Probabilistic, Structure-Aware Algorithms for Improved Variety, Accuracy, and Coverage of AMR Alignments

Austin Blodgett **Nathan Schneider**
Georgetown University
{ajb341, nathan.schneider}@georgetown.edu

Abstract

We present algorithms for aligning components of Abstract Meaning Representation (AMR) graphs to spans in English sentences. We leverage unsupervised learning in combination with heuristics, taking the best of both worlds from previous AMR aligners. Our unsupervised models, however, are more sensitive to graph substructures, without requiring a separate syntactic parse. Our approach covers a wider variety of AMR substructures than previously considered, achieves higher coverage of nodes and edges, and does so with higher accuracy. We will release our LEAMR datasets and aligner for use in research on AMR parsing, generation, and evaluation.

1 Introduction

Research with the Abstract Meaning Representation (AMR; Banarescu et al., 2013), a broad-coverage semantic annotation framework in which sentences are paired with directed acyclic graphs, must contend with the lack of gold-standard alignments between words and semantic units in the English data. A variety of rule-based and statistical algorithms have sought to fill this void, with improvements in alignment accuracy often translating into improvements in AMR parsing accuracy (Pourdamghani et al., 2014; Naseem et al., 2019; Liu et al., 2018). Yet current alignment algorithms still suffer from limited coverage and less-than-ideal accuracy, constraining the design and accuracy of parsing algorithms. Where parsers use latent alignments (e.g., Lyu and Titov, 2018; Cai and Lam, 2020), explicit alignments can still facilitate evaluation and error analysis. Moreover, AMR-to-text generation research and applications using AMR stand to benefit from accurate, human-interpretable alignments.

We present **L**inguistically **E**nriched **AMR** (LEAMR) alignment, which achieves full graph coverage via four distinct types of aligned structures: subgraphs, relations, reentrancies, and duplicate subgraphs arising from ellipsis. This formulation lends itself to unsupervised learning of alignment models. Advantages of our algorithm and released alignments include: (1) much improved coverage over previous datasets, (2) increased variety of the substructures aligned, including alignments for all relations, and alignments for diagnosing reentrancies, (3) alignments are made between spans and connected substructures of an AMR, (4) broader identification of spans including named entities and verbal and prepositional multiword expressions.

Contributions are as follows:

- A novel *all-inclusive* formulation of AMR alignment in terms of mappings between spans and connected subgraphs, including spans aligned to multiple subgraphs; mappings between spans and inter-subgraph edges; and characterization of reentrancies. Together these alignments fully cover the nodes and edges of the AMR graph (§3).
- An algorithm combining rules and EM to align English sentences to AMRs without supervision (§5), achieving higher coverage and quality than existing AMR aligners (§7).
- A corpus with automatic alignments for LDC2020 and *Little Prince* data as well as a few hundred manually annotated sentences for tuning and evaluation (§4).

We release this dataset of alignments for over 60,000 sentences along with our aligner code to facilitate more accurate models and greater interpretability in future AMR research.

2 Related Work

The main difficulty presented by AMR alignment is that it is a many-to-many mapping problem, with gold alignments often mapping multiple tokens to

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 3310–3321
August 1–6, 2021.

multiple nodes while preserving AMR structure. Previous systems use various strategies for aligning. They also have differing approaches to what types of substructures of AMR are aligned—whether they are nodes, subgraphs, or relations—and what they are aligned to—whether individual tokens, token spans, or syntactic parses. Two main alignment strategies remain dominant, though they may be combined or extended in various ways: rule-based strategies as in Flanigan et al. (2014), Flanigan et al. (2016), Liu et al. (2018), and Szubert et al. (2018), and statistical strategies using Expectation-Maximization as in Pourdamghani et al. (2014).

JAMR. The JAMR system (Flanigan et al., 2014, 2016) aligns token spans to subgraphs using iterative application of an ordered list of 14 rules which include exact and fuzzy matching. JAMR alignments form a connected subgraph of the AMR by the nature of the rules being applied. A disadvantage of JAMR is that it lacks a method for resolving ambiguities, such as repeated tokens, or of learning novel alignment patterns.

ISI. The ISI system (Pourdamghani et al., 2014) produces alignments between tokens and nodes and between tokens and relations via an Expectation-Maximization (EM) algorithm in the style of IBM Model 2 (Brown et al., 1988). First, the AMR is linearized; then EM is applied using a symmetrized scoring function of the form $P(a \mid t) + P(t \mid a)$, where a is any node or edge in the linearized AMR and t is any token in the sentence. Graph connectedness is not enforced for the elements aligning to a given token. Compared to JAMR, ISI produces more novel alignment patterns, but also struggles with rare strings such as dates and names, where a rule-based approach is more appropriate.

Extensions and Combinations. TAMR (Tuned Abstract Meaning Representation; Liu et al., 2018) uses the JAMR alignment rules, along with two others, to produce a set of candidate alignments for the sentence. Then, the alignments are "tuned" with a parser oracle to select the candidates that correspond to the oracle parse that is most similar to the gold AMR.

