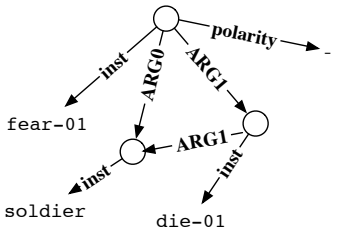
Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, Jonathan May
Information Sciences Institute
Computer Science Department
University of Southern California
{pust, ulf, knight, marcu, jonmay}@isi.edu

Abstract

We present a parser for Abstract Meaning Representation (AMR). We treat English-to-AMR conversion within the framework of string-to-tree, syntax-based machine translation (SBMT). To make this work, we transform the AMR structure into a form suitable for the mechanics of SBMT and useful for modeling. We introduce an AMR-specific language model and add data and features drawn from semantic resources. Our resulting AMR parser significantly improves upon state-of-the-art results.



The soldier was not afraid of dying.
The soldier was not afraid to die.
The soldier did not fear death.

Figure 1: An Abstract Meaning Representation (AMR) with several English renderings.

1 Introduction

Abstract Meaning Representation (AMR) is a compact, readable, whole-sentence semantic annotation (Banarescu et al., 2013). It includes entity identification and typing, PropBank semantic roles (Kingsbury and Palmer, 2002), individual entities playing multiple roles, as well as treatments of modality, negation, etc. AMR abstracts in numerous ways, e.g., by assigning the same conceptual structure to *fear* (v), *fear* (n), and *afraid* (adj). Figure 1 gives an example.

AMR parsing is a new research problem, with only a few papers published to date (Flanigan et al., 2014; Wang et al., 2015) and a publicly available corpus of more than 10,000 English/AMR pairs.[1] New research problems can be tackled either by developing new algorithms/techniques (Flanigan et al., 2014; Wang et al., 2015) or by adapting existing algorithms/techniques to the problem at hand. In this paper, we investigate the second approach.

The AMR parsing problem bears a strong formal resemblance to syntax-based machine translation (SBMT) of the string-to-tree variety, in that a string is transformed into a nested structure in both cases. Because of this, it is appealing to apply the substantial body of techniques already invented for SBMT[2] to AMR parsing. By re-using an SBMT inference engine instead of creating custom inference procedures, we lose the ability to embed some task-specific decisions into a custom transformation process, as is done by Flanigan et al. (2014) and Wang et al. (2015). However, we reap the efficiency gains that come from working within a tested, established framework. Furthermore, since production-level SBMT systems are widely available, anyone wishing to generate AMR from text need only follow our recipe and retrain an existing framework with relevant data to quickly obtain state-of-the-art results.

Since SBMT and AMR parsing are, in fact, distinct tasks, as outlined in Figure 2, to adapt the SBMT parsing framework to AMR parsing, we develop novel representations and techniques. Some of our key ideas include:

1. Introducing an AMR-equivalent representation that is suitable for string-to-tree SBMT rule extraction and decoding (Section 4.1).

[1] LDC Catalog number 2014T12

[2] See e.g. the related work section of Huck et al. (2014).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1143–1154,
Lisbon, Portugal, 17-21 September 2015.

	SBMT	AMR parsing
Target	tree	graph
Nodes	labeled	unlabeled
Edges	unlabeled	labeled
Alignments	words to leaves	words to leaves + words to edges
Children	ordered	unordered
Accuracy Metric	BLEU (Papineni et al., 2002)	Smatch (Cai and Knight, 2013)

Figure 2: Differences between AMR parsing and syntax-based machine translation (SBMT).

2. Proposing a target-side reordering technique that leverages the fact that child nodes in AMR are unordered (Section 4.4).
3. Introducing a hierarchical AMR-specific language model to ensure generation of likely parent-child relationships (Section 5).
4. Integrating several semantic knowledge sources into the task (Section 6).
5. Developing tuning methods that maximize Smatch (Cai and Knight, 2013) (Section 7).

By applying these key ideas, which constitute lightweight changes on a baseline SBMT system, we achieve state-of-the-art AMR parsing results. We next describe our baseline, and then describe how we adapt it to AMR parsing.

2 Syntax-Based Machine Translation

Our baseline SBMT system proceeds as follows. Given a corpus of (source string, target tree, source-target word alignment) sentence translation training tuples and a corpus of (source, target, score) sentence translation tuning tuples:

1. **Rule extraction:** A grammar of string-to-tree rules is induced from training tuples using the GHKM algorithm (Galley et al., 2004; Galley et al., 2006).
2. **Local feature calculation:** Statistical and indicator features, as described by Chiang et al. (2009), are calculated over the rule grammar.
3. **Language model calculation:** A Kneser-Ney-interpolated 5-gram language model (Chen and Goodman, 1996) is learned from the yield of the target training trees.
4. **Decoding:** A beamed bottom-up chart decoder (Pust and Knight, 2009; Hopkins and Langmead, 2010) calculates the optimal derivations given a source string and feature parameter set.
5. **Tuning:** Feature parameters are optimized using the MIRA learning approach (Chiang et al., 2009) to maximize the objective, typically BLEU (Papineni et al., 2002), associated with a tuning corpus.

