Unsupervised AMR-Dependency Parse Alignment

Wei-Te Chen
Department of Computer Science
University of Colorado Boulder
weite.chen@colorado.edu

Martha Palmer
Department of Linguistics
University of Colorado Boulder
martha.palmer@colorado.edu

Abstract

In this paper, we introduce an Abstract Meaning Representation (AMR) to Dependency Parse aligner. Alignment is a preliminary step for AMR parsing, and our aligner improves current AMR parser performance. Our aligner involves several different features, including named entity tags and semantic role labels, and uses Expectation-Maximization training. Results show that our aligner reaches an 87.1% F-Score score with the experimental data, and enhances AMR parsing.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic representation that expresses the logical meaning of English sentences with rooted, directed, acylic graphs. AMR associates semantic concepts with the nodes on a graph, while the relations are the label edges between concept nodes. Meanwhile, AMR relies heavily on predicate-argument relations from PropBank (Palmer et al., 2005), which share several edge labels. The representation also encodes rich information, like semantic roles (all the "ARGN" tags from PropBank), named entities (NE) ("person", "location", etc., concepts), wiki-links (":wiki" tags), and co-reference (reuse of variables, e.g., p). An example AMR in PENMAN format (Matthiessen and Bateman, 1991) is shown in Figure 1.

```
(j / join-01
  :ARG0 (p / person :wiki -
    :name (p2 / name
      :op1 "Pierre" :op2 "Vinken")
    :age (t / temporal-quantity :quant 61
      :unit (y / year)))
  :ARG1 (b / board
    :ARG1-of (h / have-org-role-91
      :ARG0 p
      :ARG2 (d2 / director
        :mod (e / executive :polarity -))))
  :time (d / date-entity :month 11 :day 29))
```

Figure 1: The AMR annotation of sentence "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29." in PENMAN format

The design of an AMR to English sentence aligner is the first step for implementation of an AMR parser, since AMR annotation does not contain links between each AMR concept and the original span of words. The basic alignment strategy is to link the AMR tokens (either concepts or edge labels) with their corresponding span of words. Another strategy is to find the alignment from an AMR concept to a word node in a dependency parse tree, the goal of this paper. A dependency parse tree is a good structure for attaching more information, e.g. named entity tags, lemma, and semantic role labels, etc., and provides richer syntactic information than the span of words. An alignment between an AMR concept and a dependency node represents a correspondence between the meaning of this concept and its child concepts and the phrase governed by the dependency node (i.e., head word). An example alignment is shown in Figure 2. For example, the word node "Vinken" on the dependency parse side in Figure 2 links to the lexical concept of "Vinken" and, furthermore, links to the "*p2/name*" and the "*p/person*" concepts since "Vinken" is the head of the named entity "Pierre Vinken" and the head of the whole noun phrase "Pierre Vinken, 61 years old.". In our work, we use Expectation-Maximization(EM) (Dempster et al., 1977) to train different feature probabilities, including rule-based features, lexical forms, relation labels, named entity tags, semantic role

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 558–567,
Valencia, Spain, April 3-7, 2017.

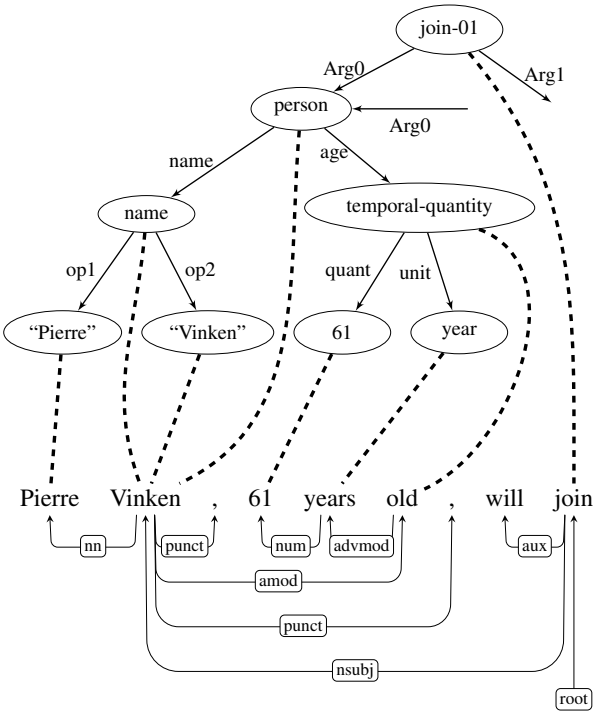


Figure 2: The alignment between a subgraph of an AMR (top) and a dependency parse (bottom) for the "Pierre Vinken" sentence. Dashed lines link dependency parse nodes and corresponding concepts.

labels, and global features, etc. Then EM processing incorporates all the individual probabilities and estimates the final alignments.