Some AMR parsers (Naseem et al., 2019; Fernandez Astudillo et al., 2020) use alignments which are a union of alignments produced by the JAMR and ISI systems. The unioned alignments achieve greater coverage, improving parser performance.

Syntax-based. Several alignment systems attempt to incorporate syntax into AMR alignments.

	nodes	edges	reentrancies
JAMR	91.1	✗	✗
ISI	78.7	9.8	✗
TAMR*	94.9	✗	✗

Table 1: Coverage and types of previous alignment systems. Scores are evaluated on 200 gold test sentences. *TAMR is evaluated on a subset of 91 sentences.

Chen and Palmer (2017) perform unsupervised EM alignment between AMR nodes and tokens, taking advantage of a Universal Dependencies (UD) syntactic parse as well as named entity and semantic role features. Szubert et al. (2018) and Chu and Kurohashi (2016) both produce hierachical (nested) alignments between AMR and a syntactic parse. Szubert et al. use a rule-based algorithm to align AMR subgraphs with UD subtrees. Chu and Kurohashi use a supervised algorithm to align AMR subgraphs with constituency parse subtrees.

Word Embeddings. Additionally, Anchiêta and Pardo (2020) use an alignment method designed to work well in low-resource settings using pretrained word embeddings for tokens and nodes.

Graph Distance. Wang and Xue (2017) use an HMM-based aligner to align tokens and nodes. They include in their aligner a calculation of graph distance as a locality constraint on predicted alignments. This is similar to our use of projection distance as described in §5.

Drawbacks of Current Alignments. Alignment methods vary in terms of components of the AMR that are candidates for alignment. Most systems either align nodes (e.g., ISI) or connected subgraphs (e.g., JAMR), with incomplete coverage. Most current systems do not align *relations* to tokens or spans, and those that do (such as ISI) do so with low coverage and performance. None of the current systems align reentrancies, although Szubert et al. (2020) developed a rule-based set of heuristics for identifying reentrancy types. Table 1 summarizes the coverage and variety of prominent alignment systems.

3 An All-Inclusive Formulation of AMR Alignment

Aligning AMRs to English sentences is a vexing problem not only because the English training data lacks gold alignments, but also because AMRs—unlike many semantic representations—are not *designed* with a derivational process of form–function subunits in mind. Rather, each AMR graph represents the full-sentence meaning, and AMR anno-

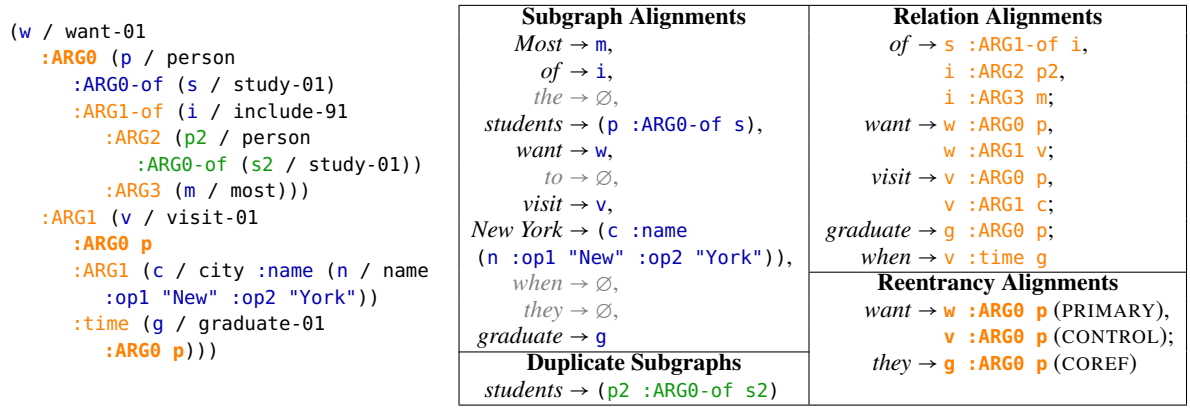

Figure 1: AMR and alignments for the sentence "*Most of the students want to visit New York when they graduate.*" Alignments are differentiated by colors: blue (subgraphs), green (duplicate subgraphs), and orange (relations). Relations that also participate in reentrancy alignments are bolded.

tation conventions can be opaque with respect to the words or surface structure of the sentence, e.g., by unifying coreferent mentions and making explicit certain elided or pragmatically inferable concepts and relations. Previous efforts toward general tools for AMR alignment have considered mapping tokens, spans, or syntactic units to nodes, edges, or subgraphs (§2). Other approaches to AMR alignment have targeted specific compositional formalisms (Groschwitz et al., 2018; Beschke, 2019; Blodgett and Schneider, 2019).

We advocate here for a definition of alignment that is *principled*—achieving full coverage of the graph structure—while being *framework-neutral* and *easy-to-understand*, by aligning graph substructures to shallow token spans on the form side, rather than using syntactic parses. We do use structural considerations to constrain alignments on the meaning side, but by using spans on the form side, we ensure the definition of the alignment search space is not at the mercy of error-prone parsers.