We initially use this system with no modifications and pretend that English–AMR is a language pair indistinct from any other.

Corpus	Sentences	Tokens
Training	10,313	218,021
Development	1,368	29,484
Test	1,371	30,263

Table 1: Data splits of AMR 1.0, used in this work. Tokens are English, after tokenization.

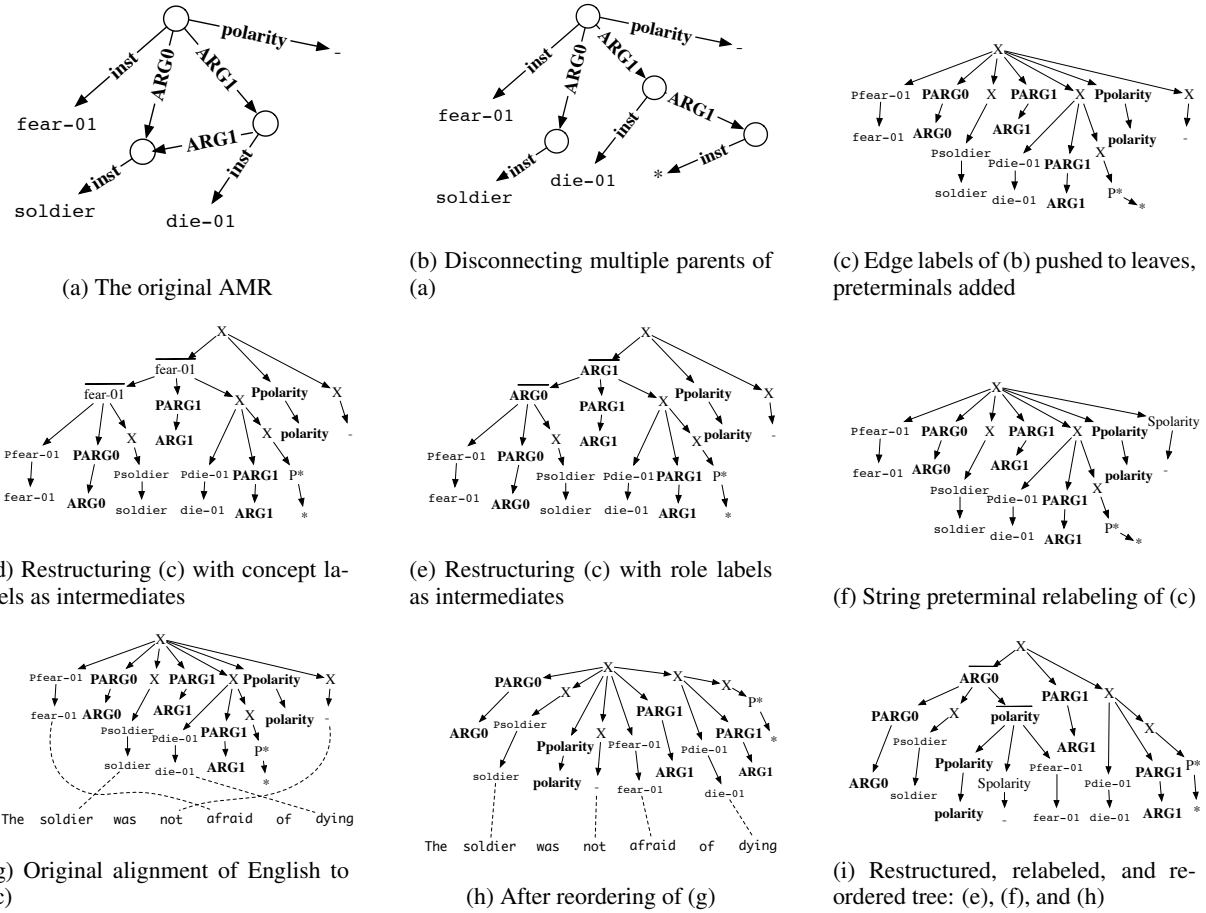
3 Data and Comparisons

We use English–AMR data from the AMR 1.0 corpus, LDC Catalog number 2014T12. In contrast to narrow-domain data sources that are often used in work related to semantic parsing (Price, 1990; Zelle, 1995; Kuhlmann et al., 2004), the AMR corpus covers a broad range of news and web forum data. We use the training, development, and test splits specified in the AMR corpus (Table 1). The training set is used for rule extraction, language modeling, and statistical rule feature calculation. The development set is used both for parameter optimization and qualitatively for hill-climbing. The test set is held out blind for evaluation. We preprocess the English with a simple rule-based tokenizer and, except where noted, lowercase all data. We obtain English–AMR alignments by using the unsupervised alignment approach of Pourdamghani et al. (2014), which linearizes the AMR and then applies the models of Brown et al. (1993) with an additional symmetrization constraint.

