VERSE: Event and Relation Extraction in the BioNLP 2016 Shared Task

Jake Lever and Steven JM Jones
Canada's Michael Smith Genome Sciences Centre
570 W 7th Ave, Vancouver
BC, V5Z 4S6, Canada
{jlever,sjones}@bcgsc.ca

Abstract

We present the Vancouver Event and Relation System for Extraction (VERSE)[1] as a competing system for three subtasks of the BioNLP Shared Task 2016. VERSE performs full event extraction including entity, relation and modification extraction using a feature-based approach. It achieved the highest F1-score in the Bacteria Biotope (BB3) event subtask and the third highest F1-score in the Seed Development (SeeDev) binary subtask.

1 Introduction

Extracting knowledge from biomedical literature is a huge challenge in the natural language parsing field and has many applications including knowledge base construction and question-answering systems. Event extraction systems focus on this problem by identifying specific events and relations discussed in raw text.

Events are described using three key concepts, entities, relations and modifications. Entities are spans of text that describe a specific concept (e.g. a gene). Relations describe a specific association between two (or potentially more) entities. Together entities and relations describe an event or set of events (such as complex gene regulation). Modifications are changes made to events such as speculation.

The BioNLP Shared Tasks have encouraged research into new techniques for a variety of important NLP challenges. Occurring in 2009, 2011 and 2013, the competitions were split into several subtasks (Kim et al., 2009; Kim et al., 2011; Nédellec et al., 2013). These subtasks provided annotated texts (commonly abstracts from PubMed) of entities, relations and events in a particular biomedical domain. Research groups were then challenged to generate new tools to better predict new relations and events in test data.

The BioNLP 2016 Shared Task contains three separate parts, the Bacteria Biotope subtask (BB3), the Seed Development subtask (SeeDev) and the Genia Event subtask (GE4). The BB3 and SeeDev subtasks have separate parts that specialise in entity recognition and relation extraction. The GE4 subtask focuses on full event extraction of NFkB related gene events.

Previous systems for relation and event extraction have taken two main approaches: rule-based and feature-based. Rule-based methods learn specific patterns that fit different events, for instance, the word "expression" following a gene name generally implies an expression event for that gene. The pattern-based tool BioSem (Bui et al., 2013) in particular performed well in the Genia Event subtask of the BioNLP'13 Shared Task. Feature-based approaches translate the textual content into feature vectors that can be analysed with a traditional classification algorithm. Support vector machines (SVMs) have been very popular with successful relation extraction tools such as TEES (Björne and Salakoski, 2013).

We present the Vancouver Event and Relation System for Extraction (VERSE) for the BB3 event subtask, the SeeDev binary subtask and the Genia Event subtask. Utilising a feature-based approach, VERSE builds on the ideas of the TEES system. It offers control over the exact semantic features to use for classification, allows feature selection to reduce the size of feature vectors and uses a stochastic optimisation strategy with k-fold cross-validation to identify the best parameters. We examine the competitive results for the various subtasks and also analyse VERSE's capability to predict relations across sentence boundaries.

[1] http://www.github.com/jakelever/VERSE

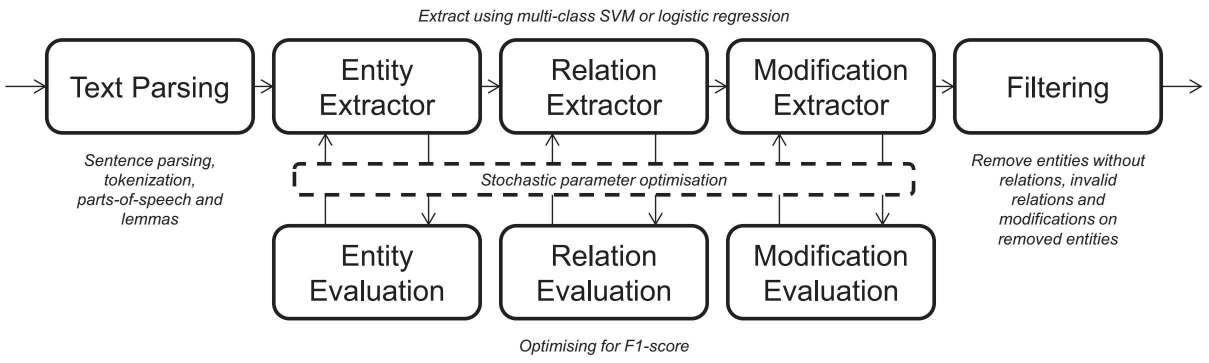

Extract using multi-class SVM or logistic regression
Text Parsing
Entity Extractor
Relation Extractor
Modification Extractor
Filtering
Sentence parsing, tokenization, parts-of-speech and lemmas
Stochastic parameter optimisation
Remove entities without relations, invalid relations and modifications on removed entities
Entity Evaluation
Relation Evaluation
Modification Evaluation
Optimising for F1-score