Definitions. Given a tokenized sentence **w** and its corresponding AMR graph $\mathcal{G}$, a complete alignment assumes a segmentation of **w** into spans **s**, each containing one or more contiguous tokens; and puts each of the nodes and edges of $\mathcal{G}$ in correspondence with some span in **s**. A span may be aligned to one or more parts of the AMR, or else is null-aligned. Individual alignments for a sentence are grouped into four **layers**: subgraph alignments, duplicate subgraph alignments, relation alignments, and reentrancy alignments. These are given for an example in figure 1.

All alignments are between a single span and a substructure of the AMR. A span may be aligned in multiple layers which are designed to capture different information. Within the subgraph layer, alignments are mutually exclusive with respect to both spans and AMR components. The same holds true within the relation layer. Every node will be aligned exactly once between the subgraph and duplicate subgraph layers. Every edge will be aligned exactly once between the subgraph and relation layers, and may additionally have a secondary alignment in the reentrancy layer.

3.1 Subgraph Layer

Alignments in this layer generally reflect the lexical semantic content of words in terms of connected,[1] directed acyclic subgraphs of the corresponding AMR. Alignments are mutually exclusive (disjoint) on both the form and meaning sides.

3.2 Duplicate Subgraph Layer

A span may be aligned to multiple subgraphs if one is a duplicate of the others, with a matching concept. This is often necessary when dealing with ellipsis constructions, where there is more semantic content in the AMR than is pronounced in the sentence and thus several identical parts of the AMR must be aligned to the same span. In this case, a single subgraph is chosen as the primary alignment (whichever is first based on depth-first order) and is aligned in the subgraph alignment layer, and any others are represented in the duplicates alignment

[1] Nodes aligned to a span must form a connected subgraph with two exceptions: (1) duplicate alignments are allowed and are separated into subgraph and duplicate layers; (2) a span may be aligned to two terminal nodes that have the same parent. For example, *never* aligns to `:polarity - :time ever`, two nodes and two edges which share the same parent.

layer. For example, verb phrase ellipsis, as in *I swim and so do you*, would involve duplication of the predicate *swim*, with distinct ARG0s. Similarly, in figure 1, *Most of the students* involves a subset-superset structure where the subset and superset correspond to separate nodes. Because *student* is represented in AMR like *person who studies*, there are two 2-node subgraphs aligned to *student*, one with the variables p and s, and the duplicate with p2 and s2. The difficulty that duplicate subgraphs pose for parsing and generation makes it convenient to put these alignments in a separate layer.

3.3 Relation Layer

This layer includes alignments between a span and a single relation—such as *when* → :time—and alignments mapping a span to its argument structure—such as *give* → :ARG0 :ARG1 :ARG2. All edges in an AMR that are not contained in a subgraph fit into one of these two categories.

English function words such as prepositions and subordinators typically function as connectives between two semantically related words or phrases, and can often be identified with the semantics of AMR relations. But many of these function words are highly ambiguous. Relation alignments make their contribution explicit. For example, *when* in figure 1 aligns to a :time relation.

For spans that are aligned to a subgraph, incoming or outgoing edges attached to that subgraph may also be aligned to the span in the relation layer. These can include core or non-core roles as long as they are evoked by the token span. For example, figure 1 contains *visit* → :ARG0 :ARG1.

3.4 Reentrancy Layer

A **reentrant** node is one with multiple incoming edges. In figure 1, for example, p appears three times: once as the ARG0 of w (the wanter), once as the ARG0 of v (the visitor), and once as the ARG0 of g (the graduate). The p node is labeled with the concept person—in the PENMAN notation used by annotators, each variable's concept is only designated on one occurrence of the variable, the choice of occurrence being, in principle, arbitrary. These three ARG0 relations are aligned to their respective predicates in the relation layer. But there are many different causes of reentrancy, and AMR parsers stand to benefit from additional information about the nature of each reentrant edge, such as the fact that the pronoun *they* is associated with one of the ARG0 relations.

The reentrancy layer "explains" the cause of each reentrancy as follows: for the incoming edges of a reentrant node, one of these edges is designated as PRIMARY—this is usually the first mention of the entity in a local surface syntactic attachment, e.g. the argument of a control predicate like *want* doubles as an argument of an embedded clause predicate. The remaining incoming edges to a reentrant node are aligned to a **reentrancy trigger** and labeled with one of 8 **reentrancy types**: *coref, repetition, coordination, control, adjunct control, unmarked adjunct control, comparative control,* and *pragmatic*. These are illustrated in table 2. These types, adapted from Szubert et al.'s (2020) classification, correspond to different linguistic phenomena leading to AMR reentrancies—anaphoric and non-anaphoric coreference, coordination, control, etc. The trigger is the word that most directly signals the reentrancy phenomenon in question. For the example in figure 1, the control verb *want* is aligned to the embedded predicate–argument relation and typed as CONTROL, while the pronoun *they* serves as the trigger for the third instance of p in *when they graduate*.

3.5 Validation

To validate the annotation scheme we elicited two gold-standard annotations for 40 of the test sentences described in §4 and measured interannotator agreement.[2] Interannotator exact-match F1 scores were 94.54 for subgraphs, 90.73 for relations, 76.92 for reentrancies, and 66.67 for duplicate subgraphs (details in appendix A).

