CMU at SemEval-2016 Task 8: Graph-based AMR Parsing with Infinite Ramp Loss

Jeffrey Flanigan[♠] **Chris Dyer**[♠] **Noah A. Smith**[♡] **Jaime Carbonell**[♠]
[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
{jflanigan,cdyer,jgc}@cs.cmu.edu, nasmith@cs.washington.edu

Abstract

We present improvements to the JAMR parser as part of the SemEval 2016 Shared Task 8 on AMR parsing. The major contributions are: improved concept coverage using external resources and features, an improved aligner, and a novel loss function for structured prediction called infinite ramp, which is a generalization of the structured SVM to problems with unreachable training instances.

1 Introduction

Our entry to the SemEval 2016 Shared Task 8 is a set of improvements to the system presented in Flanigan et al. (2014). The improvements are: a novel training loss function for structured prediction, which we call "infinite ramp," new sources for concepts, improved features, and improvements to the rule-based aligner in Flanigan et al. (2014). The overall architecture of the system and the decoding algorithms for concept identification and relation identification are unchanged from Flanigan et al. (2014), and we refer readers seeking a complete understanding of the system to that paper.

2 New Concept Fragment Sources and Features

The concept identification stage relies on a function called $clex$ in Section 3 of Flanigan et al. (2014) to provide candidate concept fragments. In that work, $clex$ has three sources of concept fragments: a lexicon extracted from the training data, rules for named entities identified by the named entity tagger, and rules for time expressions. We augment these sources with five additional sources:

- **Frame file lookup**: for every word in the input sentence, if the lemma matches the name of a frame in the AMR frame files (with sense tag removed), we add the lemma concatenated with "-01" as a candidate concept fragment.
- **Lemma**: for every word in the input sentence, we add the lemma of the word as a candidate concept fragment.
- **Verb pass-through**: for every word in the input sentence, if the word is a verb, we add the lemma concatenated with "-00" as a candidate concept fragment.
- **Named entity pass-through**: for every span of words of length 1 until 7 in the input, we add the concept fragment "(thing :name (name :op1 word1 ... :opn word$n$)" as a candidate concept fragment, where n is the length of the span, and "word1" and "word$n$" are the first and last words in the fragment.

We use the following features for concept identification:

- **Fragment given words**: Relative frequency estimates of the probability of a concept fragment given the sequence of words in the span.
- **Length** of the matching span (number of tokens).
- **Bias**: 1 for any concept graph fragment.

Proceedings of SemEval-2016, pages 1202–1206,
San Diego, California, June 16-17, 2016.

- **First match**: 1 if this is the first place in the sentence that matches the span.
- **Number**: 1 if the span is length 1 and matches the regular expression "[0-9]+".
- **Short concept**: 1 if the length of the concept fragment string is less than 3 and contains only upper or lowercase letters.
- **Sentence match**: 1 if the span matches the entire input sentence.
- **; list**: 1 if the span consists of the single word ";" and the input sentence is a ";" separated list.
- **POS**: the sequence of POS tags in the span.
- **POS and event**: same as above but with an indicator if the concept fragment is an event concept (matches the regex ".*-[0-9][0-9]").
- **Span**: the sequence of words in the span if the words have occurred more than 10 times in the training data as a phrase with no gaps.
- **Span and concept**: same as above concatenated with the concept fragment in PENMAN notation.
- **Span and concept with POS**: same as above concatenated with the sequence of POS tags in the span.
- **Concept fragment source**: indicator for the source of the concept fragment (corpus, NER tagger, date expression, frame files, lemma, verb-pass through, or NE pass-through).
- **No match from corpus**: 1 if there is no matching concept fragment for this span in the rules extracted from the corpus.

The new sources of concepts complicate concept identification training. The new sources improve concept coverage on held-out data but they do not improve coverage on the training data since one of the concept sources is a lexicon extracted from the training data. Thus correctly balancing use of the training data lexicon versus the additional sources to prevent overfitting is a challenge.

To balance the training data lexicon with the other sources, we use a variant of cross-validation. During training, when processing a training example in the training data, we exclude concept fragments extracted from the same section of the training data. This is accomplished by keeping track of the training instances each phrase-concept fragment pair was extracted from, and excluding all phrase-concept fragment pairs within a window of the current training instance. In our submission the window is set to 20.

While excluding phrase-concept fragment pairs allows the learning algorithm to balance the use of the training data lexicon versus the other concept sources, it creates another problem: some of the gold standard training instances may be unreachable (cannot be produced), because of the phrase-concept pair need to produce the example has been excluded. This can cause problems during learning. To handle this, we use a generalization of structured SVMs which we call "infinite ramp." We discuss this in the general framework of structured prediction in the next section.