Figure 1: Overview of VERSE pipeline

2 Pipeline

VERSE breaks event extraction into five steps outlined in the pipeline shown in Figure 1. Firstly the input data is passed through a text processing tool that splits and tags text and associates the parsed results with the provided annotations. This parsed data is then passed through three separate classifications steps for entities, relations and modifications. Finally, the results are filtered to make sure that all relations and modifications fit the expected types for the given task.

2.1 Text processing

VERSE can accept input in the standard BioNLP-ST format or the PubAnnotation JSON format (Kim and Wang, 2012). Both formats are standoff, as they contain the text and annotations separately. The annotations describe entities in the text as spans of text and relations and modifications of these entities.

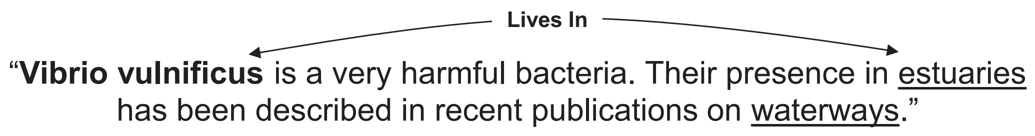
The input files for the shared subtasks are initially processed using the Stanford CoreNLP toolset. The texts are split into sentences and tokenized. Parts-of-speech and lemmas are identified and a dependency parse is generated for each sentence. CoreNLP also returns the exact positions of each token. Using this data, an interval tree is created to identify intersections of text with entities described in the associated annotation. The specific sentence and locations of each associated word are then stored for each entity. Relations and modifications described in the associated annotations are also loaded, retaining information on which entities are involved.

The entities in the BB3 and SeeDev subtasks are generally sets of full words but can be non-contiguous. Entities are stored as a set of associated words rather than a span of words. The GE4 task also contains entities that contain only partial words, for example, ”PTEN” is tagged as an entity within ”PTEN-deficient”. A list of common prefixes and suffixes from the GE4 task is used to separate these words into two words so that the example would become ”PTEN deficient”. Furthermore, any annotation that divides a word that contains a hyphen or forward slash causes the word to be separate into two separate words.

For easier interoperability, the text parsing code was developed in Jython (Developers, 2008) (a version of Python that can load Java libraries, specifically the Stanford CoreNLP toolset). This Jython implementation is then able to export easily processed Python data structures. Due to incompatibility between Jython and various numerical libraries, a separate Python-only implementation loads the generated data structures for further processing and classification.

2.2 Candidate generation

For all three classifications steps (entities, relations and modifications), the same machine learning framework is used. All possibles candidates are generated for entities, relations or modifications. For relations, this means all pairs of entities are found (within a certain sentence range). For the training step, the candidates are associated with a known class (i.e. the type of relation), or the negative class if the candidate is not annotated in the training set. For testing, the classes are unknown. Candidates can contain one argument (for entity extraction and modification) or two arguments (for relation extraction). These arguments are stored as references to sentences and the indices of the associated words.


Lives In
"Vibrio vulnificus is a very harmful bacteria. Their presence in estuaries has been described in recent publications on waterways."

Lives_In	Argument 1	Argument 2
No	vibrio vulnificus	waterways
Yes	vibrio vulnificus	estuaries
No	waterways	vibrio vulnificus
No	waterways	waterways
No	estuaries	vibrio vulnificus
No	estuaries	estuaries

(a) Ignoring argument types

Lives_In	Argument 1 (Bacteria)	Argument 2 (Habitat)
No	vibrio vulnificus	waterways
Yes	vibrio vulnificus	estuaries

(b) Filtering using argument types

Figure 2: Relation candidate generation for the example text which contains a single Lives_In relation (between bacteria and habitat). The bacteria entity is shown in bold and the habitat entities are underlined. Relation example generation creates pairs of entities that will be vectorised for classification. (a) shows all pairs matching without filtering for specific entity types (b) shows filtering for entity types of bacteria and habitat for a potential Lives_In relation

2.2.1 Entity extraction

Entity extraction aims to classify individual or sets of words as a certain type of entity, given a set of training cases. Entities may contain non-contiguous words. The set of all possible combinations of words that could compose an entity is too large for the classification system. Hence VERSE filters for only combinations of words that are identified as entities in the training set.