4 Released Data

We release a dataset[3] of the four alignment layers reflecting correpondences between English text and various linguistic phenomena in gold AMR graphs—subgraphs, relations (including argument structures), reentrancies (including coreference, control, etc.), and duplicate subgraphs.

Automatic alignments cover the ≈60,000 sentences of the LDC2020T02 dataset (Knight et al., 2020) and ≈1,500 sentences of *The Little Prince*.

We manually created **gold alignments** for evaluating our automatic aligner, split into a development set (150 sentences) and a test set (200 sen-

[2] Both annotators are Ph.D. students with backgrounds in linguistics. One annotator aligned all development and test sentences; the other aligned a subset of 40 test sentences.

[3] https://github.com/ablodge/leamr

Type	Triggered by	*Example*
COREF	a pronoun (including possessive or reflexive) (anaphora)	*I love my* ***house***
REPETITION	a repeated name or non-pronominal phrase (non-anaphoric coreference)	*The U.S. promotes American* ***goods***
COORDINATION	coordination of two or more phrases sharing an argument	*They cheered and* ***celebrated***
CONTROL	control verbs, control nouns, or control adjectives	*I was afraid to* ***speak up***
ADJUNCT CONTROL	control within an adjunct phrase	*I left to* ***buy*** *some milk; Mary cooked while* ***listening*** *to music*
UNMARKED ADJUNCT CONTROL	control within an adjunct phrase with only a bare verb and no subordinating conjunction	*Mary did her homework* ***listening*** *to music*
COMPARATIVE CONTROL	a comparative construction	*Be as* ***objective*** *as possible*
PRAGMATIC	Reentrancies that must be resolved using context	*John met up with a* ***friend***

Table 2: Reentrancy types with examples. For each reentrant node, one of its incoming edges is labeled PRIMARY and the others are labeled with one of the above reentrancy types. In the examples, the word aligned to an edge labeled with the specified type is underlined, and the word aligned to the parent of that edge is bolded.

```
(h / have-degree-91
    :ARG1 (h2 / house :location (l / left))
    :ARG2 (b / big)
    :ARG3 (m / more)
    :ARG4 (h3 / house :location (r / right)))
```

Figure 2: AMR for the sentence "The house$_1$ on the left is bigger than the house$_2$ on the right."

tences).[4] The test sentences were annotated from scratch; the development sentences were first automatically aligned and then hand-corrected. We stress that no preprocessing apart from tokenization is required to prepare the test sentences and AMRs for human annotation. We also release our annotation guidelines as a part of our data release.

5 LEAMR Aligner

We formulate statistical models for the alignment layers described above—**subgraphs**, **duplicate subgraphs**, **relations**, and **reentrancies**—and use the Expectation-Maximization (EM) algorithm to estimate probability distributions without supervision, with a decoding procedure that constrains aligned units to obey structural requirements. In line with Flanigan et al. (2014, 2016), we use rule-based preprocessing to align some substructures using string-matching, morphological features, etc.

Before delving into the models and algorithm, we motivate two important characteristics:

Structure-Preserving. Constraints on legal candidates during alignment ensure that at any point only connected substructures may be aligned to a span. Thus, while our aligner is probabilistic like the ISI aligner, it has the advantage of preserving the AMR graph structure.

Projection Distance. The scores calculated for an alignment take into account a distance metric designed to encourage locality—tokens that are close together in a sentence are aligned to subtructures that are close together in the AMR graph. We define the *projection distance dist*(`n1`, `n2`) between two neighboring nodes `n1` and `n2` to be the signed distance in the corresponding sentence between the span aligned to `n1` and the span aligned to `n2`. This motivates the model to prefer alignments whose spans are close together when aligning nodes which are close together—particularly useful when a word occurs twice with identical subgraphs. Thus, our aligner relies on more information from the AMR graph structure than other aligners (note that the ISI system linearizes the graph). Further details are given in §5.2.

5.1 Overview

Algorithm 1 illustrates our base algorithm in pseudocode. The likelihood for a sentence can be expressed as a sum of per-span alignment scores: we write the score of a full set of a sentence's subgraph alignments $\mathcal{A}$ as

$$Score(\mathcal{A} \mid \mathcal{G}, \mathbf{w}) = \prod_{i=1}^{N} score(\langle \mathbf{g}_i, s_i \rangle \mid \mathcal{G}, \mathbf{w}) \quad (1)$$

where $\mathbf{s}$ are N aligned spans in the sentence $\mathbf{w}$, and $\mathbf{g}$ are sets of subgraphs of the AMR graph $\mathcal{G}$ aligned to each span. For relations model and the reentrancies model, each $\mathbf{g}_i$ consists of relations rather than

[4] Our test set consists of sentences from the test set of Szubert et al. (2018) but with AMRs updated to the latest release version. This test set contains a mix of English sentences drawn from the LDC data and *The Little Prince*—some sampled randomly, others hand-selected—as well as several sentences constructed to illustrate particular phenomena.

subgraphs. Henceforth we assume all alignment scores are conditioned on the sentence and graph and omit $\mathbf{w}$ and $\mathcal{G}$ for brevity. The $score(\cdot)$ component of eq. (1) is calculated differently for each of the three models detailed below.

