Extracting Biomedical Events and Modifications Using Subgraph Matching with Noisy Training Data

Andrew MacKinlay[♢], **David Martinez**[♢], **Antonio Jimeno Yepes**[♢], **Haibin Liu**[♠], **W. John Wilbur**[♠] **and Karin Verspoor**[♢]
[♢] NICTA Victoria Research Laboratory, University of Melbourne, Australia
{andrew.mackinlay, david.martinez}@nicta.com.au
{antonio.jimeno, karin.verspoor}@nicta.com.au
[♠] National Center for Biotechnology Information, Bethesda, MD, USA
haibin.liu@nih.gov, wilbur@ncbi.nlm.nih.gov

Abstract

The Genia Event (GE) extraction task of the BioNLP Shared Task addresses the extraction of biomedical events from the natural language text of the published literature. In our submission, we modified an existing system for learning of event patterns via dependency parse subgraphs to utilise a more accurate parser and significantly more, but noisier, training data. We explore the impact of these two aspects of the system and conclude that the change in parser limits recall to an extent that cannot be offset by the large quantities of training data. However, our extensions of the system to extract modification events shows promise.

1 Introduction

In this paper, we describe our submission to the Genia Event (GE) information extraction subtask of the BioNLP Shared Task. This task requires the development of systems that are capable of identifying bio-molecular events as those events are expressed in full-text publications. The task represents an important contribution to the broader problem of converting unstructured information captured in the biomedical literature into structured information that can be used to index and analyse bio-molecular relationships.

This year's task builds on previous instantiations of this task (Kim et al., 2009; Kim et al., 2012), with only minor changes in the task definition introduced for 2011. The task organisers provided full text publications annotated with mentions of biological entities including proteins and genes, and asked participants to provide annotations of simple events including gene expression, binding, localization, and protein modification, as well as higher-order regulation events (e.g., positive regulation of gene expression). In our submission, we built on a system originally developed for the BioNLP-ST 2011 (Liu et al., 2011) and extended in more recent work (Liu et al., 2013a; Liu et al., 2013b). This system learns to recognise subgraphs of syntactic dependency parse graphs that express a given bio-molecular event, and matches those subgraphs to new text using an algorithm called Approximate Subgraph Matching.

Due to the method's fundamental dependency on the syntactic dependency parse of the text, in this work we set out to explore the impact of substituting the previously employed dependency parsers with a different parser which has been demonstrated to achieve higher performance than other commonly used parsers for full-text biomedical literature (Verspoor et al., 2012).

In addition, we aimed to address the relatively lower recall of the method through incorporation of large quantities of external training data, acquired through integration of previously automatically extracted bio-molecular events available in a web repository of such extracted events, EVEX (Van Landeghem et al., 2011; Van Landeghem et al., 2012), and additional bio-molecular events generated from a large sample of full text publications using one of the state-of-the-art event extraction systems, TEES (Björne and Salakoski, 2011). Since the performance of the subgraph matching method, as an instance-based learning strategy (Alpaydin, 2004), is dependent on having good training examples that express the events in a range of syntactic structures, the motivation underlying this was to increase the amount of training data available to the system, even if that data was derived from a less-than-perfect source. The augmentation of training corpora with external unlabelled data that is automatically processed to generate additional labels has been explored for retraining the same system, in an approach known as *self-training*. This approach has been shown to be

Proceedings of the BioNLP Shared Task 2013 Workshop, pages 35–44,
Sofia, Bulgaria, August 9 2013.

very effective for improving parsing performance (McClosky et al., 2006; McClosky and Charniak, 2008). Self-training of the TEES system has been previously explored (Bjorne et al., 2012), with somewhat mixed results, but with evidence suggesting it could be useful with an appropriate strategy for selecting training examples. Here, rather than training our system with its own output over external data, we explore a semi-supervised learning approach in which we train our system with the outputs of a different system (TEES) over external data.

2 Methodology

2.1 Base Event Extraction System

The event extraction algorithm is essentially the same as the one used in Liu et al. (2013b). A fuller description can be found there, but we summarise the most important aspects of it here.