2.2.2 Relation extraction

VERSE can predict relations between two entities, also known as binary relations. Candidates for each possible relation are generated for every pair of entities that are within a fixed sentence range. Hence when using the default sentence range of 0, only pairs of entities within the same sentence are analysed. VERSE can optionally filter pairs of entities using the expected types for a set of relations as shown in Figure 2.

Each candidate is linked with the locations of the two entities. If the two entities are already annotated to be in a relation, then they are labelled with the corresponding class. Otherwise, the binary relation candidate is annotated with the negative class.

2.2.3 Modification extraction

VERSE supports modification of entities in the form of event modification but currently does not support modification of individual relations. A modification candidate is created for all entities that form the base of an event. These entities are often known as the triggers of the event. In the JSON format, these entities traditionally have IDs that start with "E". If a modification exists in the training set for that entity, the appropriate class is associated with it. Individual binary classifiers are generated for each modification type. This allows an event to be classified with more than one modification.

2.3 Classification

All candidates are vectorized using the same framework, whether for candidates with one or two arguments with minor changes. The full set of features is outlined in Section 3. These vectorized candidates are then used for training a traditional classifier. The vectors may be reduced using feature selection. Most importantly, the parameters used for the feature generation and classifier can easily be varied to find the optimal results. Classification uses the scikit-learn Python package (Pedregosa et al., 2011).

2.3.1 Feature selection

VERSE implements optional feature selection using a chi-squared test on individual parameters against the class variable. The highest ranking features are then filtered based on the percentage of features desired.

2.3.2 Classifier parameters

Classification uses either a support vector machine (SVM) or logistic regression. When using the SVM, the linear kernel is used due to lower time complexity. The multi-class classification uses a one-vs-one approach. The additional parameters of the SVM that are optimised are the penalty parameter C, class weighting approach and whether to use the shrinking heuristic. The class weighting is important as the negative samples greatly outnumber the positive samples for most problems.

2.3.3 Stochastic parameter optimisation

VERSE allows adjustment of the various parameters including the set of features to generate, the classifier to use and the associated classification parameters. The optimisation strategy involves initially seeding 100 random parameter sets. After this initial set, the top 100 previous parameter sets are identified each iteration and one is randomly selected. This parameter set is then tweaked as follows. With a probability of 0.05, an individual parameter is changed. In order to avoid local maxima, an entirely new parameter set is generated with a probability of 0.1. For the subtasks, a 500 node cluster using Intel X5650s was used for optimisation runs.

The optimal parameters are determined for the entity extraction, relation extraction and each possible modification individually. In order to balance precision and recall equally at each stage, the F1-score is used.

2.4 Filtering

Final filtering is used to remove any predictions that do not fit into the task specification. Firstly all relations are checked to see that the types of the arguments are appropriate. Any entities that are not included in relations are removed. Finally, any modifications that do not have appropriate arguments or were associated with removed entities are also trimmed.

Feature Name	Target
unigrams	Entire Sentence
unigrams & parts-of-speech	Entire Sentence
bigrams	Entire Sentence
skipgrams	Entire Sentence
path edges type	Dependency Path
unigrams	Dependency Path
bigrams	Dependency Path
unigrams	Each Entity
unigrams & parts-of-speech	Each Entity
nearby path edge types	Each Entity
lemmas	Each Entity
entity types	Each Entity
unigrams of windows	Each Entity
is relation across sentences	N/A

Table 1: Overview of the various features that VERSE can use for classification

2.5 Evaluation

An evaluation system was created that generates recall, precision, and associated F1-scores for entities, relations and modifications. The system works conservatively and requires exact matches. It should be noted that our internal evaluation system gave similar but not exactly matching results to the online evaluation system for the BB3 and SeeDev subtasks.

K-fold cross-validation is used in association with this evaluation system to assess the success of the system. The entity, relation and modification extractors are trained separately. For the BB3 and SeeDev subtasks, two-fold cross-validation is used, using the provided split of training and development sets as the training sets for the first and second fold respectively. For the GE4 task, five-fold cross-validation is used. The average F1-score of the multiple folds is used as the metric of success.