All parsing results reported in this work are obtained with the Smatch 1.0 software (Cai and Knight, 2013). We compare our results to those of Flanigan et al. (2014) on the AMR 1.0 data splits; we run that work's JAMR software according to the provided instructions.[3] We also compare our results to published scores in the recent work of Wang et al. (2015). Their work uses slightly different data than that used here [4] but in practice we have not seen significant variation in results.

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

[4] LDC2013E117, a pre-released version of LDC2014T12 that is not generally available.



(a) The original AMR

(b) Disconnecting multiple parents of (a)

(c) Edge labels of (b) pushed to leaves, preterminals added

(d) Restructuring (c) with concept labels as intermediates

(e) Restructuring (c) with role labels as intermediates

(f) String preterminal relabeling of (c)

(g) Original alignment of English to (c)

(h) After reordering of (g)

(i) Restructured, relabeled, and reordered tree: (e), (f), and (h)

Figure 3: Transformation of AMR into tree structure that is acceptable to GHKM rule extraction (Galley et al., 2004; Galley et al., 2006) and yields good performance.

4 AMR Transformations

In this section we discuss various transformations to our AMR data. Initially, we concern ourselves with converting AMR into a form that is amenable to GHKM rule extraction and string to tree decoding. We then turn to structural transformations designed to improve system performance. Figure 3 progressively shows all the transformations described in this section; the example we follow is shown in its original form in Figure 3a. We note that all transformations are done internally; the input to the final system is a sentence and the output is an AMR. We further observe that all transformations are data-driven and language agnostic.

4.1 Massaging AMRs into Syntax-Style Trees

The relationships in AMR form a directed acyclic graph (DAG), but GHKM requires a tree, so we must begin our transformations by discarding some information. We arbitrarily disconnect all but a single parent from each node (see Figure 3b). This is the only lossy modification we make to our AMR data. As multi-parent relationships occur 1.05 times per training sentence and at least once in 48% of training sentences, this is indeed a regrettable loss. We nevertheless make this modification, since it allows us to use the rest of our string-to-tree tools.

AMR also contains labeled edges, unlike the constituent parse trees we are used to working with in SBMT. These labeled edges have informative content and we would like to use the alignment procedure of Pourdamghani et al. (2014), which aligns words to edges as well as to terminal nodes. So that our AMR trees are compatible with both our desired alignment approach and our desired rule extraction approach, we propagate edge labels to terminals via the following procedure:

1. For each node n in the AMR tree we create a corresponding node m with the all-purpose symbol 'X' in the SBMT-like tree. Outgoing edges from n come in two flavors: *concept*

edges, labeled 'inst,' which connect n to a terminal *concept* such as `fear-01`, and *role edges*, which have a variety of labels such as **ARG0** and **name**, and connect n to another instance or to a *string*.[5] A node has one instance edge and zero or more role edges. We consider each type of edge separately.

2. For each outgoing role edge we insert two unlabeled edges into the corresponding transformation; the first is an edge from m to a terminal bearing the original edge's role label (a so-called *role label edge*), and the second (a *role filler edge*) connects m to the transformation of the original edge's target node, which we process recursively. String targets of a role receive an 'X' preterminal to be consistent with the form of role filler edges.
3. For the outgoing concept edge we insert an unlabeled edge connecting m and the concept. It is unambiguous to determine which of m's edges is the concept edge and which edges constitute role label edges and their corresponding role filler edges, as long as paired label and filler edges are adjacent.
4. Since SBMT expects trees with preterminals, we simply replicate the label identities of concepts and role labels, adding a marker ('P' in Figure 3) to distinguish preterminals.

The complete transformation can be seen in Figure 3c. Apart from multiple parent ancestry, the original AMR can be reconstructed deterministically from this SBMT-compliant rewrite.

4.2 Tree Restructuring

While the transformation in Figure 3c is acceptable to GHKM, and hence an entire end-to-end AMR parser may now be built with SBMT tools, the resulting parser does not exhibit very good performance (Table 3, first line). The trees we are learning on are exceedingly flat, and thus yield rules that do not generalize sufficiently. Rules produced from the top of the tree in Figure 3c, such as that in Figure 4a, are only appropriate for cases where `fear-01` has exactly three roles: **ARG0** (agent), **ARG1** (patient), and **polarity**.

We follow the lead of Wang et al. (2010), who in turn were influenced by similar approaches in monolingual parsing (Collins, 1997; Charniak, 2000), and re-structure trees at nodes with more than three children (i.e. instances with more than one role), to allow generalization of flat structures.