2.1.1 Event Extraction with ASM

The principal method used in event extraction is Approximate Subgraph Matching, or ASM (Liu et al., 2013a). Broadly, we learn subgraph patterns from the event structures in the training data, and then apply them by looking for matches with the patterns of the learned rules, using ASM to allow for non-exact matches of the patterns.

The first stage in this is learning the rules which link subgraphs to associated patterns. The input is a set of dependency-parsed articles (the setup is described in §2.1.2), and a set of gold-standard annotations of proteins and events in the shared task format. Using the standoff annotations in the training data, every protein and trigger is mapped to one or more nodes in the corresponding dependency graphs. In addition, the textual content of every protein is replaced with a generic string enabling abstraction over individual protein names. Then, for each event annotation in the training data, we retrieve the nodes from the graph corresponding to the associated trigger and protein entities. We determine the shortest path (or paths, in case of a tie) connecting the graph trigger to each of the event argument nodes. For arguments which are themselves events (e.g., for regulatory events), the node corresponding to the trigger of the event argument is used instead of a protein node. Where there are multiple arguments, we take the union of the shortest paths to each individual argument.

This path is then used as the pattern component of an event rule. The rule also consists of an event type, and a mapping from event arguments to nodes from the pattern graph, or to an event type/node pair for nested event arguments. After processing all training documents, we get on the order of a few thousand rules; this can be decreased slightly by removing rules with subgraphs that are isomorphic to those of other rules.

In principle, this set of rules could then be directly applied to the test documents, by searching for any matching subgraphs. However, in practice doing so leads to very low recall, since the patterns are not general enough to get a broad range of matches on new data. We can alleviate this by relaxing the strictness of the subgraph matching process. Most basically, we relax node matching. Instead of requiring an exact match between both the token and the part-of-speech of the nodes of the sentence graph and those from the rule subgraph, we also allow a match on the basis of the lemma (according to BioLemmatizer (Liu et al., 2012)), and a coarse-grained POS-tag (where there is only one POS-tag for nouns, verbs and adjectives).

More importantly, we also relax the requirements on how closely the graphs must match, by using ASM. ASM defines distances measures between subgraphs, based on structure, edge labels and edge directions, and uses a set of specified weights to combine them into an overall subgraph distance. We have a pre-configured set of distance thresholds for each event type, and for each sentence/rule pairing, we extract events for any rules with subgraphs under the given threshold.

The problem with this approximate matching is that some rules now match too broadly, and precision is reduced. This is mitigated by adding an iterative optimisation phase. In each iteration, we run the event extraction using the current rule set over some dataset – usually the training set, or a subset of it. We check the contribution of each rule in terms of postulated events and actual events which match the gold standard. If the ratio of matched to postulated events is too low (for the work reported here, the threshold is 0.25), the rule is discarded. This process is repeated until no more rules are discarded. This can take multiple iterations since the rules are interdependent due to the presence of nested event arguments.

The optimisation step is by far the most time-consuming step of our process, especially for the large rule sets produced in some configurations.

We were able to improve optimisation times somewhat by parallelising the event extraction, and temporarily removing documents with long extraction times from the optimisation process until as late as possible, but it remained the primary bottleneck in our experimentation.

2.1.2 Parsing Pipeline

In our parsing pipeline, we first split sentences using the JULIE Sentence Boundary Detector, or JSBD (Tomanek et al., 2007). We then parse using a version of clearnlp[1] (Choi and McCallum, 2013), a successor to ClearParser (Choi and Palmer, 2011), which was shown to have state-of-the-art performance over the CRAFT corpus of full-text biomedical articles (Verspoor et al., 2012). We use dependency and POS-tagging models trained on the CRAFT corpus (except where noted); these pre-trained models are provided with clearnlp. Our fork of clearnlp integrates token span marking into the parsing process, so the dependency nodes can easily be matched to the standoff annotations provided with the shared task data. This pipeline is not dependent on any pre-annotated data, so can thus be trivially applied to extra data not provided as part of the shared task. In addition the parsing is fast, requiring roughly 46 wall-clock seconds (processing serially) to parse the 5059 sentences from the training and development sets of the 2013 GE task – an average of 9 ms per sentence. The ability to apply the same parsing configuration to new text was useful for adding extra training data, as discussed in §2.2.

