# Inter-Event Dependencies support Event Extraction from Biomedical Literature

Roman Klinger[1], Sebastian Riedel[2], and Andrew McCallum[2]

[1] Department of Bioinformatics
Fraunhofer Institute Algorithms and
Scientific Computing (SCAI)
Schloss Birlinghoven
53754 Sankt Augustin, Germany
klinger@scai.fraunhofer.de
[2] Information Extraction and Synthesis Lab
University of Massachusetts
Amherst, MA 01003, USA
{riedel,mccallum}@cs.umass.edu

**Abstract.** The description of events in biomedical literature often follows discourse patterns. For example, authors may firstly mention the transcription of a gene, and then go on to describe how this transcription is regulated by another gene. Capturing such patterns can be beneficial when we want to extract event mentions from literature. For instance, detecting the mention of a transcription of gene A gives us a hint to actively look for mentions of regulations involving A. With this hint we could find such mentions even if they follow unseen lexical or syntactic patterns. To exploit such hints we need to perform event extraction in a cross sentence manner.
It is shown that imperatively defined factor graphs (IDF) are an intuitive way to build Markov Networks that model inter-dependencies between mentions of events within sentences, and across sentence-boundaries. Small pieces of procedural code define the graph structure, feature functions and hooks for efficient inference. Empirically, this leads to an efficient cross-sentence event extractor with very competitive results on the BioNLP shared task. One of our inter-event features shows an impact of 1.94 points in $F_1$ for the class of regulation events.

## 1 Introduction

Finding relevant information is one of the most important challenges in our time. In particular in life science a huge amount of new publications, research reports and patents is produced every year. For users of very large text corpora like MEDLINE[3], which contains 19 million citations as of June 2010, document categorization, ranking and finding entity-related information is an important help in their daily research and work life. Especially gene and protein names are of high interest, and methods for their recognition and normalization as well as protein-protein interaction identification are developed since several years [2,5,6]. These tasks typically target the reconstruction of flat relations between entities.

[3] http://www.ncbi.nlm.nih.gov/pubmed/

The BioNLP Shared Task 2009 [8] aims to extract nested bio-molecular events from research abstracts. Events of interest are *gene expressions*, *transcriptions*, *protein catabolisms*, *phosphorylation* and *localization*, which can have exactly one gene as argument (the theme of the event). *Bindings* can have multiple genes as themes, and *regulations*, optionally specified as *positive* or *negative*, are declared by a theme and a cause, while these arguments can be genes as well as other events. These descriptions follow the definition of the gene ontology (GO, [1]). An example of such structure is shown in Figure 1.

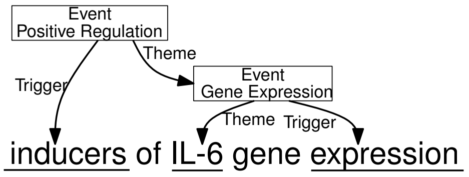


Fig. 1: Example regulation structure from training data [24]

So far, work on this problem has primarily focused on two aspects: effective features for the tasks of clue/trigger word detection and argument attachment [3], and joint models of the full event extraction task [18,16,19,12,22]. What has mostly been neglected are correlations between pairs of events.

Why would we want to model event-event dependencies? An intriguing case for event-event correlations becomes apparent when we consider that events are not just expressed in single sentences. Instead, they are usually introduced and further explained in a *discourse*. For example, an author may begin with the observation that “protein A regulates protein B”. In the course of the document the author then elaborates that A regulates the gene expression of B.

The above intuition can be measured by considering the sequence of events within a document that concern a given protein. We have observed that for this sequence, strong correlations in terms of event types hold. For example, if one sentence mentions a transcription of gene A, then there is a $44\,\%$ probability that the next mention of this gene concerns a positive regulation of A.

We have chosen to use Imperatively Defined Factor Graphs (IDF,[10,11]) to model such event-event interactions. When building IDFs, the probabilistic modeler uses small pieces of imperative code instead of declarative formulae to define the structure of a graphical model (as necessary in Markov Logic [17]). Crucially, during Markov chain monte carlo (MCMC) inference this code only unrolls the network relevant to the current variable assignment. Working with this partial network is extremely efficient—to the extent that large scale joint inference can still remain tractable [21]. In addition, when working with IDFs we can plug-in tailor-made proposal functions that allow for even faster inference.