3 Features

For each generated candidate, a variety of features (controllable through a parameter) is calculated. The features focus on characteristics of the full sentence, dependency path or individual entities. The full-set is shown in Table 1. It should also be noted that a term frequency-inverse document frequency (TFIDF) transform is also an option for all bag-of-words based features.

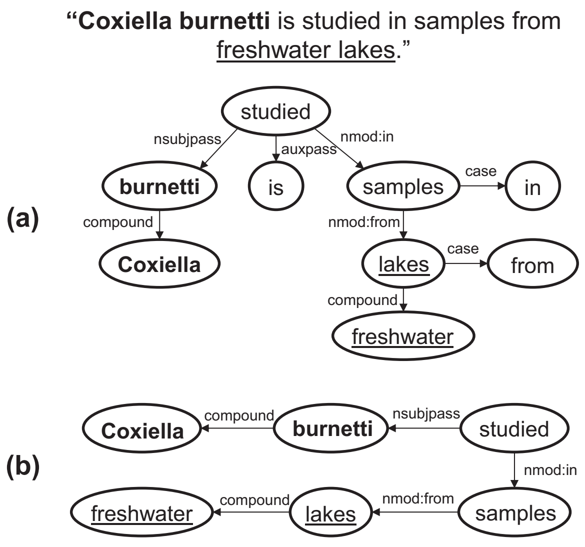


Figure 3: Dependency parsing of the shown sentence provides (a) the dependency graph of the full sentence which is then reduced to (b) the dependency path between the two multi-word terms. This is achieved by finding the subgraph which contains all entity nodes and the minimum number of additional nodes.

3.1 Full sentence features

N-grams features (unigrams and bigrams) use a bag-of-words approach to count the word occurrences across the whole sentence. The words are transformed to lowercase but notably are not filtered for stop words. A version combining the individual words with part-of-speech information is also used. A bag-of-words vector is also generated for lemmas of all words in the sentence. Skip-gram-like features are generated using two words separated by a fixed window of words are also used to generate features. Hence the terms "regulation of EGFR" and "regulation with EGFR" would match the same features of "regulation * EGFR".

3.2 Dependency path features

The dependency path is the shortest path between the two entities in a dependency parse graph and has been shown to be important for relation extraction (Bunescu and Mooney, 2005). Features generated from the set of edges and nodes of the dependency graph include a unigrams and bigrams representation. The specific edge types in the dependency path are also captured with a bag-of-words vector. In order to give specific information about the location of the entity in the dependency path, the types of the edges leaving the entity nodes are recorded separately for each entity.

Interestingly an entity may span multiple nodes in the dependency graph. An example of a dependency path with the multi-word entities "coxiella burnetii" and "freshwater lakes" is shown in Figure 3. In this case, the minimal subgraph that connects all entity nodes in the graph is calculated. This problem was transformed into a minimal spanning tree problem as follows and solved using the NetworkX Python package (Hagberg et al., 2008). The shortest paths through the graph were found for all pairs of entity nodes (nodes associated with the multi-word entities). The path distance between each pair was totalled and used to generate a new graph containing only the entity nodes. The minimal spanning tree was calculated and the associated edges recovered to generate the minimal subgraph. This approach would allow for a dependency path-like approach for relations between more than two entities.

3.3 Entity features

The individual entities are also used to generate specific features. Three different vectorised versions use a unigrams approach, a unigrams approach with parts-of-speech information and lemmas respectively. A one-hot vector approach is used to represent the type of each entity. Unigrams of words around each entity within a certain window size are also generated.

3.4 Multi-sentence and single entity features

VERSE is also capable of generating features for relations between two entities that are in different sentences. In this case, all sentence features are generated for both sentences together and no changes are made to the entity features.

The dependency path features are treated differently. The dependency path for each entity is created as the path from the entity to the root of the dependency graph, generally the main verb of the sentence. This then creates two separate paths, one per sentence and the features are generated in similar ways using these paths. Finally, a simple binary feature is generated for relation candidates that span multiple sentences.

For relation and modifications, candidates contain only a single argument. The dependency path is created in a similar manner to candidates of relations that span across sentences.