The usage of clearnlp as the parser is the primary point of difference between our system and that of Liu et al. (2013b), who use the Charniak-Johnson parser with the McClosky biomedical model (CJM; McClosky and Charniak (2008)), although there are other minor differences in tokenisation and sentence splitting. We expected that the higher accuracy of clearnlp over biomedical text would translate into increased accuracy of event detection in the shared task; we consider this question in some detail below.

2.2 Adding Noisy Training Data

One of the limitations of the ASM approach is that the high precision comes at the cost of lower recall. Our hypothesis is that adding extra training instances, even if some are errors, will raise recall and improve overall performance. We utilised two sources of automatically-annotated data: the EVEX database, and running an automatic event annotator over documents from PubMed Central (PMC) and MEDLINE.

To test our hypothesis, we utilise one of the best performing automatic event extractors in previous BioNLP tasks: TEES (Turku Event Extraction System)[2] (Björne et al., 2011). We expand our pool of training examples by adding the highest-confidence events TEES identifies in unlabelled text. We explored different approaches to ranking events based on classifier confidence empirically.

TEES relies on multi-class SVMs both for trigger and event classification, and produces confidence scores for each prediction. We explored ranking events according to: (i) score of the trigger prediction, (ii) score of the event-type prediction, and (iii) sum of trigger and event type predictions. We also compared the performance when selecting the top-k events overall, versus choosing the top-k events for each event type. We also tested adding as many instances per event-type as there were in the manually-annotated dataset, with different multiplying factors. Finally, we evaluated the effect of using different splits of the data for the evaluation and optimisation steps of ASM. This is the full list of parameters that we tested over held-out data:

- Original confidence scores: we ranked events according to the three SVM scores mentioned above: trigger prediction, event-type prediction, and combined.
- Overall top-k: we selected the top 1,000, 5,000, 10,000, 20,000, 30,000, 40,000, and 50,000 for the different experimental runs.
- Top-k per type: for each event type, we selected the top 400, 1,000, and 2,000.
- Training bias per type: we add as many instances from EVEX per type as there are in the manually annotated data. We experiment with adding up to 6 times as many as in manually annotated data.
- Training/optimisation split: we combine manually and automatically annotated data for training. For optimisation we tested different options: manually annotated only, manual + automatic, manual + top-100 events, and manual + top-1000 events.

[1] https://code.google.com/p/clearnlp/

[2] http://jbjorne.github.com/TEES/

We did not explore all these settings exhaustively due to time constraints, and we report here the most promising settings. It is worth mentioning that most of the configurations contributed to improve the baseline performance. We only observed drops when using automatically-annotated data in the optimisation step.

2.2.1 Data from EVEX

Conveniently, the developers of TEES have released the output of their tool over the full 2009 collection of MEDLINE, consisting of abstracts of biomedical articles, in a collection known as the EVEX dataset. We used the full EVEX dataset as provided by the University of Turku, and explored different ways of ranking the full list of events as described above.

2.2.2 Data from TEES

To augment the training data, we annotated two data sets with TEES based on MEDLINE and PubMed Central (PMC). The developers of TEES released a trained model for the GE 2013 training data that we utilised.

Due to the long pre-processing time of TEES, which includes gene named entity recognition, part-of-speech tagging and parsing, we used the EVEX pre-processed MEDLINE, which required some adaptation of the EVEX XML to the XML format accepted by TEES. Once this adaptation was finished, the files were processed by TEES.

Then, we have selected articles from PMC using a query containing specific MeSH headings related to the GE task and limiting the result to only the Open Access part of PMC. From the almost 600k articles from the PMC Open Access set, we reduced the total number of articles to around 155k. The PMC query is the following:

> *(Genetic Phenomena[MH] OR Metabolic Phenomena[MH] OR Cell Physiological Phenomena[MH] OR Biochemical Processes[MH]) AND open access[filter]*

Furthermore, the articles were split into sections and specific sections from the full text like *Introduction*, *Background* and *Methods* were removed to reduce the quantity of text to be annotated by TEES. The PMC files produced by this filtering were processed by TEES on the NICTA cluster.

2.3 Modification Detection

To evaluate the utility of ASM for a diverse range of tasks, we also applied it to the task of detecting modification (SPECULATION or NEGATION) of events. In event detection, triggers are explicitly annotated, so the linguistic cue which indicates that an event is occurring is easy to identify. As described in Section 3.2, these triggers are important for learning event patterns.

NEGATION cues

- **Basic:** *not*, *no*, *never*, *nor*, *only*, *neither*, *fail*, *cease*, *stop*, *terminate*, *end*, *lacking*, *missing*, *absent*, *absence*, *failure*, *negative*, *unlikely*, *without*, *lack*, *unable*
- **Data-derived:** *any*, *prevention*, *prevent*, *disrupt*, *disruption*

SPECULATION cues:

- **Basic:** *analysis*, *whether*, *may*, *should*, *can*, *could*, *uncertain*, *questionable*, *possible*, *likely*, *probable*, *probably*, *possibly*, *conceivable*, *conceivably*, *perhaps*, *address*, *analyze*, *analyse*, *assess*, *ask*, *compare*, *consider*, *enquire*, *evaluate*, *examine*, *experiment*, *explore*, *investigate*, *test*, *research*, *study*, *speculate*
- **Data-derived:** *measure*, *measurement*, *suggest*, *suggestion*, *value*, *quantify*, *quantification*, *determine*, *determination*, *detect*, *detection*, *calculate*, *calculation*

Table 1: Modification cues

The event extraction method is based on paths between dependency graph nodes, so it is necessary to have at least two relevant graph nodes before we can determine a path between them. For learning modification rules, one graph node is the trigger of the event which is subjec to modification. However here we needed a method to determine another node in the sentence which provided evidence that NEGATION or SPECULATION was occurring, and could thus form an endpoint for a semantically relevant graph pattern. To achieve this, we specified a set *cue lemmas* for NEGATION and SPECULATION. The basic set of cue lemmas came from a variety of sources. Some were manually specified and some were derived from previous work on modification detection (Cohen et al., 2011; MacKinlay et al., 2012). We manually expanded this cue list to include obvious derivational variants. This gave us a basic set of 34 SPECULATION and 21 NEGATION cues.

We also used a data-driven strategy to find additional lemmas indicative of modification. We adapted the method of Rayson and Garside (2000) which uses log-likelihood for finding words that characterise differences between corpora. Here, the "corpora" are sentences attached to all events in the training set, and sentences attached to events which are subject to NEGATION or SPECULATION (treated separately). We build a frequency distribution over lemmas in each set of sentences, and calculate the log-likelihood for all lemmas, us-

ing the observed frequency from the modification events and the expected frequency over all events. Sorting by decreasing log-likelihood, we get a list of lemmas which are most strongly associated with NEGATION or SPECULATION. We manually examined the highest-ranked lemmas from these two lists and noted lemmas which may occur, according to human judgment, in phrases which would denote the relevant modification type. We found seven extra SPECULATION cues and three extra NEGATION cues. Expanding with morphological variants as described above yielded 47 SPECULATION cues and 26 NEGATION cues total. These cues are shown, divided into basic and data-derived, in Table 1.

For every node N with a lemma in the appropriate set of cue lemmas, we create a rule based on the shortest path between the cue lemma node N and the event trigger node. The trigger lemmas are replaced with generic lemmas which only reflect the POS-tag of the trigger, to broaden the range of possible matches. Each rule thus consists of the POS-tag of an event trigger, and a subgraph pattern including the abstracted event trigger node.

At modification detection time, the rules are applied in a similar way to the event rules. After detecting events, we look for matches of each extracted event with every modification rule. A rule R is considered to match if the event trigger node POS tag matches the POS tag of the rule, and the subgraph pattern of the rule matches the graph of the sentence, including a node corresponding to the event trigger node. If R is found to match for a given event and sentence, any events which have the trigger defined in the rule are marked as SPECULATION or NEGATION as appropriate. As in event extraction, we use ASM to allow a looser match between graphs, but initial experimentation showed that increasing the match thresholds beyond a relatively small distance was detrimental. We have not yet added an optimisation phase for modification, which might allow larger ASM distance threshold to have more benefit.

