AMR Parsing as Graph Prediction with Latent Alignment

AMR Parsing as Graph Prediction with Latent Alignment

Chunchuan Lyu[1] **Ivan Titov**[1,2]
[1]ILCC, School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam

Abstract

Abstract meaning representations (AMRs) are broad-coverage sentence-level semantic representations. AMRs represent sentences as rooted labeled directed acyclic graphs. AMR parsing is challenging partly due to the lack of annotated alignments between nodes in the graphs and words in the corresponding sentences. We introduce a neural parser which treats alignments as latent variables within a joint probabilistic model of concepts, relations and alignments. As exact inference requires marginalizing over alignments and is infeasible, we use the variational auto-encoding framework and a continuous relaxation of the discrete alignments. We show that joint modeling is preferable to using a pipeline of align and parse. The parser achieves the best reported results on the standard benchmark (73.6% on LDC2016E25).

1 Introduction

Abstract meaning representations (AMRs) (Banarescu et al., 2013) are broad-coverage sentence-level semantic representations. AMR encodes, among others, information about semantic relations, named entities, co-reference, negation and modality. AMR can be represented as a rooted labeled directed acyclic graph (see Figure 1). As AMR abstracts away from details of surface realization, it is potentially beneficial in many semantic related NLP tasks, including text summarization (Liu et al., 2015; Dohare and Karnick, 2017), machine translation (Jones et al., 2012) and question answering (Mitra and Baral, 2016).

AMR parsing has recently received a lot of attention (e.g., (Flanigan et al., 2014; Artzi et al., 2015; Konstas et al., 2017)). One distinctive aspect of AMR annotation is the lack of explicit alignments between nodes in the graph (*concepts*) and words in the sentences. Though this arguably simplified the annotation process (Banarescu et al., 2013), it is not straightforward to produce an effective parser without relying on an alignment. Most AMR parsers (Damonte et al., 2017; Flanigan et al., 2016; Werling et al., 2015; Wang and Xue, 2017; Foland and Martin, 2017) use a pipeline where the aligner training stage precedes training a parser. The aligners are not directly informed by the AMR parsing objective and may produce alignments suboptimal for this task.

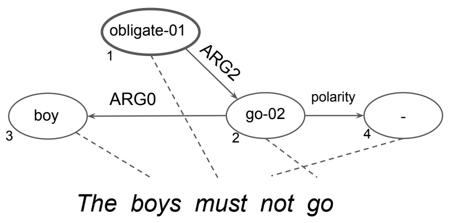

Figure 1: An example of AMR, the dashed lines denote latent alignments, *obligate-01* is the root. Numbers indicate depth-first traversal order.

In this work, we demonstrate that the alignments can be treated as latent variables in a joint probabilistic model and induced in such a way as to be beneficial for AMR parsing. Intuitively, in our probabilistic model, every node in a graph is assumed to be aligned to a word in a sentence: the corresponding concept is predicted based on the corresponding RNN state. Similarly, graph edges (i.e. relations) are predicted based on representations of concepts and aligned words (see Figure 2). As alignments are latent, exact inference requires marginalizing over latent alignments, which is infeasible. Instead we use variational inference, specifically the variational autoencoding framework of Kingma and Welling (2014). Using discrete latent variables in deep learning has proven to be challenging (Mnih and Gregor, 2014; Bornschein and Bengio, 2015). We use a continuous relaxation of the alignment problem, relying on the recently introduced Gumbel-Sinkhorn construction (Mena et al., 2018). This yields a computationally-efficient approximate method for estimating our joint probabilistic model of concepts, relations and alignments.

We assume injective alignments from concepts to words: every node in the graph is aligned to a single word in the sentence and every word is aligned to at most one node in the graph. This is necessary for two reasons. First, it lets us treat concept identification as sequence tagging at test time. For every word we would simply predict the corresponding concept or predict *NULL* to signify that no concept should be generated at this position. Secondly, Gumbel-Sinkhorn can only work under this assumption. This constraint, though often appropriate, is problematic for certain AMR constructions (e.g., named entities). In order to deal with these cases, we re-categorized AMR concepts. Similar strategies have been adopted in previous work (Foland and Martin, 2017; Peng et al., 2017).

The resulting parser achieves 73.6% ± 0.3% Smatch score on the standard test set when using LDC2016E25 training set, an improvement of 2.6% over the previous best result (van Noord and Bos, 2017). We also demonstrate that inducing alignments within the joint model is indeed beneficial. When, instead of inducing alignments, we rely on predictions of JAMR aligner (Flanigan et al., 2016), the performance drops by 1.6% Smatch. Our main contributions can be summarized as follows:

- we introduce a joint probabilistic model for alignment, concept and relation identification;
- we demonstrate how a continuous relaxation can be used to effectively estimate the model;
- the model achieves the best reported results.[1]

2 Probabilistic Model

In this section we describe our probabilistic model and the estimation technique. In section 3, we describe preprocessing and post-processing (including concept re-categorization, sense disambiguation, wikification and root selection).