Parameter	BB3 event	SeeDev binary
Features	unigrams unigrams POS bigrams of dependency path unigrams of dependency path path edges types entity types entity lemmas entity unigrams POS path edges types near entities	unigrams unigrams POS path edges types path edges types near entities entity types
Feature Selection	No	Top 5%
Use TFIDF	Yes	Yes
Sentence Range	0	0
SVM Kernel	linear	linear
SVM C Parameter	0.3575	1.0 (default)
SVM Class Weights	Auto	5 for positive and 1 for negative
SVM Shrinking	No	No

Table 2: Parameters used for BB3 and SeeDev subtasks

4 Results and discussion

The VERSE tool as described was applied to three subtasks: the BB3 event subtask, the SeeDev binary subtask and the GE4 subtask.

4.1 Datasets

The BB3 event dataset provided by the BioNLP-ST 16 organizers contains a total of 146 documents (with 61, 34 and 51 documents in the training, development and test sets respectively). These documents are annotated with entities of the following types and associated total counts: bacteria (932), habitat (1,861) and geographical (110). Only a single relation type (Lives_In) is annotated which must be between a bacteria and habitat or a bacteria and a geographical entity.

The dataset for the SeeDev binary subtask contains 20 documents with a total of 7,082 annotated entities and 3,575 relations. There are 16 entity types and 22 relation types.

The GE4 dataset focuses on NFkB gene regulation and contains 20 documents. After filtering for duplicates and cleanup, it contains 13,012 annotated entities of 15 types. These entities are in 7,232 relations of 5 different types. It also contains 81 negation and 121 speculation modifications for events. Coreference data is also provided but was not used.

4.2 Cross-validated results

Both BB3 event and SeeDev binary subtasks required only relation extraction. VERSE was trained through cross-validation using the parameter optimising strategy and the optimal parameters are outlined in Table 2. Both tasks were split into training and development sets by the competition organisers. The training set contained roughly twice as many annotations as the development set. We used this existing split for the two-fold cross-validation. A linear kernel SVM was found to perform the best in both tasks. For both subtasks, relation candidates were generated ignoring the argument types as shown in Figure 2.

The classifiers for the two tasks use two very different sizes of feature vectors. The BB3 parameter set has a significant amount of repeated unigrams data, with unigrams for the dependency path and whole sentence with and without parts of speech. This parameter set also does not use feature selection, meaning that the feature vectors are very large (14,862 features). Meanwhile, the SeeDev parameters use feature selection to select the top 5% of features which reduces the feature vector from 7,140 features down to only 357. This size difference is very interesting and warrants further exploration of feature selection for other tasks.

Unfortunately, both classifiers performed best with a sentence range of zero, meaning that only relations within sentences could be detected. Ta-

	Fold 1	Fold 2	Average
Recall	0.552	0.610	0.581
Precision	0.469	0.582	0.526
F1-score	0.507	0.596	0.552

Table 3: Cross-validated results of BB3 event subtask using optimal parameters in Table 2

	Fold 1	Fold 2	Average
Recall	0.363	0.386	0.375
Precision	0.261	0.246	0.254
F1-score	0.303	0.301	0.302

Table 4: Cross-validated results of SeeDev binary subtask using optimal parameters in Table 2

bles 3 and 4 show the optimal cross-validated results that were found with these parameters. Notably, the F1-scores for the two folds of the SeeDev dataset are very similar, which is surprising given that the datasets are different sizes.

For the GE4 subtask, the cross-validation based optimisation strategy was used to find parameters for the entity, relation and modification extractions independently. Due to the larger dataset, filtering was applied to the argument types of relation candidates as shown in Figure 2. Table 5 outlines the resulting F1-scores from the five-fold cross-validations. As these extractors are trained separately, their performance in the full pipeline would be expected to be worse. This is explained as any errors during entity extraction are passed onto relation and modification extraction.

4.3 Competition results

The official results for the BB3 and SeeDev tasks are shown in Table 6. VERSE performed well in both tasks and was ranked first for the BB3 event subtask and third for the SeeDev binary subtask. The worse performance for the SeeDev dataset may be explained by the added complexity of many additional relation and entity types.