We will describe AMR-English sentence alignment in general, and review related work, in Section 2. Then the descriptions of our AMR-dependency parse features and alignment model are in Section 3. Our beam-search docoder is described in Section 4. Our experimental results are presented in Section 5, followed by our conclusion and discussion of future work (Section 6).

2 AMR-English Sentence Aligner

A preliminary step for an AMR parser is aligning AMR concepts and the original spans of words. JAMR (Flanigan et al., 2014) includes a heuristic alignment algorithm between AMR concepts and words or phrases from the original sentence. They use a set of alignment rules, like named entity, fuzzy named entity, data entity, etc., with a greedy strategy to match the alignments. This aligner achieves a 90% F_1 score on hand aligned AMR-sentence pairs. On the other hand, the ISI Aligner (Pourdamghani et al., 2014) presents a generative model to align AMR graphs to sentence strings. They propose a string-to-string alignment model which transfers the AMR expression to a linearized string representation as the initial step. Their training method is based on the IBM word alignment model (Brown et al., 1993) but they modify the objective function of the alignment model. IBM Model-4 with a symmetric method reaches the highest F_1 score, 83.1%. When separating the alignments into roles (edge labels) and non-roles (concepts), F_1 scores are 49.3% and 89.8%, respectively. In Werling's AMR parser (Werling et al., 2015), they conceive of the alignment task as a linear programming relaxation of a boolean problem. The objective function is to maximize the sum of action reliability. Each concept is constrained to align to exactly one token in a sentence. This ensures that only adjacent nodes or nodes that share the same title refer to the same token. They hand-annotate 100 AMR parses, and their aligner achieves an accuracy of 83.2%. By providing alternative alignments to their graph-based AMR parser, their aligner achieves a better Smatch score than JAMR's aligner.

However, two transition-based parsers which parse dependency parse tree structures into AMRs, e.g., the CAMR system (Wang et al., 2015; Wang et al., 2016) and the RIGA system (Barzdins and Gosko, 2016), tie for the best results in SemEval-2016 task 8 (May, 2016). It is important to note that the JAMR aligner was not designed to align between a dependency word node and an AMR concept where its alignment F_1 score is only 69.8% (see Section 5.2). In order to deal with this problem, (Chen, 2015) proposed a preliminary aligner which estimates alignments by learning the feature probabilities of lexical (surface) forms, relations, named entities and semantic roles jointly. Besides the objective to obtain alignment between AMR concepts and original word spans, the estimation of these feature probabilities is also useful for further development of the AMR parser with these initial models. In our paper, we extend their previous work by adding rule-based and global features, and adding a beam-search algorithm at decoding time.

3 AMR-Dependency Parse Aligner

Our approach is an AMR-to-Dependency parse aligner, which represents one AMR as a list of

Concepts $C = \langle c_1, c_2, \ldots, c_{|C|} \rangle$, and the corresponding dependency parse as a list of dependency word nodes $D = \langle d_1, d_2, \ldots, d_{|D|} \rangle$. An alignment function a is designed to produce exactly one alignment to a dependency node d_{c_j} for each concept c_j, within a single sentence. Alternatively, we can view a as a mapping function that accepts one input variable concept c_j and outputs a dependency node d_{c_j} with which c_j is aligned. A is the alignment set that contains all different a_l that cover possible alignments within C and D. Our model adopts an asymmetric alignment direction, where one concept maps to exactly one dependency parse node, and each dependency parse node can be aligned by zero to multiple concepts. We denote dependency node d_c linked by concept c as $d_c = a(c)$. c^p is the parent concept of concept c, while $c^{s_1}, c^{s_2}, ..., c^{s_k}$ are the k child concepts of concept c.

3.1 Features

3.1.1 Basic Features

Several of the AMR concepts use the word form directly. For example, the concept "join-01" in Figure 1 would align to the dependency node "join" naturally. Similarly, the leaf concepts usually align to identical terms in the dependency parse. In Figure 1, the names "Pierre" and "Vinken" are aligned to their word forms on the dependency parse leaves. Therefore, we design a straightforward rule-based probability, P_{rule}, which catches the appearance of the surface form. $P_{rule}(c, d_c)$ is defined as the probability that the matching type for a given concept c and dependency node d_c are linked. The different types of rules, e.g., word, lemma, numbers, and date, etc., and their proportional applicability to both AMR concepts and leaves are listed in Table 1. For example, the rule "Date" type aligns concept "11" with word node "November" in Figure 1, while "Numbers" aligns concept "5" with word node "five". P_{rule} decides which match type to apply by following a greedy matching strategy.

