Getting the Most out of AMR Parsing

Chuan Wang and **Nianwen Xue**
Brandeis University
{cwang24,xuen}@brandeis.edu

Abstract

This paper proposes to tackle the AMR parsing bottleneck by improving two components of an AMR parser: concept identification and alignment. We first build a Bidirectional LSTM based concept identifier that is able to incorporate richer contextual information to learn sparse AMR concept labels. We then extend an HMM-based word-to-concept alignment model with graph distance distortion and a rescoring method during decoding to incorporate the structural information in the AMR graph. We show integrating the two components into an existing AMR parser results in consistently better performance over the state of the art on various datasets.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic representation where the meaning of a sentence is encoded as a rooted, directed graph. A number of AMR parsers have been developed in recent years (Flanigan et al., 2014; Wang et al., 2015b; Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Zhou et al., 2016; Goodman et al., 2016a), and the initial benefit of AMR parsing has been demonstrated in various downstream applications such as Information Extraction (Pan et al., 2015; Huang et al., 2016), Machine Comprehension (Sachan and Xing, 2016), and Language Generation (Flanigan et al., 2016b; Butler, 2016). However, AMR parsing parsing accuracy is still in the high 60%, as measured by the SMatch score (Cai and Knight, 2013), and a significant improvement is needed in order for it to positively impact a larger number of applications.

Previous research has shown that concept identification is the bottleneck to further improvement of AMR parsing. For example, JAMR (Flanigan et al., 2014), the first AMR parser, is able to achieve an F-score of 80% (close to the inter-annotator agreement of 83) if gold concepts are provided. Its parsing accuracy drops sharply to 62.3% when the concepts are identified automatically.

One of the challenges in AMR concept identification is data sparsity. A large portion of AMR's concepts are either word lemmas or sense-disambiguated lemmas drawn from Propbank (Palmer et al., 2005). Since the AMR Bank is relatively small, many of the concept labels in the development or test set only occur a few times or never appear in the training set. Werling et al. (2015) addresses this problem by defining a set of *generative* actions that maps words in the sentence to their AMR concepts and use a local classifier to learn these actions. Given such sparse data, making full use of contextual information is crucial to accurate concept labeling. Bidirectional LSTM has shown its success on many sequence labeling tasks since it is able to combine contextual information from both directions and avoid manual feature engineering. However, it is non-trivial to formalize concept identification as a sequence labeling problem because of the large concept label set. Inspired by Foland and Martin (2016; 2017), who first apply the Bidirectional LSTM to AMR concept identification by categorizing the large labels into a finite set of predefined types, we propose to address concept identification using Bidirectional LSTM with **Factored Concept Labels (FCL)**, where we re-group the concept label set based on their shared graph structure. This makes it possible for different concepts to be represented by one common label that captures the shared semantics of these concepts.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1266–1277
Copenhagen, Denmark, September 7–11, 2017.

Accurate concept identification also crucially depends on the word-to-AMR-concept alignment. Since there is no manual alignment in the AMR annotation, typically either a rule-based or unsupervised aligner is applied to the training data to extract the mapping between words and concepts. This mapping will then be used as reference data to train concept identification models. The JAMR aligner (Flanigan et al., 2014) greedily aligns a span of words to graph fragments using a set of heuristics. While it can easily incorporate information from additional linguistic sources such as WordNet, it is not adaptable to other domains. Unsupervised aligners borrow techniques from Machine Translation and treat sentence-to-AMR alignment as a word alignment problem between a source sentence and its linearized AMR graph (Pourdamghani et al., 2014) and solve it with IBM word alignment models (Brown et al., 1993). However, the distortion model in the IBM models is based on the linear distance between source side words while the linear order of the AMR concepts has no linguistic significance, unlike word order in natural language. A more appropriate sentence-to-AMR alignment model should be one that takes the hierarchical structure of the AMR into account. We develop a Hidden Markov Model (HMM)-based sentence-to-AMR alignment method with a novel **Graph Distance** distortion model to take advantage of the structural information in AMR, and apply a structural constraint to re-score the posterior during decoding time.

We present experimental results that show incorporating these two improvements to CAMR (Wang et al., 2016), a state-of-the-art transition-based AMR parser, results in consistently better Smatch scores over the state of the art on various datasets. The rest of paper is organized as follows. Section 2 describes related work on AMR parsing. Section 3 describes our improved LSTM based concept identification model, and Section 4 describes our alignment method. We present experimental results in Section 5, and conclude in Section 6.

