Biomedical Event Annotation with CRFs and Precision Grammars

Andrew MacKinlay, David Martinez and **Timothy Baldwin**
NICTA Victoria Research Laboratories
University of Melbourne, VIC 3010, Australia
{amack,davidm,tim}@csse.unimelb.edu.au

Abstract

This work describes a system for the tasks of identifying events in biomedical text and marking those that are speculative or negated. The architecture of the system relies on both Machine Learning (ML) approaches and hand-coded precision grammars. We submitted the output of our approach to the event extraction shared task at BioNLP 2009, where our methods suffered from low recall, although we were one of the few teams to provide answers for task 3.

1 Introduction

We present in this paper our techniques for the tasks 1 and 3 of the event extraction shared task at BioNLP 2009. We make use of both Machine Learning (ML) approaches and hand-coded precision grammars in an architecture that combines multiple dedicated modules. In the third task on negation/speculation, we extract extract rich linguistic features resulting from our HPSG high-precision grammar to train an ML classifier.

2 Methodology

2.1 Task 1: Shallow Features and CRFs

Our system consists of two main modules, the first of which is devoted to the detection of event trigger words, and the second to event–theme analysis.

2.1.1 Trigger-word detection

We developed two separate systems to perform trigger word detection, and also a hybrid system which combines their outputs. The first system is a simple dictionary-based look-up tagger; the second system learns a structured model from the training data using conditional random fields (CRFs). For pre-processing, we relied on the domain-specific token and sentence splitter from the JULIE Lab (Tomanek et al., 2007) and the GENIA tagger for lemmatisation, POS tagging, chunking, and protein detection (Tsuruoka et al., 2005).

The look-up tagger operates by counting the occurrences in the training data of different event tags for a given term. Over the development and test data, each occurrence of a given term is assigned the event class with the highest prior in the training data. We experimented with a frequency cut-off that allows us to explore the precision/recall trade-off.

Our second system relies on CRFs, as implemented in the CRF++ toolkit (Sha and Pereira, 2003). CRFs provide a discriminative framework for building structured models to segment and label sequence data. CRFs have the well-known advantage that they both model sequential effects and support the use of large numbers of features. In our experiments we used the following feature types: word-forms, lemmas, POS, chunk tags, protein annotation, and grammatical dependencies. For dependency annotation, we used the Bikel parser and GDep as provided by the organisers. This information was provided as a feature that expresses the grammatical function of the token. We explored window sizes of ± 3 and ± 4.

Finally, we tested combining the outputs of the look-up tagger and CRF, by selecting all trigger words from both outputs.

2.1.2 Event-theme construction

We constructed the output for task 1 by differentiating among three types of events, according to their expected themes: basic events, binding events, and regulation events. We applied a simple strategy, assigning the closest events or proteins within a given sentence as themes.

For the basic events, we simply assigned the closest protein, an approach that we found to perform well over the training and development data. For binding events, we estimated the maximum distance away from the event word(s) for themes, and the maximum number of themes. For regulation events, we had to choose between proteins or events as themes, and the CAUSE field was also required. Again, we relied on a maximum distance threshold, and gave priority to events over proteins as themes. We removed regulation events as theme candidates, since our basic approach could not indicate the direction of the regulation. We also tested predicting the CAUSE by relying on the protein closest to the regulation event.

2.2 Task 3: Deep Parsing and Maximum Entropy classification

For task 3 we ran a syntactic parser over the abstracts and used the outputs to construct feature vectors for a machine learning algorithm. We built two classifiers (possibly with overlapping sets of feature vectors) for each training run: one to identify speculation and one for negation. We deliberately built a separate binary classifier for each task instead of a single four-class classifier, since the problem naturally decomposes this way. Speculation and negation are independent of one another (informally, not statistically) and it enables us to focus on feature engineering for each subtask.

2.2.1 Deep Parsing with the ERG

It seemed likely that syntactico-semantic analysis would be useful for task 3. To identify negation or speculation with relatively high precision, it is probable that knowledge of the relationships of possibly distant elements (such as the negation particle *not*) to a particular target word would provide valuable information for classification.

Further to this, it was our intention to evaluate the utility of deep parsing in such an approach, rather than a shallower annotation such as the output of a dependency parser. With this in mind, we selected the English Resource Grammar[1] (ERG: Copestake and Flickinger (2000)), an open-source, broad-coverage high-precision grammar of English in the HPSG framework.

While the ERG is relatively robust across different domains, it is a general-purpose resource, and there are some aspects of the language used in the biomedical abstracts that cause difficulties; unknown word handling is especially important given the nature of terms in the domain. Fortunately we can make some optimisations to mitigate this. The GENIA tagger mentioned in Section 2.1.1 provides both POS and named entity annotations, which we used to constrain the input to the ERG in two ways:

- Biological named entities identified by the GENIA tagger are flagged as such, and the parser does not attempt to decompose them.

- POS tags are appended to each input token to constrain the token to an appropriate category if it is absent from the ERG lexicon.

With these modifications to the parser, as well as preprocessing to handle differences in the tokenisation expected by the ERG to the output of the tagger, we were able to obtain a spanning parse for 72% of the training sentences. This still leaves 28% of the sentences inaccessible – the need for a fallback strategy is discussed further in Section 4.2.

2.2.2 Feature Extraction from RMRSs

Rather than outputting syntactic parse trees, the ERG can also produce output in particular semantic formalisms: Minimal Recursion Semantics (MRS: Copestake et al. (2005)) and the closely related Robust Minimal Recursion Semantics (RMRS: Copestake (2004)). For our feature generation here we make use of the latter.

Figure 1 shows an example RMRS obtained from one of the training documents. While there is insufficient space to give a complete treatment here, we highlight several aspects for expository purposes.

[1]Specifically the July 2008 version, downloadable from `http://lingo.stanford.edu/ftp/test/`

```
l1,
{ l3: _thus_a_1⟨62:67⟩(e5, ARG1: h4),
  l16: generic_unk_nom_rel⟨68:78⟩(x11, CARG: 'nf- kappa _b'),
  l6: udef_q_rel⟨68:89⟩(x9, RSTR: h8, BODY: h7),
  l10: compound_rel⟨68:89⟩(e12, ARG1: x9, ARG2: x11),
  l13: udef_q_rel⟨68:89⟩(x11, RSTR: h15, BODY: h14),
  l101: _activation_n_1⟨79:89⟩(x9),
  l17: neg_rel⟨94:97⟩(e19, ARG1: h18),
  l20: _require_v_1⟨98:106⟩(e2, ARG1: u21, ARG2: x9),
  l102: parg_d_rel⟨98:106⟩(e22, ARG1: e2, ARG2: x9),
  l103: _for_p⟨107:110⟩(e24, ARG1: e2, ARG2: x23),
  l34: generic_unk_nom_rel⟨111:129⟩(x29,
    CARG: 'neuroblastoma cell'),
  l25: udef_q_rel⟨111:146⟩(x23, RSTR: h27, BODY: h26),
  l28: compound_rel⟨111:146⟩(e30, ARG1: x23, ARG2: x29),
  l31: udef_q_rel⟨111:146⟩(x29, RSTR: h33, BODY: h32),
  l104: _differentiation_n_of⟨130:146⟩(x23, ARG1: u35) },
{ h4 qeq l17, h8 qeq l10, h15 qeq l16, h18 qeq l20, h27 qeq l28,
    h33 qeq l34 },
{ l10 in-g l101, l20 in-g l102, l20 in-g l103, l28 in-g l104 }
```

Figure 1: RMRS representation of the sentence *Thus NF-kappa B activation requires neuroblastoma cell differentiation* showing, in order, elementary predicates, qeq-constraints, and in-g constraints

The primary component of an RMRS is bag of *elementary predicates*, or EPs. Each EP shown has: (a) a label, such as 'l104'; (b) a predicate name, such as '_differentiation_n_1' (where 'n' indicates the part-of-speech); (c) character indices to the source sentence; and (d) a set of arguments. The first argument is always *ARG0* and is afforded special status, generally referring to the variable introduced by the predicate. Subsequent arguments are labelled according to the relation of the argument to the predicate. Arguments can be variables such as 'e30' or 'x23' (where the first letter indicates the nature of the variable – 'e' referring to events and 'x' to entities), or *handles* such as 'h33'.

These handles are generally used in the *qeq constraints*, which relate a handle to a label, indicating a particular kind of outscoping relationship between the handle and the label – either that the handle and label are equal or that the handle is equal to the label except that one or more quantifiers occur between the two (the name is derived from 'equality modulo quantifiers'). Finally there are in-g constraints which indicate that labels can be treated as equal. For our purposes this simply affects which qeq constraints they participate in – for example from the in-g constraint 'l28 in-g l104' and the qeq constraint 'h27 qeq l28', we can also infer that 'h27 qeq l104'.

In constructing features, we make use of:

- The *outscopes* relationship (specifically qeq-outscopes) – if EP A has a handle argument which qeq-outscopes the label of EP B, A is said to immediately outscope B; *outscopes* is the transitive closure of this.
- The *shared-argument* relationship, where EPs C and D refer to the same variable in one or more of their argument positions. We also in some cases make further restrictions on the types of arguments (*ARG0*, *RSTR*, etc) that may be shared on either end of the relationship.

2.2.3 Feature Sets and Classification

Feature vectors for a given event are constructed on the basis of the trigger word for the particular event, which we assume has already been identified; a natural consequence is that all events with the same trigger words have identical feature vectors. We use the term *trigger EPs* to describe the EP(s) which correspond to that trigger word – i.e. those whose character span encompasses the trigger word. We have a potentially large set of related EPs (with the kinds of relationships described above), which we filter to create the various feature sets, as outlined below.

We have several feature sets targeted at identifying negation:

- NEGOUTSCOPE2: If any EPs in the RMRS have predicate names in {*_no_q*, *_but+not_c*, *_nor_c*, *_only_a*, *_never_a*, *_not+as+yet_a*, *_not+as+yet_a*, *_unable_a*, *neg_rel*}, and that EP outscopes a trigger EP, set a general feature as well as a specific one for the particle.
- NEGCONJINDEX: If any EPs in the RMRS have predicate names in {*_not_c*, *_but+not_c*, *_nor_c*}, the *R-INDEX* (RHS of a conjunction) of that EP is the *ARG0* a trigger EP, set a general feature as well as a specific one for the particle – capturing the notion that these conjunctions are semantically negative for the particle on the right. This also had a corresponding feature for the *L-INDEX* of *_nor_c*, corresponding to the LHS of the *neither...nor* construction.

- ARG0NEGOUTSCOPEESA: For any EPs which have an argument that matches the *ARG0* of a trigger EP, if they are outscoped by an EP whose predicate name is in the list { *_only_a*, *_never_a*, *_not+as+yet_a*, *_not+as+yet_a*, *_unable_a*, *neg_rel*}, set a general feature to true, as well as features for the name of the outscoping and outscoped EPs. This is designed to catch trigger EP which are nouns, where the verb of which they are subject or object (or indeed an adjective/preposition to which they are linked) is semantically negated.

And several targeted at identifying speculation:

- SPECVOBJ2: if a verb is a member of the set {*_investigate*, *_study*, *_examine*, *_test*, *_evaluate*, *_observe*} and its *ARG2* (which corresponds to the verb object) is the *ARG0* of a trigger EP. This has a general feature for if any of the verbs match, and a feature which is specific to each verb in the target list.

- SPECVOBJ2+WN: as above, but augment the list of seed verbs with a list of WordNet sisters (i.e. any lemmas from any synsets for the verb), and add a feature which is set for the seed verbs which gave rise to other sister verbs.

- MODALOUTSCOPE: modal verbs (*can*, *should*, etc) may be strong indicators of speculation; this sets a value when the trigger EP is outscoped by any predicate corresponding to a modal, both as a general feature and a specific feature for the particular modal.

- ANALYSISSA: the *ARG0* of the trigger EP is also an argument of an EP with the predicate name *_analysis_n*. Such constructions involving the word *analysis* are relatively frequent in speculative events in the data.

And some general features, aiming to see if the learning algorithm could pick up other patterns we had missed:

- TRIGPREDPROPS: Set a feature value for the predicate name of each trigger EP, as well as the POS of each trigger EP.

- TRIGOUTSCOPES: Set a feature value for the predicate name and POS of each EP that is outscoped by the trigger EP.

- MODADJ: Set a feature value for any EPs which have an *ARG1* which matches the *ARG0* of the trigger EP if their POS is marked as adjective or adverb.

- +CONJ: This is actually a variant on the feature extraction method, which attempts to abstract away the effect of conjunctions. If the trigger EP is a member of a conjunction (i.e. shares an *ARG0* with the *L-INDEX* or *R-INDEX* of a conjunction), also treat the EPs which are conjunction parents (and their conjunctive parents if they exist) as trigger EPs in the feature construction.

2.2.4 Implementation

To produce training data to feed into a classifier, we parsed as many sentences as possible using the ERG, and used the output RMRSs to create training data using various combinations of the feature sets described above. The construction of features, however, presupposes annotations for the events and trigger words. For producing training data, we used the provided trigger annotations. For the test phase, we simply use the outputs of the classifier we built in phase 1, selecting the combination with the best performance over the development set. This pipeline architecture places limits on annotation performance – in particular, the recall in task 1 is an upper bound on task 3 recall. We used a maximum entropy classification algorithm for the ML component here – it has a low level of parameterization and is a solid performer in NLP tasks. The implementation we used was Zhang Le's Maxent Toolkit.[2]

3 Development experiments

3.1 Task 1

We devised a set of experiments over the trial, training, and development data in order to estimate the parameters for our final submission. Using the trial data, we performed manual error analysis on the rules used to construct events. With the training

[2]`http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

data, we performed our own evaluation based on cross-validation to detect trigger words and construct events. For the experiments over the development data, we relied on the evaluation interface provided by the organisation. We focused on testing the following modules: look-up tagger, CRF, combined system, and event construction.

First, we tuned the parameters of our look-up tagger over the training data. We used a threshold on the minimum number of term occurrences required to use the class information for that term from the training data. We evaluated thresholding on raw frequencies, and also on the percentage of occurrences of the term that were linked to the majority event. In cross-validation over the training data, we found that the raw-frequency threshold worked best, achieving a maximum F-score of 38.86%, as compared to 30.81% for the percentage approach (the results are shown in the bottom part of Table 1). We also estimated the frequency threshold as ≥ 25, and observed that most of the terms identified consisted of a single word, due to data sparseness in the training set.

Our next experiments are devoted to the CRF system, focusing on feature engineering. The results over the training data for: (a) the full feature set, and (b) removing one feature type at a time, are shown in Table 1, for windows of size ± 3 and ± 4. We can see that the best F-score is achieved by the ± 3 word-window system when removing the syntactic dependencies from Bikel's parser. These results improved over the look-up system.

As a final experiment on feature combinations and window size, we used the development evaluation interface. We submitted the best combinations shown in the above experiment, and also syntactic dependencies extracted with GDep. We observed the same behaviour as in training data, with the ± 3 word window obtaining the best F-score, and syntactic dependencies harming performance. These results are shown in the upper part of Table 2. Our final CRF system used this configuration (± 3 word window and all feature types except syntactic dependencies).

Our next step was to test the integration of the look-up tagger and CRF into a single system. We observed that by combining the outputs directly we could improve over the recall of CRF, and achieve higher F-score. This approach is referred to as "Combined (All)" in Table 2. We also tested the results when choosing either the look-up tagger or CRF depending on their performance over each event in the training data. The results of this system ("Combined (Best)") show a slight improvement over the basic combination.

Finally, we analysed the results of the event construction step. We used the gold-standard trigger annotation over the trial data and analysed the errors of our rules. We found out that there were three main types of error: (1) incorrect assignation of regulation themes; (2) trigger words having multiple themes; and (3) themes crossing sentence boundaries. We plan to address these problems in future work. We also observed that predicting CAUSE for the regulatory events caused the F-score to drop, resulting in us removing this functionality from the system.

W. size	Feats.	Rec.	Prec.	FSc.
±3	All	30.28	64.44	41.20
±3	−synt. dep.	30.20	65.01	**41.24**
±3	−protein NER	28.04	65.73	39.31
±3	−chunking	30.13	65.16	41.20
±3	−POS	29.68	65.25	40.80
±3	−lemma	27.96	62.60	38.66
±3	−word form	29.98	63.81	40.79
±4	All	28.86	66.15	40.19
±4	−synt. dep.	29.75	**67.06**	41.22
±4	−protein NER	28.11	66.73	39.56
±4	−chunking	28.56	66.61	39.98
±4	−POS	28.19	66.67	39.62
±4	−lemma	26.55	65.20	37.73
±4	−word form	28.19	65.28	39.38
Look-up (freq.)		**52.14**	30.97	38.86
Look-up (perc.)		38.20	25.82	30.81

Table 1: Trigger-word detection performance over training data. Results for the look-up tagger and CRFs with the full feature set and when removing one feature type at a time, for 3 and 4 word windows. The best results per column are shown in bold.

W. size	Feats.	Rec.	Prec.	FSc.
±3	All - synt.	17.55	56.17	26.75
±4	All - synt.	17.38	56.75	26.62
±3	All (GDep)	15.48	**58.69**	24.50
Combined (All)		**26.94**	27.83	27.38
Combined (Best)		21.24	39.92	**27.73**

Table 2: Performance of selected feature and window-size combinations over development data. Best results per column are given in bold.

N1:	NegOutscope2+Conj, NegConjIndex
N2:	*N1*, TrigPredProps
N3:	*N1*, Arg0NegOutscopeeSA
N4:	*N3*, TrigPredProps, NegVOutscope
N5:	*N3*, NegVOutscope
S1:	SpecVObj2+WN+Conj, AnalysisSA
S2:	*S1*, TrigPredProps
S3:	*S1*, ModAdj, ModalOutscope
S4:	*S3*, TrigOutscopes
S5:	SpecVObj2+WN+Conj, ModAdj, ModalOutscope,TrigOutscopes
B^{+y}_{-x}:	Context window of lemmatized tokens: x preceding and y following.

Table 3: Task 3 feature sets

3.2 Task 3

We evaluated the classification performance of various feature sets (including some not described here) using 10-fold cross-validation over the training data in the initial stages. We ran various combinations of the most promising features over the development data and evaluated their relative performance in an attempt to avoid overfitting.

To evaluate the performance boost we got in task 3 relative to more naive methods, we also experimented with feature sets based on a bag-of-words approach with a sliding context window of lemmatised tokens on either side. We evaluated all combinations of preceding and following context window sizes from 0 to 3. There are features for tokens that precede the trigger, follow the trigger, or lie anywhere within the context window, as well as for the trigger itself. A ‘token’ here may also be a named biological entity (protein etc) produced by GENIA tagger in our preprocessing phase, which would not be lemmatised. For comparability we only evaluate these features for sentences which we were able to parse. For the best performing baseline and RMRS-based feature sets, we also tested them in combination to see whether the features produced were complementary.

In Table 4 we present the results over the development data, using the provided gold-standard annotations of trigger words, as well as some selected results for our other task 1 outputs. The gold-standard figures are unrealistically high compared to what we would expect to achieve against the test data, but they are indicative at least of what we could achieve with a perfect event classifier. Similar to task 1, our system shows reasonable precision but suffers badly in recall. The substantially poorer performance when using our own annotations for the input events is discussed in more detail in Section 4.2

Task 1	Mod	Feats.	Rec.	Prec.	FSc.
Gold	Spec	B^{+2}_{-2}	23.2	40.0	29.3
Gold	Spec	B^{+3}_{-3}	22.1	47.7	30.2
Gold	Spec	S2	15.8	83.3	26.5
Gold	Spec	S3	18.9	78.3	30.5
Gold	Spec	S3,B^{+2}_{-2}	21.1	58.8	31.0
Gold	Spec	S3,B^{+3}_{-3}	23.2	57.9	33.1
Comb(best)	Spec	S3	4.2	21.0	7.0
Gold	Spec	S4	17.9	94.4	30.1
Gold	Spec	S5	17.9	100.0	30.4
Gold	Neg	B^{+0}_{-2}	14.0	33.3	19.7
Gold	Neg	B^{+1}_{-3}	15.0	30.2	20.0
Gold	Neg	N2	19.6	61.8	29.8
Comb(best)	Neg	N2	0.9	7.7	1.7
Gold	Neg	N3	15.9	68.0	25.8
Gold	Neg	N4	19.6	67.7	30.4
Gold	Neg	N4,B^{+1}_{-3}	22.4	52.2	31.4
Gold	Neg	N4,B^{+0}_{-2}	24.3	68.4	35.9
Gold	Neg	N5	16.8	69.2	30.1

Table 4: Results (exact match) over development data for task 3 using gold-standard event/trigger annotations and selected other annotations for task 1. Feature sets described in Table 3

One area where we could improve is to go after the 30% of sentences for which we do not have a spanning parse and resultant RMRS. To reuse existing infrastructure, we could produce RMRS output from an alternative processing component with broader coverage but less precision. Several methods exist to do this – e.g. producing RMRS output from RASP (Briscoe et al., 2006) is described in Frank (2004). However there is clearly room for improvement in the remaining 70% of sentences which we can parse – our results in Table 4 are still well below the limit of roughly 70% recall.[3]

Additional lexical resources beyond WordNet, particularly domain-specific ones, are likely to be useful in boosting performance since they will help maximally utilise the training data. Additionally, we have not yet made use of other event annotations apart from the trigger words – features based on characteristics such as the event class or properties of the event arguments could also be useful.

[3]We have not performed any analysis to verify whether the number of events per sentence differs between parseable and unparseable sentences.

System	Rec.	Prec.	FSc.
Combined (Best)	**17.44**	**39.99**	**24.29**
Combined (All)	24.36	30.87	27.23
CRF	12.23	62.24	20.44
CRF (+ synt feats)	12.01	61.91	20.11
Look-Up	22.88	29.67	25.84
Look-Up (freq >= 20)	23.26	26.74	24.88
Look-Up (freq >= 30)	21.37	30.50	25.13

Table 5: Task 1 results with approximate span matching, recursive evaluation (our final submission is in bold)

4 Results

4.1 Task 1

Our experiments on the training and development set showed that our CRF++ was biased towards precision at the cost of recall, and for the look-up system the best F-score was obtained when aiming for high recall at the cost of lower precision. The best results were obtained when combining both approaches, and this was the composition of the system we submitted.

For our final submission, the CRF++ approach had a ± 3 word window, and all the features except for syntactic dependencies, which were found to harm performance. Our final look-up system relied on raw frequencies to choose candidate terms, and those above 24 occurrences in training data were included in the dictionary. For the combination, we observed that for most events the look-up system performed better (although the overall F-score was lower), and we decided to use the CRF++ output only for the events that showed better performance than the look-up system (TRANSCRIPTION, GENE EXPRESSION, and POSITIVE REGULATION).

The results over the test data for our final submission and the main variants we explored are shown in Table 5. We can see that the CRF performed poorly, with very low recall over the test set, in contrast with the development results, where the higher recall resulted in a higher F-score than the look-up approach. The best of our systems was the full combination of CRF and the look-up tagger, with a 27.23% F-score.

The results for each event separately are given in Table 6. The system performs much worse on regulation events, due to the difficulty of having to correctly identify other events in the near context.

Event Class	Rec.	Prec.	FSc.
Localization	25.86	65.22	37.04
Binding	17.00	28.92	21.42
Gene-expression	45.71	69.18	55.05
Transcription	34.31	26.26	29.75
Protein-catabolism	42.86	85.71	57.14
Phosphorylation	45.19	64.21	53.04
EVT-TOTAL	35.84	53.15	42.81
Regulation	15.46	13.24	14.26
Positive-regulation	13.84	14.82	14.31
Negative-regulation	12.14	20.44	15.23
REG-TOTAL	13.73	15.31	14.48
ALL-TOTAL	24.36	30.87	27.23

Table 6: Results for the different events from our combined system. Averaged scores for single events, regulations, and all.

4.2 Task 3

For testing, we repurposed all of the development data as training data and retrained our classifiers. The results in Table 7 were somewhat disappointing, but a drop in recall versus the equivalent run over the development data using oracle task 1 annotations was unsurprising and the ratio of this drop is within the bounds of what we would expect. The substantial drop in precision can similarly be explained by flow-on effects from our task 1 classification, a natural consequence of our pipeline architecture. It is quite possible for our system to identify false positive events as being modified; in the online evaluation system, these classifications of non-existent events reduce our precision in task 3.

In the feature engineering stage, we primarily used the oracle data for task 1 to maximise the amount of training data available. We felt that if we were to use our task 1 classifications for events and trigger words, the effectively lower number of training instances would only hurt performance. However this possibly led to bias towards features which were more useful for classifying events that we couldn't successfully classify in task 1. The development set shows similar performance drops under these conditions in Table 4.

It is also possible that our features work reasonably but that our classification engine trained over the oracle data simply learnt the wrong parameters

Task 1	**Mod**	**Fts.**	**Rec.**	**Prec.**	**FSc.**
Comb(Best)	Spc	B^{+3}_{-3}	2.88	12.24	4.67
Comb(Best)	Spc	S2	4.33	37.50	7.76
Comb(Best)	Spc	**S3**	**4.81**	**30.30**	**8.30**
Comb(All)	Spc	S3	5.29	26.19	8.80
Comb(Best)	Spc	S3,B^{+3}_{-3}	4.81	14.08	7.17
Comb(Best)	Spc	S4	3.85	27.59	6.75
Comb(Best)	Spe	S5	3.85	27.59	6.75
Comb(Best)	Neg	B^{+3}_{-1}	3.96	25.00	6.84
Comb(Best)	Neg	**N2**	**5.29**	**34.48**	**9.17**
Comb(All)	Neg	N2	5.73	30.00	9.62
Comb(Best)	Neg	N3	5.29	27.78	8.88
Comb(Best)	Neg	N4	5.29	34.48	9.17
Comb(Best)	Neg	N4,B^{+0}_{-2}	4.85	28.12	8.27
Comb(All)	Neg	N4	5.73	27.27	9.47
Comb(Best)	Neg	N5	5.29	29.41	8.96

Table 7: Results over test data for task 3 using gold-standard event annotations (approx recursive matching), showing which set of trigger word classifications from task 1 was used as input (submitted results in bold). Feature sets described in table 3

for the events we had identified correctly in task 1. We could check this by training a classifier using our task 1 event classifications combined with the gold-standard trigger annotations. However combining the gold-standard annotations for task 3 with the classifier outputs of task 1 is non-trivial and was not attempted due to time constraints. It also would have been instructive to calculate a ceiling on our task 3 performance given our performance in task 1 – i.e. how many modifications we could have correctly identified with a perfect task 3 classifier, but we were not able to show this for similar reasons.

5 Conclusions

Our analysis of task 1 seemed to indicate that the scarcity of training instances was the main reason for the low recall of CRFs. The look-up system contributed to increase the recall, but at the cost of lower precision. In order to improve this module we plan to find ways to extend the training data automatically in a bootstrapping process.

Another limitation of our system is the event-construction module, which follows simple rules and performs poorly on regulation events. For this subtask we plan to extend the rule set and apply optimisation techniques, following the lessons learned in error analysis.

In task 3 we investigated the application of a precise, general-purpose grammar over this domain, and were relatively successful. However, while the parse coverage for task 3 is very respectable for a precision grammar on comparatively difficult material, it is clearly unwise to throw away 30% of sentences, so a method to extract features from these is desirable. Further sources of data would also be useful, such as data from the event annotations themselves, and additional lexical resources tailored to the biomedical domain.

We have also shown the syntactico-semantic output of a deep parser, in the form of an RMRS, can be beneficial in such a task compared with a more naive approach based on bags of words within a sliding context window. From Table 4, for negation, the syntactic features provided substantial performance gains over the best set of baseline parameters we could find. For speculation the evidence here is less compelling, with similar scores from both approaches. Over test data in Table 7, the the deep methods showed superior performance, albeit over a smaller number of instances. Regardless, the RMRS still has some advantages, giving (unsurprisingly) higher precision than the baseline methods. Combining naive and deep features does tend to give slightly higher performance than either of the inputs over the development data (although not over the test data, perhaps due to the poorer performance of naive methods), suggesting that the two approaches identify slightly different kinds of modification.

Our system suffered from the pipeline approach – there was no way to recover from an incorrect classification in task 1, resulting in greatly reduced precision and recall in task 3. It is possible that a carefully constructed integrated system could annotate events for trigger words and argument at the same time as modification, with features shared between the two, which may avoid some of these issues.

Acknowledgements

We wish to thank Rebecca Dridan, Dan Flickinger and Lawrence Cavedon for their advice. 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.

References

Edward Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Poster System*, pages 77–80, Sydney, Australia.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *International Conference on Language Resources and Evaluation*.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, pages 281–332.

Ann Copestake. 2004. Report on the design of RMRS. Technical Report D1.1a, University of Cambridge, Cambridge, UK.

Anette Frank. 2004. Constraint-based RMRS construction from shallow grammars. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1269, Morristown, NJ, USA. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141.

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.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Advances in Informatics - 10th Panhellenic Conference on Informatics*, pages 382–392.