2.1 Notation and setting

We will use the following notation throughout the paper. We refer to words in the sentences as $\mathbf{w} = (w_1, \ldots, w_n)$, where n is sentence length, $w_k \in \mathcal{V}$ for $k \in \{1 \ldots, n\}$. The concepts (i.e. labeled nodes) are $\mathbf{c} = (c_1, \ldots, c_m)$, where m is the number of concepts and $c_i \in \mathcal{C}$ for $i \in \{1 \ldots, m\}$. For example, in Figure 1, $\mathbf{c} = (\textit{obligate}, \textit{go}, \textit{boy}, \text{-})$.[2] Note that senses are predicted at post-processing, as discussed in Section 3.2 (i.e. *go* is labeled as *go-02*).

A relation between 'predicate concept' i and 'argument concept' j is denoted by $r_{ij} \in \mathcal{R}$; it is set to *NULL* if j is not an argument of i. In our example, $r_{2,3} = \textit{ARG0}$ and $r_{1,3} = \textit{NULL}$. We will use R to denote all relations in the graph.

To represent alignments, we will use $\mathbf{a} = \{a_1, \ldots, a_m\}$, where $a_i \in \{1, \ldots, n\}$ returns the index of a word aligned to concept i. In our example, $a_1 = 3$.

All three model components rely on bi-directional LSTM encoders (Schuster and Paliwal, 1997). We denote states of BiLSTM (i.e. concatenation of forward and backward LSTM states) as $\mathbf{h}_k \in \mathbb{R}^d$ $(k \in \{1, \ldots, n\})$. The sentence encoder takes pre-trained fixed word embeddings, randomly initialized lemmas, suffix (last two characters) and named-entity tag embeddings.

2.2 Method overview

We believe that using discrete alignments, rather than attention-based models (Bahdanau et al., 2014) is crucial for AMR parsing. AMR banks are a lot smaller than parallel corpora used in machine translation (MT) and hence it is important to inject a useful inductive bias. We constrain our alignments from concepts to words to be injective. First, it encodes the observation that concepts are mostly triggered by single words (especially, after re-categorization, Section 3.1). Second, it implies that each word corresponds to at most one concept (if any). This encourages competition: alignments are mutually-repulsive. In our example, *obligate* is not lexically similar to the word *must* and

[1]The code will be made publicly available, should the paper get accepted.

[2]The probabilistic model is invariant to the ordering of concepts, though the order affects the inference algorithm (see Section 2.5). We use depth-first traversal of the graph to generate the ordering.

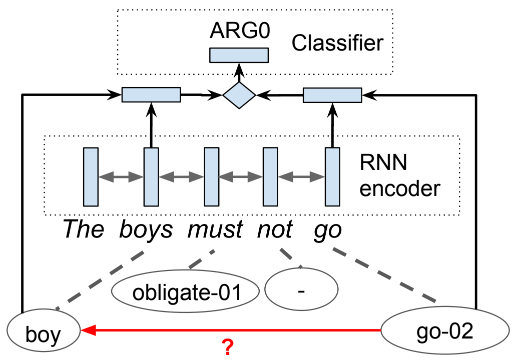

Figure 2: Relation identification: predicting a relation between *boy* and *go-02* relying on the two concepts and corresponding RNN states.

may be hard to align. However, given that other concepts are easy to predict, alignment candidates other than *must* and *the* will be immediately ruled out. We believe that these are the key reasons for why attention-based neural models do not achieve competitive results on AMR (Konstas et al., 2017) and why state-of-the-art models rely on aligners. Our goal is to combine best of two worlds: to use alignments (as in state-of-the-art AMR methods) and to induce them while optimizing for the end goal (similarly to the attention component of encoder-decoder models).

Our model consists of three parts: (1) the concept identification model $P_\theta(\mathbf{c}|\mathbf{a}, \mathbf{w})$; (2) the relation identification model $P_\phi(R|\mathbf{a}, \mathbf{w}, \mathbf{c})$ and (3) the alignment model $Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w})$.[3] Formally, (1) and (2) together with the uniform prior over alignments $P(\mathbf{a})$ form the generative model of AMR graphs. In contrast, the alignment model $Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w})$, as will be explained below, is approximating the intractable posterior $P_{\theta,\phi}(\mathbf{a}|\mathbf{c}, R, \mathbf{w})$ within that probabilistic model.