2 Related Work

Existing AMR parsers are either transition-based or graph-based. Transition-based AMR parsers (Wang et al., 2015b,a; Goodman et al., 2016a,b), focus on modeling the correspondence between the dependency tree and the AMR graph of a sentence by designing a small set of actions that transform the dependency tree into the AMR graph. Pust et al. (2015) formulates AMR parsing as a machine translation problem in which the sentence is the source language input and the AMR is the target language output. AMR parsing systems that focus on modeling the graph aspect of the AMR includes JAMR (Flanigan et al., 2014, 2016a; Zhou et al., 2016), which treats AMR parsing as a procedure for searching for the Maximum Spanning Connected Subgraphs (MSCGs) from an edge-labeled, directed graph of all possible relations. Parsers based on Hyperedge Replacement Grammars (HRG) (Chiang et al., 2013; Björklund et al., 2016; Groschwitz et al., 2015) put more emphasis on modeling the formal properties of the AMR graph. One practical implementation of HRG-based parsing is that of (Peng et al., 2015; Peng and Gildea, 2016). The adoption of Combinatory Categorical Grammar (CCG) in AMR parsing has also been explored in (Artzi et al., 2015; Misra and Artzi, 2016), where a number of extensions have been proposed to enable CCG to work on the broad-coverage AMR corpus.

More recently, Foland and Martin (2016; 2017) describe a neural network based model that decomposes the AMR parsing task into a series of subproblems. Their system first identifies the concepts using a Bidirectional LSTM Recurrent Neural Network (Hochreiter and Schmidhuber, 1997), and then locates and labels the arguments and attributes for each predicate, and finally constructs the AMR using the concepts and relations identified in previous steps. (Barzdins and Gosko, 2016) first applies the sequence-to-sequence model (Sutskever et al., 2014) typically used in neural machine translation to AMR parsing by simply treating the pre-order traversal of AMR as foreign language strings. (Peng et al., 2017) also adopts the sequence-to-sequence model for neural AMR parsing and focuses on reducing data sparsity in neural AMR parsing with categorization of the concept and relation labels. In contrast, (Konstas et al., 2017) adopts a different approach and tackles the data sparsity problem with a self-training procedure that can utilize a large set of unannotated external corpus. (Buys and Blunsom, 2017) design a generic transition-based system for semantic graph parsing and apply sequence-to-sequence framework to learn the

transformation from natural language sequences to action sequences.

3 Concept Identification with Bidirectional LSTM

In this section, we first introduce how we categorize AMR concepts using **Factored Concept Labels**. We then integrate character-level information into a Bidirectional LSTM through Convolutional Neural Network (CNN)-based embeddings.

3.1 Background and Notation

Given a pair of AMR graph G and English sentence S, a look-up table M is first generated which maps a span of tokens to concepts using an aligner. Although there are differences among results generated by different aligners, in general, the aligned AMR concepts can be classified into the following types:

- PREDICATE. Concepts with sense tags, which are frames borrowed from Propbank, belong to this case. Most of the tokens aligned to this type are verbs and nouns that have their own argument structures.
- NON-PREDICATE. This type of concepts are mostly lemmatized word tokens from the original English sentences.
- CONST. Most of the numerical expressions in English sentences are aligned to this type, where AMR concepts are normalized numerical expressions.
- MULTICONCEPT. In this type, one or more word tokens in an English sentence are aligned to multiple concepts that form a sub-structure in an AMR graph. The most frequent case is named entity subgraphs. For example, in Figure 1, *"Mr. Vinken"* is aligned to subgraph `(p / person :name (m / name :op1 "Mr." :op2 "Vinken")`.

3.2 Factored Concept Labels

To be able to fit AMR's large concept label space into a sequence labeling framework, redefining the label set is necessary in order to make the learning process feasible. While it is trivial to categorize the PREDICATE, NON-PREDICATE, CONST cases, there is no straightforward way to deal with the MULTICONCEPT type. Foland and Martin (2016) only handle named entities, which constitute the majority of the MULTICONCEPT cases, where they adopt BIOES tags to detect the boundary and use an additional Bidirectional LSTM to learn the fine-grained named entity concept types. For other MULTICONCEPT cases, they only use the leaf concepts and ignore other parts of the subgraphs. Figure 2 shows the concept label distribution on development set of LDC2015E86, where we can see nearly half of the MULTICONCEPTcases are not named entities.

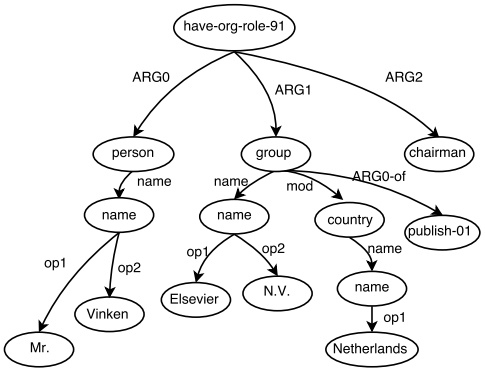


Figure 1: An example AMR graph for sentence: "*Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.*".