Alignment Pipeline. Alignment proceeds in the following phases, with each phase depending on the output of the previous phase:

1. *Preprocessing:* Using external tools we extract lemmas, parts of speech, and coreference.
2. *Span Segmentation:* Tokens are grouped into spans using a rule-based procedure (appendix B).
3. *Align Subgraphs & Duplicate Subgraphs:* We greedily identify subgraph and duplicate subgraph alignments in the same alignment phase (§5.2).
4. *Align Relations:* Relations not belonging to a subgraph are greedily aligned in this phase, using POS criteria to identify legal candidates (§5.3).
5. *Align Reentrancies:* Reentrancies are aligned in this phase, using POS and coreference in criteria for identifying legal candidates (§5.4).

The three main alignment phases use different models with different parameters; they also have their own preprocessing rules used to identify some alignments heuristically (appendices C to E).[5] In training, parameters for each phase are iteratively learned and used to align the entire training set by running EM to convergence before moving on to the next phase. At test time, the pipeline can be run sentence-by-sentence.

Decoding. The three main alignment phases all use essentially the same greedy, substructure-aware search procedure. This searches over node–span candidate pairs based on the scoring function modeling the compatibility between a subgraph (or relation) g and span s, which we denote $score(\langle g, s\rangle)$. For each unaligned node (or edge), we identify a set of legal candidate alignments using phase-specific criteria. The incremental score improvement of adding each candidate—either extending a subgraph/set of relations already aligned to the span, or adding a completely new alignment—is calculated as as $\Delta score = score(\langle g_0 \cup \{n\}, s\rangle) - score(\langle g_0, s\rangle)$, where g_0 is the current aligned subgraph, s is the span, and n is an AMR component being considered. Of the candidates for all unaligned nodes, the node–span pair giving the best score improvement is then greedily selected to add to the alignment. This is repeated until all nodes have been aligned (even if the last ones decrease the score). The procedure is detailed in algorithm 1 for subgraphs; the relations phase and the reentrancies phase use different candidates (respectively: unaligned edges; reentrant edges), different criteria for legal candidates, and different scoring functions.

[5] 79% of nodes and 89% of edges are aligned by rules. We believe this is why in practice, EM performs well without random restarts.

5.2 Aligning Subgraphs

The score assigned to an alignment between a span and subgraph is calculated as $score(\langle g, s\rangle) =$

$$P_{\text{align}}(g \mid s; \theta_1) \cdot \prod_{d_i \in D} P_{\text{dist}}(d_i; \theta_2)^{\frac{1}{|D|}} \cdot IB(g, s) \quad (2)$$

where g is a subgraph, s is a span, d_i is the projection distance of g with its ith neighboring node, and θ_1 and θ_2 are model parameters which are updated after each iteration. The subgraph g is represented in the model as a bag of concept labels and (parent concept, relation, child concept) triples.

The distributions P_{align} and P_{dist} are inspired by IBM Model 2 (Brown et al., 1988), and can be thought of as graph-theoretic extensions of translation (align) and alignment (dist) probabilities. *IB* stands for *inductive bias*, explained below.

Legal Candidates. For each unaligned node n, the model calculates a score for spans of three possible categories: 1) unaligned spans; 2) spans aligned to a neighboring node (in this case, the aligner considers adding n to an existing subgraph if the resulting subgraph would be connected); 3) spans aligned to a node with the same concept as n (this allows the aligner to identify duplicate subgraphs—candidates in this category receive a score penalty because duplicates are quite rare, so they are generally the option of last resort).

Limiting the candidate spans in this way ensures only connected, plausible substructures of the AMR are aligned. To form a multinode subgraph alignment $t_1 \rightarrow$ `n1 :rel n2`, the aligner could first align `n1` to an unaligned span t_1, then add `n2`, which is a legal candidate because t_1 is aligned to a neighboring node of `n2` (ensuring a connected subgraph).

Distance. We model the probability of the projection distance $P_{\text{dist}}(d; \theta_2)$ using a Skellam distribution, which is the difference of two Poisson distributed random variables $D = N_1 - N_2$ and can be positive or negative valued. Parameters are updated based on alignments in the previous iteration. For each aligned neighbor n_i of a subgraph g, we calculate $P_{\text{dist}}(dist(g, n_i); \theta_2)$ and take the geometric mean of probabilities as P_{dist}.

Algorithm 1 Procedure for greedily aligning all nodes to spans using a scoring function that decomposes over (span, subgraph) pairs. (Scores are expressed in real space but the implementation is in log space.)

```
function AlignSubgraphs(spans, amr)
    alignments ← dict()                                  ▷map from span to an ordered list of aligned subgraphs
    unaligned_nodes ← get_unaligned_nodes(amr, alignments)
    while |unaligned_nodes| > 0 do
        Δscores ← []
        candidate_s_g_pairs ← []
        for n ∈ unaligned_nodes do
            candidate_spans ← get_legal_alignments(n, alignments)
            for span, i_subgraph ∈ candidate_spans do    ▷either there is an edge between n and the indicated subgraph
already aligned to span, or i_subgraph would be a new subgraph consisting of n
                current_aligned_nodes ← alignments[span][i_subgraph]        ▷∅ if this would be a new subgraph
                new_aligned_nodes ← current_aligned_nodes ∪ {n}
                Δscore ← get_score(span, new_aligned_nodes, alignments)
                       − get_score(span, current_aligned_nodes, alignments)   ▷change from adding n into a subgraph
aligned to span; get_score queries score(⟨g,s⟩) and multiplies λ_dup if i_subgraph > 1
                Δscores.add(Δscore)
                candidate_s_g_pairs.add((span, new_aligned_nodes, i_subgraph))
        span*, subgraph*, i_subgraph* ← candidate_s_g_pairs[argmax(Δscores)]  ▷update having the best impact on score
(equivalently, maximizing sum of scores across individual aligned spans)
        alignments[span*][i_subgraph*] ← subgraph*
        unaligned_nodes ← get_unaligned_nodes(amr, alignments)
    return alignments
```

Null alignment. The aligner models the possibility of a span being unaligned using a fixed heuristic:

$$P_{\text{align}}(\varnothing \mid s) = \max\{rank(s)^{-\frac{1}{2}}, 0.01\} \quad (3)$$

where *rank* assigns 1 to the most frequent word, 2 to the 2nd most frequent, etc. Thus, the model expects that very common words are more likely to be null-aligned and rare words should almost always be aligned.[6]

Factorized Backoff. So that the aligner generalizes to unseen subgraph–span pairs, where $P_{\text{align}}(g \mid s) = 0$, we use a backoff factorization into components of the subgraph. In particular, the factors are empirical probabilities of (i) an AMR concept given a span string in the sentence, and (ii) a relation and child node concept given the parent node concept and span string. These cooccurrence probabilities $\hat{p}$ are estimated directly from the training sentence/AMR pairs (irrespective of latent alignments). The product is scaled by a factor λ. E.g., for a subgraph `n1 :rel1 n2 :rel2 n3`, where $\texttt{c}_\texttt{n}$ is the concept of node `n`, we have

$$P_{\text{factorized}}(g \mid s) = \lambda \cdot \hat{p}(\texttt{c}_{\texttt{n1}} \mid s) \cdot \hat{p}(\texttt{:rel1}, \texttt{c}_{\texttt{n2}} \mid \texttt{c}_{\texttt{n1}}, s) \cdot \hat{p}(\texttt{:rel2}, \texttt{c}_{\texttt{n3}} \mid \texttt{c}_{\texttt{n1}}, s) \quad (4)$$

Inductive bias. Lastly, to encourage good initialization, the score function includes an inductive bias which does not depend on EM-trained parameters. This inductive bias is based on the empirical probability of a node occurring in the same AMR with a span in the training data. We calculate inductive bias as an average of exponentiated PMIs $\frac{1}{N}\sum_i \exp(PMI(n_i, s))$, where N is the number of nodes in g, n_i is the ith node contained in the subgraph, and PMI is the PMI of n_i and s.

Aligning Duplicate Subgraphs. On rare occasion a span should be aligned to multiple subgraphs (§3.2). To encourage the model to align a different span where possible, there is a constant penalty λ_{dup} for each additional subgraph aligned to a span beyond the first. Thus the score for a span and its subgraphs is computed as:

$$score(\langle \mathbf{g}, s \rangle) = \lambda_{\text{dup}}^{|\mathbf{g}|-1} \prod_{g \in \mathbf{g}} score(\langle g, s \rangle) \quad (5)$$

5.3 Aligning Relations

For a given relation alignment between a span and a collection of edges, we calculate a score as follows:

$$score(\langle a, s \rangle) = P_{\text{align}}(a \mid s; \theta_3) \cdot \prod_{d_i \in D_1} P_{\text{dist}}(d_i; \theta_4)^{\frac{1}{|D_1|}} \cdot \prod_{d_j \in D_2} P_{\text{dist}}(d_j; \theta_5)^{\frac{1}{|D_2|}} \quad (6)$$

where a is the argument structure (the collection of aligned edges), s is a span, D_1 is the projection distances of each edge and its parent, and D_2 is

[6] We allow several exceptions. For punctuation, words in parentheses, and spans that are coreferent to another span, the probability is 0.5. For repeated spans, the probability is 0.1.