3.1.2 External Features

To capture alignments for concepts which do not match any of the above basic rules, we design the following four external feature probabilities:

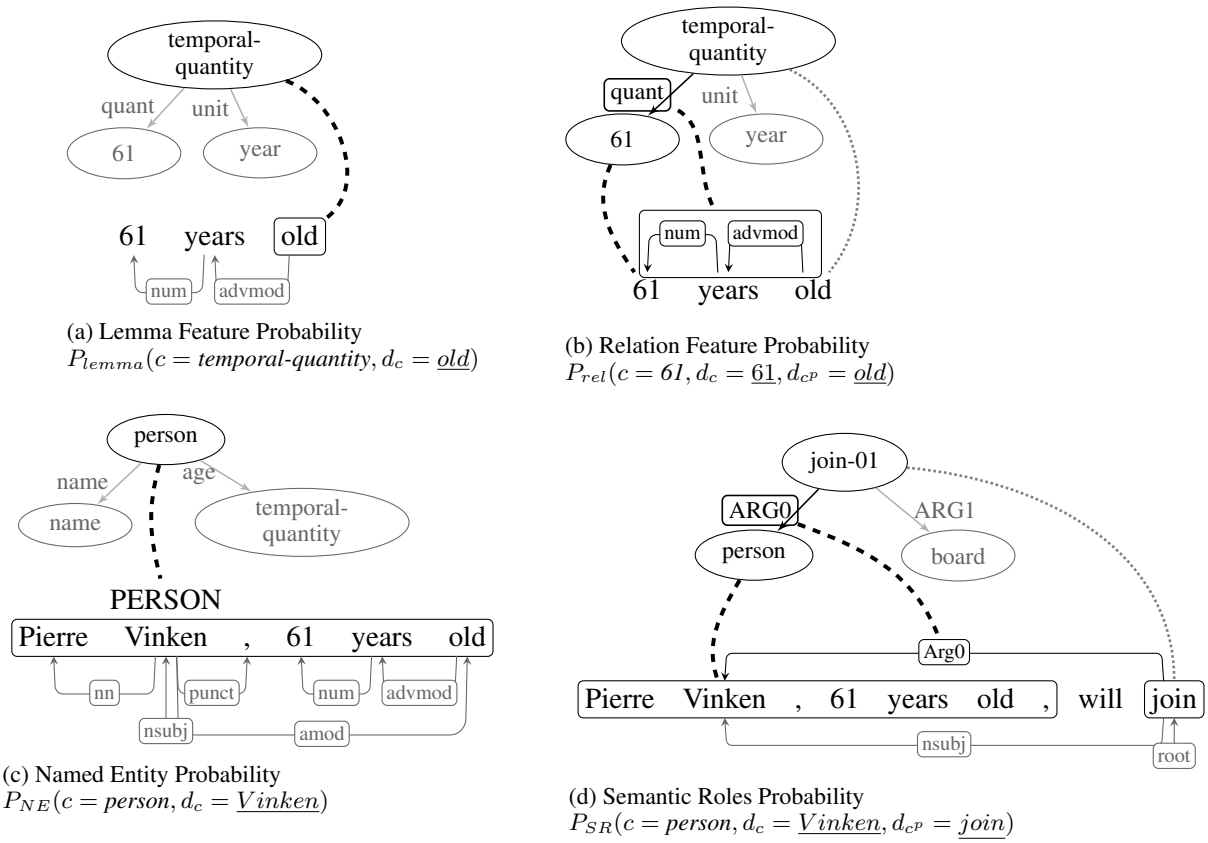
$P_{Lemma}(c, d_c) = P(c|\textit{Word}(d_c))$
Lemma Probability represents the likelihood that a concept c aligns to a dependency word d_i. For example, in Figure 3a, the concept $c =$"*temporal-quantity*" is highly likely to align to the word node $d_c =$ "old" since "old" is usually the head word of a phrase expressing age ("61 years old" here). Also, *have-org-role-91* can align to the word node "director" since "director" appears quite often with *have-org-role-91* (defined as roles in organizations). Besides, some special leaf concepts, like "*:polarity -*" (negative), and "*:mode expressive*" (which is used to mark exclamational word), also rely on this feature rather than the basic rules.

	Match Type	at Concept	at Leaf
(1)	Word	45.2%	73.4%
(2)	Word (case insensitive)	-	0.9%
(3)	Lemma (case insensitive)	10.8%	0.3%
(4)	Partial match with word	6.1%	8.2%
(5)	Partial match with lemma	0.2%	0.3%
(6)	Numbers	-	3.1%
(7)	Ordinal Numbers	-	2.8%
(8)	Date	-	4.3%
(9)	Others	37.7%	6.5%

Table 1: The rules and distribution of basic match types

$P_{rel}(c, d_c, d_{c^p}) = P(\textit{AMRLabel}(c)|\textit{Path}(d_c, d_{c^p}))$
Relation Probability is the conditional probability of the AMR relation label of c, given the parse tree path between d_c and d_{c^p}, where d_c and d_{c^p} represent the dependency nodes that are aligned by c and c^p, respectively. Parse tree path is the concatenation of all dependency tree and direction labels through the tree path between d_c and d_{c^p}. For example, the relation probability of $c =$ *61*, $d_c =$ 61, and $d_{c^p} =$ *old* in Figure 3b is $P(quant|advmod \downarrow num \downarrow)$. A parse tree path is a useful feature for extracting relations between any two tree nodes, e.g., Semantic Role Labeling (SRL) (Gildea and Jurafsky, 2002) and relation extraction (Bunescu and Mooney, 2005; Kambhatla, 2004; Xu et al., 2015), so we add relation probability to our model.

$P_{NE}(c, d_c) = P(c|\textit{NamedEntity}(d_c))$
Named Entity Probability is the probability of the concept c conditioned on different named entity types (e.g., PERSON, DATE, ORGANIZA-



(a) Lemma Feature Probability
$P_{lemma}(c = \textit{temporal-quantity}, d_c = \underline{old})$

(b) Relation Feature Probability
$P_{rel}(c = \textit{61}, d_c = \underline{61}, d_{c^p} = \underline{old})$

(c) Named Entity Probability
$P_{NE}(c = \textit{person}, d_c = \underline{Vinken})$

(d) Semantic Roles Probability
$P_{SR}(c = \textit{person}, d_c = \underline{Vinken}, d_{c^p} = \underline{join})$

Figure 3: The sample of feature probabilities that are used in our aligner. Dashed lines link AMR concepts (top) and corresponding dependency parse nodes (bottom), while dense dashed lines link the parent AMR concepts and its corresponding dependency parse nodes.

TION, etc.). $NamedEntity(d)$ indicates the named entity type of the phrase with d as the head word. For example, after named entity recognition (NER) tagging, the label assigned to "PERSON" is the dependency parse tree node "Vinken". So the named entity probability of $P_{NE}(c = \textit{person}, d_c = \underline{Vinken})$ in Figure 3c is $P(\textit{person} \mid \underline{\text{PERSON}})$. Since AMR contains a large amount of named entity information, we assume that a feature based on an external named entity module should improve the alignment accuracy.

$P_{SR}(c, d_c, d_{c^p})$
$= P(\textit{AMRLabel}(c) | \textit{SemanticRole}(d_{c^p}, d_c))$

Semantic Role Probability is the conditional probability of the AMR relation label of c, given the semantic role d_c if d_{c^p} is a predicate and d_c is d_{c^p}'s argument. If a predicate-argument structure does not exist between d_{c^p} and d_c, the semantic role probability is omitted. For example, in Figure 3d, the semantic role probability of $P_{SR}(c = \textit{person}, d_c = \underline{Vinken}, d_{c^p} = \underline{join})$ is equal to $P(\textit{ARG0} | \underline{Arg0})$. Since AMR depends heavily on predicate-argument relations, external predicate-argument information from an external SRL system should enhance the overall alignment accuracy.

The above four feature probabilities are learned by the EM algorithm (Section 3.2).

3.1.3 Global Feature

The above basic and external features capture local alignment information. However, to make sure that a concept is aligned to the correct phrase head word which represents the same sub-meaning, we need a global feature to calculate coverage. The design of our concept coverage feature is as follows:

$R_{CC}(c)$ **Overlapping Ratio** of the child concept aligned phrases to their parent concept aligned phrases plus the non-covered penalty. This ratio is defined as:

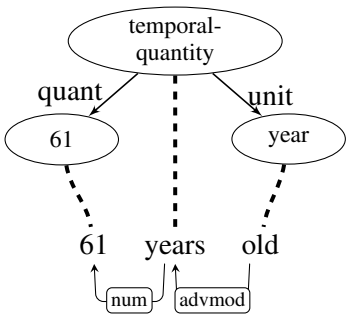


Figure 4: A sample of incorrect alignment. We use this sample to calculate its overlapping ratio (R_{cc})
let c = *temporal-quantity*
$W(c) = \{\underline{61}, \underline{year}\}$
$W_{child}(c) = \{\underline{61}\} \cup \{\underline{61}, \underline{year}, \underline{old}\}$
$= \{\underline{61}, \underline{year}, \underline{old}\}$
$W_{child}(c) \cap W(c) = \{\underline{61}, \underline{year}\}$
$pen(c) = exp(-|\{\underline{old}\}|) = 0.37$
$R_{cc}(c) = (\frac{2}{2}) \times pen(c) = 0.37$

$$R_{CC}(c) = \frac{|W_{child}(c) \cap W(c)|}{|W(c)|} \times pen(c)$$
$$W(c) = d_c$$

