Meaning Representation Parsing

Avikalp Kumar Gupta & Anshul Goyal

November 13, 2015

Abstract

This project is aimed at solving the task 8 of SemEval 2016, in which, any English sentence had to be converted into its Abstract Meaning Representation (AMR) [ISI, 2015]. All the data and the code for the baseline algorithm was provided by the organizer of the task. In the baseline algorithm provided for the same [Flanigan et al., 2014], we noticed that the AMR graph generation hugely depended on the *focus* of the sentence. And the baseline failed to recognize the focus quite often. So we used an LSTM to recognize the focus separately and use this new focus for generating the rest of the graph.

1 Abstract Meaning Representation

"Abstract Meaning Representation (AMR) is a compact, readable, whole-sentence semantic annotation. Annotation components include entity identification and typing, PropBank semantic roles, individual entities playing multiple roles, entity grounding via wikification, as well as treatments of modality, negation, etc." [May, 2015]

1. Who is doing what to whom in a sentence
2. Different from a parse tree, it is abstract
3. AMR does not say anything about how it wants to be processed.
4. It is not an interlingua.

Figure 1: An ideal AMR graph for the sentence "The boy wants the girl to believe him."

1.1 Representation

AMR is a rooted, directed acyclic graph with labels on edges (relations) and leaves (concepts), as shown in figure 1.

The representation of the above graph in the output is:

```
1. (w / want-01
2.       :ARG0 (b / boy)
3.       :ARG1 (b2 / believe-01
4.             :ARG0 (g / girl)
5.             :ARG1 b))
```

1.2 More Logical than Syntax

A single AMR can be expressed in various ways in English. For example, figure 1 can be expressed in the following ways:

- The boy wants the girls to believe him
- The boy desires the girl to believe him.
- The boy desires to be believed by the girl.
- The boy has a desire to be believed by the girl.
- The boy's desire is for the girl to believe him.
- The boy is desirous of the girl believing him

2 Related Work

2.1 Baseline Algorithm

The method is based on a novel algorithm for finding a maximum spanning, connected subgraph, embedded within a Lagrangian relaxation of an optimization problem that imposes linguistically inspired constraints.

JAMR parsing represents an AMR parse as a graph $G = \langle V, E \rangle$; vertices and edges are given labels from sets L_V and L_E, respectively. G is constructed in two stages. The first stage identifies the **concepts** evoked by words and phrases in an input sentence $w = \langle w_1, \cdots, w_n \rangle$, each w_i a member of vocabulary W. The second stage connects the concepts by adding L_E-labeled edges capturing the **relations** between concepts, and selects a root in G corresponding to the **focus** of the sentence w.

2.1.1 Concept Identification

The concept identification stage maps spans of words in the input sentence w to concept graph fragments from F, or to the empty graph fragment ϕ. These graph fragments often consist of just one labeled concept node, but in some cases they are larger graphs with multiple nodes and edges.

Concept identification is illustrated below using the example, "The boy wants to visit New York City.":

```
1.    (c / city
2.          :name (n / name
3.                :op1 "New"
4.                :op1 "York"
5.                :op1 "City"))
```

Let the concept lexicon be a mapping $clex : W \rightarrow 2^F$ that provides candidate graph fragments for sequences of words. (The construction of F and $clex$ is discussed below.) Formally, a concept labeling is

1. a segmentation of w into contiguous spans represented by boundaries b, giving spans $\langle w_{b_0:b_1}, w_{b_1:b_2}, \cdots w_{b_{k1}}$ with $b_0 = 0$ and $b_k = n$
2. an assignment of each phrase $w_{b_{i1}:b_i}$ to a concept graph fragment $c_i \in clex(w_{b_{i1}:b_i}) \cup \phi$.

A sequence of spans b and a sequence of concept graph fragments c, both of arbitrary length k, are scored using the following locally decomposed, linearly parameterized function:

$$score(b, c; \theta) = \sum_{i=1}^{k} \theta^T f(w_{b_{i1}:b_i}, b_{i1}, b_i, c_i) \tag{1}$$

where f is a feature vector representation of a span and one of its concept graph fragments in context. The features are:

- **Fragment given words:** Relative frequency estimates of the probability of a concept graph fragment given the sequence of words in the span. This is calculated from the concept-word alignments in the training corpus (2.1.3).
- **Length** of the matching span (number of tokens).
- **NER:** 1 if the named entity tagger marked the span as an entity, 0 otherwise.
- **Bias:** 1 for any concept graph fragment from F and 0 for ϕ.

