Modeling Exclusion with a Differentiable Factor Graph Constraint

Jason Naradowsky [1] **Sebastian Riedel** [2]

Abstract

With the adoption of general neural network architectures, many researchers have opted to trade informative priors for powerful models and big data. However, for many structured prediction tasks the complex relationships between variables in the output space are often difficult to learn from the available data alone.

Such relationships often centre around the notion of *exclusion*: that predicting one structure prohibits the prediction of others. In this work we formulate a differentiable factor graph exclusion constraint to incorporate this prior belief into neural end-to-end architectures. We demonstrate the effectiveness of this method in the context of extracting event information from clusters of related news articles, and introduce meta-inference learning to determine the ideal number of inference iterations to simulate.

1. Introduction

Exclusion is a fundamental aspect of many structured prediction problems in NLP. In a constituent tree, the presence of one span *excludes* the existence of those that would cross it. In a semantic role frame, labeling an argument with a specific role (almost always) prohibits its ability to perform the same role in a different frame. And, as we explore in this paper, when extracting events from news articles, a single value is not often associated with more than one field.

Factor graphs provide an intuitive framework for incorporating such structured constraints, and can be paired with well-studied message-passing inference algorithms. Earlier work on factor graph models introduced hard/combinatorial constraint factors (Smith & Eisner, 2008). Both enforce hard constraints on their neighbouring variables, the former using discrete-valued potential tables, and the latter replacing the standard out-going message-passing computations with specialised algorithms.

In this work we show how to utilise such factor graph models to incorporate useful exclusion constraints into an end-to-end differentiable machine reading system for event extraction from news clusters. For each event, a schema specifies which types of information to extract. Here exclusion plays an important role: each value should be associated with only one slot in the schema, and each slot should take only one value. We model this behaviour using a set of combinatorial factors. Following recent work in computer vision, inference under this constraint is treated as an RNN, each move in time corresponding to one iteration of loopy belief propagation (LBP). Enforcing this constraint improves extraction by 3.6 F_1 over the unconstrained model, which itself is a significant improvement over the previous state-of-the-art.

2. Model

We begin by defining terminology. A news cluster c is a set of documents, $\{d_1, ..., d_{|c|}\} \in c$, and is described by a sequence of words, $d = (w_1, ..., w_{|d|})$. Documents within a cluster pertain to the same event, and the task is to predict a value for each of a number of pre-defined slots, describing the event. In our domain slots refer to pieces of information which describe a plane crash event (SURVIVORS, OPERATOR, CRASH-SITE, etc., with values such as '68', 'Trans American', and 'Chicago').

A *mention* is an occurrence of a value in its textual context. For each value $v \in V$, there are potentially many mentions of v in the cluster, defined as $m \in M(v)$.

Our system is an aggregation-based neural reading model (Kadlec et al., 2016), which (1) constructs a low-dimensional representation of each mention m in context, (2) computes a $\phi_{\text{mention}}(m, s)$ score assessing the compatibility of each mention with each slot s, and (3) maps them, through aggregation, into value-level scores, $\phi_{\text{value}}(v, s)$, where a global constraint can optionally be applied prior to calculating the loss or decoding.

Representation For each mention m we construct a d-dimensional representation $r(m) \in \mathbb{R}^d$ of the mention in its

[1]Department of Theoretical and Applied Linguistics, University of Cambridge, Cambridge, UK [2]University College London, London, UK. Correspondence to: Jason Naradowsky <jrn39@cam.ac.uk>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

context. This representation functions as a general "reading" or encoding of the mention, irrespective of the type of slots for which it will later be considered.

We first construct an embedding matrix $E \in \mathbb{R}^{n \times e}$, using pre-trained word embeddings, where e is the dimensionality of the embeddings and n the number of words in the cluster. These are held fixed during training. All mentions are masked and receive the same one-hot vector in place of a pretrained embedding.