$$W_{child}(c) = \bigcup_{c^{s_i} \in child(c)} d_{c^{s_i}}$$
$$pen(c) = exp(-|W_{child}(c) \setminus (W_{child}(c) \cap W(c))|)$$

where W refers to the set of words that the aligned dependency word node contains. The first term of R_{CC} ensures the child concepts contain the largest possible subspans of the parent concept span. The non-covered penalty term (pen) is to prevent a child concept from aligning to a word node that contains a larger word span than the child's parent concept. The pen term will increase exponentially if child concepts align to a larger word span. The back slash term "\" refers to set subtraction. We take Figure 4 as an example of an incorrect alignment example where the concept "*temporal-quantity*" aligns to "year" and the concept "*year*" aligns to "old", the overlapping ratio of this alignment is 0.37 since it suffers a penalty. As we compare it with the correct alignment in Figure 3b, the overlapping ratio of this alignment is 0.67, which is much higher than the incorrect one.

3.2 Training with EM Algorithm

The objective function of our AMR-to-Dependency Parse aligner is listed as follows: Since our long term goal is to design a dependency parse to AMR parser, we define the objective function L_θ as the probability that dependency parses transfer to AMR graphs for the AMR-to-Dependency Parse aligner:

$$\theta = \text{argmax}\, L_\theta(\text{AMR}|\text{DEP}) \quad (1)$$

$$L_\theta(\text{AMR}|\text{DEP}) = \prod_{\substack{(C,D,A) \\ \in \mathbb{S}}} P(C|D) = \prod_{\substack{(C,D,A) \\ \in \mathbb{S}}} \sum_{a \in A} P(C, a|D) \quad (2)$$

$$P(C, a|D) = \prod_{j=1}^{|C|} P(c_j | d_{c_j} = a(c_j), d_{c_j^p} = a(c_j^p)) \quad (3)$$

where $\theta = (P_{lemma}, P_{rel}, P_{NE}, P_{SR})$ is the set of feature probabilities (parameters) we want to estimate, alignment set A is the latent variable we want to observe, and $\mathbb{S}$ is the training sample that contains a set of tuples (C, D, A), where C and D are a $\langle$*AMR*, *dependency parse*$\rangle$ pair and A is their alignment combination set. In equation (3), the probability that dependency tree D translates to AMR C with an alignment combination a is equal to the product of all probabilities that concept c_j in C aligns to dependency node d_{c_j} and c_j^p aligns to dependency node $d_{c_j^p}$.

3.2.1 Expectation-Step

The E-Step estimates all the different alignment probabilities of an input AMR and dependency parse pair by giving the product of feature probabilities. The alignment probability can be calculated using:

$$P(a|C, D) = \prod_{j=1}^{|C|} \frac{P(c_j | d_{c_j}, d_{c_j^p})}{\sum_{l=1}^{|D|} \sum_{i=1}^{|D|} P(c_j | d_i, d_l)} \quad (4)$$

$$P(c_j|d_i, d_l) = P_{rule}(c_j, d_i) \times P_{lemma}(c_j, d_i) \times P_{rel}(c_j, d_i, d_l) \times P_{NE}(c_j, d_i) \times P_{SR}(c_j, d_i, d_l) \quad (5)$$

The alignment probability is equal to the product of all tuple (c, d_c, d_{c^p})'s aligning probabilities.

P_{rule} is obtained by a simple calculation from the development set, while P_{lemma}, P_{rel}, P_{NE}, and P_{SR} are initialized uniformly before the first round of E-step. And these feature probabilities will be updated during the M-step.

3.2.2 Maximization-Step

In the M-Step, feature probabilities are re-estimated by collecting the count of all AMR-dependency parse pairs. The count of lemma (cnt_{lemma}), relation (cnt_{rel}), named entity (cnt_{NE}), and semantic role (cnt_{SR}) features are the normalized counts that are collected from the accumulating probability of all possible alignments from the E-step. Here we take the derivation of cnt_{lemma} as an example. cnt_{rel}, cnt_{NE}, and cnt_{SR} can be obtained with similar equations:

$$cnt_{lemma}(c|Word(d_c); C, D) = \sum_{a \in A} \frac{P(c|d_c, d_{c_p})}{\sum_{i=0}^{|D|} \sum_{l=0}^{|D|} P(c|d_i, d_l)}$$

After we collect all counts for different features, the four feature probabilities, $P_{lemma}, P_{rel}, P_{NE}$, and P_{SR}, are updated with their feature counts. Here we show the update of P_{lemma} as an example. The rest of feature probability updates can be derived in the same way:

$$P_{lemma}(c, d) \leftarrow \sum_{\substack{C \in AMR, \\ D \in DEP}} \frac{cnt_{lemma}(c|Word(d); C, D)}{\sum_{j=1}^{|C|} cnt_{lemma}(c_j|Word(d); C, D)}$$

After this, we apply the newer feature probabilities to recalculate alignment probabilities in the E-step again. EM iterates the E and M-steps until convergence or certain criteria are met.