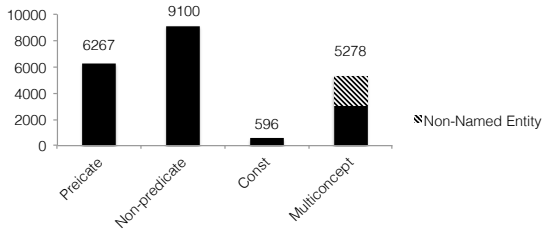


Figure 2: AMR concept label distribution for development set of LDC2015E86

Based on the observation that many of the MULTICONCEPT cases are actually similarly structured subgraphs that only differ in the lexical items, we choose to factor the lexical items out of the subgraph fragments and use the skeletal structure as the fine-grained labels, which we refer as **Factored Concept Label (FCL)**.

Figure 4 shows that although English words "visitor" and "worker" have been aligned to different subgraph fragments, after replacing the lexical items, in this case the leaf concepts `visit-01` and `work-01` with a placeholder "x", we are able to arrive at the same FCL. The strategy for determining the FCL for a word is simple: for each English word w and the subgraph s it aligns to, if the length of the longest overlapping substring be-

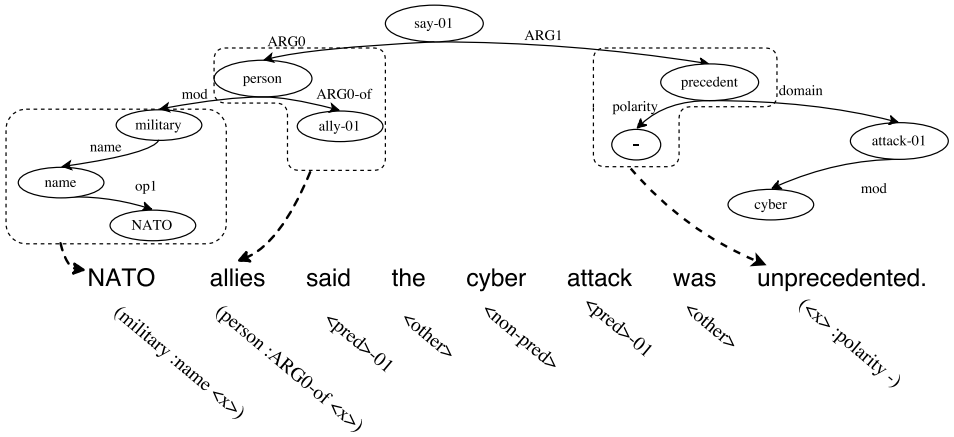


Figure 3: One example of generating FCL for sentence "*NATO allies said the cyber attack was unprecedented.*"

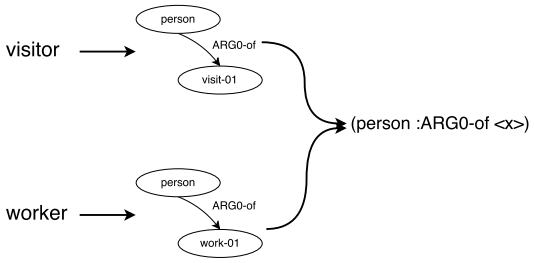


Figure 4: One example of generating FCL

tween w and the leaf concept c of s is greater than 4, we replace c with a placeholder.

Despite this simple strategy, our results show that it can capture a wide range of MULTICONCEPT cases while keeping the new label space manageable. While the named entity can be easily categorized using FCL, it can also cover some other common cases such as morphological expressions of negation (e.g., "inadequate") and comparatives (e.g., "happier"). Setting a frequency threshold to prune out the noisy labels, we are able to extract 91 canonical FCL labels on the training set. Our empirical results show that this canonical FCL label set can cover 96% of the MULTICONCEPT cases on the development set. Figure 3 gives one full example of FCLs generated for one sentence. For the PREDICATE cases, following (Foland and Martin, 2016), we only use the sense tag as its label.

We use label $\langle other \rangle$ to label stop words that do not map to AMR concepts. The MULTICONCEPT cases are handled by FCL. The FCL label set generated by this procedure can be treated as an abstraction of the original AMR concept label space, where it groups concepts that have similar AMR subgraphs into the same category.

3.3 CNN-based Character-level Embedding

After constructing the new label set with FCL, we set up a baseline Bidirectional LSTM using the concatenation of word and NER embeddings as the input. For each input word w and its NER tag t, their embeddings e_{w} and e_{t} are extracted from a word embedding matrix $W_{\text{wd}} \in \mathbb{R}^{d_{\text{wd}} \times |V_{\text{wd}}|}$ and a NER tag embedding matrix $W_{\text{t}} \in \mathbb{R}^{d_{\text{t}} \times |V_{\text{t}}|}$ respectively, where d_{wd} and d_{t} are the dimensions of the word and NER tag embedding matricies, $|V_{\text{wd}}|$ and $|V_{\text{t}}|$ are the sizes of the word and NER tag vocabulary.

Although this architecture is able to capture long-range contextual information, it fails to extract information originating from the word form itself. As we have discussed above, in some of the MULTICONCEPT cases the concepts are associated with the word forms themselves and won't benefit from its contextual information. For example, in "*unprecedented*", the prefix "*un*" itself already gives enough information to predict the FCL label $\langle x \rangle : polarity$ –, which indicates negative polarity. In order to incorporate such morphological and shape information, we choose to add a convolutional layer to extract character-level representations. A similar technique has been applied to Named Entity Recognition (Santos and Guimaraes, 2015; Chiu and Nichols, 2015) and we only provide a brief description of the architecture here. For a word w composed of characters $\{c_1, c_2, \ldots, c_l\}$, where l is the length of word w, we learn a character embedding matrix $W_{\text{c}} \in \mathbb{R}^{d_{\text{c}} \times |V_{\text{c}}|}$, where d_{c} is the character embedding dimension defined by the user and V_{c} is character vocabulary size. After retrieving the character embedding ch_i for each character c_{i} in word w, we

obtain a sequence of vectors $\{ch_1, ch_2, \ldots, ch_l\}$. This serves as the input to convolutional layer.

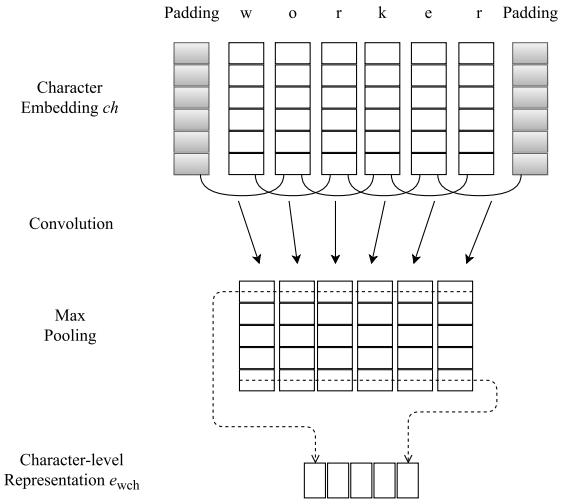


Figure 5: The architecture of the CNN-based character-level embedding.

The convolutional layer applies a linear transformation to the local context of a character in the input sequence, where the local context is parameterized by window size k. Here we define the local context of the character embedding ch_i to be:

$$f_i = (ch_{i-(k-1)/2}, \ldots, ch_{i+(k-1)/2})^\top \quad (1)$$

The j-th element of the convolutional layer output vector e_{wch} is computed by element-wise max-pooling (Ranzato et al., 2007):

$$[e_{\text{wch}}]_j = \max_{1 \le i \le l} [W^0 f_i + b^0]_j \quad (2)$$

W^0 and b^0 are the parameters of the convolutional layer. And the output vector e_{wch} is the character level representation of the word w. The architecture of the model is shown in Figure 5.

The final input to the Bidirectional LSTM is the concatenation of three embeddings $[e_w, e_t, e_{\text{wch}}]$ for each word position.

4 Aligning an English Sentence to its AMR graph

Given an AMR graph G and English sentence $\boldsymbol{e} = \{e_1, e_2, \ldots, e_i, \ldots, e_I\}$, in order to fit them into the traditional word alignment framework, the AMR graph G is normally linearized using depth first search by printing each node as soon as it it visited. The re-entrance node is printed but not expanded to preserve the multiple mentions of concept. The relation (also called AMR role token) between concepts are preserved in the unsupervised aligner (Pourdamghani et al., 2014) because they also try to align relations to English words. We ignore the relations here since we focus on aligning concepts. Therefore the linearized concept sequences can be represented as $\boldsymbol{g} = \{g_1, g_2, \ldots, g_j, \ldots, g_J\}$. However, although this configuration makes it easy to adopt existing word alignment models, it also ignores the structural information in the AMR graph.

In this section, we proposes a method that incorporates the structural information in the AMR graph through a distortion model inside an HMM-based word aligner. We then further improve the model with a re-scoring method during decoding time.