JAMR highest-scoring b and c using a dynamic programming algorithm: the zeroth-order case of inference under a semiMarkov model [Janssen and Limnios, 1999]. Let $S(i)$ denote the score of the best labeling of the first i words of the sentence, $w_{0:i}$; it can be calculated using the recurrence:

$$S(0) = 0$$

$$S(i) = \max_{\substack{j:0j<i, \\ c \in clex(w_{j:i}) \cup \phi}} \{S(j) + \theta^T f(w_{j:i}, j, i, c)\}$$

The best score will be $S(n)$, and the best scoring concept labeling can be recovered using back-pointers, as in typical implementations of the Viterbi algorithm. Runtime is $O(n^2)$.

clex is implemented as follows. When *clex* is called with a sequence of words, it looks up the sequence in a table that contains, for every word sequence that was labeled with a concept fragment in the training data, the set of concept fragments it was labeled with. *clex* also has a set of rules for generating concept fragments for named entities and time expressions. It generates a concept fragment for any entity recognized by the named entity tagger, as well as for any word sequence matching a regular expression for a time expression. *clex* returns the union of all these concept fragments.

2.1.2 Relation Identification

The relation identification stage adds edges among the concept sub-graph fragments identified in the first stage (2.1.1), creating a graph. We frame the task as a constrained combinatorial optimization problem.

Consider the fully dense labeled multigraph $D = \langle V_D, E_D \rangle$ that includes the union of all labeled vertices and labeled edges in the concept graph fragments, as well as every possible labeled edge $u \xrightarrow{l} v$, for all $u, v \in V_D$ and every $l \in L_E$.

We require a subgraph $G = \langle V_G, E_G \rangle$ that respects the following constraints:

1. **Preserving:** all graph fragments (including labels) from the concept identification phase are subgraphs of G.

2. **Simple:** for any two vertices u and $v \in V_G$, E_G includes at most one edge between u and v. This constraint forbids a small number of perfectly valid graphs, for example for sentences such as "John hurt himself"; however, we see that $< 1\%$ of training instances violate the constraint. We found in preliminary experiments that including the constraint increases overall performance.

3. **Connected:** G must be weakly connected (every vertex reachable from every other vertex, ignoring the direction of edges). This constraint follows from the formal definition of AMR and is never violated in the training data.

4. **Deterministic:** For each node $u \in V_G$, and for each label $l \in L_E$, there is at most one outgoing edge in E_G from u with label l.
   Linguistically, the determinism constraint enforces that predicates have at most one semantic argument of each type.

One constraint we do not include is **acyclicity**, which follows from the definition of AMR. In practice, graphs with cycles are rarely produced by JAMR. In fact, none of the graphs produced on the test set violate acyclicity.

Given the constraints, we seek the maximum-scoring subgraph. We define the score to decompose by edges, and with a linear parameterization:

$$score(E_G; \psi) = \sum_{e \in E_G} \psi^T g(e) \tag{2}$$

The features are shown in Table 2.1.2.

The solution to maximizing the score to Eq. 2, subject to the constraints, uses a Lagrangian relaxation that iteratively adjusts the edge scores to an algorithm which ignores the constraint 4 (but respects the others) so as to enforce the constraint 4. For details about the mathematics involved, please go through [Flanigan et al., 2014].

2.1.2.1 Focus Identification

In AMR, one node must be marked as the focus of the sentence. We notice this can be accomplished within the relation identification step: we add a special concept node root to the dense graph D, and add an edge from root to every other node, giving each of these edges the label FOCUS. We require that root have at most one outgoing FOCUS edge. Our system has two feature types for this edge: the concept it points to, and the shortest dependency path from a word in the span to the root of the dependency tree.

Name	Description
Label	For each $l \in L_E$, 1 if the edge has that label
Self edge	1 if the edge is between two nodes in the same fragment
Tail fragment root	1 if the edge's tail is the root of its graph fragment
Head fragment root	1 if the edge's head is the root of its graph fragment
Path	Dependency edge labels and parts of speech on the shortest syntactic path between any two words in the two spans
Distance	Number of tokens (plus one) between the two concepts' spans (zero if the same)
Distance indicators	A feature for each distance value, that is 1 if the spans are of that distance
Log distance	Logarithm of the distance feature plus one.
Bias	1 for any edge.