However, our trees are unlike syntactic constituent trees in that they do not have labeled nonterminal nodes, so we have no natural choice of an intermediate ("bar") label. We must choose a meaningful label to characterize an instance and its roles. We initially choose the concept label, resulting in trees like that in Figure 3d, in which a chain of fear-01 nodes is used to unflatten the root, which has instance `fear-01`.

This attempt at re-structuring yields rules like that in Figure 4b, which are general in form but are tied to the concept context in which they were extracted. This leads to many redundant rules and blows up the nonterminal vocabulary size to approximately 8,000, the size of the concept vocabulary. Furthermore, the rules elicited by this procedure encourage undesirable behavior such as the immediate juxtaposition of two rules generating **ARG1**.

We next consider restructuring with the immediately dominant role labels, resulting in trees like that in Figure 3e and rules like that in Figure 4c. The shape of the structure added is the same as in Figure 3d but the bar nodes now take their labels from their second children. This approach leads to more useful rules with fewer undesirable properties.

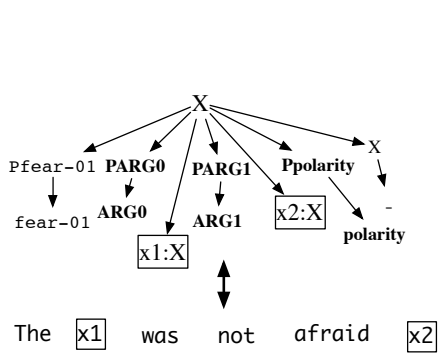
4.3 Tree Relabeling

AMR strings have an effective preterminal label of 'X,' which allows them to compete with full AMR instances at decode time. However, whether or not a role is filled by a string or an instance is highly dependent on the kind of role being filled. The **polarity** and **mode** roles, for instance, are nearly always filled by strings, but **ARG0** and **ARG1** are always filled by instances. The **quant** role, which is used for representation of numerical quantities, can be filled by an instance (e.g. for approximate quantities such as 'about 3') or a string. To capture this behavior we relabel string preterminals of the tree with labels indicating role identity and string subsumption. This relabeling, replaces, for example, one 'X' preterminal in Figure 3c with "Spolarity," as shown in Figure 3f.

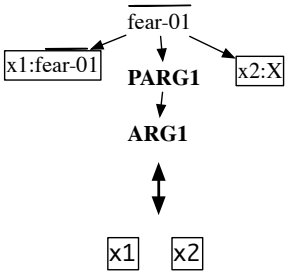
4.4 Tree Reordering

Finally, let us consider the alignments between English and AMR. As is known in SBMT, nonmonotone alignments can lead to large, unwieldy

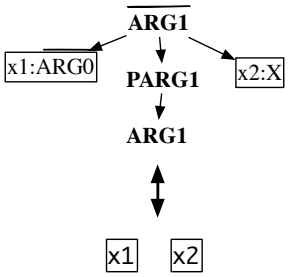
[5] In Figure 3 the negative polarity marker '-' is a string. Disconnected referents labeled '*' are treated as AMR instances with no roles.



(a) A rule extracted from the AMR tree of Figure 3c. All roles seen in training must be used.



(b) A rule from the AMR tree of Figure 3d. Many nearly identical rules of this type are extracted, and this rule can be used multiple times in a single derivation.



(c) A rule from the AMR tree of Figure 3e. This rule can be used independent of the concept context it was extracted from and multiple reuse is discouraged.

Figure 4: Impact of restructuring on rule extraction.

rules and in general make decoding more difficult (May and Knight, 2007). While this is often an unavoidable fact of life when trying to translate between two natural languages with different syntactic behavior, it is an entirely artificial phenomenon in this case. AMR is an *unordered* representation, yet in order to use an SBMT infrastructure we must declare an order of the AMR tree. This means we are free to choose whatever order is most convenient to us, as long as we keep role label edges immediately adjacent to their corresponding role filler edges to preserve conversion back to the edge-labeled AMR form. We thus choose the order that is as close as possible to that of the source yet still preserves these constraints. We use a simple greedy bottom-up approach that permutes the children of each internal node of the unrestructured tree so as to minimize crossings. This leads to a 79% overall reduction in crossings and is exemplified in Figure 3g (before) and Figure 3h (after). We may then restructure our trees, as described above, in an instance-outward manner. The final restructured, relabeled, and reordered tree is shown in Figure 3i.

5 AMR Language Models

We now turn to language models of AMRs, which help us prefer reasonable target structures over unreasonable ones.

Our first language model is unintuitively simple—we pretend there is a language called *AMRese* that consists of yields of our restructured AMRs. An example AMRese string from Figure 3i is '**ARG0** `soldier polarity -` `fear-01` **ARG1** `die-01` **ARG1** *.' We then build a standard n-gram model for AMRese.