4.1 HMM-based Aligner with Graph Distance Distortion

Given a sequence pair $(\boldsymbol{e}, \boldsymbol{g})$, the HMM-based word alignment model assumes that each source word is assigned to exactly one target word, and defines an asymmetric alignment for the sentence pair as $\boldsymbol{a} = \{a_1, a_2, \ldots, a_i, \ldots, a_I\}$, where each $a_i \in [0, J]$ is an alignment from source position i to target position a_i, $a_i = 0$ means that e_i is not aligned to any target words. Note that in the AMR to English alignment context, both the alignment and the graph structure is asymmetric, since we only have AMR graph annotation on in linearized AMR sequence g. Unlike the traditional word alignment for machine translation, here we will have different formulas for each translation direction. In this section, we only discuss the translation from English (source) to linearized AMR concepts (target) and we will discuss the AMR to English direction in the following section.

The HMM-based model breaks the generative alignment process into two factors:

$$P(\boldsymbol{e}, \boldsymbol{a} \mid \boldsymbol{g}) = \prod_{i=1}^{I} P_d(a_i \mid a_{i-1}, J) P_t(e_i | g_{a_i}) \quad (3)$$

where P_d is the distortion model and P_t is the translation model. Traditionally, the distortion probability $P_d(j \mid j', J)$ is modeled to depend only on the jump width $(j - j')$ (Vogel et al., 1996) and is defined as:

$$P_d(j \mid j', J) = \frac{ct(j - j')}{\sum_{j''=1}^{J} ct(j'' - j')} \quad (4)$$

where $ct(j - j')$ is the count of jump width. This formula simultaneously satisfies the normalization constraint and captures the *locality* assumption that words that are adjacent in the source sentence tend to align to words that are closer in the target sentence.

As the linear *locality* assumption does not hold among linearized AMR concepts, we choose instead to encode the distortion probability through graph distance, which is given by:

$$P_{\text{gd}}(j \mid j', G) = \frac{ct(d(j, j'))}{\sum_{j''} ct(d(j'', j'))} \quad (5)$$

The graph distance $d(j, j')$ is the length of shortest path on AMR graph G from concept j to concept j'. Note that we have to artificially normalize $P_{\text{gd}}(j \mid j', G)$, because unlike the linear distance between word tokens in a sentence, there can be multiple concepts that can have the same distance from the j'-th concept in the AMR graph, as pointed out in (Kondo et al., 2013).

During training, just like the original HMM-based aligner, an EM algorithm can be applied to update the parameters of the model.

4.2 Improved Decoding with Posterior Rescoring

So far, we have integrated the graph structure information into the forward direction (English to AMR). To also improve the reverse direction model (AMR to English), we choose to use the graph structure to rescore the posterior during decoding time.

Compared with Viterbi decoding, posterior thresholding has shown better results in word alignment tasks (Liang et al., 2006). Given threshold γ, for all possible alignments, we select the final alignment based on the following criteria:

$$\boldsymbol{a} = \{(i, j) : p(a_j = i \mid \boldsymbol{g}, \boldsymbol{e}) > \gamma\} \quad (6)$$

where the state probability $p(a_j = i \mid \boldsymbol{g}, \boldsymbol{e})$ is computed using the forward-backward algorithm. The forward algorithm is defined as:

$$\alpha_{j,i} = \sum_{i'} \alpha_{j-1,i'} p(a_j = i \mid a_{j-1} = i') p(g_j \mid e_{a_j}) \quad (7)$$

To incorporate the graph structure, we rescale the distortion probability in reverse direction model as:

$$p_{\text{new}}(a_j = i \mid a_{j-1} = i') = p(a_j = i \mid a_{j-1} = i')^{e^{\Delta d}} \quad (8)$$

where the scaling factor $\Delta d = d_j - d_{j-1}$ is the graph depth difference between the adjacent AMR concepts g_j and g_{j-1}. We also apply the same procedure for the backward computation. Note that since the model is in reverse direction, the distortion $p(a_j = i \mid a_{j-1} = i')$ here is still based on English word distance, jump width.

This rescaling procedure is based on the intuition that after we have processed the last concept g_{j-1} in some subgraph, the next concept g_j's aligned English position i is not necessarily related to the last aligned English position i'. Figure 6 illustrates this phenomenon: Although `we` and `current` are adjacent concepts in linearized AMR sequence, they are actually far away from each other in the graph (with a graph depth difference of -2). However, the distortion based on the English word distance mostly tends to choose the closer word, which may yield a very low probability for our correct answer here (the jump width between "Currently" and "our" is -6). By applying the exponential scaling factor, we are able to reduce the differences between different distortion probabilities. On the contrary, when the distortion probability is reliable (the absolute value of the graph depth difference is small), the model chooses to trust the distortion and picks the closer English word.

The rescaling factor can be viewed as a selection filter for decoding, where it relies on the graph depth difference Δd to control the effect of learned distortion probability. Note that after the rescaling, the resulting distortion probability no longer satisfies the normalization constraint. However, we only apply this during decoding time and experiments show that the typical threshold $\gamma = 0.5$ still works well for our case.