4 Decoding

At decoding time, we want to find the most likely alignment a for the given $\langle C, D \rangle$. By applying Equations (4) and (5), we define the search for alignments as follows:

$$\underset{a}{\operatorname{argmax}} P(a|C, D) = \underset{a}{\operatorname{argmax}} \prod_{j=1}^{|C|} R_{CC}(c_j) * P(c_j|d_i = a(c_j), d_l = a(c_j^p))$$

This decoding problem finds the alignment a that maximizes the likelihood, which we define in Equation (5). The overlapping ratio(R_{CC}) is introduced to the likelihood function to ensure that a parent concept covers a wider word span range than its child concepts. A beam search algorithm is designed to extract the target alignment without exhaustively searching all of the candidate alignments (which has a complexity of $O(|D|^{|C|})$.) The beam search starts from leaf concepts and then walks through parent concepts after their child concepts have been traversed. When we go through concept c_j, we need to consider all the following likelihoods: 1) the accumulated likelihood for aligning to any dependency word node d_{c_j} from all the child concepts of c_j, and 2) the product of P_{lemma}, P_{NE}, P_{rel}, P_{SR}, and R_{CC} for c_j. Instead of using during training, R_{CC} is only applied during decoding time. The probabilities are obtained simply from the product of all the above likelihoods. We keep the top-$|b|$ alignment probabilities and their aligned dependency node d_{c_j} for each c_j until we reach the root concept, where $|b|$ is the beam size. Finally we can trace back and find the most likely alignment.

The running time for the beam search algorithm is $O(|b| * |C| * |D|^2)$.

		Sent.	Token	# of NE	# of Arg.
Gold	train	8,276	176,422	3,750	58,520
	dev.	409	8,695	415	2,574
	test	415	8,786	401	3,107
All	train	39,260	649,219	43,715	260,979
	dev.	409	8,695	580	2,574
	test	415	8,786	401	3,107

Table 2: The data split of the LDC DEFT AMR corpus. *Gold* refers to the sentences also appearing in OntoNotes 5.0 with gold annotations, while *All* refers to all sentences in DEFT AMR corpus with dependency parses, named entities, and semantic roles generated by ClearNLP. Number of tokens, named entities, and arguments(Arg.) in each data set are also presented

5 Experiments and Results

5.1 Experimental Data

The LDC DEFT Phase 2 AMR Annotation Release 2.0[1] consists of AMRs with English sentence

[1]LDC DEFT Phase 2 AMR Annotation Release 2.0, Release date: March 10th, 2016. https://catalog.ldc.upenn.edu/LDC2016E25

pairs. Annotated selections from various genres (including newswire, discussion forum, other web logs, and television transcripts) are available, for a total of 39,260 sentences. This release uses the PropBank Unification frame files (Bonial et al., 2014; Bonial et al., 2016). To generate automatic dependency parses for all DEFT AMR Release data, we use ClearNLP (Choi and Mccallum, 2013) to produce dependency parses. ClearNLP also labels semantic roles and named entity tags automatically on the generated dependency parses. This data set is named "All". To compare the effect of applying automatic dependency parses to our aligner with gold dependency parses, we select the sentences which appear in the OntoNotes 5.0[2] release as well. The OntoNotes data contains TreeBanking, PropBanking, and Named Entity annotations. OntoNotes 5.0 also uses PropBank Unification frame files for PropBanking. This data set, containing a total of 8,276 of selected AMRs and their dependency parses from OntoNotes, is named "Gold". To generate the development and test set, we manually align the AMR concepts and dependency word nodes. Since the manual alignment is time-consuming, "Gold" and "All" data share the same development/test set. Table 2 presents the statistics for the experimental data.[3]

5.2 Experiment Results

We run EM for 50 iterations and ensure the EM model converges. Afterwards, we use our decoding algorithm to find the alignments that maximize the likelihood. The test set data is used to evaluate performance.

We first evaluate the performance of our system with the external features added incrementally. Table 3 indicates the results. By running with the "Gold"‘ data, the only feature that improves significantly over the baseline (rule-based and lexicon features only) is the semantic role feature. The named entity feature actually hurts performance. On the other hand, all the features contribute to the F-Score incrementally for "All". Again, the semantic role feature still has the most positive impact against other features, and a significant improvement over the baseline.