It also seems sensible to judge the correctness of an AMR by calculating the empirical probability of the concepts and their relations to each other. This is the motivation behind the following model of an AMR:[6]

We define an AMR instance $i = (c, R)$, where c is a *concept* and R is a set of *roles*. We define an AMR role $r = (l, i)$, where l is a role label, and i is an AMR instance labeled l. For an AMR instance i let $\hat{c}_i$ be the concept of i's parent instance, and $\hat{l}_i$ be the label of the role that i fills with respect to its parent. We also define the special instance and role labels ROOT and STOP. Then, we define $P_{\text{AMR}}(i|\hat{l}_i, \hat{c}_i)$, the conditional probability of AMR instance i given its ancestry as $P_{\text{AMR}}(i = (c, R)|\hat{l}_i, \hat{c}_i) = P(c|\hat{l}_i, \hat{c}_i) \prod_{r \in R} P_{\text{Role}}(r|c) \times P(\text{STOP}|c)$, where $P_{\text{Role}}(r = (l, i)|c) = P(l|c)P_{\text{AMR}}(i|l, c)$.

We define $P(c|\hat{l}_i, \hat{c}_i)$, $P(l|c)$, and $P(\text{STOP}|c)$ as empirical conditional probabilities, Witten-Bell interpolated (Witten and Bell, 1991) to lower-order models by progressively discarding context from the right.[7] We model exactly one STOP event per instance. We define the probability of a full-sentence AMR i as $P_{\text{AMR}}(i|\text{ROOT})$ where ROOT in this case serves as both parent concept and role label.

[6] This model is only defined over AMRs that can be represented as trees, and not over all AMRs. Since tree AMRs are a prerequisite of our system we did not yet investigate whether this model could be sufficiently generalized.

[7] That is, $P(c|\hat{l}_i, \hat{c}_i)$ is interpolated with $P(c|\hat{l}_i)$ and then $P(c)$.

System	Tune	Test
AMRese n-gram LM	61.7	59.7
AMR LM	59.1	57.1
both LMs	62.3	60.6

Table 2: The effect of AMRese n-gram and AMR LMs on Smatch quality.

As an example, the instance associated with concept `die-01` in Figure 3b has $\hat{l}_i$ = **ARG1** and $\hat{c}_i$ = `fear-01`, so we may score it as $P(\texttt{die-01}|\textbf{ARG1}, \texttt{fear-01}) \times P(\textbf{ARG1}|\texttt{die-01}) \times P(\text{STOP}|\texttt{die-01}) \times P(*|\textbf{ARG1}, \texttt{die-01})$

In Table 2 we compare the effect of varying LMs on Smatch quality. The AMR LM by itself is inferior to the AMRese n-gram LM, but combining the two yields superior quality.

6 Adding External Semantic Resources

Although we are engaged in the task of semantic parsing, we have not yet discussed the use of any semantic resources. In this section we rectify that omission.

6.1 Rules from Numerical Quantities and Named Entities

While the majority of string-to-tree rules in SBMT systems are extracted from aligned parallel data, it is common practice to dynamically generate additional rules to handle the translation of dates and numerical quantities, as these follow common patterns and are easily detected at decode-time. We follow this practice here, and additionally detect person names at decode-time using the Stanford Named Entity Recognizer (Finkel et al., 2005). We use cased, tokenized source data to build the decode-time rules. We add indicator features to these rules so that our tuning methods can decide how favorable the resources are. We leave as future work the incorporation of named-entity rules for other classes, since most available named-entity recognition beyond person names is at a granularity level that is incompatible with AMR (e.g. we can recognize 'Location' but not distinguish between 'City' and 'Country').

6.2 Hierarchical Semantic Categories

In order to further generalize our rules, we modify our training data AMRs once more, this time replacing the identity preterminals over concepts

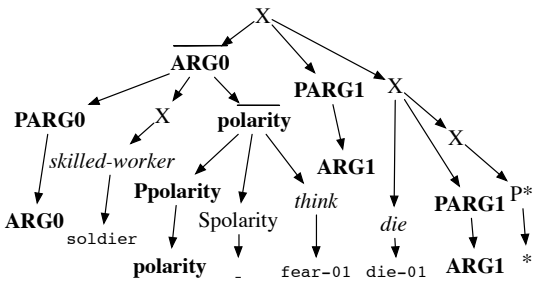


Figure 5: Final modification of the AMR data; semantically clustered preterminal labels are added to concepts.

with preterminals designed to enhance the applicability of our rules in semantically similar contexts. For each concept c expressed in AMR, we consult WordNet (Fellbaum, 1998) and a curated set of gazetteers and vocabulary lists to identify a hierarchy of increasingly general semantic categories that describe the concept. So as not to be overwhelmed by the many fine-grained distinctions present in WordNet, we pre-select around 100 salient semantic categories from the WordNet ontology. When traversing the WordNet hierarchy, we propagate a smoothed count[8] of the number of examples seen per concept sense,[9] combining counts when paths meet. For each selected semantic category s encountered in the traversal, we calculate a weight by dividing the propagated example count for c at s by the frequency s was proposed over all AMR concepts. We then assign c to the highest scoring semantic category s. An example calculation for the concept `computer` is shown in Figure 7.

We apply semantic categories to our data as replacements for identity preterminals of concepts. This leads to more general, more widely-applicable rules. For example, with this transformation, we can parse correctly not only contexts in which "soldiers die," but also contexts in which other kinds of "skilled workers die." Figure 5 shows the addition of semantic preterminals to the tree from Figure 3i. We also incorporate semantic categories into the AMR LM. For concept c, let s_c be the semantic category of c. Then we reformu-

[8] We use very simple smoothing, and add 0.1 to the provided example counts.

[9] Since WordNet senses do not correspond directly to PropBank or AMR senses, we simply use a lexical match and must consider all observed senses for that match.

System	Sec.	Tune	Test
flat trees	4.1	51.6	49.9
concept restructuring	4.2	57.2	55.3
role restructuring (rr)	4.2	**60.8**	**58.6**
rr + string preterminal relabeling (rl)	4.3	**61.3**	**59.7**
rr + rl + reordering (ro)	4.4	**61.7**	**59.7**
rr + rl + ro + AMR LM	5	**62.3**	**60.6**
rr + rl + ro + AMR LM + date/number/name rules (dn)	6.1	**63.3**	**61.3**
rr + rl + ro + AMR LM + dn + semantic categories (sc)	6.2	**66.2**	**64.3**
rr + rl + ro + AMR LM + dn + sc + morphological normalization (mn)	6.3	**67.3**	**65.4**
rr + rl + ro + AMR LM + dn + sc + mn, rule-based alignments	6.4	**68.3**	**66.3**
rr + rl + ro + AMR LM + dn + sc + mn, rule-based + unsupervised alignments	6.4	**69.0**	**67.1**
JAMR (Flanigan et al., 2014)	9	58.8	58.2
dependency parse-based (Wang et al., 2015)	9	N/A	63

Table 3: AMR parsing Smatch scores for the experiments in this work. We provide a cross-reference to the section of this paper that describes each of the evaluated systems. Entries in **bold** are improvements over JAMR (Flanigan et al., 2014). Test entries underlined are improvements over the dependency-based work of Wang et al. (2015). Human inter-annotator Smatch performance is in the 79-83 range (Cai and Knight, 2013).

English	AMRese
tigers	`tiger`
asbestos	`asbestos`
quietly	`quiet`
nonexecutive	`executive` **polarity** '-'
broke up	`break-up-08`

Table 4: Lexical conversions to AMRese form due to the morphological normalization rules described in Section 6.3.

late $P_{\text{AMR}}(i|\hat{l}_i, \hat{c}_i)$ as $P_{\text{AMR}}(i = (c, R)|\hat{l}_i, \hat{c}_i) = P(s_c|\hat{l}_i, s_{\hat{c}_i}, \hat{c}_i)P(c|s_c, \hat{l}_i, s_{\hat{c}_i}, \hat{c}_i) \prod_{r \in R} P_{\text{Role}}(r|c) \times P(\text{STOP}|s_c, c)$, where $P_{\text{Role}}(r = (l, i)|c) = P(l|s_c, c) \times P_{\text{AMR}}(i|l, c)$.

6.3 Morphological Normalization

While we rely heavily on the relationships between words in-text and concept nodes expressed in parallel training data, we find this is not sufficient for complete coverage. Thus we also include a run-time module that generates AMRese base forms at the lexical level, expressing relationships such as those depicted in Table 4. We build these dictionary rules using three resources:

1. An inflectional morphological normalizing table, comprising a lexicon with 84,558 entries, hand-written rules for regular inflectional morphology, and hand-written lists of irregular verbs, nouns, and adjectives.
2. Lists of derivational mappings (e.g. 'quietly' → 'quiet', 'naval' → 'navy').
3. PropBank framesets, which we use, e.g., to map the morphologically normalized 'break up' (from 'broke up') into a sense match, such as `break-up-08`.

6.4 Semantically informed Rule-based Alignments

For our final incorporation of semantic resources we revisit the English-to-AMR alignments used to extract rules. As an alternative to the unsupervised approach of Pourdamghani et al. (2014), we build alignments by taking a linguistically-aware, supervised heuristic approach to alignment:

First, we generate a large number of potential links between English and AMR. We attempt to link English and AMR tokens after conversion through resources such as a morphological analyzer, a list of 3,235 pertainym pairs (e.g. adj-'gubernatorial' → noun-'governor'), a list of 2,444 adverb/adjective pairs (e.g. 'humbly' → 'humble'), a list of 2,076 negative polarity pairs (e.g. 'illegal' → 'legal'), and a list of 2,794 known English-AMR transformational relationships (e.g. 'asleep' → `sleep-01`, 'advertiser' → `person` **ARG0-of** `advertise-01`, 'Greenwich Mean Time' → `GMT`). These links are then culled based on context and AMR structure. For example, in the sentence "The big fish ate the little fish," ini-

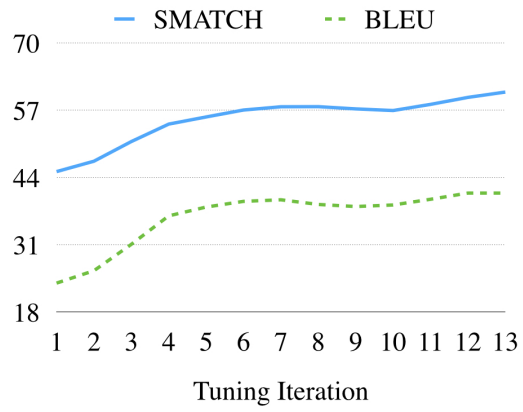


Figure 6: BLEU of AMRese and Smatch correlate closely when tuning.

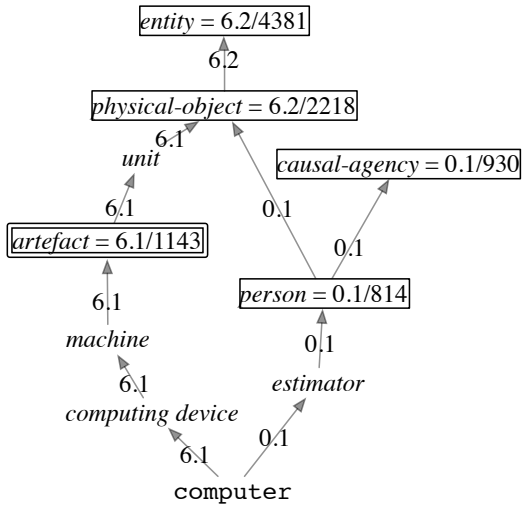


Figure 7: WordNet hierarchy for `computer`. Pre-selected salient WordNet categories are boxed. Smoothed sense counts are propagated up the hierarchy and re-combined at join points. Scores are calculated by dividing propagated sense count by count of the category's prevalence over the set of AMR concepts. The double box indicates the selection of *artefact* as the category label for `computer`.

tially both English 'fish' are aligned to both AMR `fish`. However, based on the context of 'big' and 'little' the spurious links are removed.

In our experiments we explore both replacing the unsupervised alignments of Pourdamghani et al. (2014) with these alignments and concatenating the two alignment sets together, essentially doubling the size of the training corpus. Because the different alignments yield different target-side tree reorderings, it is necessary to build separate 5-gram AMRese language models.[10] When using both alignment sets together, we also use both AMRese language models simultaneously.

7 Tuning

We would like to tune our feature weights to maximize Smatch directly. However, a very convenient alternative is to compare the AMRese yields of candidate AMR parses to those of reference AMRese strings, using a BLEU objective and forest-based MIRA (Chiang et al., 2009). Figure 6 shows that MIRA tuning with BLEU over AMRese tracks closely with Smatch. Note that, for experiments using reordered AMR trees, this requires obtaining similarly permuted reference tuning AMRese and hence requires alignments on the development corpus. When using unsupervised alignments we may simply run inference on the trained alignment model to obtain development alignments. The rule-based aligner runs one sentence at a time and can be employed on the development corpus. When using both sets of alignments, each approach's AMRese is used as a development reference (i.e. each development sentence has two possible reference translations).

[10]The AMR LM is insensitive to reordering so we do not need to vary it when varying alignments.

8 Results and Discussion

Our AMR parser's performance is shown in Table 3. We progressively show the incremental improvements and compare to the systems of Flanigan et al. (2014) and Wang et al. (2015). Purely transforming AMR data into a form that is compatible with the SBMT pipeline yields suboptimal results, but by adding role-based restructuring, relabeling, and reordering, as described in Section 4, we are able to surpass Flanigan et al. (2014). Adding an AMR LM and semantic resources increases scores further, outperforming Wang et al. (2015). Rule-based alignments are an improvement upon unsupervised alignments, but concatenating the two alignments is even better. We compare rule set sizes of the various systems in Table 5; initially we improve the rule set by removing numerous overly brittle rules but then successive changes progressively add useful rules. The parser is available for public download and use at `http://amr.isi.edu`.