Table 1: Features used in relation identification. In addition to the features above, the following conjunctions are used (Tail and Head concepts are elements of L_V): Tail concept $\wedge$ Label, Head concept $\wedge$ Label, Path $\wedge$ Label, Path $\wedge$ Head concept, Path $\wedge$ Tail concept, Path $\wedge$ Head concept $\wedge$ Label, Path $\wedge$ Tail concept $\wedge$ Label, Path $\wedge$ Head word, Path $\wedge$ Tail word, Path $\wedge$ Head word $\wedge$ Label, Path $\wedge$ Tail word $\wedge$ Label, Distance $\wedge$ Label, Distance $\wedge$ Path, and Distance $\wedge$ Path $\wedge$ Label. To conjoin the distance feature with anything else, we multiply by the distance.

2.1.3 Automatic Alignments

In order to train the parser, we need alignments between sentences in the training data and their annotated AMR graphs. More specifically, we need to know which spans of words invoke which concept fragments in the graph. To do this, we built an automatic aligner and tested its performance on a small set of alignments we annotated by hand.

The automatic aligner uses a set of rules to greedily align concepts to spans. The list of rules is given in Table 2.1.3. The aligner proceeds down the list, first aligning named-entities exactly, then fuzzy matching named-entities, then date-entities, etc. For each rule, an entire pass through the AMR graph is done. The pass considers every concept in the graph and attempts to align a concept fragment rooted at that concept if the rule can apply. Some rules only apply to a particular type of concept fragment, while others can apply to any concept. For example, rule 1 can apply to any `NAME` concept and its `OP` children. It searches the sentence for a sequence of words that exactly matches its `OP` children and aligns them to the `NAME` and `OP` children fragment.

Concepts are considered for alignment in the order they are listed in the AMR annotation (left to right, top to bottom). Concepts that are not aligned in a particular pass may be aligned in subsequent passes. Concepts are aligned to the first matching span, and alignments are mutually exclusive. Once aligned, a concept in a fragment is never realigned. However, more concepts can be attached to the fragment by rules 8-14.

We use WordNet to generate candidate lemmas, and we also use a fuzzy match of a concept, de- fined to be a word in the sentence that has the longest string prefix match with that concept's label, if the match length is 4. If the match length is < 4, then the concept has no fuzzy match. For example the fuzzy match for `ACCUSE-01` could be "accusations" if it is the best match in the sentence. WordNet lemmas and fuzzy matches are only used if the rule explicitly uses them. All tokens and concepts are lowercased before matches or fuzzy matches are done.
The aligner was trained on hand-aligned data (200 sentences [Flanigan et al., 2014]).

1. **(Named Entity)** Applies to `name` concepts and their `opn` children. Matches a span that exactly matches its `opn` children in numerical order.
2. **(Fuzzy Named Entity)** Applies to `name` concepts and their `opn` children. Matches a span that matches the fuzzy match of each child in numerical order.
3. **(Date Entity)** Applies to `date-entity` concepts and their `day, month, year` children (if exist). Matches any permutation of day, month, year, (two digit or four digit years), with or without spaces.
4. **(Minus Polarity Tokens)** Applies to - concepts, and matches "no", "not", "non."
5. **(Single Concept)** Applies to any concept. Strips off trailing '-[0-9]+' from the concept (for example `run-01` → `run`), and matches any exact matching word or WordNet lemma.
6. **(Fuzzy Single Concept)** Applies to any concept. Strips off trailing '-[0-9]+', and matches the fuzzy match of the concept.
7. **(U.S.)** Applies to `name` if its `op1` child is `united` and its `op2` child is `states`. Matches a word that matches "us", "u.s." (no space), or "u. s." (with space).
8. **(Entity Type)** Applies to concepts with an outgoing `name` edge whose head is an aligned fragment. Updates the fragment to include the unaligned concept. Ex: `continent` in `(continent :name (name :op1 "Asia"))` aligned to "asia."
9. **(Quantity)** Applies to `.*-quantity` concepts with an outgoing `unit` edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `distance-quantity` in `(distance-quantity :unit kilometer)` aligned to "kilometres."
10. **(Person-Of, Thing-Of)** Applies to `person` and `thing` concepts with an outgoing `.*-of` edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `person` in `(person :ARG0-of strike-02)` aligned to "strikers."
11. **(Person)** Applies to `person` concepts with a single outgoing edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `person` in `(person :poss (country :name (name :op1 "Korea")))`
12. **(Goverment Organization)** Applies to concepts with an incoming `ARG.*-of` edge whose tail is an aligned `government-organization` concept. Updates the fragment to include the unaligned concept. Ex: `govern-01` in `(government-organization :ARG0-of govern-01)` aligned to "government."
13. **(Minus Polarity Prefixes)** Applies to - concepts with an incoming `polarity` edge whose tail is aligned to a word beginning with "un", "in", or "il." Updates the fragment to include the unaligned concept. Ex: - in `(employ-01 :polarity -)` aligned to "unemployment."
14. **(Degree)** Applies to concepts with an incoming `degree` edge whose tail is aligned to a word ending is "est." Updates the fragment to include the unaligned concept. Ex: `most` in `(large :degree most)` aligned to "largest."

