Model Combination for Event Extraction in BioNLP 2011

Sebastian Riedel[a], **David McClosky**[b], **Mihai Surdeanu**[b],
Andrew McCallum[a], and **Christopher D. Manning**[b]
[a] Department of Computer Science, University of Massachusetts at Amherst
[b] Department of Computer Science, Stanford University
{riedel,mccallum}@cs.umass.edu
{mcclosky,mihais,manning}@stanford.edu

Abstract

We describe the FAUST entry to the BioNLP 2011 shared task on biomolecular event extraction. The FAUST system explores several stacking models for combination using as base models the UMass dual decomposition (Riedel and McCallum, 2011) and Stanford event parsing (McClosky et al., 2011b) approaches. We show that using stacking is a straightforward way to improving performance for event extraction and find that it is most effective when using a small set of stacking features and the base models use slightly different representations of the input data. The FAUST system obtained 1st place in three out of four tasks: 1st place in Genia Task 1 (56.0% f-score) and Task 2 (53.9%), 2nd place in the Epigenetics and Post-translational Modifications track (35.0%), and 1st place in the Infectious Diseases track (55.6%).

1 Introduction

To date, most approaches to the BioNLP event extraction task (Kim et al., 2011a) use a single model to produce their output. However, model combination techniques such as voting, stacking, and reranking have been shown to consistently produce higher performing systems by taking advantage of multiple views of the same data. The Netflix Prize (Bennett et al., 2007) is a prime example of this. System combination essentially allows systems to regularize each other, smoothing over the artifacts of each (c.f. Nivre and McDonald (2008), Surdeanu and Manning (2010)). To our knowledge, the only previous example of model combination for the BioNLP shared task was performed by Kim et al. (2009). Using a weighted voting scheme to combine the outputs from the top six systems, they obtained a 4% absolute f-score improvement over the best individual system.

This paper shows that using a straightforward model combination strategy on two competitive systems produces a new system with substantially higher accuracy. This is achieved with the framework of stacking: a *stacking* model uses the output of a *stacked* model as additional features.

While we initially considered voting and reranking model combination strategies, it seemed that given the performance gap between the UMass and Stanford systems that the best option was to include the predictions from the Stanford system into the UMass system (e.g., as in Nivre and McDonald (2008)). This has the advantage that one model (Umass) determines how to integrate the outputs of the other model (Stanford) into its own structure, whereas in reranking, for example, the combined model is required to output a complete structure produced by only one of the input models.

2 Approach

In the following we briefly present both the stacking and the stacked model and some possible ways of integrating the stacked information.

2.1 Stacking Model

As our stacking model, we employ the UMass extractor (Riedel and McCallum, 2011). It is based on a discriminatively trained model that jointly predicts trigger labels, event arguments and protein pairs in

binding. We will briefly describe this model but first introduce three types of binary variables that will represent events in a given sentence. Variables $e_{i,t}$ are active if and only if the token at position i has the label t. Variables $a_{i,j,r}$ are active if and only if there is an event with trigger i that has an argument with role r grounded at token j. In the case of an entity mention this means that the mention's head is j. In the case of an event j is the position of its trigger. Finally, variables $b_{p,q}$ indicate whether or not two entity mentions at p and q appear as arguments in the same binding event.

Two parts form our model: a scoring function, and a set of constraints. The scoring function over the trigger variables $\mathbf{e}$, argument variables $\mathbf{a}$ and binding pair variables $\mathbf{b}$ is

$$s(\mathbf{e},\mathbf{a},\mathbf{b}) \stackrel{\text{def}}{=} \sum_{e_{i,t}=1} s_{\text{T}}(i,t) + \sum_{a_{i,j,r}=1} s_{\text{R}}(i,j,r) + \sum_{b_{p,q}=1} s_{\text{B}}(p,q)$$

with local scoring functions $s_{\text{T}}(i,t) \stackrel{\text{def}}{=} \langle \mathbf{w}_{\text{T}}, \mathbf{f}_{\text{T}}(i,t) \rangle$, $s_{\text{R}}(i,j,r) \stackrel{\text{def}}{=} \langle \mathbf{w}_{\text{R}}, \mathbf{f}_{\text{R}}(i,j,r) \rangle$ and $s_{\text{B}}(p,q) \stackrel{\text{def}}{=} \langle \mathbf{w}_{\text{B}}, \mathbf{f}_{\text{B}}(p,q) \rangle$.