As we compare the F_1 score on training with "All" and "Gold" data set, training with "All" outperforms training with "Gold" data in all different feature combinations. We believe there are two reasons for this. First, the "All" data contains richer information than the "Gold" data. "All" has double the sentence size of "Gold", and proportionally more named entity labels. Second, the automatic dependency parses do not hurt the performance of our aligner very much. We believe that our unsupervised alignment model works better with more data, even without access to gold standard dependency parses.

Data	Feature	P	R	F-Score
Gold	L	84.0	85.0	84.5
	L + S	85.2	86.3	**85.7**
	L + S + R	82.8	83.8	83.3
	L + S + R + N	80.9	81.9	81.4
All	L	84.9	85.4	85.1
	L + S	85.7	87.4	86.5
	L + S + R	85.8	87.7	86.7
	L + S + R + N	86.3	88.0	**87.1**

Table 3: Incremental Feature Contributions for different features: L: lemma; R: relation; N: NE; S: semantic role.

We then compare our aligner with three other aligners: JAMR, another version of unsupervised alignment (Chen, 2015), and ISI. To make them fit our test data, we design a heuristic method to force every unaligned concept (e.g., named entity and "‘:polarity -"’ concepts) to align to a dependency word node according to rule-based and global features (see Section 3.1). The alignment is counted as a correct match when the concept aligns to either the head word or the partial word span of a phrase. The alignments from concept relation to word span (apply in ISI) are discarded in our task. The results of the experiment are shown in Table 4. Our aligner achieves the best F_1 score in both the "All" and "Gold" data sets, as it should, since it is designed to align AMRs to dependency parses, as was the Chen aligner. Our aligner performs better than the Chen aligner by around 28% in F_1 score. We can conclude that the addition of rule-based feature, global features, and beam-search in decoding time helps the alignment task substantially.

5.3 Apply to AMR Parsing

To evaluate how alignment can enhance AMR parsing, we compare the parsing performance of

[2]LDC OntoNotes Release 5.0, Release date: October 16th, 2013 https://catalog.ldc.upenn.edu/LDC2013T19

[3]The manually aligned data and our aligner will be available after this paper gets accepted

Data	Aligner	P	R	F-Score
Gold	Chen 2015	61.1	53.4	57.0
	JAMR	78.5	62.8	69.8
	ISI	78.6	71.4	74.9
	Ours	**85.2**	**86.3**	**85.7**
All	Chen 2015	62.4	55.5	58.7
	JAMR	80.2	65.9	72.4
	ISI	80.4	74.9	77.6
	Ours	**86.3**	**88.0**	**87.1**

Table 4: Results of different alignment models

Data	Aligner	P	R	F-Score
Gold	JAMR	62.2	61.0	61.1
	ISI	65.3	63.9	64.5
	Ours	**68.6**	**64.2**	**66.4**
All	JAMR	64.2	63.0	63.1
	ISI	66.1	65.1	65.6
	Ours	**68.1**	**64.7**	**66.7**

Table 5: Comparison of using different alignments with CAMR Parser.

the CAMR parser with different alignments produced by JAMR, ISI, and our aligner. To make the alignments fit the CAMR parser, we convert both ISI and our alignments to the original JAMR alignment format, word span to AMR concept. We get rid of the ":wiki" tag, which links the named entity to its Wikipedia page, to simplify the parsing task since we think the Wikify task (Mihalcea and Csomai, 2007) is basically different from the AMR parsing task. Smatch v2.0.2 is used to evaluate AMR parsing performance (Cai and Knight, 2012). The evaluation script is obtained from the SemEval 2016 Task 8 website[4].

A comparison of parsing results is given in Table 5. We first train the parser with "Gold" Standard dependency parses and alignments from the different aligners. Results show that our aligner improves by a 2% F_1 score over the two other aligners. Then we train the AMR Parser system with the "All" data set. The dependency parses attached with semantic roles and named entities generated by ClearNLP are also provided to CAMR as training data. CAMR use dependency parsing results from Stanford dependency parser (Klein and Manning, 2003) by default. Our aligner still achieves slightly better performance than the other two. Modifying the AMR parser to take advantage of parse node-concept alignments could potentially result in greater improvement, since CAMR takes the input alignments as word span to AMR concept.

5.4 Error Analysis

To further understand the advantages and the disadvantages of our model, we go through all incorrect alignments and manually categorize 40% of them into different error types, with their proportion:

Automatic Parsing Errors - 3.8%: ClearNLP has a 92.96% unlabeled attachment score on the Penn English Treebank evaluation set (Marcus et al., 1993), Section 23, for dependency parsing. Therefore, when training our aligner on the "All" data set with dependency parses, named entities, and semantic roles generated by ClearNLP, incorrect parses occasionally show up. Since NE and semantic roles are attached to dependency parses, incorrect dependency parses cause additional NE and semantic roles alignment errors, on top of the dependency parse alignment errors.

