Painless Relation Extraction with Kindred

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

Relation extraction methods are essential for creating robust text mining tools to help researchers find useful knowledge in the vast published literature. Easy-to-use and generalizable methods are needed to encourage an ecosystem in which researchers can easily use shared resources and build upon each others' methods. We present the Kindred Python package[1] for relation extraction. It builds upon methods from the most successful tools in the recent BioNLP Shared Task to predict high-quality predictions with low computational cost. It also integrates with PubAnnotation, PubTator, and BioNLP Shared Task data in order to allow easy development and application of relation extraction models.

1 Introduction

Modern biomedical research is beginning to rely on text mining tools to help search and curate the ever-growing published literature and to interpret large numbers of electronic health records. Many text mining tools employ information extraction (IE) methods to translate knowledge discussed in free text into a form that can be easily searched, analyzed and used to build valuable biomedical databases. Examples of applications of IE methods include building protein-protein interaction networks (Donaldson et al., 2003) and automatically retrieving information about proteins (Rebholz-Schuhmann et al., 2007).

Information extraction relies on several key technologies including relation extraction. Relation extraction focuses on understanding the relation between two of more biomedical terms in a stretch of text. This may be understanding how one protein interacts with another protein, whether a drug treats or causes a particular symptom and many other uses. Most methods assume that entities (e.g. gene and drug names) in the sentence have already been identified, either through a named entity recognition tools (e.g. BANNER (Leaman et al., 2008)) or basic dictionary matching against a word list. The method must then use linguistic cues within the sentence to predict whether or not a relation exists between each pair or group of entities and exactly which type of relation it is.

The BioNLP Shared Task has catalyzed research in relation extraction tools by providing an environment for friendly competition between different relation extraction approaches. The organizers of the relation extraction subtasks provide text from published literature with entities and relations annotated. The participating researchers build relation extraction models and predicted relations on a test set. The participants' predictions are then analyzed by the organizers and the results presented to all. The BioNLP Shared Task has been held in 2009, 2011, 2013 and recently in 2016. The recent 2016 relation extraction problems focused on two areas: bacteria biotopes (BB3 subtask) and seed development (SeeDev subtask). The BB3 subtask required participants to predict relations between bacteria and their habitats. The SeeDev subtask involved prediction of over twenty different relation types related to seed development.

Two main approaches to relation extraction have been taken, a rule-based method and a vector-based method. A rule-based approach identifies common patterns that capture a relation. For instance, two gene names with the word "regulates" between them generally implies a regulation relation between the two entities. The BioSem method

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

Proceedings of the BioNLP 2017 workshop, pages 176–183,
Vancouver, Canada, August 4, 2017.

(Bui et al., 2013) identifies common patterns of words and parts-of-speech between biomedical terms and performed well in the BioNLP Shared Task in 2013.

The vector-based approach transforms a span of text and candidate relation into a numerical vector that can be used in a traditional machine learning classification approach. Support vector machines (SVM) have commonly been used. The TEES (Björne and Salakoski, 2013) and VERSE (Lever and Jones, 2016) methods, which were successful in many of the shared tasks, use this approach with different approaches for creating the vectors and selecting the parameters for classification.

Deep learning, already very popular in natural language processing (LeCun et al., 2015), has begun to be used in the biomedical text mining field with one entry in the BioNLP Shared Task using a recurrent neural network approach (Mehryary et al., 2016). The paper examined the use of long short-term memory (LSTM) networks for relation extraction, especially in situations with small training dataset sizes. Given such a complicated model, the problem of overfitting becomes very large. They proposed approaches to reduce overfitting and the entry performed very well, coming second in the competition.