Our model scores all parts of the structure in isolation. It is a joint model due to the nature of the constraints we enforce: First, we require that each active event trigger must have at least one `Theme` argument; second, only regulation events (or Catalysis events for the EPI track) are allowed to have `Cause` arguments; third, any trigger that is itself an argument of another event has to be labelled active, too; finally, if we decide that two entities p and q are part of the same binding (as indicated by $b_{p,q} = 1$), there needs to be a binding event at some trigger i that has p and q as arguments. We will denote the set of structures $(\mathbf{e},\mathbf{a},\mathbf{b})$ that satisfy these constraints as $\mathcal{Y}$.

Stacking with this model is simple: we only need to augment the local feature functions $\mathbf{f}_{\text{T}}(i,t)$, $\mathbf{f}_{\text{R}}(i,j,r)$ and $\mathbf{f}_{\text{B}}(p,q)$ to include predictions from the systems to be stacked. For example, for every system S to be stacked and every pair of event types (t', t_S) we add the features

$$f_{S,t',t_S}(i,t) = \begin{cases} 1 & h_S(i) = t_S \wedge t' = t \\ 0 & \text{otherwise} \end{cases}$$

to $\mathbf{f}_{\text{T}}(i,t)$. Here $h_S(i)$ is the event label given to token i according to S. These features allow different weights to be given to each possible combination of type t' that we want to assign, and type t_S that S predicts.

Inference in this model amounts to maximizing $s(\mathbf{e},\mathbf{a},\mathbf{b})$ over $\mathcal{Y}$. Our approach to solving this problem is dual decomposition (Komodakis et al., 2007; Rush et al., 2010). We divide the problem into three subproblems: (1) finding the best trigger label and set of outgoing edges for each candidate trigger; (2) finding the best trigger label and set of incoming edges for each candidate trigger; (3) finding the best pairs of entities to appear in the same binding. Due to space limitations we refer the reader to Riedel and McCallum (2011) for further details.

2.2 Stacked Model

For the stacked model, we use a system based on an event parsing framework (McClosky et al., 2011a) referred to as the Stanford model in this paper. This model converts event structures to dependency trees which are parsed using MSTParser (McDonald et al., 2005).[1] Once parsed, the resulting dependency tree is converted back to event structures. Using the Stanford model as the stacked model is helpful since it captures tree structure which is not the focus in the UMass model. Of course, this is also a limitation since actual BioNLP event graphs are DAGs, but the model does well considering these restrictions. Additionally, this constraint encourages the Stanford model to provide different (and thus more useful for stacking) results.

Of particular interest to this paper are the four possible decoders in MSTParser. These four decoders come from combinations of feature order (first or second) and whether the resulting dependency tree is required to be projective.[2] Each decoder presents a slightly different view of the data and thus has different model combination properties. Projectivity constraints are not captured in the UMass model so these decoders incorporate novel information.

To produce stacking output from the Stanford system, we need its predictions on the training, devel-

[1] `http://sourceforge.net/projects/mstparser/`

[2] For brevity, the second-order non-projective decoder is abbreviated as 2N, first-order projective as 1P, etc.

	UMass			FAUST+All		
	R	P	F1	R	P	F1
GE T1	48.5	64.1	55.2	49.4	64.8	56.0
GE T2	43.9	60.9	51.0	46.7	63.8	53.9
EPI (F)	28.1	41.6	33.5	28.9	44.5	35.0
EPI (C)	57.0	73.3	64.2	59.9	80.3	68.6
ID (F)	46.9	62.0	53.4	48.0	66.0	55.6
ID (C)	49.5	62.1	55.1	50.6	66.1	57.3

Table 1: Results on test sets of all tasks we submitted to. T1 and T2 stand for task 1 and 2, respectively. C stands for CORE metric, F for FULL metric.

opment and test sets. For predictions on test and development sets we used models learned from the the complete training set. Predictions over training data were produced using crossvalidation. This helps to avoid a scenario where the stacking model learns to rely on high accuracy at training time that cannot be matched at test time.