In this work, we present an IDF approach to biomedical event extraction. Notably, to our knowledge it is the first that captures document-wide interactions between pairs of events. We show that by modeling such interactions error reductions of $1.2$ points in $F_1$ measure over a strong baseline are possible. This leads to results that tie the best ever reported numbers on the BioNLP shared task development set, and to the third best results on the test set. Moreover, we show that despite the added complexity, inference and learning in our model are still efficient.

## 2 Methods

### 2.1 Imperatively Defined Factor Graphs

A factor graph [9] is a bipartite graph over factors and variables. Let factor graph $G$ define a probability distribution over a set of output variables $\boldsymbol{y}$ conditioned on input variables $\boldsymbol{x}$. A factor $\Psi_i$ computes a scalar value over the subset of variables $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ that are neighbors of $\Psi_i$ in the graph. Often this real-valued function is defined as the exponential of an inner product over sufficient statistics $\{f_{ik}(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ and parameters $\{\theta_{ik}\}$, where $k \in [1, K_i]$ and $K_i$ is the number of parameters for factor $\Psi_i$. Let $Z(\boldsymbol{x})$ be the data-dependent partition function used for normalization. The probability distribution can be written as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{\Psi_i \in G} \exp\left( \sum_{k=1}^{K_i} \theta_{ik} f_{ik}(\boldsymbol{x}_i, \boldsymbol{y}_i) \right).$$

In practice, factor graphs often use the same parameters for several factors, this is termed *parameter tying*. A *factor template* $T_j$ consists of parameters $\{\theta_{jk}\}$, sufficient statistic functions $\{f_{jk}\}$, and a description of an arbitrary relationship between variables, yielding a set of satisfying tuples $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}$. For each of these variable tuples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ that fulfil this relationship, the factor template instantiates a factor that shares $\{\theta_{jk}\}$ and $\{f_{jk}\}$ with all other instantiations of $T_j$. Let $\mathcal{T}$ be the set of factor templates. In this case the probability distribution is

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{T_j \in \mathcal{T}} \prod_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in T_j} \exp\left( \sum_{k=1}^{K_j} \theta_{jk} f_{jk}(\boldsymbol{x}_i, \boldsymbol{y}_i) \right).$$

The process of instantiating individual factors from their templates is termed *unrolling*. For a factor $\Psi_i$ that is an instantiation of factor template $T_j$, the inner product of $\{f_{jk}(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ and parameters $\{\theta_{jk}\}$ is termed the *score* of the factor.

Imperatively-defined factor graphs (IDFs) are an approach to probabilistic programming that preserves the *declarative* semantics of factor graphs, while leveraging *imperative* constructs (pieces of procedural programming) to greatly aid both efficiency and natural intuition in specifying model structure, inference, and learning.

FACTORIE[4] [10,11] is an implementation of IDFs in the context of Markov chain Monte Carlo (MCMC) inference, a common approach for inference in very large graph structures [4,17,13]. It only requires to represent a single world at one time by proposing a change to the current world and accepting that change depending on the ratio of post- and pre-proposal model scores. In this framework, factors that touch unchanged variables do not need to be evaluated.

IDF programming consists of four stages: (1) representing data through variables, (2) designing templates that define the graphical structure of the network, (3) optionally providing application specific hooks for efficient inference, (4) reading in the data, learning parameters (using Sample-Rank in this paper [20]), testing, and evaluating.

These steps, for the case of biological event extraction, are described in the following sections, the fourth stage is part of Section 3.