3 Infinite Ramp Loss

The infinite ramp is a new loss function for structured prediction problems. It is useful when the training data contains outputs that the decoder cannot produce given their inputs (we refer to these as "**unreachable examples**"). It is a direct generalization of the SVM loss and latent SVM loss.

Let x be the input, $\mathcal{Y}(x)$ be the space of all possible outputs given the input x, and $\hat{y}$ be the predicted output. Let $\mathbf{f}(x, y')$ denote the feature vector for the output y' with the input x, which is the sum of the local features. (In concept identification, the local features are the features computed for each span, and $\mathbf{f}$ is the sum of the features for each span.) Let $\mathbf{w}$ be the parameters of a linear model, used to make predictions as follows:

$$\hat{y} = \arg\max_{y' \in \mathcal{Y}(x)} \mathbf{w} \cdot \mathbf{f}(x, y')$$

To train the model parameters $\mathbf{w}$, a function of the training data is minimized with respect to $\mathbf{w}$. This function is a sum of individual training examples' losses L, plus a regularizer:

$$L(\mathcal{D}; \mathbf{w}) = \sum_{(x_i, y_i) \in \mathcal{D}} L(x_i, y_i; \mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

$$L(x_i, y_i; \mathbf{w}) = -\left(\lim_{\alpha \to \infty} \max_{y \in \mathcal{Y}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, y) + \alpha \cdot \left[\overbrace{\min_{y'' \in \mathcal{Y}(x_i)} cost(y_i, y'')}^{C(x_i, y_i)} - cost(y_i, y)\right]\right)\right)$$
$$+ \max_{y' \in \mathcal{Y}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, y') + cost(y_i, y')\right) \quad (1)$$

Figure 1: Infinite ramp loss.

Typical loss functions are the **structured perceptron loss** (Collins, 2002):

$$L(x_i, y_i; \mathbf{w}) = -\mathbf{w} \cdot \mathbf{f}(x_i, y_i) + \max_{y \in \mathcal{Y}(x_i)} \mathbf{w} \cdot \mathbf{f}(x_i, y) \quad (2)$$

and the **structured SVM loss** (Taskar et al., 2003; Tsochantaridis et al., 2004), which incorporates margin using a cost function:[1]

$$L(x_i, y_i; \mathbf{w}) = -\mathbf{w} \cdot \mathbf{f}(x_i, y_i) + \max_{y \in \mathcal{Y}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, y) + cost(y_i, y)\right) \quad (3)$$

Both (2) and (3) are problematic if example i is unreachable, i.e., $y_i \notin \mathcal{Y}(x_i)$, due to imperfect data or an imperfect definition of $\mathcal{Y}$. In this case, the model is trying to learn an output it cannot produce. In some applications, the features $\mathbf{f}(x_i, y_i)$ cannot even be computed for these examples. This problem is well known in machine translation: some examples cannot be produced by the phrase-table or grammar. It also occurs in AMR parsing.

To handle unreachable training examples, we modify (3), introducing the **infinite ramp loss**, shown in Eq. 1 in Fig. 1. The term labeled $C(x_i, y_i)$ is present only to make the limit well-defined in case $\min_{y \in \mathcal{Y}(x_i)} cost(y_i, y) \neq 0$. In practice, we set α to be a very large number (10^{12}) instead of taking a proper limit, and set $C(x_i, y_i) = 0$.

The intuition behind Eq. 1 is the following: for very large α, the first max picks a y that minimizes $cost(y_i, y)$, using the model score $\mathbf{w} \cdot \mathbf{f}(x_i, y)$ to break any ties. This is what the model updates *towards* in subgradient descent-style updates, called the "hope derivation" by Chiang (2012). The second max is the usual cost augmented decoding that gives a margin in the SVM loss, and is what the model updates *away from* in subgradient descent, called the "fear derivation" by Chiang (2012).

Eq. 1 generalizes the structured SVM loss. If y_i is reachable and the minimum over $y \in \mathcal{Y}(x_i)$ of $cost(y, y_i)$ occurs when $y = y_i$, then the first max in Eq. 1 picks out $y = y_i$ and Eq. 1 reduces to the structured SVM loss.

The infinite ramp is also a generalization of the latent structured SVM (Yu and Joachims, 2009), which is a generalization of the structured SVM for hidden variables. This loss can be used when the output can be written $y_i = (\tilde{y}_i, h_i)$, where $\tilde{y}_i$ is observed output and h_i is latent (even at training time). Let $\tilde{\mathcal{Y}}(x_i)$ be the space of all possible observed outputs and $\mathcal{H}(x_i)$ be the hidden space for the example x_i. Let $\tilde{c}$ be the cost function for the observed output. The **latent structured SVM loss** is:

$$L(x_i, y_i; \mathbf{w}) = -\max_{h \in \mathcal{H}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, \tilde{y}_i, h)\right)$$
$$+ \max_{\tilde{y} \in \tilde{\mathcal{Y}}(x_i)} \max_{h' \in \mathcal{H}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, \tilde{y}, h) + \tilde{c}(\tilde{y}_i, \tilde{y})\right) \quad (4)$$

If we set $cost(y_i, y) = \tilde{c}(\tilde{y}_i, \tilde{y})$ in Eq. 1, and the minimum of $\tilde{c}(\tilde{y}_i, \tilde{y})$ occurs when $\tilde{y} = \tilde{y}_i$, then minimizing Eq. 1 is equivalent to minimizing Eq. 4.