Note that, unlike Stanford's individual submission in this shared task, the stacked models in this paper do not include the Stanford reranker. This is because it would have required making a reranker model for each crossvalidation fold.

We made 19 crossvalidation training folds for Genia (GE) (Kim et al., 2011b), 12 for Epigenetics (EPI), and 17 for Infectious Diseases (ID) (Kim et al., 2011b; Ohta et al., 2011; Pyysalo et al., 2011, respectively). Note that while ID is the smallest and would seem like it would have the fewest folds, we combined the training data of ID with the training and development data from GE. To produce predictions over the test data, we combined the training folds with 6 development folds for GE, 4 for EPI, and 1 for ID.

3 Experiments

Table 1 gives an overview of our results on the test sets for all four tasks we submitted to. Note that for the EPI and ID tasks we show the CORE metric next to the official FULL metric. The former is suitable for our purposes because it does not measure performance for negations, speculations and cellular locations—all of these we did not attempt to predict.

We compare the UMass standalone system to the FAUST+All system which stacks the Stanford 1N, 1P, 2N and 2P predictions. For all four tasks we observe substantial improvements due to stacking. The increase is particular striking for the EPI track, where stacking improves f-score by more than 4.0 points on the CORE metric.

System	SVT	BIND	REG	TOTAL
UMass	74.7	**47.7**	42.8	54.8
Stanford 1N	71.4	38.6	32.8	47.8
Stanford 1P	70.8	35.9	31.1	46.5
Stanford 2N	69.1	35.0	27.8	44.3
Stanford 2P	72.0	36.2	32.2	47.4
FAUST+All	**76.9**	43.5	44.0	**55.9**
FAUST+1N	76.4	45.1	43.8	55.6
FAUST+1P	75.8	43.1	**44.6**	55.7
FAUST+2N	74.9	42.8	43.8	54.9
FAUST+2P	75.7	46.0	44.1	55.7
FAUST+All (triggers)	76.4	41.2	43.1	54.9
FAUST+All (arguments)	76.1	41.7	43.6	55.1

Table 2: BioNLP f-scores on the development section of the Genia track (task 1) for several event categories.

To analyze the impact of stacking further, Table 2 shows a breakdown of our results on the Genia development set. Presented are f-scores for simple events (SVT), binding events (BIND), regulation events (REG) and the set of all event types (TOTAL). We compare the UMass standalone system, various Stanford-standalone models and stacked versions of these (FAUST+X).

Remarkably, while there is a 7 point gap between the best individual Stanford system and the standalone UMass systems, integrating the Stanford prediction still leads to an f-score improvement of 1. This can be seen when comparing the UMass, Stanford 1N and FAUST+All results, where the latter stacks 1N, 1P, 2N and 2P. We also note that stacking the projective 1P and 2P systems helps almost as much as stacking all Stanford systems. Notably, both 1P and 2P do not do as well in isolation when compared to the 1N system. When stacked, however, they do slightly better. This suggests that projectivity is a missing aspect in the UMass standalone system.

The FAUST+All (triggers) and FAUST+All (arguments) lines represent experiments to determine whether it is useful to incorporate only portions of

the stacking information from the Stanford system. Given the small gains over the original UMass system, it is clear that stacking information is only useful when attached to triggers and arguments. Our theory is that most of our gains come from when the UMass and Stanford systems disagree on triggers and the Stanford system provides not only its triggers but also their attached arguments to the UMass system. This is supported by a pilot experiment where we trained the Stanford model to use the UMass triggers and saw no benefit from stacking (even when both triggers and arguments were used).