[4] `http://factorie.cs.umass.edu`

Table 1: Variables to represent an event structure on sentences.

| Variable | Description |
|---|---|
| Document | Sequence of sentences |
| Sentence | Sequence of tokens together with a sequence of spans |
| Token | Member of Sentence and Span, never changed |
| Span | Contains consecutive tokens in sentence, associated with event or gene, member of Sentence |
| Event | Attribute of span, multiple events on one span possible, contains clue word and arguments |
| Gene | Attribute of span, never changed |
| Argument | Points to a gene or an event, can be a theme or a cause |

## 2.2 Data and Variable Representation

For any kind of probabilistic modeling we need to represent data, or *possible worlds*, through variables. In the case of IDFs we use an object-oriented model that uses classes, member fields and methods to achieve the same. In fact, we can use the full repertoire of constructs in a modern programming language—in the case of FACTORIE this language is Scala.[5]

In our representation, each document is divided into *sentences* which are sequences of *tokens*. Each sentence has *spans* over tokens that are associated with the entities these spans express. Entities can either be *genes/proteins*, or *events*. In the case of the latter, the corresponding span is an *event clue*. Each event entity can have a set of *arguments*. An argument is either a gene or an event, and is labelled with a *theme* or *cause* role. The variables are summarized in Table 1.

For our actual experiments we restrict us to a set of candidate spans that are allowed to become event clues during inference. We fill this set with all spans of token length one, except for those that are preceded by a hyphen (e. g. "up - regulation"). In this case we instead add a span that includes the token itself and its preceding two tokens. No spans are generated on tokens whose words never appear as clues in the training data.

## 2.3 Templates

As described in Section 2.1, templates define the sets of variables that form factors (i. e., the graphical structure), the sufficient statistics/features defined on these factors, and the parameters associated with them. In FACTORIE, templates are Scala classes implementing *unroll* methods to define the connectivity of the graphical model. This method returns all factors the current templates associates with a given variable. This is very suitable for MCMC inference which requires, for each proposed jump, all factors associated with a changed variable.

Sufficient statistics/features are implemented through methods of a template class as well. Given a factor, a *statistics* method returns a feature vector representation of this factor.

[5] `http://www.scala-lang.org/`

In the following, we will distinguish between two types of templates. The first one defines factors that touch a single event (described in Section 2.4). The second one covers factors that assess pairs of events (described in Section 2.5).

## 2.4 Single Event Template

Given an event as input, the unroll method of these templates returns factor containing only the event itself.

The sufficient statistics/features assess three aspects of an event: (a) its clue, (b) the combination of clue and each of the event's arguments, and (c) each pair of its arguments.

We describe these features as patterns. The corresponding feature vector is binary and has active components corresponding to the patterns present in the event variable that touches the factor.

Features measuring the *event clue* are created by conjoining different representations of the clue word with different representations of the event type. Table 2 shows the representations used. *String* yields the type name or clue word as is. *Stem* refers to the clue stem and *Dict* to test the membership in a dictionary. *Pre-Hyphen* returns the word at token $i-2$ if a hyphen is at $i-1$. *Normalized* yields "Regulation" for each regulation type, and the type name as is for each other type. *Any* fires for each possible type or clue. Features for event clues are generated from the cross-product of the above representations.

To assess the compatibility of event type and *clue* with an event *argument*, representations of the clue word, event type (as before), the dependency path between clue and argument head, the argument token itself, and the role of the argument are generated. These representations are then conjoined to form features. Table 2 lists the representations for each of these components under "Clue Argument". *Stem/Prot*, which returns a "PROT" if the argument is a protein, and the argument stem word otherwise. For dependency path representations we use: *Full*, which amounts to the full path (including directions) between clue and argument; *Start* and *End* which yield partial paths which start at the clue or end at the argument, respectively; *Length*, which returns the length of the path; *Norm.*, which normalizes the dependency path by replacing "conj_X" with "conj" labels (to generalize over conjunctions), and removes "appos" and "abbrev" edges from the path (as they are representing equivalence). Again, features are created from the cross-product of the above representations, and for each clue-argument pair in the event.

The above features cannot assess how compatible the *arguments* are to each other. For example, they cannot differentiate between a binding event at clue T with two arguments A and B, and two binding events at T, one with argument A and one with argument B. Therefore, argument pair features are created by conjoining different representations of the two arguments, and of the path between the two arguments. Table 2 shows the used representations under "Argument Pair". We include *1/2 Gram* representations of the dependency path, and *Rel Position*, which yields the relative positions of the first and second argument with respect to the clue word. The latter representation is motivated as follows. In a sentence such as "A binds with B and C" two events are expressed: A binds B and A binds C. Here arguments on the same side of the clue do not belong together, but arguments on opposite sides do. Features are again created from the cross-product, and for each unique (unordered) pair of arguments.

## 2.5 Event Pair Templates

There are two types of templates for factors between events. The first measures compatibility between certain pairs of events in the same sentence, the second compatibility between events in different sentences of the same document.

There are dependencies between regulation events and their argument events. For example, it is less likely to see positive regulations of negative regulations than it is to have positive regulations of gene expressions. Moreover, regulations tend to not share arguments with their themes, and if they do, these shared arguments would rather be themes than causes. This is assessed by a *parent-child* template. To capture these dependencies we define factors between regulation events (the parents) and the events they regulate (the children). For any event $e$ the method returns all pairs $(p, e)$ where $p$ is a parent of $e$, and all pairs $(e, c)$ where $c$ is a child of $e$. The sufficient statistics of this template measure the compatibility of the event types of parent and child, as well as their arguments and relative position to each other. In particular, the features test whether or not parent and child share an argument.

The idea behind the second of the event pair templates, the *document wide* template is to capture how the description of a gene changes throughout the document. To this end it connects events that concern the same genes (as determined by simple string match). Instead of pairwisely connecting all such events, we only connect those of directly succeeding gene mentions. This allows to model the transitions we mentioned in Section 1, and describe in more detail in Section 3.

For a given event $e$ the document wide template returns the pair $(p, e)$, where $p$ is an event that concerns a preceding mention of a protein in $e$. It also returns the corresponding pairs in forward direction. Note that the IDF approach helps us to make unrolling for this template very efficient: we can simply add an index to our data-structure that maps gene mentions to preceding mentions of the same gene. As sufficient statistics the template returns the conjunction of event types of both events. This allows to model transitions such as the one in Figure 2 that progresses from *Gene Expression* to *Positive Regulation*.

Table 2: Features defined on single events

| Event Clue | |
|---|---|
| Clue | Stem,Dict,Any,Pre-Hyphen |
| Type | String, Normalized |

| Clue Argument | |
|---|---|
| Type | String, Normalized |
| Clue | String, Stem, Dict |
| Path | Full, Split, Length, Norm. |
| Role | String |
| Arg | String/Prot, Stem, Dict |

| Argument Pair | |
|---|---|
| Arg1/2 | String/Prot |
| Type | String |
| Path | Full, 1/2 Gram, Length, Rel Position |

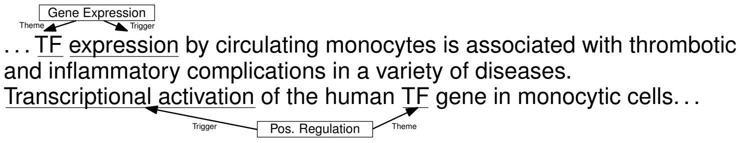

Gene Expression
Theme
Trigger
. . . TF expression by circulating monocytes is associated with thrombotic and inflammatory complications in a variety of diseases.
Transcriptional activation of the human TF gene in monocytic cells. . .
Trigger
Pos. Regulation
Theme

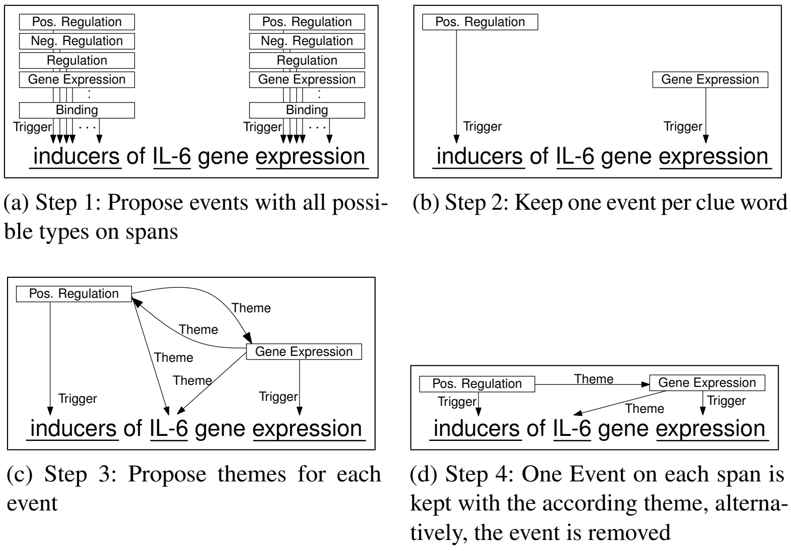
Fig. 2: Example text snippet depicting the transition of the event type from gene expression to positive regulation of the gene TF [15].

## 2.6 Sampling

We need to know how to move from one possible world (set of events) to other likely possible worlds. This is required for finding the MAP configuration, as well as for learning the parameters of our model. We will refer to this process as sampling, although technically it may be closer to *local search* because our models are discriminatively trained, and we do not take into consideration backward probabilities of our moves.

One iteration of the sampler consists of two phases. The first one processes each candidate clue span and proposes events. The second one considers existing events and proposes changes to these.

At the beginning, each span is either mentioning a gene or is empty. The sampler starts by proposing the generation of events with all possible event types (gene expression, phosphorylation, regulation etc.) on each span as a clue. Depending on model score, one of them is accepted. Next, we propose to attach one of available events or genes as an argument to the newly generated event. We take care to not propose invalid events such



(a) Step 1: Propose events with all possible types on spans

(b) Step 2: Keep one event per clue word

(c) Step 3: Propose themes for each event

(d) Step 4: One Event on each span is kept with the according theme, alternatively, the event is removed

Fig. 3: Proposing the generation of events with themes.

as gene expressions with multiple themes. Moreover, we take into account that events can only take other events as arguments if their event type is positive regulation, negative regulation or regulation. The sampler is forced to accept this event with one argument, or to remove it completely from the data structure. In this manner we guarantee that after the end of phase two there are no "dangling" events without themes. The above sampling steps are visualized in Figure 3.

In the latter phase, events which already have a theme can be further altered. This can happen in different ways: If the event to be changed is a (positive, negative, or general) regulation, a cause can be added which is a gene or event. If the event is a binding, more genes can be added as themes. Similarly, an argument can be proposed to be removed provided that at least one theme is remaining.

For each existing event we additionally propose to change the type while respecting the current configuration of arguments. If multiple themes are currently attached, a change is not possible because only bindings can be in such state. If a cause is currently attached and the type is to be changed to a non-regulation, the cause needs to be removed (because only regulations can have causes). Finally, we also propose to remove the complete event.

## 2.7 Learning and Objective Function

In order to learn the parameters of our model we apply SampleRank [23] . This allows to make parameter updates *within* inference and hence to speed-up convergence. A crucial component in the SampleRank framework is the objective function.

Our objective function consists of two parts. The first part assesses event types and their arguments in isolation. This means that getting the type wrong, but one argument right, still gives partial credit. Let $e$ be the event to be evaluated and $\mathbf{1}_{\mathrm{ClueType}} = 1$ iff the clue and type is correct. Let $\mathrm{TP}_{\mathrm{ArgClue}}$ the number of correct arguments provided that clue and type are correct, and $\mathrm{TP}_{\mathrm{Arg}}$ the number of correct argument pairs regardless of the correct clue and type. Note that the latter can differentiate between two events with one argument each, and one event with both of the arguments.

With the above definitions we use as a measure for the correctness of an event $\mathrm{TP}(e)$ ($\mathrm{FP}(e)$ analogously):

$$\mathrm{TP}(e) = \mathbf{1}_{\mathrm{ClueType}} + \mathrm{TP}_{\mathrm{ArgClue}} + \mathrm{TP}_{\mathrm{Arg}} \, .$$

The objective function $f(e)$ to be evaluated on event $e$ is then

$$f_1(e) = \mathrm{TP}(e) - \mathrm{FP}(e) \, .$$

This objective function cannot measure the global goodness of events. For example, it rewards duplicate true positives. Therefore, we additionally use an objective function which is to be evaluated on a per-sentence basis. This objective is simply

$$f_2(s) = \mathrm{TP}_{\mathrm{ClueTypeTheme}} - \mathrm{FP}_{\mathrm{ClueTypeTheme}}$$

and rewards the true events respecting the clue, event type and theme. In particular, this addresses the generation of duplicate events. If a proposal changes sentence and event, objective is $f(\cdot) = f_1(\cdot) + f_2(\cdot)$.