Table 2: Rules used in the automatic aligner

2.2 Our Inspiration

Zhou and Xu, in their work [Zhou and Xu, 2015] have done a similar thing. They also created a graph based on the meaning of the statement. But they did not have any formalism in the representation of their output.
Their goal was semantic role labeling (SRL), which they performed using a Deep Bi-directional Long Sort-Term Memory (DB-LSTM).

2.2.1 A Brief about the paper

Semantic role labeling (SRL) is a form of shallow semantic parsing whose goal is to discover the predicate-argument structure of each predicate in a given input sentence.
Given a sentence, for each target verb (predicate) all the constituents in the sentence which fill a semantic role of the verb have to be recognized. Typical semantic arguments include Agent, Patient, Instrument, etc., and also adjuncts such as Locative, Temporal, Manner, Cause, etc..

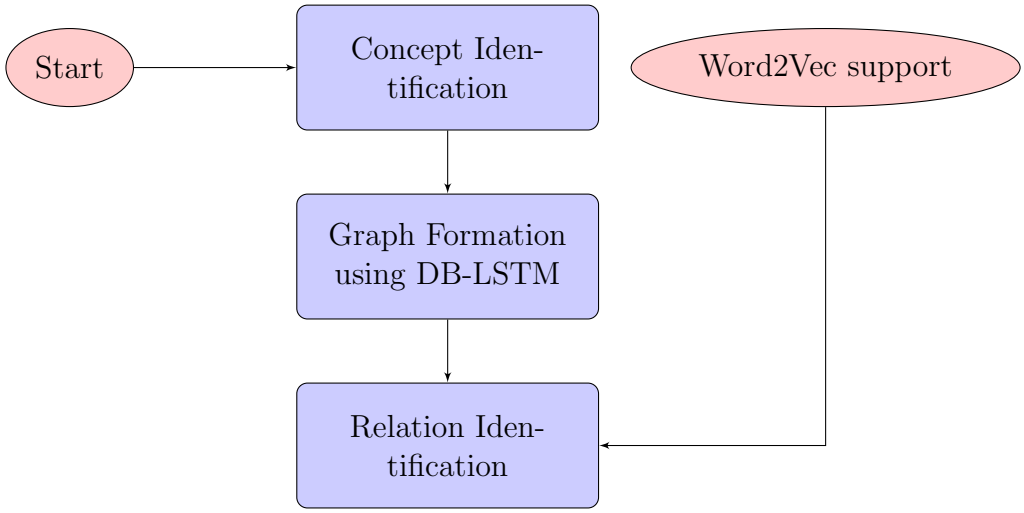
SRL is considered as a supervised machine learning problem. In traditional methods, linear classifier such as SVM is often employed to perform this task based on features extracted from the training corpus.

To address the above issues, [Collobert et al., 2011] proposed a unified neural network architecture using word embedding and convolution. They applied their architecture on four standard NLP tasks: Part-Of-Speech tagging (POS), chunking (CHUNK), Named Entity Recognition (NER) and Semantic Role Labeling (SRL). They were able to reach the previous state-of-the-art performance on all these tasks except for SRL. They had to resort to parsing features in order to make the system competitive with state-of-the-art performance.