Table 7 shows the final results for the test set

	Entities	Relations	Mods
Recall	0.703	0.695	0.374
Precision	0.897	0.736	0.212
F1-score	0.786	0.715	0.266

Table 5: Averaged cross-validated F1-score results of GE4 event subtask with entities, relations and modifications trained separately

	BB3 event	SeeDev binary
Recall	0.615	0.458
Precision	0.510	0.273
F1-score	0.558	0.342

Table 6: Final reported results for the BB3 event and SeeDev binary subtasks

	Entities	Relations	Mods
Recall	0.71	0.23	0.11
Precision	0.94	0.60	0.38
F1-score	0.81	0.33	0.17

Table 7: Final reported results for GE4 subtask split into entity, relations and modifications results

for the Genia Event subtask using the online evaluation tool. As expected, the F1-scores of the relation and modification extraction are lower for the full pipeline compared to the cross-validated results. Nevertheless, the performance is very reasonable given the more challenging dataset.

4.4 Multi-sentence analysis

29% of relations span sentence boundaries in the BB3 event dataset and 4% in the SeeDev dataset. Most relation extraction systems do not attempt to predict these multi-sentence relations. Given the higher proportion in the BB3 set, we use this dataset for further analysis of VERSE's ability to predict relations that span sentence boundaries. It should be noted that some of these relations may be artifacts due to false identification of sentence boundaries by the CoreNLP pipeline.

Using the optimal parameters for the BB3 problem, we analysed prediction results using different values for the sentence range parameter. The performance, shown in Figure 4, is similar for relations within the same sentence using different sentence range parameters. However, as the distance of the relation increases, the classifier predicts larger ratios of false positives to true positives. With sentence range = 3, the overall F1-score for the development set has dropped to 0.326 from 0.438 when sentence range = 1.

The classifier is limited by the small numbers of multi-sentence relations to use as a training set. With a suitable amount of data, it would be worthwhile exploring the use of separate classifiers for relations that are within sentences and those that span sentences as they likely depend on different features.

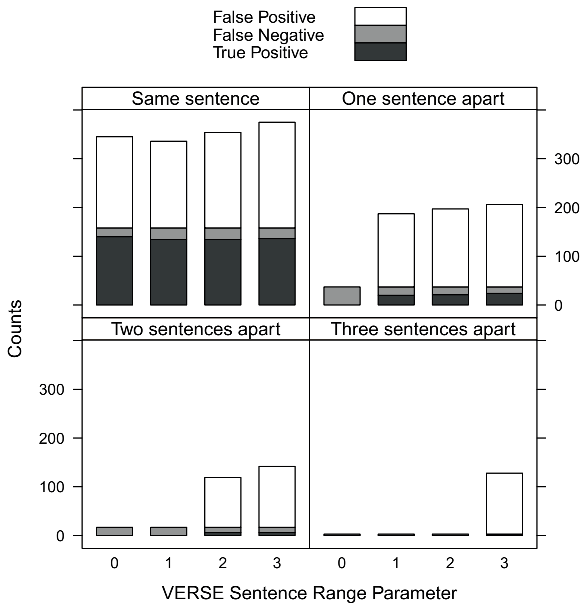


Figure 4: Analysis of performance on binary relations that cross sentence boundaries. The classifier was trained on the BB3 event training set and evaluated using the corresponding development set.

5 Conclusion

We have presented VERSE, a full event extraction system that performed very well in the BioNLP 2016 Shared Task. The VERSE system builds upon the success of previous systems, particularly TEES, in several important ways. It gives full control of the specific semantic features used to build the classifier. In combination with the stochastic optimisation strategy, this control has been shown to be important given the differing parameter sets found to be optimal for the different subtasks. Secondly, VERSE allows for feature selection which is important in reducing the size of the large sparse feature vectors and avoid overfitting. Lastly, VERSE can predict relations that span sentence boundaries, which is certain to be an important avenue of research for future relation extraction tasks. We hope that this tool will become a valuable asset in the biomedical text-mining community.

Acknowledgments

This research was supported by a Vanier Canada Graduate Scholarship. The authors would like to thank ComputeCanada for computational resources used in this project.

References

Jari Björne and Tapio Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 Shared Task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25.

Quoc-Chinh Bui, David Campos, Erik van Mulligen, and Jan Kors. 2013. A fast rule-based approach for biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 104–108. Association for Computational Linguistics.

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*, pages 724–731. Association for Computational Linguistics.

Jython Developers. 2008. Jython implementation of the high-level, dynamic, object-oriented language python written in 100% pure Java. Technical report, Technical report (1997-2016), http://www. jython. org/(accessed May 2016).

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August.

Jin-Dong Kim and Yue Wang. 2012. PubAnnotation: a persistent and sharable corpus and annotation repository. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 202–205. 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*, pages 1–9. 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*, pages 1–6. Association for Computational Linguistics.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

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.