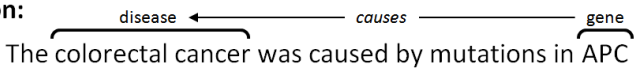
The VERSE method came first in the BB3 event subtask and third in the SeeDev binary subtask in the BioNLP Shared Task 2016. An analysis of the two systems that outperformed VERSE in the SeeDev subtask points to interesting directions for further development. The SeeDev subtask differs greatly from the BB3 subtask as there are 24 relation types compared to only 1 in BB3 and the training set size for each relation is drastically smaller. The LitWay approach, which came first, uses a hybrid approach of rule-based and vector-based (Li et al., 2016). For "simpler" relations, defined using a custom list, a rule-based approach is used using a predefined set of patterns. The UniMelb approach created individual classifiers for each relation type and was able to predict multiple relations for a candidate relation (Panyam et al., 2016). This approach of treating relation types differently suggests that there may be large differences in how a relation should be treated in terms of the linguistic cues used to identify it and the best algorithm approach to identify it.

There are several shortcomings in the approaches to the BioNLP Shared Tasks, the greatest of all is the poor number of participants that provide code. It is also clear that the advantages of some of the most successful tools are tailored specifically to these datasets and may not be able to generalize easily to other relation extraction tasks. Some tools that do share code such as TEES and VERSE have a large number of dependencies, though TEES ameliorates this problem with an excellent installing tool that manages dependencies. These tools can also be computationally costly, with both TEES and VERSE taking a parameter optimization strategy that requires a cluster for reasonable performance.

The biomedical text mining community is endeavoring to improve consistency and ease-of-use for text mining tools. In 2012, the Biocreative BioC Interoperability Initiative (Comeau et al., 2014) encouraged researchers to develop biomedical text mining tools around the BioC file format (Comeau et al., 2013). More recently, one of the Biocreative BeCalm tasks focuses on "technical interoperability and performance of annotation servers" for a named entity recognition systems. This initiative encourages an ecosystem of tools and datasets that will make text mining a more common tool in biology research. PubAnnotation (Kim and Wang, 2012), which is part of this approach, is a public resource for sharing annotated biomedical texts. The hope of this resource is to provide data to improve biomedical text mining tools and as a launching point for future shared tasks. The PubTator tool (Wei et al., 2013b) provides PubMed abstracts with various biomedical entities annotated using several named entity recognition tools including tmVar (Wei et al., 2013a) and DNorm (Leaman et al., 2013).

In order to overcome some of the challenges in the relation extraction community in terms of ease-of-use and integration, we present Kindred. Kindred is an easy-to-install Python package for relation extraction using a vector-based approach. It abstracts away much of the underlying algorithms in order to allow a user to easily start extracting biomedical knowledge from sentences. However, the user can easily use individual components of Kindred in conjunction with other parsers or machine learning algorithms. It integrates seamlessly with PubAnnotation and PubTator to allow easy access to training data and text to be applied to. Furthermore, we show that it per-

Example Relation:



Standoff Format:

```
example.txt:
The colorectal cancer was caused by
mutations in APC
example.a1:
T1  disease 4 21 colorectal cancer
T2  gene 49 52   APC
example.a2:
E1  causes subj:T2 obj:T1
```

BioC Format:

```
<?xml version='1.0' encoding='UTF-8'?><!DOCTYPE
collection SYSTEM 'BioC.dtd'>
<collection>
  <document>
      <offset>0</offset>
      <text>The colorectal cancer was caused by
mutations in APC</text>
      <annotation id="T1">
        <infon key="type">disease</infon>
        <location offset="4" length="17"/>
        <text>colorectal cancer</text>
      </annotation>
      <annotation id="T2">
        <infon key="type">gene</infon>
        <location offset="49" length="3"/>
        <text>APC</text>
      </annotation>
      <relation id="R1">
        <infon key="type">causes</infon>
        <node refid="T2" role="subj"/>
        <node refid="T1" role="obj"/>
      </relation>
    </passage>
  </document>
</collection>
```

JSON Format:

```
{
  "text": "The colorectal cancer was
caused by mutations in APC",
  "denotations":
     [{"id":"T1", "obj":"disease",
       "span":{"begin":4,"end":21}},
      {"id":"T2", "obj":"gene",
       "span":{"begin":49,"end":52}}],
  "relations":
     [{"id":"R1","pred":"causes",
       "subj":"T2", "obj":"T1"}]
}
```

Simple Tag Format:

```
The <disease id="T1">colorectal cancer</disease> was caused by mutations in <gene
id="T2">APC</gene><relation type="causes" subj="T2" obj="T1" />
```

Figure 1: An example of a relation between two entities in the same sentence and the representations of the relation in four input/output formats that Kindred supports.

forms very well on the BioNLP Shared Task 2016 relation subtasks.

2 Methods

Kindred is a Python package that builds upon the Stanford CoreNLP framework (Manning et al., 2014) and the scikit-learn machine learning library (Pedregosa et al., 2011). The decision to build a package was based on the understanding that each text mining problem is different. It seemed more valuable to make the individual features of the relation extraction system available to the community than a bespoke tool that was designed to solve a fixed type of biomedical text mining problem. Python was selected due to the excellent support for machine learning and the easy distribution of Python packages.

The ethos of the design is based on the scikit-learn API that allows complex operations to occur in very few lines of code, but also gives detailed control of the individual components. Individual computational units are encapsulated in separate classes to improve modularity and allow easier testing. Nevertheless, the main goal was to allow the user to download annotated data and build a relation extraction classifier in as few lines of code as possible.

2.1 Package development

The package has been developed for ease-of-use and reliability. The code for the package is hosted on Github. It was also developed using the continuous integration system Travis CI in order to improve the robustness of the tool. This allows regular tests to be run whenever code is committed to the repository. This will enable further development of Kindred and ensure that it continues to work with both Python 2 and Python 3. Coveralls and the Python coverage tool are used to evaluate code coverage and assist in test evaluation.

These approaches were in line with the recent recommendations on improving research software (Taschuk and Wilson, 2017). We hope these techniques will allow for and encourage others to make use of and contribute to the Kindred package.

2.2 Data Formats

As illustrated in Figure 1, Kindred accepts data in four different formats: the standoff format used by BioNLP Shared Tasks, the JSON format used by PubAnnotation, the BioC format (Comeau et al., 2013) and a simple tag format. The standoff format uses three files, a TXT file that contains the raw text, an A1 file that contains information on the tagged entities and an A2 file that contains information on the relations between the entities. The JSON, BioC and simple tag formats integrate this information into single files. The input text in each of these formats must have already been annotated for entities.

The simple tag format was implemented primarily for simple illustrations of Kindred and for easier testing purposes. It is parsed using an XML parser to identify all tags. A relation tag should contain a "type" attribute that denotes the relation type (e.g. causes). All other attributes are assumed to be arguments for the relation and their values should be IDs for entities in the same text. A non-relation tag is assumed to be describing an entity and should have an ID attribute that is used for associating relations.

2.3 Parsing and Candidate Building

The text data is loaded, and where possible, the annotations are checked for validity. In order to prepare the data for classification, the first step is sentence splitting and tokenization. We use the Stanford CoreNLP toolkit for this which is also used for dependency parsing for each sentence.

Once parsing has completed, the associated entity information must then be matched with the corresponding sentences. An entity can contain non-contiguous tokens as was the case for the BB3 event dataset in the BioNLP 2016 Shared Task. Therefore each token that overlaps with an annotation for an entity is linked to that entity.

Any relations that occur entirely within a sentence are associated with that sentence. The decision to focus on relations contained within sentence boundaries is based on the poor performance of relation extraction systems in the past. The VERSE tool explored predicting relations that spanned sentence boundaries in the BioNLP Shared Task and found that the false positive rate was too high. The sentence is also parsed to generate a dependency graph which is stored as a set of triples $(token_i, token_j, dependency_{ij})$ where $dependency_{ij}$ is the type of edge in the dependency graph between tokens i and j. The edge types use the Universal Dependencies format (Nivre et al., 2016).

Relation candidates are then created by finding every possible pair of entities within each sentence. The candidates that are annotated relations are stored with a class number for use in the multiclass classifier. The class zero denotes no relation. All other classes denote relations of specific types. The types of relations and therefore how many classes are required for the multiclass classifier are based on the training data provided to Kindred.

2.4 Vectorization

Each candidate is then vectorized in order to transform the tokenized sentence and set of entity information into a numerical vector that can be processed using the scikit-learn classifiers. In order to keep Kindred simple and improve performance, it only generates a small set of features as outlined below.

- Entity types in the candidate relation
- Unigrams between entities
- Bigrams for the full sentence
- Edges in dependency path
- Edges in dependency path that are next to each entity.

For the entity type and edge relations, they are stored in a one-hot format. For the entity specific relations, features are created for each entity. For instance, if there are three relation types for relations between two arguments, then six binary features would be required to capture the entity types.

The unigrams and bigrams use a bag-of-words approach. Term-frequency inverse-document frequency (TF-IDF) is used for all bag-of-words based features. The dependency path, using the same method as VERSE, is calculated as the minimum spanning tree between the nodes in the dependency graph that are associated with the entities in the candidate relation.

2.5 Classification

Kindred has in-built support for the support vector machine (SVM) and logistic regression classifiers implemented in scikit-learn. By default, the

SVM classifier is used with the vectorized candidate relations. The linear kernel has shown to give good performance and is substantially faster to train than alternative SVM kernels such as radial basis function or exponential.

The success of the LitWay and UniMelb entries to the SeeDev shared task suggested that individual classifiers for unique relation types may give improved performance. This may be due to the significant differences in complexity between different relation types. For instance, one relation type may require information from across the sentence for good classification, whereas another relation type may require only the neighboring word.

Using one classifier per relation type, instead of a single multiclass classifier, means that a relation candidate may be predicted to be multiple relation types. Depending on the dataset, this may be the appropriate decision as relations may overlap. Kindred offers this functionality of one classifier per relation type. However, for the SeeDev dataset, we found that the best performance was actually through a single multiclass classifier.

2.6 Filtering

The predicted set of relations is then filtered using the associated relation type and types of the entities in the relation. Kindred uses the set of relations in the training data to infer the possible argument types for each relation.

2.7 Precision-recall tradeoff

The importance of precision and recall depends on the specific text mining problem. The BioNLP Shared Task has favored the F1-score, giving an equal weighting to precision and recall. Other text mining projects may prefer higher precision in order to avoid biocurators having to manually filter out spurious results. Alternatively, projects may require higher recall in order to not miss any possibly important results. Kindred gives the user the control of a threshold for making predictions. In this case, the logistic regression classifier is used as it allows for easier thresholding. This is because the underlying predicted values can be interpreted as probabilities. We found that logistic regression achieved performance very close to the SVM classifier. By selecting a higher threshold, the classifier will become more conservative, decrease the number of false positives and therefore improve precision at the cost of recall. By using cross-validation, the user can get an idea of the precision-recall tradeoff. The tradeoffs for the BB3 and SeeDev tasks are shown in 2. This allows the user to select the appropriate threshold for their task.

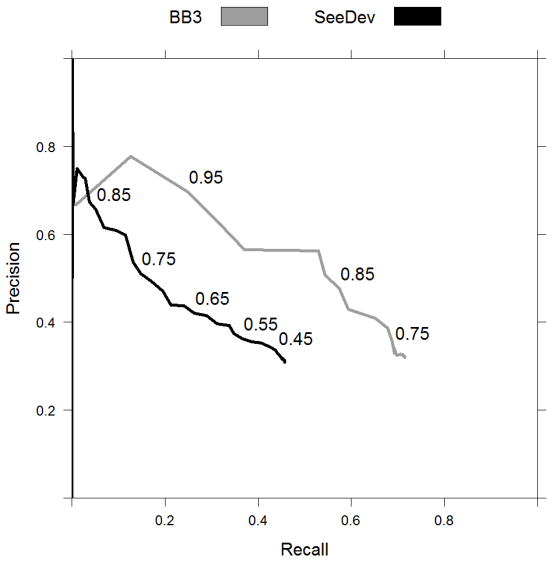


Figure 2: The precision-recall tradeoff when trained on the training set for the BB3 and SeeDev results and evaluating on the development set using different thresholds. The numbers shown on the plot are the thresholds.

