poor performance for the first stage filter and a drop in overall performance.

2. Binary classifiers: We attempted training the classifiers $C_i, i \in [1, 22]$ as binary classifiers, by modifying the triples (e_a, e_b, r_j) to $(e_a, e_b, +)$ if $j == i$ and $(e_a, e_b, -)$ if $j \neq i$. At test time, positive predictions from C_i were inferred as relations r_i. We observed that this approach of combining many subclasses into one negative class reduced precision and hence overall performance.
3. Co-occurrence: A simple approach to relation extraction is to consider all event pairs that occur within a sentence as related. We tried using this cooccurrence strategy for relation types for which SVM or Naive Bayes classifiers did not work effectively. We abandoned this strategy as we observed that the overall F1 score reduced over the development dataset, even as the recall at the relation level improved.
4. Kernel methods: We experimented with the shortest dependency path kernel (Bunescu and Mooney, 2005) and the subset tree kernels (Moschitti, 2006) for classification with SVMs. However their performance was quite low (F1 score < 0.20). It is likely that small training set sizes and multiple entity pairs in most sentences affect the performance of these kernel methods.
5. Dominant class types : In our system we adopted the strategy of only accepting predictions of the dominant class type from each classifier. That is, we filter out predictions of type r_j from classifier C_i when $j \neq i$. This strategy proved very effective when tested over the development dataset. Without this filtering step, we found that our system gets a high recall as expected (0.896) but also too many false positives resulting in low precision (0.027) and F-score (0.053).

True	Predicted relation type													
relation type	NR	BT	CP	EG	HS	IW	IF	IL	IM	IP	RA	RD	RE	RP	TO
NR	NA	**21**	**5**	**46**	**43**	**15**	**32**	**33**	**67**	**5**	**6**	**26**	**54**	**157**	**6**
BT	**14**	**7**	0	0	0	**3**	0	0	0	0	0	0	0	0	0
CP	**7**	0	**7**	0	**1**	0	0	0	0	0	0	0	0	0	0
EG	**63**	0	0	**18**	0	0	0	0	0	0	0	0	0	0	0
HS	**1**	0	**1**	0	**16**	0	**3**	0	0	0	0	0	0	0	0
IW	**23**	0	0	0	0	**7**	0	0	**1**	0	0	0	0	0	0
IF	**14**	0	**1**	0	**14**	0	**10**	0	**1**	0	0	0	0	0	0
IL	**25**	0	0	0	0	0	0	**22**	0	0	0	0	0	0	0
IM	**25**	0	0	0	**1**	0	0	0	**30**	0	0	0	0	0	0
IP	**27**	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0
RA	**19**	0	0	0	0	0	0	0	0	0	**10**	0	0	0	0
RD	**39**	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0
RE	**57**	0	0	0	0	0	0	0	0	0	0	0	**53**	0	0
RP	**86**	0	0	0	0	0	0	0	0	0	0	0	0	**92**	0
TO	**12**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Table 2: Confusion matrix for evaluation over development data using our multiclass classifiers. Rows and columns represent the relations Not Related(NR), Binds To(BT) , Composes Primary Structure(CP) , Exists In Genotype(EG) , Has Sequence Identical To(HS) , Interacts With(IW) , Is Functionally Equivalent To(IF) , Is Localized In(IL) , Is Member Of Family(IM) , Is Protein Domain Of(IP) , Regulates Accumulation(RA) , Regulates Development Phase(RD) , Regulates Expression(RE) , Regulates Process(RP) and Transcribes Or Translates To(TO).

4.2 Error analysis

In Table 4.2 we show the confusion matrix for 16 classifiers of our system, when evaluated over the development dataset. The remaining 6 classifiers were left out as they have 0 predictions and are discussed separately in Section 4.2.1. The entries of the confusion matrix $CM[i, j]$ are the number of test examples whose true type is i and its predicted label is j. From the confusion matrix we see that the primary source of errors is in predicting a relation where there is none or vice versa. Amongst the related entity pairs, the classifier for "Has Sequence Identical To" makes the most errors when the input examples are of type "Is Functionally Equivalent To". Adding more discriminatory features or keywords to discriminate between these two classes is likely to improve performance. Better handling of unrelated entity pairs is likely to be achieved with more syntactic or dependency parse related features, that specifically target the entity mentions in the sentence.

4.2.1 Unsuccessful classifiers

In Table 3.2, the F-score for some of the relation types has been recorded as not available("NA") as our classifiers failed to predict any relations. Studying the confusion matrix at the classifier level confirms that the classifier did not have enough evidence to detect a relation in many cases. Also, for most of these unsuccessful relation types we observed that the primary class type is under-represented in their training set. For example, the training sets for the classifier for "Exists At Stage" has $3X$ more examples of type "Regulates Development Phase" than examples of type "Exists At Stage". Better ways of handling class imbalance may improve performance.

5 Conclusion

SeeDev-Binary event extraction was shown to be an important but challenging problem in the BioNLP-Shared Task 2016. This task is also unusual as it calls for the extraction of multiple relation types amongst multiple entity types, often cooccurring in a single sentence. In this paper, we describe our system, which was ranked second with an F1 score of 0.364 in the official results of the task. Our solution was based on a series of supervised classifiers and a rich feature set that contributes to effective relation extraction.

References

Asma Ben Abacha and Pierre Zweigenbaum. 2011. Automatic extraction of semantic relations between medical entities: a rule based approach. *Journal of biomedical semantics*, 2(5):1.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Estelle Chaix, Bertrand Dubreucq, Abdelhak Fatihi, Dialekti Valsamou, Robert Bossy, Mouhamadou Ba, Louise Delger, Pierre Zweigenbaum, Philippe Bessires, Loc Lepiniec, and Claire Ndellec. 2016. Overview of the regulatory network of plant seed development (seedev) task at the bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task workshop*, Berlin, Germany, August. Association for Computational Linguistics.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, BioNLP Shared Task '11, pages 1–6, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haibin Liu, Karin Verspoor, Donald C Comeau, Andrew MacKinlay, and W John Wilbur. 2013. Generalizing an approximate subgraph matching-based system to extract events in molecular biology and cancer genetics. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 76–85.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(01):131–146.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning.

Tomoko Ohta, Sampo Pyysalo, Makoto Miwa, and Jun'ichi Tsujii. 2011. Event extraction for dna methylation. *Journal of Biomedical Semantics*, 2(5):1–15.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.