Table 3 shows our results on the development set of the ID task, this time in terms of recall, precision and f-score. Here the gap between Stanford-only results, and the UMass results, is much smaller. This seems to lead to more substantial improvements for stacking: FAUST+All obtains a f-score 2.2 points larger than the standalone UMass system. Also note that, similarly to the previous table, the projective systems do worse on their own, but are more useful when stacked.

Another possible approach to stacking *conjoins* all the original features of the stacking model with the predicted features of the stacked model. The hope is that this allows the learner to give different weights to the stacked predictions in different contexts. However, incorporating Stanford predictions by conjoining them with all features of the UMass standalone system (FAUST+2P-Conj in Table 3) does not help here.

We note that for our results on the ID task we augment the training data with events from the GE training set. Merging both training sets is reasonable since there is a significant overlap between both in terms of events as well as lexical and syntactic patterns to express these. When building our training set we add each training document from GE once, and each ID training document twice—this lead to substantially better results than including ID data only once.

4 Discussion

Generally stacking has led to substantial improvements across the board. There are, however, some exceptions. One is binding events for the GE task. Here the UMass model still outperforms the best stacked system (see Table 2). Likewise, for full papers in the Genia test set, the UMass model still does slightly better with 53.1 f-score compared to 52.7 f-score. This suggests that a more informed combination of our systems (e.g., metaclassifiers) could lead to better performance.

System	Rec	Prec	F1
UMass	46.2	51.1	48.5
Stanford 1N	43.1	49.1	45.9
Stanford 1P	40.8	46.7	43.5
Stanford 2N	41.6	53.9	46.9
Stanford 2P	42.8	48.1	45.3
FAUST+All	47.6	**54.3**	**50.7**
FAUST+1N	45.8	51.6	48.5
FAUST+1P	47.6	52.8	50.0
FAUST+2N	45.4	52.4	48.6
FAUST+2P	**49.1**	52.6	**50.7**
FAUST+2P-Conj	48.0	53.2	50.4

Table 3: Results on the development set for the ID track.

5 Conclusion

We have presented the FAUST entry to the BioNLP 2011 shared task on biomolecular event extraction. It is based on stacking, a simple approach for model combination. By using the predictions of the Stanford entry as features of the UMass model, we substantially improved upon both systems in isolation. This helped us to rank 1st in three of the four tasks we submitted results to. Remarkably, in some cases we observed improvements despite a 7.0 f-score margin between the models we combined.

In the future we would like to investigate alternative means for model combination such as reranking, union, intersection, and other voting techniques. We also plan to use dual decomposition to encourage models to agree. In particular, we will seek to incorporate an MST component into the dual decomposition algorithm used by the UMass system.

Acknowledgments

We thank the BioNLP shared task organizers for setting this up and their quick responses to questions. This work was supported in part by the Center for Intelligent Information Retrieval. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181.

References

James Bennett, Stan Lanning, and Netflix. 2007. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on BioNLP: Shared Task*, pages 1–9. Association for Computational Linguistics.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Jun'ichi Tsujii. 2011a. Overview of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011b. Overview of the Genia Event task in BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*.

David McClosky, Mihai Surdeanu, and Chris Manning. 2011a. Event extraction as dependency parsing. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies 2011 Conference (ACL-HLT'11), Main Conference*, Portland, Oregon, June.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011b. Event extraction as dependency parsing in BioNLP 2011. In *BioNLP 2011 Shared Task*.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*. The Association for Computational Linguistics.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

Tomoko Ohta, Sampo Pyysalo, and Jun'ichi Tsujii. 2011. Overview of the Epigenetics and Post-translational Modifications (EPI) task of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Dan Sullivan, Chunhong Mao, Chunxia Wang, Bruno Sobral, Jun'ichi Tsujii, and Sophia Ananiadou. 2011. Overview of the Infectious Diseases (ID) task of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *BioNLP 2011 Shared Task*.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proc. EMNLP*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference (NAACL-2010)*, Los Angeles, CA, June.