2.8 Parameter optimization

TEES took a grid-search approach to parameter optimization and focused on the parameters of the SVM classifier. VERSE had a significantly larger selection of parameters and grid search was not computationally feasible so a stochastic approach was used. Both approaches are computationally expensive and generally need a computer cluster.

Kindred takes a much simpler approach to parameter optimization and can work out of the box with default values. To improve performance, the user can choose to do minor parameter optimization. The only parameter optimized by Kindred is the exact set of features used for classification. This decision was made with the hypothesis that some relations potentially require words from across the sentence and other need only the information from the dependency parse.

The feature choice optimization uses a greedy algorithm. It calculates the F1-score using cross validation for each feature type. It then selects the best one and tries adding the remaining feature types to it. It continues growing the feature

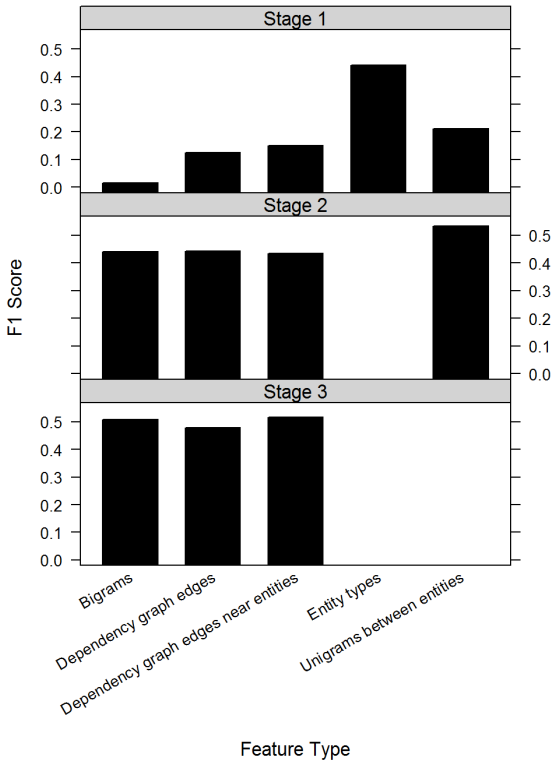


Figure 3: An illustration of the greedy approach to selecting feature types for the BB3 dataset.

set until the cross-validated F1 score does not improve.

Figure 3 illustrates the process for the BB3 subtask using the training set and evaluating on the development set. At the first stage, the entity types feature is selected. This is understandable as the types of entity are highly predictive of whether a candidate relation is reasonable for a particular candidate type, e.g. two gene entities are unlikely to be associated in a 'IS_TREATMENT_FOR' relation. At the next stage, the unigrams between entities feature is selected. And on the third stage, no improvement is made. Hence for this dataset, two features are selected. We use this approach for the BB3 dataset but found that the default feature set performed best for the SeeDev dataset.

2.9 Dependencies

The main dependencies of Kindred are the scikit-learn machine learning library and the Stanford CoreNLP toolkit. Kindred will check for a locally running CoreNLP server and connect if possible. If none is found, then the CoreNLP archive file will be downloaded. After checking the SHA256 checksum to confirm the file integrity, it is extracted. It will then launch CoreNLP as a background process and wait until the toolkit is ready before proceeding to send parse requests to it. It also makes sure to kill the CoreNLP process when the Kindred package exits. Kindred also depends on the wget package for easy downloading of files, the IntervalTree python package for identifying entity spans in text and NetworkX for generating the dependency path (Schult and Swart, 2008).