	Exact Align			Partial Align			Spans	Coverage
	P	R	F1	P	R	F1	F1	
Subgraph Alignments (N = 1707)								
Our system	93.91	94.02	93.97	95.69	95.81	95.75	96.05	100.0
JAMR	87.21	83.06	85.09	90.29	85.99	88.09	92.38	91.1
ISI	71.56	68.24	69.86	78.03	74.54	76.24	86.59	78.7
TAMR (91 sentences)	85.68	83.38	84.51	88.62	86.24	87.41	93.64	94.9
Relation Alignments (N = 1263)								
Our system	85.67	85.37	85.52	88.74	88.44	88.59	95.41	100.0
ISI	59.28	8.51	14.89	66.32	9.52	16.65	83.09	9.8
Reentrancy Alignments (N = 293)								
Ours (labeled)	55.75	54.61	55.17	-	-	-	-	100.0
Ours (unlabeled)	62.72	61.43	62.07	-	-	-	-	100.0
Duplicate Subgraph Alignments (N = 17)								
Our system	66.67	58.82	62.50	70.00	61.76	65.62	-	100.0

Table 3: Main results on the test set. N represents the denominator of exact alignment recall. There are 2860 gold spans in total, 41% of which are null-aligned and 0.6% of which are aligned to multiple subgraphs. 95% of the spans consist of a single token, and 49% of spans are aligned to a single subgraph consisting of a single node.

the projection distances of each edge and its child. The collection of edges a is given a normalized label which represents the relations contained in the alignment (distinguishing incoming versus outgoing relations, and normalizing inverse edges).

Legal Candidates. There are two kinds of candidate spans for relation alignment. First, previously unaligned spans[7] (with no relation or subgraph alignments), e.g. prepositions and subordinating conjunctions such as *in* → `:location` or *when* → `:time`. Second, any spans aligned to the relation's parent or child in the subgraph layer: this facilitates alignment of argument structures such as *give* → `:ARG0 :ARG1 :ARG2`. Additionally, we constrain certain types of edges to only align with the parent and others to only align with the child.

Distance. For relations there are potentially two distances of interest—the projected distance of the relation from its parent and the projected distance of the relation from its child. We model these separately as *parent distance* and *child distance* with distinct parameters. To see why this is useful, consider the sentence "Should we meet at the restaurant or at the office?", where each *at* token should be aligned to a `:location` edge. In English, prepositions like *at* precede an object and follow a governor. Thus parent distance tends to be to the left (negative valued) while child distance tends to be to the right (positive valued).

[7] We constrain these to particular parts of speech: prepositions (IN), infinitival to (TO), possessives (POS), and possessive pronouns (PRP$). Additionally, only spans that are between the spans aligned to the parent and any descendent of child nodes of the relation (and are not between the child's aligned span and any of its descendants' spans) are allowed. This works well in practice for English.

5.4 Aligning Reentrancies

The probability of a reentrancy alignment is similar to eq. (6), but with an extra variable for the reentrancy type: $score(\langle r, s, type \rangle)$ =

$$P_{\text{align}}(r, type \mid s; \theta_6) \cdot P_{\text{dist}}(d_1; \theta_7) \cdot P_{\text{dist}}(d_2; \theta_8) \quad (7)$$

where r is the role label of the reentrant edge.

Legal Candidates. There are 8 reentrancy types (§3.4). For each type, a rule-based test determines if a span and edge are permitted to be aligned. The 8 tests use part of speech, the structure of the AMR, and subgraph and relation alignments. A span may be aligned (rarely) to multiple reentrancies, but these alignments are scored separately.

6 Experimental Setup

Sentences are preprocessed with the Stanza library (Qi et al., 2020) to obtain lemmas, part-of-speech tags, and named entities. We identify token spans using a combination of named entities and a fixed list of multiword expressions (details are given in appendix B). Coreference information, which is used to identify legal candidates in the reentrancy alignment phase, is obtained using NeuralCoref.[8] Lemmas are used in each alignment phase to normalize representation of spans, while parts of speech and coreference are used to restrict legal candidates in the relation and reentrancy alignment phases. We tune hyperparameters, including penalties for duplicate alignments and our factorized backoff probability, on the development set.

[8] https://github.com/huggingface/neuralcoref

	Exact Align		
	P	R	F1
Relation Alignments Breakdown			
Our system: all (1163)	85.67	85.37	85.52
...single relations (121)	53.49	56.56	54.98
...argument structures (1042)	89.67	88.73	89.20
ISI: all (1163)	59.28	8.51	14.89
...single relations (121)	82.89	52.07	63.96
...argument structures (1042)	39.56	3.45	6.35
Reentrancy Alignments Breakdown			
Our system: all (293)	62.37	61.09	61.72
...primary (128)	79.37	78.12	78.74
...coref (41)	57.14	58.54	57.83
...control (36)	73.08	52.78	61.29
...coordination (29)	57.14	58.54	57.83
...pragmatic (25)	20.93	36.00	26.47
...adjunct control (15)	100.00	6.67	12.50
...repetition (13)	60.00	46.15	52.17
...comparative control (5)	0.0	0.0	0.0
...unmarked adjunct control (1)	0.0	0.0	0.0

Table 4: Detailed results for relation alignments and reentrancy alignments.

7 Results

Table 3 describes our main results on the 200-sentence test set (§4), reporting exact-match and partial-match alignment scores as well as span identification F1 and coverage.[9] The partial alignment evaluation metric is designed to be more forgiving of arbitrary or slight differences between alignment systems. We argue that this metric is more comparable across alignment systems. It assigns partial credit equal to the product of Jaccard indices $\frac{|N_1 \cap N_2|}{|N_1 \cup N_2|} \cdot \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$ for nodes (or edges) and tokens respectively. This partial credit is calculated for each gold alignment and the closest matching predicted alignment with nodes (or edges) N_1 and N_2 and tokens T_1 and T_2. Coverage is the percentage of relevant AMR components that are aligned.