3 Results

We present our results over development data, and the official test. We report the Approximate Span/Approximate Recursive metric in all our tables, for easy comparison of scores. We describe the data split used for development, explain our event extraction results, and finally describe our performance in modification detection.

3.1 Data division for development

In the data provided by the task organisers, the split of data between training and development sets, with 249 and 222 article sections respectively, was fairly even. If we had used such a split, we would have had an unfeasibly small amount of data to train from during development, and possible unexpected effects when we sharply increased the amount of training data for running over the held-out test set. We instead used our own data set split during development, pooling the provided training and development sets, and randomly selecting six PMC articles (PMC IDs 2626671, 2674207, 3062687, 3148254, 3333881 and 3359311) for the development set, with the remainder available for training. We respected article boundaries in the new split to avoid training and testing on sentences taken from different sections of the same article. Results over the development set reported in this section are over this data split. We will refer to our training subset as GE13tr, and to the testing subset as GE13dev.

For our runs over the official test of this challenge, we merged all the manually annotated data from 2013 to be used as training. We also performed some experiments with adding the examples from the 2011 GE task to our training data.

3.2 Event Extraction

For our first experiment, we evaluated the contribution of the automatically annotated data over using GE13tr data only. We performed a set of experiments to explore the parameters described in Section 2.2 over two sources of extra examples: EVEX and TEES.

Using EVEX data in training resulted in clear improvements in performance when only manually annotated data was consulted for optimisation. The increase was mainly due to the better recall, with small variations in precision over the baseline for the majority of experiments. Our best run over the GE13dev data followed this setting: rank events according to trigger scores, include all top-30000 events (without considering the types of the events), and use only manually annotated data for the optimisation step. Other settings also performed well, as we will see below.

For TEES, we selected noisy examples from MEDLINE and PMC to be used as additional

System	Prec.	Rec.	F-sc.
GE13tr	60.40	27.02	37.34
+TEES	59.27	29.89	39.74
+TEES +EVEX (top5k)	46.93	30.78	37.18
+TEES +EVEX (top20k)	56.32	31.90	40.73
+TEES +EVEX (top30k)	55.34	**32.48**	**40.93**
+TEES +EVEX (pt1k)	**58.54**	30.96	40.50
+TEES +EVEX (trx4)	57.83	31.23	40.56

Table 2: Impact of adding extra training data to the ASM method. top5k,20k,30k: using the top 5,000, 20,000, and 30,000 events. pt1k: using the top 1,000 events per event-type. trx4: following the training bias of events, with a multiplying factor of four. For TEES we always use the top 10,000 events. Evaluated over GE13dev.

training data. Initial results showed that when using only MEDLINE annotated data in the training step, the performance decreased compared to not using any additional data. This might have been due to differences between the EVEX preprocessed data that we used and what TEES was expecting, so the MEDLINE set was not considered for further experimentation. Using PMC articles annotated with TEES in the training step selected by the evidence score of TEES shows an increase of recall while slightly decreasing the precision, which was expected. We selected the top 10000 events from the PMC set based on the evidence score as additional training data.

Table 2 summarises the results of combining different settings of EVEX with TEES. We achieve a considerable boost in recall, at the cost of precision for most configurations. The only setting where there is a slight drop in F-score is the experiment with only 5000 events from EVEX; in the remaining runs we are able to alleviate the drop in precision, and improve the F-score. Considering the addition of top-events according to their type, the increment in recall is slightly lower, but these runs are able to reach similar F-score to the best ones, using less training data. Results with TEES might be slightly overoptimistic since the PMC annotation is based on a TEES model trained on the 2013 GE data and our configurations are evaluated on a subset of this data.

For our next experiment, we tested the contribution of adding the dataset from the 2011 GE task to the training dataset. We use this data both in the training and optimisation steps. The results are

Train	Prec.	Rec.	F-sc.
GE13tr	**60.40**	27.02	37.34
+GE11	53.41	**32.62**	**40.50**

Table 3: Adding GE11 data to the training and optimisation steps. Evaluated over GE13dev.

Parser	Train	Prec.	Rec.	F-sc.
clearnlp	GE13	60.40	27.02	37.34
	+GE11	53.41	32.62	40.50
CJM	GE13	60.96	33.11	42.91
	+GE11	**64.11**	**38.93**	**48.44**

Table 4: Performance depending on the applied parsing pipeline (clearnlp for this work against the CJM pipeline of Liu et al. (2013b)) over GE13dev. For each run, the available data was used both in training and optimisation.

given in Table 3, where we can observe a boost in recall at the cost of precision. Overall, the improved F-score suggests that this dataset would make a useful contribution to the system.

We also compared our system to that of Liu et al. (2013b), where the primary difference (although not the only difference, as noted in §2.1.2) is the use of clearnlp instead of the CJM (Charniak-Johnson/McClosky) pipeline. It is thus somewhat surprising to see in Table 4 that the CJM pipeline outperforms our clearnlp pipeline by 5.5–8% in F-score, depending on the training data. For the smaller GE13-only training set, the gap is smaller, and the precision figures are in fact comparable. However, the recall is uniformly lower, suggesting that the rules learned from clearnlp parses are for some reason less generally applicable. Another interesting difference is that our clearnlp pipeline gets a smaller benefit from the addition of the GE11 training data. We consider possible reasons for this in §4.1.

Table 5 contains the evaluation of different experiments on the official test data. We tested the baseline system using the training and development data from 2011 and 2013 GE tasks and the addition of TEES and EVEX data. The additional data improves the recall slightly compared to not using it, while, as expected, it decreases the precision. Table 5 also shows the results for our official submission (+TEES+EVEX sub), which due to time constraints was a combination of the optimised rules of different data splits and has a lower

Train	Prec.	Rec.	F-sc.
GE11, GE13	**65.71**	32.57	43.55
+TEES+EVEX	63.67	33.50	**43.91**
+TEES+EVEX *	50.68	**36.99**	42.77

Table 5: Test set results, always optimised over gold data only. * denotes the official submission.

performance compared to the other results.

3.3 Modification Detection

We show results for selected modification detection experiments in Table 6. In all cases we used all of the available gold training data from the GE11 and GE13 datasets. To assess the impact of modification cues, we show results using the basic set as well as with the addition of the data-derived cues. It has often been noted (MacKinlay et al., 2012; Cohen et al., 2011) that modification detection accuracy is strongly dependent on the quality of the upstream event annotation, so we provide an oracle evaluation, using gold-standard event annotations rather than automatic output.

The performance over the automatically-annotated runs is respectable, given that the recall is fundamentally limited by the recall of the input event annotations, which is only around 30% for the configurations shown. With the oracle event annotations, the results improve substantially, with considerable gains in precision, and recall increasing by a factor of 4–6. This boost in recall in particular is more than we would naively expect from the roughly threefold increase in recall over the events. It seems that many of the modification rules we learned were even more effective over events which our pipeline was unable to detect. The modification rules were learned from oracle event data, but this does not fully explain the discrepancy. Regardless, our algorithm for modification detection showed excellent performance over the oracle annotations. Over the 2009 version of the BioNLP shared task data, MacKinlay et al. (2012) report F-scores of 54.6% for NEGATION and 51.7% for SPECULATION. These are not directly comparable with those in Table 6, but running our newer algorithm over the same 2009 data gives F-scores of 84.2% for NEGATION and 69.1% for SPECULATION.

For the official run, which conflates event extraction and modification detection accuracy, our system was ranked third for NEGATION and SPECULATION out of the three competing teams, although the other teams had event extraction F-scores of roughly 8% higher than our system. For SPECULATION, our system had the highest precision of 34.15%, while the F-score of 20.22% was close to the best result of 23.92%. Our NEGATION detection was less competitive, with an F-score of 20.94% – roughly 6% lower than the other teams. We cannot extrapolate directly from the oracle evaluation in Table 6, but it seems to indicate that an increase in event extraction accuracy would have flow-on benefits in modification detection.

4 Discussion

4.1 Detrimental Effects of Parser Choice

The biggest surprise here was that clearnlp, a more accurate dependency parser for the biomedical domain, as evaluated on the CRAFT treebank, gave a substantially lower event extraction F-score than the CJM parser. To determine whether preprocessing caused the differences, we replaced the existing modules (sentence-splitting from JSBD and tokenisation/POS-tagging from clearnlp) with the BioC-derived versions from the CJM pipeline, but this yielded only an insignificant decrease in accuracy.

Over the same training data, the optimised rules from CJM have an average of 2.6 nodes per subgraph path, compared to 3.9 nodes per path using clearnlp. A longer path is less likely to match than a shorter path, so this may help to explain the lower generalisability of the clearnlp-derived rules. While it is possible for a longer subgraph to match just as generally, if the test sentences are parsed consistently, in general there are more nodes and edges which can fail to match due to minor surface variations. One way to mitigate this is to raise the ASM distance thresholds to compensate for this; preliminary experiments suggest it would provide a small ($\sim 1\%$) boost in F-score but this would not close the gap between the parsers.

Both parsers produce outputs with Stanford Dependency labels (de Marneffe and Manning, 2008), so we might naively expect similar graph topology and subgraph pattern lengths. However, the CJM pipeline produces graphs in the "CCprocessed" SD format, which are simpler and denser. If a node N has a link to a node O with a conjunction link to another node P (from e.g. *and*), an extra link with the same label is added directly from N to P in the CCprocessed format. This means

Eval	Events (F-sc)	Cues	NEGATION P / R / F	SPECULATION P / R / F
Dev	GE13+TEES+EVEX (40.93)	Basic	32.69 / 13.71 / 19.32	37.04 / 14.49 / 20.83
	GE13+TEES+EVEX (40.93)	B + Data	32.69 / 12.88 / 18.48	39.71 / 17.20 / 24.00
	Oracle (100.0)	B + Data	82.48 / 71.07 / 76.35	78.79 / 67.71 / 72.83
Test	GE11+GE13 (43.55)	B + Data	39.53 / 13.99 / 20.66	50.00 / 13.85 / 21.69
	GE11+GE13+TEES+EVEX * (42.77)	B + Data	32.76 / 15.38 / 20.94	34.15 / 14.36 / 20.22

Table 6: Results for SPECULATION and NEGATION using automatically-annotated events (showing the F-score of the configuration), as well as using oracle event annotations from the gold standard, over our development set and the official test set. Rules are learned from GE13+GE11 gold data (excluding any test data). Cues for learning rules are either the basic manually-specified set (34 SPEC/21 NEG) or the augmented set with data-driven additions (47 SPEC/26 NEG). * denotes the official submission.

there are more direct links in the graph, matching the semantics more closely. The shortest path from N to P is now direct, instead of via O, which could enable the CJM pipeline to produce more general rules.

To evaluate how much this detrimentally affects the clearnlp pipeline, as a *post hoc* investigation, we implemented a conversion module. Using Stanford Dependency parser code, we replicated the CCprocessed conversion on the clearnlp graphs, reducing the average subgraph pattern length to 2.8, and slightly improving accuracy. Over our development set, compared to the results in Table 3 it gave a 0.7% absolute F-score boost over using GE13 training-data only, and 1.1% over using GE11 and GE13 training data (in both cases improving recall). Over the test set, the improvement was greater, with a P/R/F of 35.66/64.99/46.05, a 2.5% increase in F-score compared to the results in Table 5 and only 2.9% less than the official Liu et al. (2012) submission.

Clearly some of the inter-parser discrepancies are due to surface features and post-processing, and as noted above, we can also achieve small improvements by relaxing ASM thresholds, so some problems may be caused by the default parameters being suboptimal for the parser. However, the accuracy is still lower where we would expect it to be higher, and this remaining discrepancy is difficult to explain without performing a detailed error analysis, which we leave for future work.

4.2 Effect of additional data

Our initial intuition that using additional noisy training data during the training of the system would improve the performance is supported by the results in Table 2. Table 3 shows that using a larger set of manually annotated data based on 2011 GE task data also improves performance. However, these tables also indicate that adding manually annotated data produces an increase in performance comparable to adding the noisy data, despite its smaller size, and when using this manually annotated set together with the noisy data, the improvement resulting from the noisy data is smaller (Table 5). Noisy data was only used during training, which limits its effectiveness—any rule extracted from automatically acquired annotations that are not seen during optimisation of the rule set will have a lower weight. On the other hand, we found that using noisy data for optimisation seemed to decrease performance. Together, these results suggest that studying strategies, possibly self-training, for selection of events from the noisy data to be used during rule set optimisation in the ASM method are warranted.

5 Conclusion

Using additional training data, whether manually annotated or noisy, improves the performance of our baseline event extraction system. The gains that we achieved by adding training data, however, were outweighed by a loss of performance due to our parser substitution, with longer dependency subgraphs limiting rule generalisability the most likely explanation. Our experiments demonstrate that while a given parser might be 'better' in one evaluation context, that advantage may not translate to improved performance in a downstream task that depends strongly on the parser output. We presented an extension of the subgraph matching methodology to extract modification events which, when based on a good core event extraction system, shows very promising results.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. This research was supported in part by the Intramural Research Program of the NIH, NLM.

References

Ethem Alpaydin. 2004. *Introduction to Machine Learning*. MIT Press.

Jari Björne and T. Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 183–191.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2011. Extracting contextualized complex biological events with rich graph-based features sets. *Computational Intelligence*, 27(4):541–557.

Jari Bjorne, Filip Ginter, and Tapio Salakoski. 2012. University of turku in the bionlp'11 shared task. *BMC Bioinformatics*, 13(Suppl 11):S4.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria.

Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 687–692, Portland, Oregon, USA, June. Association for Computational Linguistics.

K.B. Cohen, K. Verspoor, H.L. Johnson, C. Roeder, P.V. Ogren, W.A. Baumgartner, E. White, H. Tipney, and L. Hunter. 2011. High-precision biological event extraction: Effects of system and data. *Computational Intelligence*, 27(4):681701, November.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *CrossParser '08: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Morristown, NJ, USA. 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. *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*, pages 1–9.

Jin-Dong Kim, Ngan Nguyen, Yue Wang, Jun'ichi Tsujii, Toshihisa Takagi, and Akinori Yonezawa. 2012. The genia event and protein coreference tasks of the bionlp shared task 2011. *BMC Bioinformatics*, 13(Suppl 11):S1.

H. Liu, R. Komandur, and K. Verspoor. 2011. From graphs to events: A subgraph matching approach for information eextraction from biomedical text. *ACL HLT 2011*, page 164.

Haibin Liu, Tom Christiansen, William Baumgartner, and Karin Verspoor. 2012. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *Journal of Biomedical Semantics*, 3(1):3.

Haibin Liu, Lawrence Hunter, Vlado Keselj, and Karin Verspoor. 2013a. Approximate subgraph matching-based literature mining for biomedical events and relations. *PLoS ONE*, 8(4):e60954, 04.

Haibin Liu, Karin Verspoor, Don Comeau, Andrew MacKinlay, and W. John Wilbur. 2013b. Generalizing an approximate subgraph matching-based system to extract events in molecular biology and cancer genetics. In *Proceedings of the 2013 BioNLP Workshop Companion Volume for the Shared Task*.

Andrew MacKinlay, David Martinez, and Timothy Baldwin. 2012. Detecting modification of biomedical events using a deep parsing approach. *BMC Medical Informatics and Decision Making*, 12(Suppl 1):S4.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the Association for Computational Linguistics (ACL 2008, short papers)*.

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

Paul Rayson and Roger Garside. 2000. Comparing corpora using frequency profiling. In *The Workshop on Comparing Corpora*, pages 1–6, Hong Kong, China, October. Association for Computational Linguistics.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57, Melbourne, Australia.

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.

S. Van Landeghem, K. Hakala, S. Rönnqvist, T. Salakoski, Y. Van de Peer, and F. Ginter. 2012. Exploring biomolecular literature with EVEX: Connecting genes through events, homology and indirect associations. *Advances in Bioinformatics*, Special issue Literature-Mining Solutions for Life Science Research:ID 582765.

Karin Verspoor, K. Bretonnel Cohen, Arrick Lanfranchi, Colin Warner, Helen L. Johnson, Christophe Roeder, Jinho D. Choi, Christopher Funk, Yuriy Malenkiy, Miriam Eckert, Nianwen Xue, William A. Baumgartner Jr., Michael Bada, Martha Palmer, , and Lawrence E. Hunter. 2012. A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools. *BMC Bioinformatics*.