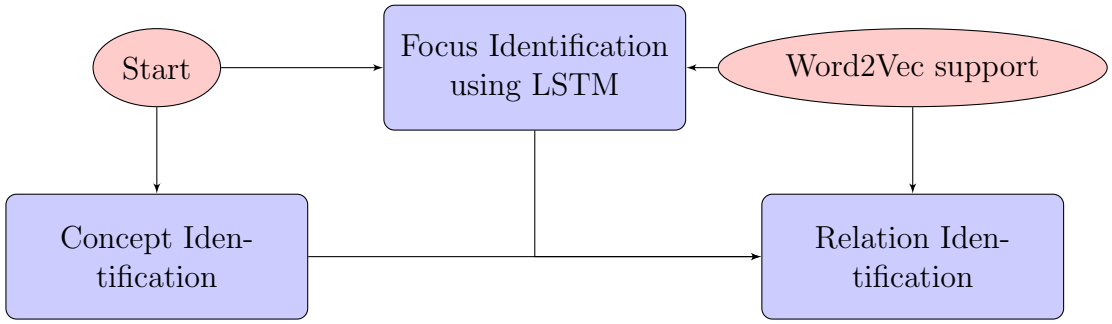
To address the above difficulties, [Zhou and Xu, 2015] proposed an end-to-end system using deep bi-directional long short-term memory (DB-LSTM) model. This model takes only original text as the input features, without any intermediate tag such as syntactic information. The input features are processed by the following 8 layers of LSTM bidirectionally. At the top locates the conditional random field (CRF) model for tag sequence prediction. It achieves the state-of-the-art performance of f-score $F_1 = 81.07$ on CoNLL-2005 shared task and $F_1 = 81.27$ on CoNLL-2012 shared task.

3 Methodology

Our idea was to implement the algorithm described in [Zhou and Xu, 2015] on the concepts created by [Flanigan et al., 2014], and then label the edges using a similar technique. We also wanted to support the relation-label identification using Word2Vec by giving greater weight to labels that occur more frequently with a particular concept.



But due to time constraints in the course project, we thought of implementing some change similar to [Zhou and Xu, 2015] that would be most critical to the algorithm. So we targetted the focus of the word, which was critical in the relation identification step.



We trained an LSTM on the data to learn the focus of a sentence, and assigned greater weight to the focus in the relation identification step.

4 Results

We evaluate the performance of the full parser using Smatch v1.0 [Cai and Knight, 2013], which counts the precision, recall and F_1 of the concepts and relations together. Using the full pipeline (concept identification and relation identification stages), our parser achieves ¡¿% F_1 on the test data (Table 3).

— Evaluation on Test —

Algorithm	**Precision**	**Recall**	**F1 Score**
JAMR claimed result	0.52	0.66	0.58
JAMR + LSTM	0.763	0.438	0.557

Table 3: Parser Performance

Test Sentence-
"However , most of the buildings in this hard - hit area did not meet these requirements"
AMR produced by:

- JAMR:

```
(r / require-01
      :ARG1 (t / thing
            :mod (t2 / this)
            :name (a / area
                  :ARG1-of (h / hit-01
                        :manner (h2 / hard))
                  :location-of (b / building
                        :quant (m / most)))))
```

- JAMR + LSTM:

- Ideal output:

```
(c / contrast-01
     :ARG2 (m / meet-01 :polarity -
           :ARG0 (b / building
                 :quant (m2 / most)
                 :location (a3 / area
                       :ARG1-of (h / hit-01
                             :manner (h2 / hard-04))
                       :mod (t5 / this)))))
           :ARG1 (t2 / thing
                 :ARG1-of (r / require-01)
                 :mod (t3 / this)))
```

5 Future Work

- AMR have only been developed for english language. Hence extension to other languages.
- Train LSTM with the sentence parsed recursively to capture more features.
- Deep Bidirectional LSTM are in general much more promising than the single layered LSTM as the earlier work by [Zhou and Xu, 2015] suggests.

References

[Cai and Knight, 2013] Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Flanigan et al., 2014] Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the abstract meaning representation.

[ISI, 2015] ISI, A. (2015). Abstract meaning representation specification. https://github.com/amrisi/amr-guidelines/blob/master/amr.md#part-ii--concepts-and-relations.

[Janssen and Limnios, 1999] Janssen, J. and Limnios, N. (1999). *Semi-Markov Models and Applications*. Springer US.

[May, 2015] May, J. (2015). Semeval 2016, task 8. http://alt.qcri.org/semeval2016/task8/.

[Zhou and Xu, 2015] Zhou, J. and Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks.