Parser for Abstract Meaning Representation using Learning to Search

Sudha Rao[1,3]*, **Yogarshi Vyas**[1,3]*, **Hal Daumé III**[1,3], **Philip Resnik**[2,3]
[1]Computer Science, [2]Linguistics, [3]UMIACS
University of Maryland
raosudha@cs.umd.edu, yogarshi@cs.umd.edu, hal@cs.umd.edu, resnik@umd.edu

Abstract

We develop a novel technique to parse English sentences into Abstract Meaning Representation (AMR) using SEARN, a Learning to Search approach, by modeling the concept and the relation learning in a unified framework. We evaluate our parser on multiple datasets from varied domains and show an absolute improvement of 2% to 6% over the state-of-the-art. Additionally we show that using the most frequent concept gives us a baseline that is stronger than the state-of-the-art for concept prediction. We plan to release our parser for public use.

1 Introduction

Abstract Meaning Representation (Banarescu et al., 2013) is a semantic representation which is a rooted, directed, acyclic graph where the nodes represent concepts (words, PropBank (Palmer et al., 2005) framesets or special keywords) and the edges represent relations between these concepts. Figure 1 shows the complete AMR for a sample sentence.

The key motivation behind developing AMR was to have a comprehensive and broad-coverage semantic formalism that puts together the best insights from a variety of semantic annotations (like named entities, co-reference, semantic relations, discourse connectives, temporal entities, etc.) in a way that would enable it to have the same kind of impact that syntactic treebanks have on natural language processing tasks. Currently, there are approximately 20,000 sentences which have been annotated with their AMRs, but for such a representation to be useful for almost any NLP task, a larger set of annotations would be needed. Algorithms that can perform automatic semantic parsing of sentences into AMR can help alleviate the problem of paucity of manual annotations.

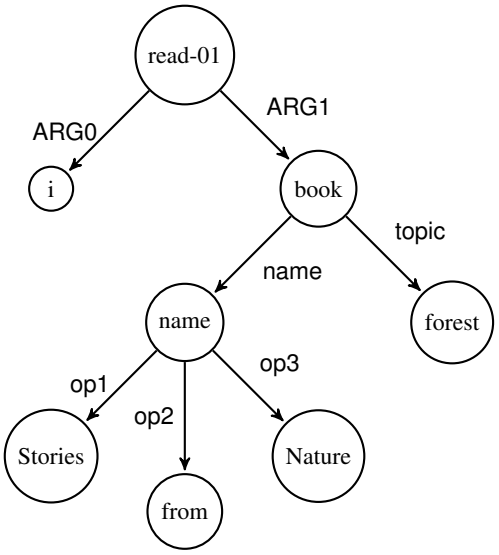
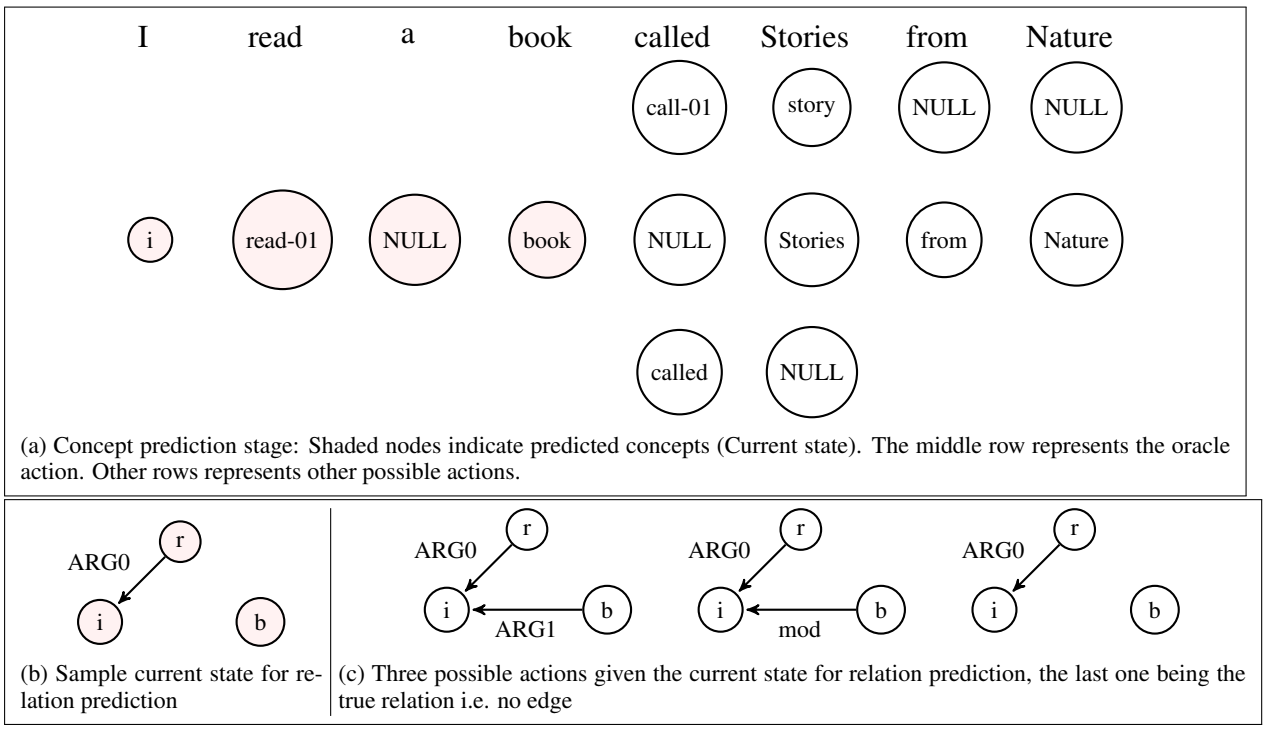