4.3 Combining Both Directions

Empirical results show that combining alignments from both directions improve the alignment quality (DeNero and Klein, 2007; Och and Ney, 2003; Liang et al., 2006). To combine the alignments, we adopt a slightly modified version of posterior thresholding, *competitive thresholding*, as proposed in (DeNero and Klein, 2007), which tends to select alignments that form a contiguous span.

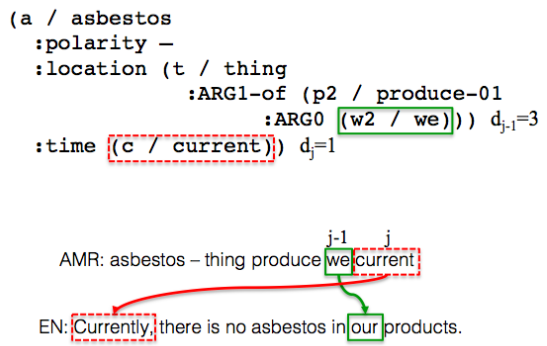


Figure 6: AMR graph annotation, linearized concepts for sentence "*Currently, there is no asbestos in our products*". The concept `we` in solid line is the $(j-1)$-th token in linearized AMR. It is aligned to English word "our" and its depth in graph d_{j-1} is 3. While the word distance-based distortion prefers an alignment near "our", the correct alignment needs a longer distortion.

5 Experiments

We first test the performance of our Bidirectional LSTM concept identifier and HMM-based aligner as standalone tasks, where we investigate the effectiveness of each component in AMR parsing. Then we report the final results by incorporating both components to CAMR (Wang et al., 2016). At the model development stage, we mainly use the dataset LDC2015E86 used in the SemEval Shared Task (May, 2016). Note that this dataset includes `:wiki` relations where every named entity concept is linked to its wikipedia entry. We remove this information in the training data throughout the development of our models. At the final testing stage, we add wikification using an off-the-shelf AMR wikifier (Pan et al., 2015) as a post-processing step. All AMR parsing results are evaluated using the Smatch (Cai and Knight, 2013) scorer.

5.1 Bidirectional LSTM Concept Identification Evaluation

In order to isolate the effects of our concept identifier, we first use the official alignments provided by SemEval. The alignment is generated by the unsupervised aligner described in (Pourdamghani et al., 2014). After getting the alignment table, we generate our FCL label set by filtering out noisy FCL labels that occur fewer than 30 times in the training data. The remaining FCL labels account for 96% of the MULTICONCEPT cases in the development set. Adding other labels that include PREDICATE, NON-PREDICATE and CONST gives us 116 canonical labels. UNK label is added to handle the unseen concepts.

In the Bidirectional LSTM, the hyperparameter settings are as follows: word embedding dimension $d_{wd} = 128$, NER tag embedding dimension $d_t = 8$, character embedding dimension $d_c = 50$, character level embedding dimension $d_{wch} = 50$, convolutional layer window size $k = 2$.

Input	P	R	F_1	Acc
word,NER	81.2	80.6	80.9	85.4
word,NER,CNN	83.3	82.7	83.0	87.0

Table 1: Performance of Bidirectional LSTM with different input.

Table 1 shows the performance on the development set of LDC2015E86, where the precision, recall and F-score are computed by treating $\langle other \rangle$ as the negative label and accuracy is calculated using all labels. We include accuracy here since correctly predicting words that don't invoke concepts is also important. We can see that utilizing CNN-based character level embedding yields an improvement of around 2 percentage points absolute for both F-score and accuracy, which indicates that morphological and word shape information is important for concept identification.

Impact on AMR Parsing In order to test the impact of our concept identification component on AMR parsing, we add the predicted concept labels as features to CAMR. Here is the detailed feature set we add to CAMR's feature templates. To clarify the notation, we refer the concept labels predicted by our concept identifier as c_{pred} and the candidate concept labels in CAMR as c_{cand}:

- `pred_label`. c_{pred} used directly as a feature.
- `is_eq_sense`. A binary feature of whether a c_{pred} and c_{cand} have the same sense (if applicable).