Long Distance Dependencies - 14.2%: Long sentences with long distance dependencies always bring difficulty to NLP parsing tasks. Experimental results show that our model runs into troubles when nearby concepts align to dependency nodes which are far from each other. Co-reference is an example that is highly likely to align to long distance dependencies, and our model can not deal with it well.

Duplicate Words - 17.4%: When two identical concepts align to different word nodes, our model is confused by duplicate words. In Figure 5, there are two "first"s in the sentence. One refers to "first 6 rounds", and the other refers to "first position". However, our model faultily aligns both *ordinal-entity* concepts to the same "first" word node. Our model did not distinguish these two ordinal-entities since the lexicon and named entity tags of the two "first"s are identical.

Meaning Coverage Errors - 40.4%: We define a good alignment as a concept that aligns to the correct phrase head word which represents the same sub-meaning. So instead of aligning to a concept's word lexicon, sometimes a concept aligns to its parent node (head word). However, the lexicon features dominate the alignment probability in our

[4]http://alt.qcri.org/semeval2016/task8/index.php?id=data-and-tools

```
(a / and
  :op1 (o / occupy-01
    :ARG0 (p3 / person
      :name (n / name
        :op1 "Mingxia" :op2 "Fu"))
    :ARG1 (p4 / position
      :ord (o3 / ordinal-entity :value 1))
  :op2 (o2 / occupy-01
    :ARG0 (p / person
      :name (n2 / name
        :op1 "Bin" :op2 "Chi"))
    :ARG1 (p2 / position
      :ord (o4 / ordinal-entity :value 3))
  :mod (r / respective)
  :time (r3 / round-05 :quant 6
    :ARG1 (c / compete-01)
      :ord (o5 / ordinal-entity :value 1)))
```

Figure 5: The AMR annotation of sentence "In the first 6 rounds of competition, Mingxia Fu and Bin Chi are occupying the first and third positions respectively"

E-M calculation. That causes our model to tend to align a concept with its word form instead of its head word. For example, English light verb constructions (LVCs), e.g., *take a bath*, are thought to consist of a semantically general verb and a noun that denotes an event or state. AMR representation always drops light verb and uses eventive noun as concept. Our model sometimes aligns this eventive noun concept to its nominal word node, which is incorrect since the light verb on dependency parse covers the same sub-meaning and should be aligned.

6 Conclusion and Future Work

In this paper, we present an AMR-Dependency Parse aligner, which estimates the feature probabilities by running the EM algorithm. It can be used directly by AMR parser. Results show that our aligner performs better than other aligners, and improves AMR parser performance. The latent probabilities that we obtain during training, i.e., all the external feature sets, could also potentially benefit a parser. We plan to develop our own AMR parser, which will apply these external feature sets as the basic model. We also plan to continue to perfect our aligner via tuning the feature weights and learning techniques, and adding new features, like word embeddings and WordNet features.

Acknowledgments

We gratefully acknowledge the support of the National Science Foundation Grants 0910992 IIS:RI: Richer Representations for Machine Translation, and NSF IIA-0530118 PIRE (a subcontract from Johns Hopkins) for the 2014 Frederick Jelinek Memorial Workshop for Meaning Representations in Language and Speech Processing, and funding under 2016 Summer Graduate School Fellowship of University of Colorado at Boulder. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

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.

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. *CoRR*, abs/1604.01278.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Claire Bonial, Kathryn Conger, Jena D. Hwang, Aous Mansouri, Yahya Aseri, Julia Bonn, Timothy OGorman, and Martha Palmer. 2016. Current directions in english and arabic propbank. In Nancy Ide and James Pustejovsky, editors, *The Handbook of Linguistic Annotation*. Springer, Berlin.

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

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2012. Smatch: an evaluation metric for semantic feature structures. submitted.

Wei-Te Chen. 2015. Learning to map dependency parses to abstract meaning representations. In *Proceedings of the ACL-IJCNLP 2015 Student Research Workshop*, pages 41–46, Beijing, China, July. Association for Computational Linguistics.

Jinho D. Choi and Andrew Mccallum. 2013. Transition-based dependency parsing with selectional branching. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

J. Flanigan, S. Thomson, J. Carbonell, C. Dyer, and N. A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*, Baltimore, Maryland, June. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June.

C. Matthiessen and J.A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Communication in artificial intelligence. Pinter.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California, June. Association for Computational Linguistics.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

Martha Palmer, Dan Guildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, March.

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. Association for Computational Linguistics.

Chuan Wang, Xue Nianwen, and Pradhan Sameer. 2015. 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*. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, Sameer Pradhan Pradhan, Xiaoman Pan, and Heng Ji. 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)*. Association for Computational Linguistics, June.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *CoRR*, abs/1506.03139.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing*.