## 3 Results

The data provided by the BioNLP Shared Task 2009 organizers is divided into several sets: A training set with 800 abstracts, a development set with 150 abstracts, and a test set with 260 abstracts. The gold data for the latter is not available but predictions on the test abstracts can be evaluated online once on a day on the data publishers website.[6] In this way, an adaption of a system to the test set is not possible.

We evaluate different subsets of templates using the development set. Of special interest is the impact of our proposed global features that capture (a) correlations between parent and child events and (b) the transitions of event types for events with the same gene argument throughout the text. We also assess the impact of our argument pair features. To our knowledge, these have not yet been applied in the context of biomedical event extraction.[7]

To get a better sense of the document-wide transitions between events that concern the same genes, we show an event type transition matrix on the development set in Table 3. As expected, in general self-transitions from and to the same state are the most probable for most of the event types (*Gene Expression*, *Localization*, *Phosphorylation*, *Protein Catabolism*, and *Negative Regulation*).

More interesting are transitions between *different states* that have a high probability. Transitions from *Regulation* to *Positive Regulation* are frequent (probability $0.44$) which point to concretions in the text. Interestingly, the transition to *Negative Regulation* is comparatively low (with only $0.08$). Eye-catching is also the transition between *Transcription* and *Positive Regulation* with $0.44$.

The results on the development set (using the approximate span, recursive evaluation used in the competition to rank the participants) are shown in Table 4. Our best configuration including the templates with features explained in Sections 2.4 and 2.5

[6] `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/eval-test.shtml`

[7] Note that they do resemble sibling features in dependency parsing. In this same context parent-child features are closely related to so called grandparent features.

Table 3: Markov Chain-like transitions describing the change of the events concerning the same gene throughout the document. Transitions to or from gene mentions outside any event are not shown.

| | GE | L | Ph | PC | T | B | R | PR | NR |
|---|---|---|---|---|---|---|---|---|---|
| Gene expression | **0.53** | 0.02 | 0.01 | 0.01 | 0.05 | 0.10 | 0.04 | 0.19 | 0.06 |
| Localization | 0.20 | **0.43** | | | | 0.13 | 0.03 | 0.10 | 0.10 |
| Phosphorylation | 0.03 | | **0.57** | 0.07 | | 0.27 | 0.03 | 0.03 | |
| Protein Catabolism | 0.08 | | 0.33 | **0.42** | | 0.17 | | | |
| Transcription | 0.22 | 0.07 | | | 0.11 | 0.02 | 0.13 | **0.44** | 0.02 |
| Binding | 0.10 | 0.03 | 0.03 | 0.02 | 0.03 | **0.60** | 0.06 | 0.07 | 0.04 |
| Regulation | 0.10 | 0.01 | 0.02 | | 0.03 | 0.23 | 0.08 | **0.44** | 0.08 |
| Positive Regulation | **0.12** | 0.01 | | | 0.04 | **0.12** | 0.11 | 0.07 | |
| Negative Regulation | 0.16 | 0.01 | | | 0.02 | 0.02 | 0.05 | 0.31 | **0.42** |

yields an $F_1$ score of 55.6 on the development set, in range of the currently best performing system [19]. The impact of the global features presented in Section 2.5 is shown by removing each template: The Argument-Pair template has an impact of 1.1 in $F_1$ (significant[8] for regulations $p \leq 0.03$), the detection of an event-argument structure between two events contributes 0.6 (significant for negative regulations, $p \leq 0.04$). Detecting the transition of event types of events which have the same gene mention has an impact of 1.2 proving such approach to be valuable on a competitive system. We observe the highest impact for regulations (*Regulation*: 0.74, *Positive Regulation*: 1.83, *Negative Regulation:* 3.46, All Regulations: 1.94, results are significant, $p \leq 0.02$ for all regulations, for all events, $p \leq 0.07$ holds).[9] This is consistent with our expectation: Especially regulations are highly probable to be the successor of several other event types.

A decrease in performance can be observed for *Protein Catabolism*. This type has the lowest number of annotations in the development set (20). Learning a global feature for such low-frequent types is generally error-prone.

The overall increase in performance does not involve a problematic runtime raise. Training on the training set with 20 iterations together with 20 iterations for prediction takes 101 minutes (on a single core of a Quad Core AMD Opteron CPU with 2.3 GHz). Without the event wide template, the runtime is still 92 minutes. The reason for this limited increase in runtime is that per change the document wide template only instantiates two factors: one for the previous event with same gene, one for the following event with same gene.

The results on the test set with the best configuration on the development set has near state-of-the-art results with an $F_1$ measure of 49.6 (cf. Table 5). Interestingly, akin to [16] who achieve 50.0 $F_1$, we observe a 6% drop from development to test set.

[8] Tested via sign-test [7].

[9] Full results per entity type for all four configurations are omitted for brevity.

Table 4: Results for different configurations on the development set and comparison with other approaches using the approximates span and recursive evaluation. The $*$ denotes a significant contribution of the template ($\alpha = 0.05$) to regulations. $F_1$ shows the overall performance, $F_1$ Reg. for all regulations.

| Configuration | Precision | Recall | $F_1$ | $F_1$ Reg. | |
|---|---|---|---|---|---|
| No Arg-Pair | 68.4 | 45.3 | 54.5 | 43.6 | $*$ |
| No parent event | 68.9 | 45.8 | 55.0 | 43.8 | |
| No doc-wide | 68.3 | 45.3 | 54.4 | 43.6 | $*$ |
| Best | 68.5 | 46.7 | 55.6 | 45.6 | |
| Miwa 2010 [14] | – | – | 55.6 | | |
| Riedel 2010 [19] | 67.9 | 51.8 | 58.7 | | |

Table 5: Results on the test set

| Event Class | Prec. | Rec. | $F_1$ |
|---|---|---|---|
| Gene Expression | 78.7 | 62.7 | 69.8 |
| Transcription | 71.0 | 16.1 | 26.2 |
| Protein Catabolism | 85.7 | 42.9 | 57.1 |
| Phosphorylation | 79.3 | 79.3 | 79.3 |
| Localization | 93.3 | 40.2 | 56.2 |
| Binding | 56.7 | 34.0 | 42.5 |
| Regulation | 45.0 | 23.0 | 30.6 |
| Positive Regulation | 56.9 | 31.8 | 40.8 |
| Negative Regulation | 51.5 | 31.1 | 38.8 |
| Total | 65.0 | 40.0 | 49.6 |

## 4 Discussion and Future Work

This paper presented the use of imperatively defined factor graphs to incorporate novel global features into an event extraction model. The global feature with the biggest gain of $1.2$ in total $F_1$ (and $3.46$ F1 for the event class of negative regulations) is based on a document-wide factor template contributing significantly to the recognition of events. It assesses progression of event types for events that concern the same gene.

A further idea would be to model the explanation of an event across a document in a more detailed fashion. For example, often the regulation of a gene A is mentioned first, and this is expanded to a regulation of an expression of gene A. In fact, we also assembled the transition matrix for this scenario, and it resembles the one in Table 3. However, we could not produce a performance gain with a template that we designed to capture the properties of this matrix. A reason may be that the transitions of interest are not as frequent as the ones in Table 3, and hence harder to exploit.

The described document-wide template concerns mentions of the same gene; until now, we used exact string match to identify these mentions. Obviously, this approach can be improved by using a more sophisticated co-reference approach, which will hopefully increase the impact of this template. So far our models have only been applied to abstracts. Moving to full papers may also increase the impact of document wide features.

## Acknowledgements

This work has been performed during a visit of Roman Klinger at the University of Massachusetts, partially financed by a scholarship of the Germany Acadamic Exchange Service (DAAD). This publication is partically funded in the context of the Positive-Spaces project (`http://www.positivespaces.eu/`) by the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 248726.