One reason why we choose to add the concept label and sense as features to predict the concept rather than using the predicted label to recover the concept directly is that the latter is not a straightforward process. For example, since we generalize all the predicates to a compact form `<pred-xx>`, for irregular verbs like "*became*" ⇒ `become-01`, simply stemming the inflected verb form will not

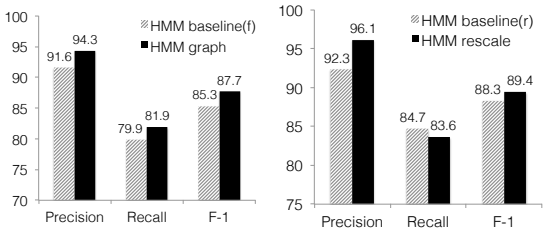
give us the correct concept even if the sense is predicted correctly. However, since CAMR uses the alignment table to store all possible concept candidates for a word, adding our predicated label as a feature could potentially help the parser to choose the correct concept. In order to take full advantage of the predicted concept labels, we also extend CAMR so that it can discover candidate concepts outside of the alignment table. To achieve this, during the FCL label generation process, we first store the string-to-concept mapping as a template. For example, when we generate the FCL label `(person :ARG0-of <x>-01)` from "worker", we also store the template `<x>er -> (person :ARG0-of <x>-01)`. Then during decoding time, we would enumerate every template and try to use the left hand side of the template (which is `<x>er`) as a regular expression to match current word. Once we find a match in all the template entries, we would substitute the placeholder in right hand side with the matched substring to get the candidate concept label. As a result, even we haven't seen "teacher", by matching teacher with the regular expression `(.*)er`, we could generate the correct answer `(person :ARG0-of teach-01)`. We refer this process as *unknown concept generation*. Table 2 summarizes the impact of our proposed methods on development set of LDC2015E86. We can see that by utilizing the unknown concept generation and features derived from c_{pred} , both precision and recall improve by about 1 percentage point, which indicates that the new feature brings richer information to the concept prediction model to help correctly score candidate concepts from the alignment table.

Parsers	P	R	F_1
CAMR (Wang et al., 2016)	72.3	61.4	66.5
CAMR-*gen*	72.1	62.0	66.6
CAMR-*gen*-c_{pred}	**73.6**	**62.6**	**67.6**

Table 2: Performance of AMR parsing with c_{pred} features without *wikification* on dev set of LDC2015E86. The first row is performance of the baseline parser. The second row adds unknown concept generation and the last row additionally extends the baseline parser with c_{pred} features.

5.2 HMM-based AMR-to-English Aligner Evaluation

To validate the effectiveness of our proposed alignment methods, we first evaluate our forward (English-to-AMR) and reverse (AMR-to-English) aligners against the baseline HMM word alignment model, which is the Berkeley aligner toolkit (DeNero and Klein, 2007). Then we combine the forward and reverse alignment results using competitive thresholding. We set the threshold γ to be 0.5 in the following experiments. To evaluate the alignment quality, we use 200 hand-aligned sentences from (Pourdamghani et al., 2014) split equally as the development and test sets. We process the English sentences by removing stop words, following similar procedure as in (Pourdamghani et al., 2014). When linearizing AMR graphs, we instead remove all the relations and only keep the concepts. For all models, we run 5 iterations of IBM Model 1 and 2 iterations of HMM on the whole dataset.



(a) HMM forward (b) HMM reverse

Figure 7: Our improved forward (graph) and reverse(rescale) model compared with HMM baseline on hand aligned development set.

From Figure 7a, we can see that our graph-distance based model improves both the precision and recall by a large margin, which indicates the graph distance distortion better fits the English-to-AMR alignment task. For the reverse model, although our HMM rescaling model loses accuracy in recall, it is able to improve the precision by around 4 percentage points, which confirms our intuition that the rescoring factor is able to keep reliable alignments and penalize unreliable ones. We then combine our forward and reverse alignment result using competitive thresholding. Table 3 shows the combined result against hand-aligned dev and test sets.

Datasets	P	R	F_1
dev	97.7	84.3	90.5
test	96.9	84.6	90.3

Table 3: Combined HMM alignment result evaluation.

Dataset	Aligner	P	R	F_1
Dev	JAMR	73.6	62.6	67.6
	ISI	72.8	**64.6**	**68.4**
	Our HMM	**73.6**	63.6	68.2
Test	JAMR	71.6	61.3	66.0
	ISI	70.6	**62.7**	66.4
	Our HMM	**72.1**	62.5	**67.0**

Table 4: AMR parsing result (without *wikification*) with different aligner on development and test of LDC2015E86, where JAMR is the rule-based aligner, ISI is the modified IBM Model 4 aligner

Impact on AMR Parsing To investigate our aligner's contribution to AMR parsing, we replace the alignment table generated by the best performing aligner (the forward and reverse combined) in the previous section and re-train CAMR with the predicted concept label features included.

From Table 4, we can see that the unsupervised aligner (ISI and HMM) generally outperforms the JAMR rule-based aligner, and our improved HMM aligner is more consistent than the ISI aligner (Pourdamghani et al., 2014), which is a modified version of IBM Model 4.