2.10 PubAnnotation integration

In order to make use of existing resources in the biomedical text mining community, Kindred integrates with PubAnnotation. This allows annotated text to be downloaded from PubAnnotation and used to train classifiers.

The PubAnnotation platform provides a RESTful API that allows easy download of annotations from a given project. Kindred will initially download the listing of all available text sources with annotation for a given project. The listing is provided as a JSON data file. It will then download the complete set of texts with annotations.

2.11 PubTator integration

Kindred can also download a set of annotated PubMed abstracts that have already been annotated with named entities through the PubTator framework using the RESTful API. This requires the user to provide a set of PubMed IDs which are then requested from the PubTator server using the JSON data format. The same loader used for PubAnnotation data is then used for the PubTator data.

2.12 BioNLP Shared Task integration

Kindred gives easy access to the data from the most recent BioNLP Shared Task. By providing the name of the test and specific data set (e.g. training, development or testing), Kindred manages the download of the appropriate archive, unzipping and loading of the data. As with the CoreNLP dependency, the SHA256 checksum of the downloaded archive is checked before unzipping occurs.

2.13 API

One of the main goals of Kindred is to open up the internal functionality of a relation extraction system to other developers. The authors are keenly aware that their specific interest in relation extraction, in order to build knowledge bases related to cancer, differs from other researchers. With this

	Precision	Recall	F1 Score
Fold 1	0.319	0.715	0.441
Fold 2	0.460	0.684	0.550
Test Set	0.579	0.443	0.502
VERSE	0.510	0.615	0.558

Table 1: Cross-validated results (Fold1/Fold2) and final test set results for Kindred predictions in Bacteria Biotope (BB3) event subtask with test set results for the top performing tool VERSE.

	Precision	Recall	F1 Score
Fold 1	0.333	0.411	0.368
Fold 2	0.255	0.393	0.309
Test Set	0.344	0.479	0.400
LitWay	0.417	0.448	0.432

Table 2: Cross-validated results (Fold1/Fold2) and final test set results for Kindred predictions in Seed Development (SeeDev) binary subtask with test set results for the top performing tool LitWay.

in mind, the API is designed to give easy access to the different modules of Kindred that may be used independently. For instance, the candidate builder or vectorizer could easily be integrated with functionality from other Python packages, which would allow for other machine learning algorithms or deep learning techniques to be tested. Other parsers could easily be integrated and tested with the other parts of the Kindred in order to understand how the parser performance affects the overall performance of the system. We hope that this ease-of-use will encourage others to use Kindred as a baseline method for comparison in future research.

3 Results and Discussion

In order to show the efficacy of Kindred, we evaluate the performance on the BioNLP 2016 Shared Task data for the BB3 event extraction subtask and the SeeDev binary relation subtask. Parameter optimization was used for BB3 subtask but not for the SeeDev subtask which used the default set of feature types. Both tasks used a single multiclass classifier. Tables 1 and 2 shows both the cross-validated results using the provided training/development split as well as the final results for the test set.

The results are in line with the best performing tools in the shared task. It is to be expected that it does not achieve the best score in either task. VERSE, which achieved the best score in the BB3 subtask, utilized a computational cluster to test out different parameter settings for vectorization as well as classification. LitWay, the winner of the SeeDev subtask, used hand-crafted rules for a number of the relation types. Given the computational speed and simplicity of the system, Kindred is a valuable contribution to the community.

These results suggest several possible extensions of Kindred. Firstly, a hybrid system that mixes a vector-based classifier with some hand-crafted rules may improve system performance. This would need to be implemented to allow customization in order to support different biomedical tasks. Kindred is also geared towards PubMed abstract text, especially given the integration with PubTator. Using PubTator's API to annotate other text would allow Kindred to easily integrate other text sources, including full-text articles where possible. Given the open nature of the API, we hope that these improvements, if desired by the community, could be easily developed and tested.