In other words, we assume the following model for generating the AMR graph:

$$P_{\theta,\phi}(\mathbf{c}, R|\mathbf{w}) = \sum_{\mathbf{a}} P(\mathbf{a}) P_\theta(\mathbf{c}|\mathbf{a}, \mathbf{w}) P_\phi(R|\mathbf{a}, \mathbf{w}, \mathbf{c})$$

$$= \sum_{\mathbf{a}} P(\mathbf{a}) \prod_{i=1}^{m} P(c_i|\mathbf{h}_{a_i}) \prod_{i,j=1}^{m} P(r_{ij}|\mathbf{h}_{a_i}, \mathbf{c}_i, \mathbf{h}_{a_j}, \mathbf{c}_j)$$

AMR concepts are assumed to be generated conditional independently relying on the BiLSTM states and surface forms of the aligned words. Similarly, relations are predicted based only on AMR concept embeddings and LSTM states corresponding to words aligned to the involved concepts. Their combined representations are fed into a bi-affine classifier (Dozat and Manning, 2017) (see Figure 2).

[3] θ, ϕ and ψ denote all parameters of the models.

The expression involves intractable marginalization over all valid alignments. As standard in variational autoencoders, VAEs (Kingma and Welling, 2014), we lower-bound the log-likelihood as

$$\begin{aligned} &\log P_{\theta,\phi}(\mathbf{c}, R|\mathbf{w}) \\ &\geq E_Q[\log P_\theta(\mathbf{c}|\mathbf{a}, \mathbf{w}) P_\phi(R|\mathbf{a}, \mathbf{w}, \mathbf{c})] \\ &- D_{KL}(Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w}) || P(\mathbf{a})), \end{aligned} \quad (1)$$

where $Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w})$ is the variational posterior (aka the inference network), $E_Q[\ldots]$ refers to the expectation under $Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w})$ and D_{KL} is the Kullback-Liebler divergence. In VAEs, the lower bound is maximized both with respect to model parameters (θ and ϕ in our case) and the parameters of the inference network (ψ). Unfortunately, gradient-based optimization with discrete latent variables is challenging. We use a continuous relaxation of our optimization problem, where real-valued vectors $\hat{\mathbf{a}}_i \in \mathbb{R}^n$ (for every concept i) approximate discrete alignment variables a_i. This relaxation results in low-variance estimates of the gradient using the parameterization trick (Kingma and Welling, 2014), and ensures fast and stable training. We will describe the model components and the relaxed inference procedure in detail in sections 2.6 and 2.7.

Though the estimation procedure requires the use of the relaxation, the learned parser is straightforward to use. Given our assumptions about the alignments, we can independently choose for each word w_k ($k = 1, \ldots, m$) the most probably concept according to $P_\theta(c|\mathbf{h}_k)$. If the highest scoring option is *NULL*, no concept is introduced. The relations could then be predicted relying on $P_\phi(R|\mathbf{a}, \mathbf{w}, \mathbf{c})$. This would have led to generating inconsistent AMR graphs, so instead we search for the highest scoring valid graph (see Section 3.2). Note that the alignment model Q_ψ is not used at test time and only necessary to train accurate concept and relation identification models.

2.3 Concept identification model

The concept identification model chooses a concept c (i.e. a labeled node) conditioned on the aligned word k or decides that no concept should be introduced (i.e. returns *NULL*). Though it can be modeled with a softmax classifier, it would not be effective in handling rare or unseen words. First, we split the decision into estimating the probability of concept category $\tau(c) \in \mathcal{T}$ (e.g. 'number', 'frame') and estimating the probability of the specific concept within the chosen category. Secondly, based on a lemmatizer and training data[4] we prepare one candidate concept e_k for each word k in vocabulary (e.g., it would propose *want* if the word is *wants*). Similar to Luong et al. (2015), our model can then either copy the candidate e_k or rely on the softmax over potential concepts of category τ. Formally, the concept prediction model is defined as

$$P_\theta(c|\mathbf{h}_k, w_k) = P(\tau(c)|\mathbf{h}_k, w_k) \times \frac{[[e_k = c]] \times \exp(\mathbf{v}_{copy}^T \mathbf{h_k}) + \exp(\mathbf{v}_c^T \mathbf{h_k})}{Z(\mathbf{h_k}, \theta)},$$

where the first multiplicative term is a softmax classifier over categories (including *NULL*); $\mathbf{v}_{copy}, \mathbf{v}_c \in \mathbb{R}^d$ (for $c \in \mathcal{C}$) are model parameters; $[[\ldots]]$ denotes the indicator function and equals 1 if its argument is true and 0, otherwise; $Z(\mathbf{h}, \theta)$ is the partition function ensuring that the scores sum to 1.