System	Rules
flat trees	1,430,124
concept restructuring	678,265
role restructuring (rr)	660,582
rr + preterminal relabeling (rl)	661,127
rr + rl + semantic categories (sc)	765,720
rr + rl + sc + reordering (ro)	790,624
rr + rl + sc + ro + rule-based alignments	908,318
rr + rl + sc + ro + both alignments	1,306,624

Table 5: Comparison of extracted rule set size on the systems evaluated in this work. Note that, as compared to Table 3, only systems that affect the rule size are listed.

9 Related Work

The first work that addressed AMR parsing was that of Flanigan et al. (2014). In that work, multiple discriminatively trained models are used to identify individual concept instances and then a minimum spanning tree algorithm connects the concepts. That work was extended and improved upon by Werling et al. (2015). Recent work by Wang et al. (2015) also uses a two-pass approach; dependency parses are modified by a tree-walking algorithm that adds edge labels and restructures to resolve discrepancies between dependency standards and AMR's specification. In contrast to these works, we use a single-pass approach and re-use existing machine translation architecture, adapting to the AMR parsing task by modifying training data and adding lightweight AMR-specific features.

Several other recent works have used a machine translation approach to semantic parsing, but all have been applied to domain data that is much narrower and an order of magnitude smaller than that of AMR, primarily the Geoquery corpus (Zelle and Mooney, 1996). The WASP system of Wong and Mooney (2006) uses hierarchical SMT techniques and does not apply semantic-specific improvements; its extension (Wong and Mooney, 2007) incorporates a target-side reordering component much like the one presented in Section 4.4. Jones et al. (2012a) cast semantic parsing as an instance of hyperedge replacement grammar transduction; like this work they use an IBM model-influenced alignment algorithm and a GHKM-based extraction algorithm. Andreas et al. (2013) use phrase-based and hierarchical SMT techniques on Geoquery. Like this work, they perform a transformation of the input semantic representation so that it is amenable to use in an existing machine translation system. However, they are unable to reach the state of the art in performance. Li et al. (2013) directly address GHKM's word-to-terminal alignment requirement by extending that algorithm to handle word-to-node alignment.

Our SBMT system is grounded in the theory of tree transducers, which were applied to the task of semantic parsing by Jones et al. (2011; 2012b).

Semantic parsing in general and AMR parsing specifically can be considered a subsumption of many semantic resolution sub-tasks, e.g. named entity recognition (Nadeau and Sekine, 2007), semantic role labeling (Gildea and Jurafsky, 2002), word sense disambiguation (Navigli, 2009) and relation finding (Bach and Badaskar, 2007).

10 Conclusion

By restructuring our AMRs we are able to convert a sophisticated SBMT engine into a baseline semantic parser with little additional effort. By further restructuring our data to appropriately model the behavior we want to capture, we are able to rapidly achieve state-of-the-art results. Finally, by incorporating novel language models and external semantic resources, we are able to increase quality even more. This is not the last word on AMR parsing, as fortunately, machine translation technology provides more low-hanging fruit to pursue.

Acknowledgments

Thanks to Julian Schamper and Allen Schmaltz for early attempts at this problem. This work was sponsored by DARPA DEFT (FA8750-13-2-0045), DARPA BOLT (HR0011-12-C-0014), and DARPA Big Mechanism (W911NF-14-1-0364).

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria, August. Association for Computational Linguistics.

Nguyen Bach and Sameer Badaskar. 2007. A Review of Relation Extraction. Unpublished. `http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf`.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

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

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 132–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

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

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October. Association for Computational Linguistics.

Matthias Huck, Hieu Hoang, and Philipp Koehn. 2014. Augmenting string-to-tree and tree-to-string translation with non-syntactic phrases. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 486–498, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2011. Formalizing semantic parsing with tree transducers. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 19–28, Canberra, Australia, December.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012a.

Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pages 1359–1376, Mumbai, India, December. The COLING 2012 Organizing Committee.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2012b. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 488–496, Jeju Island, Korea, July. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *In Language Resources and Evaluation*.

Gregory Kuhlmann, Peter Stone, Raymond J. Mooney, and Jude W. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*, July.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended ghkm algorithm for inducing lambda-scfg. In Marie desJardins and Michael L. Littman, editors, *AAAI*. AAAI Press.

Jonathan May and Kevin Knight. 2007. Syntactic re-alignment models for machine translation. In Jason Eisner and Taku Kudo, editors, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 360–368, Prague, Czech Republic, June 28 – June 30. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October. Association for Computational Linguistics.

P. J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 91–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 141–144, Boulder, Colorado, June. Association for Computational Linguistics.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2):247–277, June.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 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*, pages 366–375, Denver, Colorado, May–June. Association for Computational Linguistics.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991, Beijing, China, July. Association for Computational Linguistics.

I.H. Witten and T.C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

John Marvin Zelle. 1995. *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. Ph.D. thesis, University of Texas at Austin, Austin, TX, USA.