Eq. 1 is related to **ramp loss** (Collobert et al., 2006; Chapelle et al., 2009; Keshet and McAllester, 2011):

$$L(x_i, y_i; \mathbf{w}) =$$
$$- \max_{y \in \mathcal{Y}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, y) - \alpha \cdot cost(y_i, y)\right)$$
$$+ \max_{y' \in \mathcal{Y}(x_i)} \left(\mathbf{w} \cdot \mathbf{f}(x_i, y') + cost(y_i, y')\right) \quad (5)$$

The parameter α is often set to zero, and controls the "height" of the ramp, which is $\alpha + 1$. Taking

[1] $cost(y_i, y)$ returns the cost of mistaking y for correct output y_i.

$\alpha \to \infty$ in Eq. 5 corresponds roughly to Eq. 1, hence the name "infinite ramp loss". However, Eq. 1 also includes $C(x_i, y_i)$ term to make the limit well defined even when $\min_{y \in \mathcal{Y}(x_i)} cost(y_i, y) \neq 0$. Like infinite ramp loss, ramp loss also handles unreachable training examples (Gimpel and Smith, 2012), but we have found ramp loss to be more difficult to optimize than infinite ramp loss in practice due to local minima. Both loss functions are non-convex. However, infinite ramp loss is convex if $\arg\min_{y \in \mathcal{Y}(x_i)} cost(y_i, y)$ is unique.

4 Training

We train the concept identification stage using infinite ramp loss (1) with AdaGrad (Duchi et al., 2011). We process examples in the training data $((x_1, y_1), \ldots, (x_N, y_N))$ one at a time. At time t, we decode with the current parameters and the cost function as an additional local factor to get the two outputs:

$$h^t = \arg\max_{y' \in \mathcal{Y}(x_t)} \left(\mathbf{w}^t \cdot \mathbf{f}(x_t, y') - \alpha \cdot cost(y_i, y) \right) \quad (6)$$

$$f^t = \arg\max_{y' \in \mathcal{Y}(x_t)} \left(\mathbf{w}^t \cdot \mathbf{f}(x_t, y') + cost(y_i, y) \right) \quad (7)$$

and compute the subgradient:

$$\mathbf{s}^t = \mathbf{f}(x_t, h^t) - \mathbf{f}(x_t, f^t) - 2\lambda \mathbf{w}^t$$

We then update the parameters and go to the next example. Each component i of the parameters gets updated as:

$$w_i^{t+1} = w_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} s_i^{t'}}} s_i^t$$

5 Experiments

We evaluate using Smatch (Cai and Knight, 2013). Following the recommended train/dev./test split of LDC2015E86, our parser achieves 70% precision, 65% recall, and 67% F_1 Smatch on the LDC2015E86 test set. The JAMR baseline on this same dataset is 55% F_1 Smatch, so the improvements are quite substantial. On the SemEval 2016 Task 8 test set, our improved parser achieves 56% F_1 Smatch.

We hypothesize that the lower performance of the parser on the SemEval Task 8 test set is due to drift in the AMR annotation scheme between the production of the LDC2015E86 training data and the SemEval test set. During that time, there were changes to the concept senses and the concept frame files. Because the improvements in our parser were due to boosting recall in concept identification (and using the frame files to our advantage), our approach does not show as large improvements on the SemEval test set as on the LDC2015E86 test set.

6 Conclusion

We have presented improvements to the JAMR parser as part of the SemEval 2016 Shared Task on AMR parsing, showing substantial improvements over the baseline JAMR parser. As part of these improvements, we introduced infinite ramp loss, which generalizes the structured SVM to handle training data with unreachable training examples. We hope this loss function will be useful in other application areas as well.

Acknowledgments

This work is supported by the U.S. Army Research Office under grant number W911NF-10-1-0533. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the U.S. Army Research Office or the United States Government.

References

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*.

Olivier Chapelle, Chuong B. Do, Choon H Teo, Quoc V. Le, and Alex J. Smola. 2009. Tighter bounds for structured estimation. In *NIPS*.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *JMLR*, 13(1):1159–1187.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. 2006. Trading convexity for scalability. In *Proc. of ICML*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proc. of NAACL*.

Joseph Keshet and David A McAllester. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Advances in Neural Information Processing Systems*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *NIPS*.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. of ICML*.