2.4 Relation identification model

Most predicates have at most one argument for each relation type (e.g., there is typically at most one agent / ARG0), hence we would like to encourage competition for each role among arguments. For non-NULL relations, we have

$$P(r_{ij}|\mathbf{h}_{a_i}, c_i, \mathbf{h}_{a_j}, c_j) = P(s_{ir} = j|\mathbf{h}_{a_i}, c_i, \mathbf{h}_{a_j}, c_j) \times P(r_{ij}|\mathbf{h}_{a_i}, c_i, \mathbf{h}_{a_j}, c_j, s_{ir} = j),$$

where the first term corresponds to picking a candidate argument (i.e. predicting that concept c_j is a candidate for being argument of type r for the predicate c_i), whereas the second to deciding that it is indeed an argument. The remaining probability mass $1 - \sum_{r_{ij} \neq NULL} P(r_{ij}|\mathbf{h}_{a_i}, c_i, \mathbf{h}_{a_j}, c_{a_j})$ is reserved for $r_{ij} = NULL$.

Each term is modeled in exactly the same way: (1) for both endpoints, embedding of the concept c is concatenated with the RNN state $\mathbf{h}$; (2) they are linearly projected to a lower dimension $\mathbf{f}(\mathbf{h}, c) \in \mathbb{R}^{d_f}$; (3) a log-linear model with bilinear scores $\mathbf{f}(\mathbf{h_i}, c_i)^T C_r \mathbf{f}(\mathbf{h_j}, c_j)$, $C_r \in \mathbb{R}^{d_f \times d_f}$ is used to compute the probabilities.

[4]See supplementary materials.

2.5 Alignment model

Recall that the alignment models is only used at training, and hence it can rely both on input (states $\mathbf{h}_1, \ldots, \mathbf{h}_n$) and on the list of concepts $c_1, \ldots, c_m$.

Formally, we add $(m-n)$ *NULL* concepts to the list.[5] Aligning a word to any *NULL*, would correspond to saying that the word is not aligned to any 'real' concept. Note that each one-to-one alignment (i.e. permutation) between n such concepts and n words implies a valid injective alignment of n words to m 'real' concepts. This reduction to permutations will come handy when we turn to the Gumbel-Sinkhorn relaxation in the next section. From now on, we will assume that $m = n$.

As with sentences, we use a BiLSTM model to encode concepts $\mathbf{c}$, where $\mathbf{g}_i \in \mathcal{R}^{d_g}$, $i \in \{1, \ldots, n\}$. We use a globally-normalized alignment model:

$$Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w}) = \frac{\exp(\sum_{i=1}^n \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}))}{Z_\psi(\mathbf{c}, \mathbf{w})},$$

where $Z_\psi(\mathbf{c}, \mathbf{w})$ is the intractable partition function and the terms $\varphi(\mathbf{g}_i, \mathbf{h}_{a_i})$ score each alignment link according to a bilinear form

$$\varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) = \mathbf{g}_i^T B \mathbf{h}_{a_i}, \qquad (2)$$

where $B \in \mathbb{R}^{d_g \times d}$ is a parameter matrix.

2.6 Estimating model with Gumbel-Sinkhorn

Recall that our learning objective (1) involves expectation under the alignment model. The partition function of the alignment model is intractable, and it is tricky even to draw samples. Luckily, the recently proposed relaxation (Mena et al., 2018) lets us circumvent this issue. First, note that exact samples from a categorical distribution can be obtain using the perturb-and-max technique (Papandreou and Yuille, 2011). For our alignment model, it would correspond to adding independent noise to the score for every possible alignment and choosing the highest scoring one:

$$\mathbf{a}^\star = \operatorname*{argmax}_{\mathbf{a} \in \mathcal{P}} \sum_{i=1}^n \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) + \epsilon_\mathbf{a}, \qquad (3)$$

[5]After re-categorization (Section 3.1), $m \geq n$ holds for most cases. For exceptions, we append *NULL* to the sentence.

where $\mathcal{P}$ is the set of all permutations of n elements, $\epsilon_{\mathbf{a}}$ is a noise drawn independently for each $\mathbf{a}$ from the fixed Gumbel distribution ($\mathcal{G}(0,1)$). Unfortunately, this is also intractable, as there are $n!$ permutations. Instead, in perturb-and-max an approximate schema is used where noise is assumed factorizable. In other words, first noisy scores are computed as $\hat{\varphi}(\mathbf{g}_i, \mathbf{h}_{a_i}) = \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) + \epsilon_{i,a_i}$, where $\epsilon_{i,a_i} \sim \mathcal{G}(0,1)$ and an approximate sample is obtained by $\mathbf{a}^{\star} = \operatorname{argmax}_{\mathbf{a}} \sum_{i=1}^{n} \hat{\varphi}(\mathbf{g}_i, \mathbf{h}_{a_i})$,