Figure 1: AMR graph for the sentence *"I read a book, called Stories from Nature, about the forest."*

Automatic semantic parsing for AMR is still in a nascent stage. There have been two published approaches for automatically parsing English sentences into AMR. Flanigan et al. (2014) use a semi-Markov model to first identify the concepts, and then find a maximum spanning connected subgraph that defines the relations between these concepts. The other approach (Wang et al., 2015) uses a transition-based algorithm to convert the dependency representation of a sentence to its AMR.

In this work, we develop a novel technique for AMR parsing that uses SEARN (Daumé III et al., 2009), a Learning to Search (L2S) algorithm. SEARN and other L2S algorithms have proven to be highly effective for tasks like part-of-speech tagging, named entity recognition (Daumé III et al., 2014), and for even more complex structured prediction tasks like coreference resolution (Ma et al., 2014) and dependency parsing (He et al., 2013). Using SEARN allows us to model the learn-

*The first two authors contributed equally to this work.



(a) Concept prediction stage: Shaded nodes indicate predicted concepts (Current state). The middle row represents the oracle action. Other rows represents other possible actions.

(b) Sample current state for relation prediction

(c) Three possible actions given the current state for relation prediction, the last one being the true relation i.e. no edge

Figure 2: Using SEARN for AMR parsing

ing of concepts and relations in a unified framework which aims to minimize the loss over the entire predicted structure, as opposed to minimizing the loss over concepts and relations in two separate stages, as is done by Flanigan et al (2014).

There are three main contributions of this work. Firstly, we provide a novel algorithm based on SEARN to parse sentences into AMRs. Additionally, our parser extracts possible 'candidates' for the right concepts and relations from the entire training data, but only uses smaller sentences to train the learning algorithm. This is important since AMR annotations are easier to obtain for smaller sentences. Secondly, we evaluate our parser on datasets from various domains, unlike previous works, which have been restricted to newswire text. We observe that our parser performs better than the existing state-of-the-art parser, with an absolute improvement of 2 to 6 % over the different datasets. Finally, we show that using the most frequently aligned concept for each word in the sentence (as seen in the training data) as the predicted concept, proves to be a strong baseline for concept prediction. This baseline does better than existing approaches, and we show that our parser performs as well as the baseline at this part of the task in some datasets, and even better in some others.

The rest of this paper is organized as follows. In the next section, we briefly review SEARN and explain its various components with respect to our AMR parsing task. Section 3 describes our main algorithm along with the strategies we use to deal with the large search space of the search problem. We then describe our experiments and results (Section 4).

2 Using SEARN

The task of generating an AMR given a sentence is a structured prediction task where the structure that we are trying to predict is a singly rooted, connected directed graph with concepts (nodes) and relations (edges). In this work, we design an AMR parser that learns to predict this structure using SEARN. SEARN solves complex structured prediction problems by decomposing it into classification problems. It does so by decomposing the structured output, y, into a sequence of decisions $y_1, y_2, ..., y_m$ and then using a classifier to make predictions for each component in turn, allowing for dependent predictions. We decompose the AMR prediction problem into the three problems of predicting the concepts of the AMR, predicting the root and then predicting the relations between the predicted concepts (explained in more detail under section 3). Below, we explain how we

use SEARN, with reference to a running example in Figure 2.

SEARN works on the notion of a policy which can be defined as "what is the best next action (y_i) to take" in a search space given the current state ($s = (x, y_1, y_2, .., y_{i-1})$), where x is the input. For our problem, a state during the concept prediction phase is defined as the concepts predicted for a part of the input sentence. Similarly, a state during the relation prediction phase is defined as the set of relations predicted for certain pairs of concepts obtained during the concept prediction stage. (In Figure 2a (concept prediction), the current state corresponds to the concepts {'i', 'read-01', 'book'} predicted for a part of the sentence. In Figure 2c (relation prediction), the current state corresponds to the relation 'ARG0' predicted between 'r' and 'i')

At training time, SEARN operates in an iterative fashion. It starts with some initial policy and given an input x, makes a prediction $y = y_1, y_2, ..., y_m$ using the current policy. For each prediction y_i it generates a set of cost-sensitive multi-class classification examples each of which correspond to a possible action (a) the algorithm can take given the current state. Each example can be defined using local features and features that depend on previous predictions. The possible set of next actions in our concept prediction phase corresponds to the set of possible concepts the next word can take. The possible set of next actions in our relation prediction phase corresponds to the set of possible relations the next pair of concepts can take. (In Figure 2a (concept prediction), the next action is assigning one of {'call-01', 'called', NULL} to the word 'called'. In Figure 2c (relation prediction), the next action is assigning one of {'ARG1', 'mod', NO-EDGE} to the pair of concept 'b' and 'i').

During training, SEARN has access to an "oracle" policy which gives the true best action (a^*) given the current state . Our oracle returns the correct concept and relation labels in the concept prediction and relation prediction phase respectively. (In Figure 2a (concept prediction), the oracle will return NULL and in Figure 2c (relation prediction), the oracle will return NO-EDGE). SEARN then calculates the loss between a and a^* using a prespecified loss function. It then computes a new policy based on this loss and interpolates it with the current policy to get an updated policy, before moving on to the next iteration.

At test time, predictions are made greedily using the policy learned during training time.

3 Methodology

3.1 Learning technique

Algorithm 1

for $each\ span\ s_i$ **do**
  $c_i = predict\_concept(s_i)$
end for
$c_{root} = predict\_root([c_1, ..., c_n])$
for $each\ concept\ c_i$ **do**
  for $each\ j < i$ **do**
    $r_{(i,j)} = predict\_relation(c_i, c_j)$
    $r_{(j,i)} = predict\_relation(c_j, c_i)$
  end for
end for

We use SEARN as described in section 2 to learn a model that can successfully predict the AMR y for a sentence x. The sentence x is composed of a sequence of spans $(s_1, s_2, ..., s_n)$ each of which can be a single word or a span of words (We describe how we go from a raw sentence to a sequence of spans in Section 4.2). Given that our input has n spans, we first decompose the structure into a sequence of $n^2 + 1$ predictions $\mathbf{D} = (\mathbf{C}, \mathbf{ROOT}, \mathbf{R})$, where

$\mathbf{C} = c_1, c_2, ..., c_n$ - where c_i is the concept predicted for span s_i

ROOT is the decision of choosing one of the predicted concepts as the root (c_{root}) of the AMR

$\mathbf{R} = r_{2,*}, r_{*,2}, r_{3,*}, r_{*,3}, ..., r_{n,*}, r_{*,n}$ - where $r_{i,*}$ are the predictions for the directed relations from c_i to $c_j\ \forall j < i$, and $r_{*,i}$ are the predictions for the directed relations from c_j to $c_i\ \forall j < i$. We constrain our algorithm to not predict any incoming relations to c_{root}.

During training time, the possible set of actions for each prediction is given by the k-best list, which we will describe in Section 3.2. We use Hamming Loss as our loss function. Under Hamming Loss, the oracle policy is simply choosing the right action for each prediction. Since this loss is defined on the entire predicted output, the model learns to minimize the loss for concepts and relations jointly.

Algorithm 1 describes the sequence of predictions to be made in our problem. We learn three different policies corresponding to each of

Feature label	Description
$w_{i-2}, w_{i-i}, w_i, w_{i+1}, w_{i+2}$	Words in s_i and context
$p_{i-2}, p_{i-i}, p_i, p_{i+1}, p_{i+2}$	POS tags of words in s_i and context
NE_i	Named entity tags for words in s_i
s_i	Binary feature indicating whether w_i is(are) stopword(s)
dep_i	All dependency edges originating from words in w_i
b_c	Binary feature indicating whether c is the most frequently aligned concept with s_i or not
c_{i-2}, c_{i-1}	Predicted concepts for two previous spans
c	Concept label and its conjunction with all previous features
$frame_i$ and $sense_i$	If the label is a PropBank frame (e.g. 'see-01', use the frame ('see') and the sense('01') as additional features.

Table 1: Concept prediction features for span s_i and concept label c_i

Feature label	Description
$c_i, c_j, c_i \wedge c_j$	The two concepts and their conjunction
$w_i, w_j, w_i \wedge w_j$	Words in the corresponding spans and their conjunction
$p_i, p_j, p_i \wedge p_j$	POS tags of words in spans and their conjunction
dep_{ij}	All dependency edges with tail in w_i and head in w_j
dir	Binary feature which is true iff $i < j$
r	Relation label and its conjunction with all other features

Table 2: Relation prediction features for concepts c_i and c_j and relation label r

Feature label	Description
c_i	Concept label. If the label is a PropBank frame (e.g. 'see-01', use the frame ('see') and the sense('01') as additional features.
w_i	Words in s_i, i.e. the span corresponding to c_i
p_i	POS tags of words in s_i
$is\_dep\_root_i$	Binary feature indicating whether one of the words in s_i is the root in the dependency tree of the sentence

Table 3: Root prediction features for concept c_i

the functions $predict\_concept$, $predict\_root$ and $predict\_relation$. The learner in each stage uses features that depend on predictions made in the previous stages. Tables 1, 2 and 3 describe the set of features we use for the concept prediction, relation prediction and root prediction stages respectively.

3.2 Selecting k-best lists

For predicting the concepts and relations using SEARN, we need a candidate-list (possible set of actions) to make predictions from.

Concept candidates: For a span s_i, the candidate-list of concepts, CL-CON$_{s_i}$ is the set of all concepts that were aligned to s_i in the training data. If s_i has not been seen in the training data, CL-CON$_{s_i}$ consists of the lemmatized span, PropBank frames (for verbs) obtained using the Unified Verb Index (Schuler, 2005) and the NULL concept.

Relation candidates: The candidate list of relations for a relation from concept c_i to concept c_j, CL-REL$_{ij}$, is the union of the following three sets:

- $pairwise_{i,j}$ - All directed relations from c_i to c_j when c_i and c_j occurred in the same AMR,
- $outgoing_i$ - All outgoing relations from c_i, and
- $incoming_j$ - All incoming relations into c_j.

In the case when both c_i and c_j have not been seen in the training data, CL-REL$_{ij}$ consists of all

relations seen in the training data. In both cases, we also provide an option NO-EDGE which indicates that there is no relation between c_i and c_j.

3.3 Pruning the search space

To prune the search space of our learning task, and to improve the quality of predictions, we use two observations about the nature of the edges of the AMR of a sentence, and its dependency tree, within our algorithm.

First, we observe that a large fraction of the edges in the AMR for a sentence are between concepts whose underlying spans (more specifically, the words in these underlying spans) are within two edges of each other in the dependency tree of the sentence. Thus, we refrain from calling the $predict\_relation$ function in Algorithm 1 between concepts c_i and c_j if each word in w_i is three or more edges away from all words in w_j in the dependency tree of the sentence under consideration, and vice versa. This implies that there will be no relation r_{ij} in the predicted AMR of that sentence. This doesn't affect the number of calls to $predict\_relation$ in the worst case ($n^2 - n$, for a sentence with n spans), but practically, the number of calls are far fewer. Also, to make sure that this method does not filter out too many AMR edges, we calculated the percentage of AMR edges that are more than two edges away in dependency tree. We found this number to be only about 5% across all our datasets.

Secondly, and conversely, we observe that for a large fraction of words which have a dependency edge between them, there is an edge in the AMR between the concepts corresponding to those two words. Thus, when we observe two concepts c_i and c_j which satisfy this property, we force our $predict\_relation$ function to assign a relation r_{ij} that is not NULL.

3.4 Training on smaller sentences

For a sentence containing n spans, Algorithm 1 has to make n^2 predictions in the worst case, and this can be inhibitive for large values of n. To deal with this, we use a parameter to indicate a cut-off on the length of a sentence (C), and only use sentences whose length (number of spans) is less than or equal to C. This parameter can be varied based on the size of the training data and the distribution of the length of the sentences in the training data. Setting a higher values of C will cause the model to use more sentences for training, but spend longer time, whereas lower values will train quickly on fewer sentences. In our experiments, a C-value between 10 and 15 gave us the best balance between training time, and number of examples considered.

4 Experiments and Results

4.1 Dataset and Method of Evaluation

We use the publicly available AMR Annotation Release 1.0 (LDC2014T12) corpus for our experiments. This corpus consists of datasets from varied domains such as online discussion forums, blogs, and newswire, with about 13,000 sentence-AMR pairs. Previous works have only used one of these datasets for evaluation (proxy), but we evaluate our parser on all of them. Additionally, we also use the freely available AMRs for *The Little Prince*, (lp) [1] which is from a more literary domain. All datasets have a pre-specified training and test split (Table 4).

As stated earlier (Sections 3.2 and 3.4), we use the entire training set to extract the candidate lists for concept prediction and relation prediction, but train our learning algorithm on only a subset of the sentence-AMR pairs in the training data, which is obtained by selecting sentences having less than a fixed number of spans (C, set to 10 for all our experiments). Table 4 also mentions the number of sentences in each training dataset that are of length $\leq C$ (column **Training** ($\leq \boldsymbol{C}$)).

Dataset	Training	Training ($\leq C$)	Test
bolt	1061	119	133
proxy	6603	1723	823
xinhua	741	115	86
dfa	1703	438	229
lp	1274	584	173

Table 4: Dataset statistics. All figures represent number of sentences.

We compare our results against those of the JAMR parser [2] of Flanigan et. al (2014) [3]. We run the parser with the configuration that is specified to give the best results.

The evaluation of predicted AMRs is done using

[1] http://amr.isi.edu/download.html

[2] https://github.com/jflanigan/jamr

[3] The transition-based parser by Wang et al. () is newer, but the latest release of JAMR performs better, hence we do not compare against the former.

Smatch (Cai and Knight, 2013) [4], which compares two AMRs using precision, recall and F_1. Additionally, we also evaluate how good we are at predicting the concepts of the AMRs, by calculating precision, recall and F_1 against the gold-concepts that are aligned to the induced spans during test time.

4.2 Preprocessing

JAMR Aligner: The training data for AMR parsing consists of sentences paired with corresponding AMRs. To convert a raw sentence into a sequence of spans (as required by our algorithm), we obtain alignments between words in the sentence and concepts in the AMR using the automatic aligner of JAMR. The alignments obtained can be of three types (Examples refer to Figure 1):

- *A single word aligned to a single concept*: E.g., word 'read' aligned to concept 'read-01'.
- *Span of words aligned to a graph fragment*: E.g., span 'Stories from Nature' aligned to the graph fragment rooted at 'name'. This usually happens for named entities and multi-word expressions such as those related to date and time.
- *A word aligned to* NULL *concept*: Most function words like 'about', 'a', 'the', etc are not aligned to any particular concept. These are considered to be aligned to the NULL concept.

Forced alignments: The JAMR aligner does not align all concepts in a given AMR to a span in the sentence. We use a heuristic to forcibly align these leftover concepts and improve the quality of alignments. For every unaligned concept, we count the number of times an unaligned word occurs in the same sentence with the unaligned concept across all training examples. We then align every leftover concept in every sentence with the unaligned word in the sentence with which it has maximally coocurred.

Span identification: During training time, the aligner takes in a sentence and its AMR graph and splits each sentence into spans that can be aligned to the concepts in the AMR. However, during test time, we do not have access to the AMR graph. Hence, given a test sentence, we need to split the sentence into spans, on which we can predict concepts. We consider each word as a single span except for two cases. First, we detect possible multi-word spans corresponding to named entities, using a named entity recognizer (Lafferty et al., 2001). Second, we use some basic regular expressions to identify time and date expressions in sentences.

[4] `http://amr.isi.edu/download/smatch-v2.0.tar.gz`

4.3 Experiments

To train our model, we use SEARN as implemented in the Vowpal Wabbit machine learning library (Langford et al., 2007; Daumé III et al., 2014).

For each dataset, we run three kinds of experiments. They differ in how they get the concepts during test time. All of them use the approach described in Section 3.1 for predicting the relations.

- **Oracle Concept** - Use the true concept aligned with each span.
- **1-Best Concept** - Use the concept with which the span was most aligned in the training data.
- **Fully automatic** - Use the concepts predicted using the approach described in Section 3.1.

4.4 Connectivity

Algorithm 1 does not place explicit constraints on the structure of the AMR. Hence, the predicted output can have disconnected components. Since we want the predicted AMR to be connected, we connect the disconnected components (if any) using the following heuristic. For each component, we find its roots (i.e. concepts with no incoming relations). We then connect the components together by simply adding an edge from our predicted root c_{root} to each of the component roots. To decide what edge to use between our predicted root c_{root} and the root of a component, we get the k-best list (as described in section 3.2) between them and choose the most frequent edge from this list.

4.5 Acyclicity

The post-processing step described in the previous section ensures that the predicted AMRs are rooted, connected, graphs. However, an AMR, by definition, is also acyclic. We do not model this constraint explicitly within our learning framework. Despite this, we observe that only a very small number of AMRs predicted using our fully automatic approach have cycles in them. Out of the total 1,444 AMRs predicted in all test sets, less

Dataset	Our Results									JAMR Results		
	Oracle Concepts			1-Best Concepts			Fully Automatic			Fully Automatic		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
bolt	0.64	0.53	0.58	0.52	0.43	**0.47**	0.51	0.42	**0.46**	0.55	0.33	**0.41**
proxy	0.69	0.65	0.67	0.61	0.59	**0.60**	0.62	0.60	**0.61**	0.68	0.53	**0.59**
xinhua	0.68	0.60	0.64	0.55	0.49	**0.52**	0.56	0.50	**0.52**	0.59	0.40	**0.48**
dfa	0.62	0.47	0.54	0.48	0.37	**0.42**	0.48	0.40	**0.44**	0.52	0.15	**0.23**
lp	0.70	0.58	0.63	0.57	0.45	**0.50**	0.54	0.49	**0.52**	0.53	0.41	**0.46**

Table 5: Full Results

Dataset	Our Results						JAMR Results		
	1-Best			Fully Automatic			Fully Automatic		
	P	R	F_1	P	R	F_1	P	R	F_1
bolt	0.74	0.72	**0.73**	0.74	0.72	**0.73**	0.73	0.55	**0.63**
proxy	0.79	0.77	**0.78**	0.78	0.78	**0.78**	0.78	0.68	**0.73**
xinhua	0.74	0.77	**0.74**	0.74	0.77	**0.75**	0.69	0.57	**0.63**
dfa	0.76	0.72	**0.74**	0.74	0.76	**0.75**	0.85	0.33	**0.48**
lp	0.77	0.79	**0.78**	0.77	0.80	**0.79**	0.53	0.41	**0.46**

Table 6: Concept Prediction Results

than 5% have cycles in them. Besides, almost all cycles that are predicted consist of only two nodes, i.e. both r_{ij} and r_{ji} have non-NO-EDGE values for concepts c_i and c_j. To get an acyclic graph, we can greedily select one of r_{ij} or r_{ji}, without any loss in parser performance.

4.6 Results

Table 5 shows the result of running our parser on all five datasets. By running our fully automatic approach, we get an absolute improvement of about **2% to 6%** on most datasets as compared to JAMR. Surprisingly, we observe a large improvement of **21%** on the online discussion forum dataset (dfa). In all cases, our results indicate a more balanced output in terms of precision and recall as compared to JAMR, with consistently higher recall.

It should be noted that selecting the 1-best concept also gives better results than JAMR. This indicates that the 1-best baseline is strong, and possibly, not very easy to beat. To reinforce this, we evaluate our concept predictions separately. The results are shown in Table 6. First, observe that going from the fully learned concept prediction to the 1-best concept shows only a small (or in some cases, no) drop in performance. Second, note that we show a consistent absolute improvement of **10% to 12%** over the concept prediction results of JAMR. As in the full prediction case, we observe a large performance increase (**27%**) on the online discussion forum dataset.

5 Related work

Semantic representations and techniques for parsing them have a rich and varied history. AMR itself is based on propositional logic and neo-Davidsonian semantics (Davidson, 1967). AMR is not intended to be an interlingua, but due to the various assumptions made while creating an AMR (dropping tense, function words, morphology, etc.), it does away with language-specific idiosyncrasies and interlingual representations (Dorr, 1992) are thus, important predecessors to AMR.

Like the task of AMR parsing, there have been various attempts to parse sentences into a logical form, given raw sentences annotated with such forms (Kate et al., 2005; Wong and Mooney, 2006). The work by Zettlemoyer and Collins (Zettlemoyer and Collins, 2005) attempts to map natural language sentences to a lambda-calculus encoding of their semantics. They do so by treating the problem as a structured learning task, and use a log-linear model to learn a Probabilistic *Combinatory Categorical Grammar* (CCG) (Steedman and Baldridge, 2011), which is a gram-

mar formalism based on lambda calculus.

AMR aims to combine various semantic annotations to produce a unified annotation, but it mainly builds on top of PropBank (Palmer et al., 2005). PropBank has found extensive use in other semantic tasks such as shallow semantic parsing (Giuglea and Moschitti, 2006),

In our work we used SEARN to build an AMR parser. SEARN comes from a family of algorithms called "Learning to Search (L2S)" that solves structured prediction problems by decomposing the structured output in terms of an explicit search space and then learning a policy that can take actions in this search space in the optimal way. Incremental structured perceptron (Collins and Roark, 2004; Huang et al., 2012), DAGGER (Ross et al., 2011), AGGREVATE (Ross and Bagnell, 2014), etc. (Daumé III and Marcu, 2005; Xu and Fern, 2007; Xu et al., 2007; Ratliff et al., 2007; Syed and Schapire, 2010; Doppa et al., 2012) are other algorithms that also belong to this family.

6 Conclusion and Future work

We have presented a novel technique for parsing english sentences into AMR using a learning to search approach. We model the concept and the relation learning in a unified framework using SEARN which allows us to optimize over the loss of the entire predicted output. We evaluate our parser on multiple datasets from varied domains and show that our parser performs better than the state-of-the-art across all the datasets. We also show that a simple technique of choosing the most frequent concept gives us a baseline that is better than the state-of-the-art for concept prediction.

Currently we ensure various properties of AMR, such as connectedness and acyclicity using heuristics. In the future, we plan to incorporate these as constraints in our learning technique.

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.

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, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 748–752.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Hal Daumé III, John Langford, and Stephane Ross. 2014. Efficient programmable learning to search. *arXiv preprint arXiv:1406.1837*.

Donald Davidson. 1967. The logical form of action sentences.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2012. Output space search for structured prediction. *arXiv preprint arXiv:1206.6460*.

Bonnie J Dorr. 1992. The use of lexical semantics in interlingual machine translation. *Machine Translation*, 7(3):135–193.

Jeffrey Flanigan, Sam Thomson, Jaime G. 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, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1426–1436.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1455–1464.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1062–1068.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

John Langford, Lihong Li, and Alexander Strehl. 2007. Vowpal wabbit online learning project.

Chao Ma, Janardhan Rao Doppa, J Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich, and Prasad Tadepalli. 2014. Prune-and-score: Learning for greedy coreference resolution. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Nathan Ratliff, David Bradley, J Andrew Bagnell, and Joel Chestnutt. 2007. Boosting structured prediction for imitation learning. *Robotics Institute*, page 54.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoff J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Workshop on Artificial Intelligence and Statistics (AIStats)*.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax Oxford: Blackwell*, pages 181–224.

Umar Syed and Robert E Schapire. 2010. A reduction from apprenticeship learning to classification. In *Advances in Neural Information Processing Systems*, pages 2253–2261.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Yuehua Xu and Alan Fern. 2007. On learning linear ranking functions for beam search. In *Proceedings of the 24th international conference on Machine learning*, pages 1047–1054. ACM.

Yuehua Xu, Alan Fern, and Sung Wook Yoon. 2007. Discriminative learning of beam-search heuristics for planning. In *IJCAI*, pages 2041–2046.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.