5.3 Comparison with other Parsers

We first add the *wikification* information to the parser output using the off-the-shelf AMR wikifier (Pan et al., 2015) and compare results with the state-of-the-art parsers in 2016 SemEval Shared Task. We also report our result on the previous release (LDC2014T12), AMR annotation Release 1.0, which is another popular dataset that most of the existing parsers report results on. Note that the Release 1.0 annotation doesn't include *wiki* information.

Dataset	Parsers	P	R	F_1
SemEval Test	CAMR	70.3	63.1	66.5
	RIGA	-	-	67.2
	JAMR(2016a)	70	65	67
	Our parser	**71.7**	**64.9**	**68.1**
SemEval Blind Test	CAMR	67.4	57.3	62.0
	RIGA	68.0	57.0	62.0
	JAMR(2016a)	-	-	56
	Our parser	**68.2**	**59.5**	**63.6**

Table 5: Comparison with the winning systems in SemEval (**with** *wikification*) on test and blind test sets

CAMR and RIGA (Barzdins and Gosko, 2016) are the two best performing parsers that participated in SemEval 2016 shared task. While we use CAMR as our baseline system, the parser from RIGA is also based on a version of CAMR extended with a error-correction wrapper and an ensemble with a character-level neural sequence-to-sequence model. Our parser outperforms both systems by around 1.5 percentage points, where the improvement in recall is more significant, at around 2 percentage points.

Parsers	P	R	F_1
CAMR	71.3	62.2	66.5
(Zhou et al., 2016)	70	62	66
(Pust et al., 2015)	-	-	65.8
Our parser	**72.7**	**64.0**	**68.07**

Table 6: Comparison with the existing parsers on full test set of LDC2014T12

Table 6 shows the performance of our parser on the full test set of LDC2014T12. We include the previous best results on this dataset. The parser proposed in (Zhou et al., 2016) jointly learns the concept and relation through an incremental joint model. We also include the AMR parser by (Pust et al., 2015) that models AMR parsing as a machine translation task and incorporates various external resources. Our parser still achieves the best result without incorporating external resources other than the NER information.

6 Conclusion

In this paper, we presents work that improves AMR parsing performance by focusing on two components of the parser: concept identification and alignment. We first build a Bidirectional LSTM based concept identifier which is able to incorporate richer context and learn sparse concept labels. Then we extend the HMM-based word alignment model with a graph distance distortion and a rescoring method during decoding to incorporate the graph structure information. By integrating the two components into an existing AMR parser, our parser is able to outperform state-of-the-art AMR parsers and establish a new state of the art.

Acknowledgments

We want to thank the anonymous reviewers for their suggestions and detailed error checking.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

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. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. *arXiv preprint arXiv:1604.01278*.

Henrik Björklund, Frank Drewes, and Petter Ericson. 2016. Between a rock and a hard place–uniform parsing for hyperedge replacement dag grammars. In *International Conference on Language and Automata Theory and Applications*, pages 521–532. Springer.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Alastair Butler. 2016. Deterministic natural language generation from meaning representations for machine translation. In *Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016)*, pages 1–9. Association for Computational Linguistics.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. *CoRR*, abs/1704.07092.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Meeting of the Association of Computational Linguistics*, Sofia, Bulgaria.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *CoRR*, abs/1511.08308.

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24.

Jeffrey Flanigan, Chris Dyer, A. Noah Smith, and Jaime Carbonell. 2016a. 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, Chris Dyer, A. Noah Smith, and Jaime Carbonell. 2016b. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

William Foland and H. James Martin. 2016. CU-NLP at SemEval-2016 Task 8: AMR parsing using LSTM-based recurrent neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1197–1201. Association for Computational Linguistics.

William Foland and James Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association of the Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016a. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016b. UCL+Sheffield at SemEval-2016 task 8: Imitation learning for amr parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1167–1172. Association for Computational Linguistics.

Jonas Groschwitz, Alexander Koller, Christoph Teichmann, et al. 2015. Graph parsing with s-graph grammars. In *ACL (1)*, pages 1481–1490.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, R. Clare Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268. Association for Computational Linguistics.

Shuhei Kondo, Kevin Duh, and Yuji Matsumoto. 2013. Hidden markov tree model for word alignment. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 503–511.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR*, abs/1704.08381.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. *Proceedings of SemEval*, pages 1063–1073.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce ccg semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786, Austin, Texas. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1130–1139.

Xiaochang Peng and Daniel Gildea. 2016. UofR at SemEval-2016 Task 8: Learning synchronous hyperedge replacement grammar for AMR parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1185–1189. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

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, Valencia, Spain. Association for Computational Linguistics.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal. Association for Computational Linguistics.

M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 486–492. Association for Computational Linguistics.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

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, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *arXiv preprint arXiv:1506.03139*.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR Parsing with an Incremental Joint Model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas. Association for Computational Linguistics.