Such sampling procedure is still intractable in our case and also non-differentiable. The main contribution of Mena et al. (2018) is approximating this argmax with a simple differentiable computation $\hat{A} = S_t(\Phi, \Sigma)$ which yields an approximate (i.e. relaxed) permutation. We use Φ and Σ to denote the $n \times n$ matrices of alignment scores $\varphi(\mathbf{g}_i, \mathbf{h}_k)$ and noise variables ϵ_{ik}, respectively. Instead of returning index a_i for every concept i, it would return a (peaky) distribution over words $\hat{\mathbf{a}}_i$. The peakiness is controlled by the temperature parameter t of Gumbel-Sinkhorn which balances smoothness ('differentiability') vs. bias of the estimator. For further details and the derivation, we refer the reader to the original paper (Mena et al., 2018).

Note that Φ is a function of the alignment model Q_ψ, so we will write Φ_ψ in what follows. The variational bound (1) can now be approximated as

$$\begin{aligned} E_{\Sigma \sim \mathcal{G}(0,1)} & [\log P_\theta(c | S_t(\Phi_\psi, \Sigma), \mathbf{w}) \\ & + \log P_\phi(R | S_t(\Phi_\psi, \Sigma), \mathbf{w}, \mathbf{c})] \\ & - D_{KL}(\frac{\Phi_\psi + \Sigma}{t} || \frac{\Sigma}{t_0}) \end{aligned} \tag{4}$$

Following Mena et al. (2018), the original KL term from equation (1) is approximated by the KL term between two $n \times n$ matrices of i.i.d. Gumbel distributions with different temperature and mean. The parameter t_0 is the 'prior temperature'.

At test time we will use the model in a pipeline fashion, first predicting concepts and the relations between these concepts. Consequently, being accurate at predicting concepts is more important. In contrast, the concept prediction term in expression (4) would have minor influence on the loss, as there are n concepts to predict, whereas there are $n \times n$ relations. Consequently, we down-weight the relation identification term by the factor of $1/n$.

Our new objective is fully differentiable with respect to all parameters (i.e. θ, ϕ and ψ) and has low variance as sampling is performed from the fixed non-parameterized distribution, as in standard VAEs.

2.7 Relaxing concept and relation identification

One remaining question is how to use the soft input $\hat{A} = S_t(\Phi_\psi, \Sigma)$ in the concept and relation identification models in equation (4). In other words, we need to define how we compute $P_\theta(c|S_t(\Phi_\psi, \Sigma), \mathbf{w})$ and $P_\phi(R|S_t(\Phi_\psi, \Sigma), \mathbf{w}, \mathbf{c})$.

The standard technique would be to pass to the models expectations under the relaxed variables $\sum_{k=1}^{n} \hat{\mathbf{a}}_{ik} \mathbf{h}_k$, instead of the vectors $\mathbf{h}_{a_i}$ (Maddison et al., 2017; Jang et al., 2017). This is exactly what we do for the relation identification model.

However, the concept prediction model $\log P_\theta(c|S_t(\Phi_\psi, \Sigma), \mathbf{w})$ relies on the pointing mechanism, i.e. directly exploits the words $\mathbf{w}$ rather than relies only on biLSTM states $\mathbf{h}_k$. So instead we treat $\hat{\mathbf{a}}_i$ as a prior in a hierarchical model:

$$\begin{aligned} & \log P_\theta(c_i | \hat{\mathbf{a}}_\mathbf{i}, \mathbf{w}) \\ & \quad \approx \log \sum_{k=1}^{n} \hat{\mathbf{a}}_{ik} P_\theta(c_i | a_i = k, \mathbf{w}) \end{aligned} \tag{5}$$

As we will show in our experiments, a softer version of the loss is even more effective:

$$\begin{aligned} & \log P_\theta(c_i | \hat{\mathbf{a}}_\mathbf{i}, \mathbf{w}) \\ & \quad \approx \sum_{k=1}^{n} \hat{\mathbf{a}}_{ik}^{\alpha} P_\theta(c_i | a_i = k, \mathbf{w})^{(1-\alpha)}, \end{aligned}$$

where we set the parameter $\alpha = 0.5$. We believe that using this loss encourages the model to more actively explore the alignment space.