Kindred has several weaknesses that we hope to improve. It does not properly handle entities that lie within tokens. For example, a token "HER2+", with "HER" annotated as a gene name, denotes a breast cancer subtype that is positive for the HER2 receptor. Kindred will currently associate the full token as a gene entity and will not properly deal the "+". This is not a concern for the BioNLP Shared Task problem but may become important in other text mining tasks.

4 Conclusion

We have presented the Kindred relation extraction package. It is designed for ease-of-use to encourage more researchers to test out relation extraction in their research. By integrating a selection of file formats and connecting to a set of existing resources including PubAnnotation and PubTator, Kindred will make the first steps for a researcher must less cumbersome. We also hope that the codebase will allow researchers to build upon the methods to make further improvements in relation extraction research.

Acknowledgments

This research was supported by a Vanier Canada Graduate Scholarship. The authors would like to thank Compute Canada 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*. Association for Computational Linguistics, pages 104–108.

Donald C Comeau, Riza Theresa Batista-Navarro, Hong-Jie Dai, Rezarta Islamaj Doğan, Antonio Jimeno Yepes, Ritu Khare, Zhiyong Lu, Hernani Marques, Carolyn J Mattingly, Mariana Neves, et al. 2014. Bioc interoperability track overview. *Database* 2014.

Donald C Comeau, Rezarta Islamaj Doğan, Paolo Ciccarese, Kevin Bretonnel Cohen, Martin Krallinger, Florian Leitner, Zhiyong Lu, Yifan Peng, Fabio Rinaldi, Manabu Torii, et al. 2013. Bioc: a minimalist approach to interoperability for biomedical text processing. *Database* 2013.

Ian Donaldson, Joel Martin, Berry De Bruijn, Cheryl Wolting, Vicki Lay, Brigitte Tuekam, Shudong Zhang, Berivan Baskin, Gary D Bader, Katerina Michalickova, et al. 2003. PreBIND and Textomy–mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics* 4(1):11.

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*. Association for Computational Linguistics, pages 202–205.

Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. 2013. DNorm: disease name normalization with pairwise learning to rank. *Bioinformatics* .

Robert Leaman, Graciela Gonzalez, et al. 2008. BANNER: an executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*. volume 13, pages 652–663.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Jake Lever and Steven JM Jones. 2016. VERSE: Event and Relation Extraction in the BioNLP 2016 Shared Task. *Proceedings of the 4th BioNLP Shared Task Workshop* page 42.

Chen Li, Zhiqiang Rao, and Xiangrong Zhang. 2016. LitWay, Discriminative Extraction for Different BioEvents. *Proceedings of the 4th BioNLP Shared Task Workshop* page 32.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep Learning with Minimal Training Data: TurkuNLP Entry in the BioNLP Shared Task 2016. *Proceedings of the 4th BioNLP Shared Task Workshop* page 73.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666.

Nagesh C Panyam, Gitansh Khirbat, Karin Verspoor, Trevor Cohn, and Kotagiri Ramamohanarao. 2016. SeeDev Binary Event Extraction using SVMs and a Rich Feature Set. *Proceedings of the 4th BioNLP Shared Task Workshop* page 82.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Dietrich Rebholz-Schuhmann, Harald Kirsch, Miguel Arregui, Sylvain Gaudan, Mark Riethoven, and Peter Stoehr. 2007. EBIMedtext crunching to gather facts for proteins from Medline. *Bioinformatics* 23(2):e237–e244.

Daniel A Schult and P Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*. volume 2008, pages 11–16.

Morgan Taschuk and Greg Wilson. 2017. Ten Simple Rules for Making Research Software More Robust. *PLOS Computational Biology* 13(4).

Chih-Hsuan Wei, Bethany R Harris, Hung-Yu Kao, and Zhiyong Lu. 2013a. tmVar: a text mining approach for extracting sequence variants in biomedical literature. *Bioinformatics* .

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013b. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic acids research* .