From this matrix we embed the mention context, yielding $r(m)$, using a two-layer convolutional neural network (CNN), with a detailed discussion of the architecture parameters provided in Section 3. CNNs have been used in a similar manner for a number of information extraction and classification tasks (Kim, 2014; Zeng et al., 2015) and are capable of producing rich sentence representations (Kalchbrenner et al., 2014).

Scoring Having produced a representation $r(m)$ for each mention m, a slot-specific attention mechanism produces $\phi_{\text{mention}}(m, s) \in \mathbb{R}$, representing the compatibility of mention m with slot s. Let $R \in \mathbb{R}^{n \times d} = r(m_0) \oplus ... \oplus r(m_n)$, the concatenation of all $r(m)$, and let $k(m)$ be the index of m into R. We create a separate embedding, $\pi^s \in \mathbb{R}^r$, for each slot s, and utilize it to attend over all mentions in the cluster as follows:

$$a^s = \text{softmax}(R\pi^s) \tag{1}$$

$$\phi_{\text{mention}}(m, s) = a^s_{k(m)} \tag{2}$$

The mention-level scores reflect *an interpretation* of the value's encoding with respect to the slot. The softmax normalizes the scores over each slot, supplying the attention, and creating competition between mentions. This encourages the model to attend over mentions with the most characteristic contexts for each slot.

Aggregation For values mentioned repeatedly throughout the news cluster, mention scores must be aggregated to produce a single value-level score. If the data set is noisy with numerous spurious mentions, as it is in our case, a summation-based aggregation, as opposed to a max, may better exploit the redundancy of the true signal in the data:

$$\phi_{\text{value}}(v, s) = \sum_{m \in M(v)} \phi_{\text{mention}}(m, s) \tag{3}$$

We also utilise aggregation as a means to model the absence of a value in the text. A common solution for handling such missing values is the use of a threshold, below which the model returns null. However, the belief regarding whether a passage of text mentions a particular piece of information should be a factor of how large the passage of text is, and how much it discusses related concepts. For this reason we posit a null value whose score is the aggregation of all mentions which are not entities.

2.1. Global Exclusion Constraints

The combination of learned representations and aggregation produces an effective system in its own right, but its predictions may reflect a lack of world knowledge. For instance, we may want to discourage the model from predicting the same value for multiple slots, as this is not a common occurrence. However, the constraint must be implemented in a differentiable manner to allow gradients to propagate back down through the various model components. There are many ways to implement such a constraint, but because our training set is small and contains only 33 (often incomplete) event instances, we wish to model this behaviour with as little additional parameterisation as possible. This rules out learning such output dependencies from data alone (Kalchbrenner et al., 2016).

Exclusion as a Factor Graph A factor graph is a type of graphical model which factorizes the model function using a bipartite graph, consisting of variables and factors. Variables maintain beliefs over their values, and factors specify scores over configurations of these values for the variables they neighbor. To implement an exclusion constraint in a factor graph, factors must prohibit their neighboring variables from taking values which would violate the constraint logic.

An EXACTLY-1 factor (Smith & Eisner, 2008) constrains its neighbouring Boolean variables such that *exactly one* must be true, and can be used as the basis for such a constraint. Let the Boolean variable $x_{v,s}$ represent the probability of slot s taking the value v. Let each $x_{v,s}$ be connected to a local factor $u_{v,s}$ whose initial potential is derived from $\phi_{\text{value}}(v, s)$. An EXACTLY-1 factor, is created for (1) each slot, connected across all values, and (2) each value, connected across all slots. This is illustrated in Figure 1. Note that this architecture contains cycles, and therefore exact inference in a single forward-backward pass is not guaranteed.

Belief Propagation as an RNN The connections between backpropagation and message-passing inference are well-studied (Eisner, 2016). Tatikonda & Jordan (2002) drew connections between LBP convergence and properties of the corresponding Gibbs measures by decomposing inference into a series of simpler operations. By creating differentiable decompositions of these operations, Ross et al. (2011) and Stoyanov et al. showed that parameters could be optimised by backpropagating through belief propagation. Gormley et al. (2015) used backpropagation on mod-

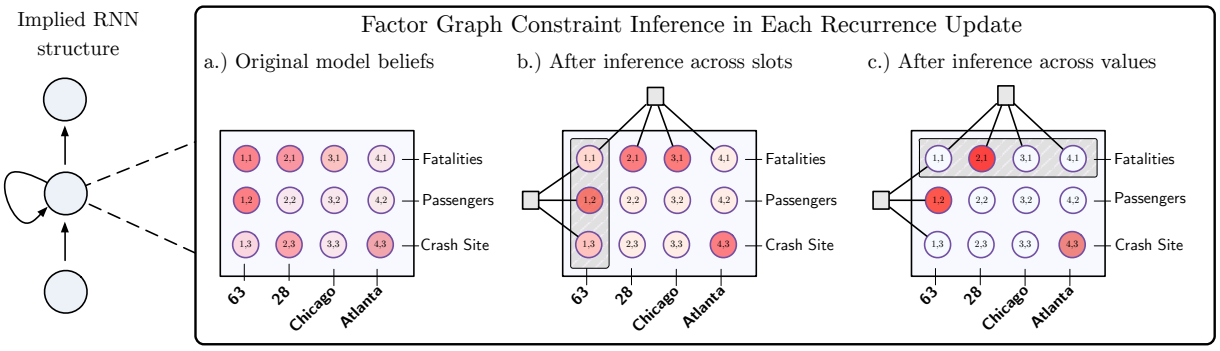

Figure 1. Belief propagation constraint inference as an RNN. Red indicates a true value. At each step of LBP inference, the belief of each variable is updated with respect to the two EXACTLY-1 factors it neighbors, pushing it closer to discrete values. In convergence (c.), values reflect the desired constraint.

els with combinatorial (tree-structured) factors, and more recently such structured factors have also been incorporated into attention mechanisms (Kim et al., 2017).

Our approach follows recent work in computer vision which casts belief propagation as an RNN (Zheng et al., 2015). The output of previous layers, a slot-by-value matrix, is fed into a specialized RNN whose hidden state reflects the model's variable beliefs, and where each time step corresponds to an iteration of LBP inference in the factor graph model.

Inference Inference is performed using two types of messages: from variables to factors and from factors to variables. The standard message a variable x sends to a factor f (denoted $\mu_{x\to f}$) is defined recursively w.r.t. to incoming messages from its neighbors $n(x)$:

$$\mu_{x\to f} = \prod_{f'\in n(x)\neq f} \mu_{f'\to x} \tag{4}$$

and conveys the information "My other neighbors jointly suggest I have the posterior distribution $\mu_{v\to u}(v)$ over my values." We replace this initial out-going message for variable $x_{v,s}$ with the corresponding output of the aggregation network:

$$\mu_{x_{v,s}\to f} = \text{sigmoid}(\phi_{\text{value}}(v,s)) \tag{5}$$

where the sigmoid moves the scores into probability space. All subsequent LBP iterations compute variable messages as in Eq. 4, incorporating the out-going factor beliefs of the previous iteration.

Exactly-1 A second message type is sent from an EXACTLY-1 factor to its neighboring variables, conveying the meaning "My other neighbors jointly suggest that you have the posterior distribution $m_{Exactly-1\to v}$ under the constraint that exactly one of my neighbors must be True". See Smith (2010) for a derivation of the out-going message computation for EXACTLY-1:

$$\neg\mu_{x\to f} = 1.0 - \mu_{x\to f} \tag{6}$$

$$Z = \prod_{x} \neg\mu_{x\to f} \tag{7}$$

$$\mu_{\text{EXACTLY-1}\to x} = \frac{Z}{\neg\mu_{x\to f}} \tag{8}$$

At each time step we perform either Eq. (4) or Eq. (5), and Eq. (6)-(8) to simulate one iteration of LBP.

Meta-Inference The view of LBP as an unrolled RNN creates interesting opportunities in meta-inference. For instance, assuming T is the number of LBP inference steps, one can introduce a set of weights $w \in \mathbb{R}^{T+1}$. Using w as a set of mixture weights over the slot-by-value matrices produced during LBP inference, and averaging the resulting weighted matrices, enables the model to learn approximately how many iterations of LBP to apply. This can be viewed as an alternative to other soft constraint methods which encourage, but do not outright restrict, constraint violations (Anzaroot et al., 2014).

3. Experiments

In all experiments we train using adaptive online gradient updates (Adam, see Kingma & Ba (2014)). Model architecture and parameter values were tuned on the development set, and are as follows (chosen values in bold):

- CNN filter width, L1: [3, 5, 8, **10**], L2: [3, **5**, 8, 10]
- CNN dim: [5, **10**, 15, 20], L2: [5, **10**, 15, 20]
- learning rate: [0.001, **0.003**, 0.005, 0.01]

- L2 regularization: [0.001, 0.003, 0.005, **0.01**]
- dropout rate: 1-[0.5, 0.6, 0.7, **0.8**, 0.9, 1.0]

Pre-trained word embeddings are 200-dimensional GLoVe embeddings (Pennington et al., 2014). We decode by taking the highest scoring value per slot.

3.1. Data

The Stanford Plane Crash Dataset (Reschke et al., 2014) is a small data set consisting of 80 plane crash events, each paired with a set of related news articles. Of these events, 40 are reserved for training, and 40 for testing, with the average cluster containing more than 2,000 mentions. Gold labels for each cluster are derived from Wikipedia infoboxes and cover up to 15 slots, of which 8 are used in evaluation.

We follow the same entity normalization procedure as (Reschke et al., 2014), limit the cluster size to the first 200 documents, and further reduce the number of duplicate documents to prevent biases in aggregation. We partition out every fifth document from the training set to be used as development data, primarily for use in an early stopping criterion.

3.2. Evaluation

We evaluate our proposed architecture in the manner set forth by Reschke et al. (2014), which is a modified form of F_1. Candidate values are proposed by the Stanford CoreNLP NER system (Manning et al., 2014) and are included in the data set release. Similarly, the second evaluation measure we present is standard precision, recall, and F_1, specifically for null values.

3.3. Results

In Table 1 we show the results of our system. The model without the constraint outperforms the previous best reported result on this data by 7.0 F_1, despite using less linguistic annotation and no hand-engineered features. This is in part attributable to how our system optimizes its representations through a cluster-wide pooling of evidence, and makes use of a more versatile cluster and slot-specific null threshold. Applying the constraint further improves performance by 3.6 F_1.

Additional iterations reduce performance. While this may appear counter-intuitive, it is the result of the constraint assumption not holding absolutely in the data. For instance, multiple slots can take the null value, and zero commonly occurs in multiple slots within the same event, yet the constraint attempts to discourage such predictions. Running constraint inference for a single iteration encourages the intended 1-to-1 mapping between values and slots, but it does not prohibit it. This result also implies that a hard heuristic decoding constraint would not be as effective.

	Score			Nulls		
Model	**P**	**R**	**F_1**	**P**	**R**	**F_1**
CRF	15.9	42.5	23.2	–	–	–
Noisy-OR	18.7	37.0	24.8	–	–	–
Searn	24.5	38.6	30.0	–	–	–
0	32.1	43.7	37.0	53.7	41.3	46.7
1	35.1	48.2	**40.6**	51.6	38.8	44.3
2	33.6	45.3	38.6	54.8	40.8	**46.7**
3	33.3	43.2	37.6	61.1	34.9	44.4
BP-Mix	35.1	48.2	**40.6**	51.6	38.8	44.3

Table 1. Results on the Stanford Plane Crash Dataset.

One solution to this problem is to soften the application of the constraint. Our preliminary results in meta-inference learning fail to improve significantly over choosing a single value for this parameter from development data. In fact, the decoded output produced the same predictions, and $w = [0.13, 0.93, 0.00, 0.03]$. To recover a more traditional interpretation of the optimal number of iterations to use, mixture weights may be treated as a discrete-valued variable with a variational approximation to the categorical distribution (Jang et al., 2016), but learning a weighted average inference stages is a unique aspect of our approach.

A noteworthy finding is that naively training such an end-to-end system performs poorly. We attribute this to the assistance that the global constraint provides in removing noise from the slot-by-value matrix, allowing the model to achieve higher scores with poorer representations. Thus we pre-train the model without the constraint for 30 iterations (approximately to the point of convergence on development data) before adopting full end-to-end training. Here the mixture model approach to inference is also applicable: w may be initialised to $[1.0, 0.0, ...]$, and slowly allowed to assign more weight to successive iterations of inference as training progresses, mitigating the strong effect of the constraint in earlier iterations of training.

4. Conclusion and Future Work

In this work we present a machine reading architecture designed to effectively read collections of news documents, and utilise differentiable exclusion constraints to promote more realistic predictions across multiple slots in an event schema. We view this as a step towards more general end-to-end training with highly-structured models, where back-propagation is used to enforce constraints that are specified in a probabilistic interpretation of the world.

References

Anzaroot, Sam, Passos, Alexandre, Belanger, David, and McCallum, Andrew. Learning soft linear constraints with application to citation field extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 593–602, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Eisner, Jason. Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the EMNLP Workshop on Structured Prediction for NLP*, Austin, TX, November 2016.

Gormley, Matthew, Dredze, Mark, and Eisner, Jason. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3, 2015.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax, 2016. URL http://arxiv.org/abs/1611.01144.

Kadlec, Rudolf, Schmid, Martin, Bajgar, Ondej, and Kleindienst, Jan. Text understanding with the attention sum reader network. In *Association for Computational Linguistics (ACL)*, 2016.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Kalchbrenner, Nal, Danihelka, Ivo, and Graves, Alex. Grid long short-term memory. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2016.

Kim, Yoon. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

Kim, Yoon, Denton, Carl, Hoang, Luong, and Rush, Alexander M. Structured attention networks. *CoRR*, abs/1702.00887, 2017. URL http://arxiv.org/abs/1702.00887.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Reschke, Kevin, Jankowiak, Martin, Surdeanu, Mihai, Manning, Christopher D., and Jurafsky, Dan. Event extraction using distant supervision. In *Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC)*, 2014.

Ross, Stephane, Munoz, Daniel, Hebert, Martial, and Bagnell, J. Andrew. Learning message-passing inference machines for structured prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Smith, David and Eisner, Jason. Dependency parsing by belief propagation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 145–156, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Smith, David A. *Efficient Inference for Trees and Alignments: Modeling Monolingual and Bilingual Syntax with Hard and Soft Constraints and Latent Variables*. PhD thesis, Johns Hopkins University, Baltimore, MD, October 2010.

Stoyanov, Veselin, Ropson, Alexander, and Eisner, Jason. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *JMLR Workshop and Conference Proceedings*, pp. 725–733.

Tatikonda, Sekhar C. and Jordan, Michael I. Loopy belief propagation and gibbs measures. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, 2002.

Zeng, Daojian, Liu, Kang, Chen, Yubo, and Zhao, Jun. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Zheng, Shuai, Jayasumana, Sadeep, Romera-Paredes, Bernardino, Vineet, Vibhav, Su, Zhizhong, Du, Dalong, Huang, Chang, and Torr, Philip. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015.