3 Pre- and post-pocessing

3.1 Re-Categorization

AMR parsers often rely on a pre-processing stage, where specific subgraphs of AMR are grouped together and assigned to a single node with a new compound category (e.g., Werling et al. (2015); Foland and Martin (2017); Peng et al. (2017)); this transformation is reversed at the post-processing stage. Our approach is very similar to the Factored Concept Label system of Wang and Xue (2017), with one important difference that we unpack our concepts before the relation identification stage, so

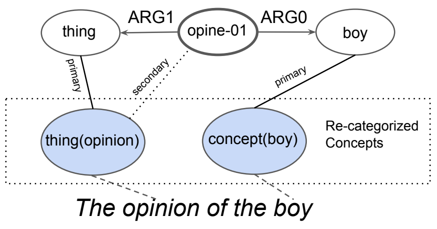

Figure 3: An example of re-categorized AMR. AMR graph at the top, re-categorized concepts in the middle, and the sentence is at the bottom.

the relations are predicted between original concepts (all nodes in each group share the same alignment distributions to the RNN states). Intuitively, the goal is to ensure that concepts rarely lexically triggered (e.g., *thing* in Figure 3) get grouped together with lexically triggered nodes. Such 'primary' concepts get encoded in the category of the concept (the set of categories is τ, see also section 2.3). In Figure 3, the re-categorized concept *thing(opinion)* is produced from *thing* and *opine-01*. We use *category* as the dummy category type. There are 8 templates in our system which extract re-categorization for fixed phrases (e.g. thing(opinion)), and deterministic system for grouping lexically flexible, but structurally stable sub-graph (e.g., named entities, *have-rel-role-91* and *have-org-role-91* concepts).

Details of the re-categorization procedure and other pre-processing are provided in appendix.

3.2 Post-processing

For post-processing, we handle sense-disambiguation, wikification and ensure legitimacy of the produced AMR graph. For sense disambiguation we pick the most frequent sense for that particular concept ('-01', if unseen). For wikification we again look-up in the training set and default to "-".

Our probability model predicts edges conditional independently and thus cannot guarantee the connectivity of AMR graph, also there are additional constraints which are useful to impose. We enforce three constraints: (1) specific concepts can have only one neighbor (e.g., 'number' and 'string'; see appendix for details); (2) each predicate concept can have at most one argument for each relation $r \in \mathcal{R}$; (3) the graph should be connected. Constraint (1) is addressed simply by keeping only the highest scoring neighbor. In order to satisfy the last two constraints we use a simple greedy procedure. First, for each edge, we pick-up the highest scoring relation and edge (possibly *NULL*). If the constraint (2) is violated, we simply keep the highest scoring edge among the duplicates and drop the rest. If the graph is not connected (i.e. constraint (3) is violated), we greedily choose edges linking the connected components until the graph gets connected (MSCG in Flanigan et al. (2014)).

Finally, we need to select a root node. Similarly to relation identification, for each candidate concept c_i, we concatenate its embedding with the corresponding LSTM state ($\mathbf{h}_{a_i}$) and use these scores in a softmax classifier over all the concepts.

4 Experiments and Discussion

4.1 Data and setting

We primarily focus on the most recent LDC2016E25 (R2) dataset, which consists of 36521, 1368 and 1371 sentences in training, development and testing sets, respectively. The earlier LDC2015E86 (R1) dataset has been used by much of the previous work. It contains 16833 training sentences, and same sentences for development and testing as R2.[6]

We used the development set to perform model selection and hyperparameter tuning. The hyperparameters, as well as information about embeddings and pre-processing, are presented in the supplementary materials.

We used Adam (Kingma and Ba, 2014) to optimize the loss (4) and to train the root classifier.

In order to make sure the model pays more attention to concept prediction, we split our training into two stages. We start by jointly training the alignment model and the concept identification model. Then we continue optimizing the entire objective but fix the concept identification model. We do early stopping on the development set scores, for both stages.

4.2 Experiments and discussion

We start by comparing our parser to previous work (see Table 1). Our model substantially outperforms all the previous models on both datasets. Specifically, it achieves 73.6% Smatch score on LDC2016E25 (R2), which is an improvement of 2.6% over character seq2seq model relying on silver data (van Noord and Bos, 2017). For LDC2015E86 (R1), we obtain 72.7% Smatch

[6]Annotation in R2 has also been slightly revised.

Model	Data	Smatch
JAMR (Flanigan et al., 2016)	R1	67.0
AMREager (Damonte et al., 2017)	R1	64.0
CAMR (Wang et al., 2016)	R1	66.5
SEQ2SEQ + 20M (Konstas et al., 2017)	R1	62.1
Mul-BiLSTM (Foland and Martin, 2017)	R1	70.7
Ours	R1	**72.7**
Neural-Pointer (Buys and Blunsom, 2017)	R2	61.9
ChSeq (van Noord and Bos, 2017)	R2	64.0
ChSeq + 100K (van Noord and Bos, 2017)	R2	71.0
Ours	R2	**73.6** ± 0.26

Table 1: Smatch scores on the test set. R2 is LDC2016E25 dataset, and R1 is LDC2015E86 dataset. Statistics on R2 are over 6 runs.

Models	A' 17	C' 16	J' 16	Ch' 17	Ours
Dataset	R1	R1	R1	R2	R2
Smatch	64	63	67	71	**73.6**±0.26
Unlabeled	69	69	69	74	**76.2**±0.32
No WSD	65	64	68	72	**74.6**±0.30
Reentrancy	41	41	42	**52**	51.5±0.32
Concepts	83	80	83	82	**84.9**±0.14
NER	**83**	75	79	79	82.3±0.56
Wiki	64	0	75	65	**75.6**±0.49
Negations	48	18	45	**62**	56.1±0.82
SRL	56	60	60	66	**68.5**±0.29

Table 2: F1 scores on individual phenomena. A'17 is AMREager, C'16 is CAMR, J'16 is JAMR, Ch'17 is ChSeq+100K.

score, which is an improvement of 2.0% over the previous best model, multi-BiLSTM parser of Foland and Martin (2017).

In order to disentangle individual phenomena, we use the AMR-evaluation tools (Damonte et al., 2017) and compare to systems which reported these scores (Table 2). We obtain the highest scores on most subtasks. The exceptions are NER (named entity recognition) and negations. For NER, the best parser is AMREager by Damonte et al. (2017) which relies on an external NER system and a gazetteer. This may suggest that the AMR bank on its own is not large enough for a system to learn an accurate NER model. For negation detection, it is also not surprising as many negations are encoded with morphology, and character models, unlike our word-level model, are able to capture predictive morphological features (e.g., detect prefixes such as "un-" or "im-").

Metric	Fixed-Align	R1	Fixed-Align	R2 mean
Smatch	70.9	72.7	72.0	**73.6**
Unlabeled	73.9	75.6	74.6	**76.2**
No WSD	71.8	73.7	73.0	**74.6**
Reentrancy	48.4	50.0	50.9	**51.5**
Concepts	83.7	84.8	84.2	**84.9**
NER	**82.7**	82.5	81.7	82.3
Wiki	67.1	74.7	67.6	**75.6**
Negations	48.4	52.3	50.6	**56.1**
SRL	66.4	67.5	67.8	**68.5**

Table 3: F1 scores of on subtasks. The left side results are from LDC2015E86 and right results are from LDC2016E25.

Now, we turn to ablation tests (see Table 3). First, we would like to see if our latent alignment framework is beneficial. In order to test this, we create a baseline version of our system ('fixed-align') which relies on the JAMR aligner (Flanigan et al., 2014), rather than induces alignments as latent variables. Recall that in our model we used training data and a lemmatizer to produce candidates for the concept prediction model (see Section 2.3, the copy function). In order to have a fair comparison, if a concept is not aligned after JAMR, we try to use our copy function to align it. If an alignment is not found, we make the alignment uniform across the unaligned words. In preliminary experiments, we considered alternatives versions (e.g., dropping concepts unaligned by JAMR or dropping concepts unaligned after both JAMR and the matching heuristic), but the chosen strategy was the most effective. We observe that using fixed alignments gives us 70.9% Smatch score on R1, a substantial drop in performance (1.8%). Interestingly, these scores of fixed-align are on par with Foland and Martin (2017). The fixed-align version is indeed similar to Foland and Martin (2017): both rely on fixed JAMR alignments and use BiLSTM encoders. These results confirm that we used a strong baseline, and that the gains in performance are primarily due to using our variational alignment framework.

We present further ablations in Table 4. We would like to confirm that our 2-stage training procedure (described in section 4.1) is necessary. When compared to using one-stage training ('full joint'), we observe that using the schedule was beneficial (+1.5%). Now, it is a natural question to ask whether alignments need to be updated at all

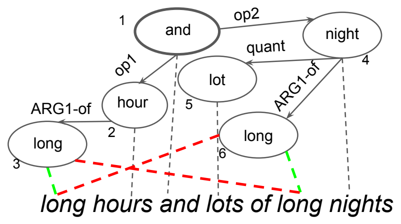

Figure 4: When modeling concepts alone, the posterior probability of the correct (green) and wrong (red) alignment links will be the same.

Ablation	Concepts	SRL	Smatch
H-relax	84.1	66.9	72.3
Full joint	83.9	66.5	72.3
Relation simple	**85.1**	67.9	73.6
No fine-tune	**85.1**	68.2	73.5
Our model	**85.1**	**68.8**	**73.8**

Table 4: Ablation studies (all on R2). The bottom three models load the same concept and alignment model before the second stage.

based on relations (i.e. on the second stage). This 'no fine-tune' version also appears weaker overall than the best approach (-0.3%). The drop is more substantial for relations ('SRL'): -0.6%. In order to see why relations are potentially useful in learning alignments, consider Figure 4. The example contains duplicate concepts *long*. The concept prediction model factorizes over concepts and does not care which way these duplicates are aligned: correctly (green edges) or not (red edges). Formally, the true posterior under the relation-only model in 'no fine-tune' assigns exactly the same probability to both configurations, and the alignment model Q_ψ will be forced to mimic it (even though it relies on a LSTM model of the graph). The spurious ambiguity will have a detrimental effect on the relation identification stage. These observations and the ablations suggest that our training regime provides a good middle ground between full-joint optimization (which fails to emphasize the importance of the concept prediction stage) and 'no fine-tune' (which makes the alignments sub-optimal for relation identification).

We perform two additional ablation tests. First, we compare our relation identification component to simply using a bilinear scoring function (i.e. not encouraging competition, Section 2.4) and observe a drop in performance (-0.2% overall Smatch, and -0.9 % in SRL). Secondly, we show that using the simple hierarchical relaxation ('h-relax', see equation (5))) results in a large drop in performance when compared to our softer relaxation (-1.5% Smatch). We hypothesize that the softer relaxation favors exploration of alignments and helps to discover better configurations.

5 Additional Related Work

Alignment performance has been previously identified as a potential bottleneck affecting AMR parsing (Damonte et al., 2017; Foland and Martin, 2017). Some recent work has focused on building aligners specifically for training their parsers (Werling et al., 2015; Wang and Xue, 2017). However, those aligners are trained independently of concept and relation identification and only used at pre-processing.

Treating alignment as discrete variables has been successful in some sequence transduction tasks (Yu et al., 2017, 2016). Our work is similar in that we also train discrete alignments jointly but the tasks, the inference framework and the decoders are very different.

For AMR parsing, another way to avoid using pre-trained aligners is to use seq2seq models (Konstas et al., 2017; van Noord and Bos, 2017). In particular, van Noord and Bos (2017) used character level seq2seq model and achieved the previous state-of-the-art result. However, their model is very data demanding as they needed to train it on additional 100K sentences parsed by other parsers. This may be due to two reasons. First, seq2seq models are often not as strong on smaller datasets. Second, recurrent decoders may struggle with predicting the linearized AMRs, as many statistical dependencies are highly non-local.

6 Conclusions

We introduced a neural AMR parser trained by jointly modeling alignments, concepts and relations. We make such joint modeling computationally feasible by using the variational auto-encoding framework and continuous relaxations. The parser achieves state-of-the-art results and ablation tests show that joint modeling is indeed beneficial. We believe that the proposed approach may be extended to other parsing tasks where alignments are latent (e.g., parsing to logical form (Liang, 2016)). Another promising direction is integrating character seq2seq to substitute the copy function. This should also improve the handling of negation and rare words.

References

Yoav Artzi, Kenton Lee, and Luke S. Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking.

Jörg Bornschein and Yoshua Bengio. 2015. Reweighted wake-sleep. *Proceedings of ICLR*.

Jan Buys and Phil Blunsom. 2017. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919. Association for Computational Linguistics.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An Incremental Parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 536–546.

Shibhansh Dohare and Harish Karnick. 2017. Text Summarization using Abstract Meaning Representation. *arXiv preprint arXiv:1706.01678*.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. *International Conference on Learning Representations*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 Task 8: Graph-based AMR Parsing with Infinite Ramp Loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Carlos de Juan Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *ACL*.

William Foland and James H. Martin. 2017. Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–472, Vancouver, Canada. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*.

Bevan K. Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *COLING*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *International Conference on Learning Representations*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman M. Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *HLT-NAACL*.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *In ACL*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning Latent Permutations with Gumbel-Sinkhorn Networks. *International Conference on Learning Representations*. Accepted as poster.

Arindam Mitra and Chitta Baral. 2016. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. In *AAAI*.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning*.

Rik van Noord and Johan Bos. 2017. Neural Semantic Parsing by Character-based Translation: Experiments with Abstract Meaning Representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

George Papandreou and Alan L Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 193–200. IEEE.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English Strings with Abstract Meaning Representation Graphs. In *EMNLP*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 Task 8: An Extended Transition-based AMR Parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California. Association for Computational Linguistics.

Chuan Wang and Nianwen Xue. 2017. Getting the Most out of AMR Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust Subgraph Generation Improves Abstract Meaning Representation Parsing. In *ACL*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017. The Neural Noisy Channel. In *International Conference on Learning Representations*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online Segment to Segment Neural Transduction. In *Proceedings of EMNLP*.