## References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature Genetics 25(1), 25–29 (May 2000)
2. Bañeres, B., Cesareni, G., Hirschman, L., Krallinger, M., Leitner, F., Valencia, A. (eds.): Proceedings of the BioCreative II.5 Workshop. CNIO, Madrid, Spain (2009)
3. Bjorne, J., Heimonen, J., Ginter, F., Airola, A., Pahikkala, T., Salakoski, T.: Extracting complex biological events with rich graph-based feature sets. In: Proceedings of the Workshop on BioNLP: Shared Task. pp. 10–18. Association for Computational Linguistics, Boulder, Colorado (June 2009)
4. Culotta, A., McCallum, A.: Tractable Learning and Inference with High-Order Representations. In: International Conference on Machine Learning (ICML) Workshop on Open Problems in Statistical Relational Learning (2006)
5. Hirschman, L., Krallinger, M., Valencia, A. (eds.): Proc. of the Second BioCreative Challenge Evaluation Workshop. Centro Nacional de Investigaciones Oncologicas, CNIO (2007)
6. Hirschman, L., Yeh, A., Blaschke, C., Valencia, A.: Overview of BioCreAtIvE: critical assessment of information extraction for biology. BMC Bioinformatics 6 Suppl 1, S1 (2005)

7. Kanji, G.K.: 100 Statistical Tests. Sage (2006)
8. Kim, J.D., Ohta, T., Pyysalo, S., Kano, Y., Tsujii, J.: Overview of BioNLP-09 Shared Task on Event Extraction. In: Proceedings of the Workshop on BioNLP: Shared Task. pp. 41–49. Association for Computational Linguistics, Boulder, Colorado (June 2009)
9. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Trans on 47(2), 498–519 (Feb 2001)
10. McCallum, A., Rohanimanesh, K., Wick, M., Schultz, K., Singh, S.: FACTORIE: Efficient Probabilistic Programming via Imperative Declarations of Structure, Inference and Learning. In: NIPS Workshop on Probabilistic Programming (2008)
11. McCallum, A., Schultz, K., Singh, S.: FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs. In: Advances on Neural Information Processing Systems (NIPS) (2009)
12. McClosky, D., Surdeanu, M., Manning, C.D.: Event extraction as dependency parsing in bionlp 2011. In: BioNLP 2011 Shared Task (2011)
13. Milch, B., Marthi, B., Russell, S.: BLOG: Relational Modeling with Unknown Objects. Ph.D. thesis, University of California, Berkeley (2006)
14. Miwa, M., Pyysalo, S., Hara, T., Tsujii, J.: Evaluating Dependency Representation for Event Extraction. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). p. 779787 (2010)
15. Oeth, P., Mackman, N.: Salicylates inhibit lipopolysaccharide-induced transcriptional activation of the tissue factor gene in human monocytic cells. Blood 86(11), 4144–4152 (Dec 1995)
16. Poon, H., Vanderwende, L.: Joint Inference for Knowledge Extraction from Biomedical Literature. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT) (2010)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
18. Riedel, S., Chun, H.W., Takagi, T., Tsujii, J.: A Markov Logic Approach to Bio-Molecular Event Extraction. In: Proceedings of the Workshop on BioNLP: Shared Task. pp. 41–49. Association for Computational Linguistics, Boulder, Colorado (June 2009)
19. Riedel, S., McCallum, A.: Fast and robust joint models for biomedical event extraction. In: Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '11) (2011), to appear
20. Rohanimanesh, K., Wick, M., McCallum, A.: Inference and Learning in Large Factor Graphs with a Rank Based Objective. Tech. Rep. UM-CS-2009-08, University of Massachusetts, Amherst (2009)
21. Singh, S., Schultz, K., McCallum, A.: Bi-directional Join Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2009)
22. Vlachos, A., Craven, M.: Search-based structured prediction applied to biomedical event extraction. In: Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL' 11) (2011)
23. Wick, M., Rohanimanesh, K., Bellare, K., Culotta, A., McCallum, A.: SampleRank: Training factor graphs with atomic gradients. In: Proceedings of the International Conference on Machine Learning (ICML) (2011)
24. Zhang, Y., Broser, M., Rom, W.N.: Activation of the interleukin 6 gene by Mycobacterium tuberculosis or lipopolysaccharide is mediated by nuclear factors NF-IL6 and NF-kappa B. Proc Natl Acad Sci U S A 91(6), 2225–2229 (Mar 1994)