Our aligner shows improvements over previous aligners in terms of coverage and accuracy even when using a partial credit metric for evaluation. We demonstrate greater coverage, including coverage of phenomena not aligned by previous systems.

Table 4 shows detailed results for relation subtypes and reentrancy subtypes. Here, we see room for improvement. In particular, ISI outperforms our system at aligning single relations. Our reentrancy aligner lacks a baseline to compare to, but the breakdown of results by type suggest there are several categories of reentrancies where scores could be improved.

Qualitative Analysis. A number of errors from our subgraph aligner resulted from unseen multiword expressions in our test data that our span preprocessing failed to recognize and our aligner failed to align. For example, the expression "on the one hand" appears in test and should be aligned to `contrast-01`. The JAMR aligner suffers without a locality bias; we notice several cases where it misaligns words that are repeated in the sentence. The ISI aligner generally does not align very frequent nodes such as `person`, `thing`, `country`, or `name`, resulting in generally lower coverage. It also frequently aligns disconnected nodes with the same concept to one token instead of separate tokens. While our relation aligner yields significantly higher coverage, we do observe that the model is overeager to align relations to extremely frequent prepositions (such as *to* and *of*), resulting in lower precision of single relations in particular.

[9] A previous draft of this work reported lower scores on relations before a constraint was added to improve the legal candidates for relation alignment.

Ablations	Exact Align		
	P	R	F1
Subgraphs	93.91	94.02	93.97
Subgraphs (−distance)	92.69	92.85	92.77
Subgraphs (−inductive bias)	93.88	93.44	93.66
Relations	85.67	85.37	85.52
Relations (−distance)	85.14	84.77	84.95
Relations (gold subgraphs)	91.21	90.59	90.90

Table 5: Results when the aligner is trained without projection distance probabilities (−distance) and without the subgraph inductive bias (−inductive bias), as well as a relation aligner with access to gold (instead of trained) subgraphs.

Ablations. Table 5 shows that projection distance is valuable, adding 1.20 points (exact align F1) for subgraph alignment and 0.57 points for relation alignment. Despite showing anecdotal benefits in early experiments, the inductive bias does not aid the model in a statistically significant way. Using gold subgraphs for relation alignment produces an improvement of over 5 points, indicating the scope of error propagation for the relation aligner.

8 Conclusions

We demonstrate structure-aware AMR aligners that combine the best parts of rule-based and statistical methods for AMR alignment. We improve on previous systems in terms of accuracy and particularly in terms of alignment coverage and variety of AMR components to be aligned.

Acknowledgments

We thank reviewers for their thoughtful feedback, Jakob Prange for assisting with annotation, and members of the NERT lab for their support.

References

Rafael Anchiêta and Thiago Pardo. 2020. Semantically inspired AMR alignment for the Portuguese language. In *Proc. of EMNLP*, pages 1595–1600, Online.

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 *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Sebastian Beschke. 2019. Exploring graph-algebraic CCG combinators for syntactic-semantic AMR parsing. In *Proc. of RANLP*, pages 112–121, Varna, Bulgaria.

Austin Blodgett and Nathan Schneider. 2019. An improved approach for semantic graph composition with CCG. In *Proc. of the 13th International Conference on Computational Semantics - Long Papers*, pages 55–70, Gothenburg, Sweden.

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. 1988. A statistical approach to language translation. In *Proc. of COLING*, pages 71–76, Budapest, Hungary.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proc. of ACL*, pages 1290–1301, Online.

Wei-Te Chen and Martha Palmer. 2017. Unsupervised AMR-dependency parse alignment. In *Proc. of EACL*, pages 558–567, Valencia, Spain.

Chenhui Chu and Sadao Kurohashi. 2016. Supervised syntax-based alignment between English sentences and Abstract Meaning Representation graphs. *arXiv:1606.02126 [cs]*.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 Task 8: Graph-based AMR parsing with infinite ramp loss. In *Proc. of SemEval*, pages 1202–1206, San Diego, California.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proc. of ACL*, pages 1426–1436, Baltimore, Maryland, USA.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proc. of ACL*, pages 1831–1841, Melbourne, Australia.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Tim O'Gorman, Martha Palmer, Nathan Schneider, and Madalina Bardocz. 2020. Abstract Meaning Representation (AMR) Annotation Release 3.0. Technical Report LDC2020T02, Linguistic Data Consortium, Philadelphia, PA.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proc. of EMNLP*, pages 2422–2430, Brussels, Belgium.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proc. of ACL*, pages 397–407, Melbourne, Australia.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding smatch: transition-based AMR parsing with reinforcement learning. In *Proc. of ACL*, pages 4586–4592, Florence, Italy.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with Abstract Meaning Representation graphs. In *Proc. of EMNLP*, pages 425–429, Doha, Qatar.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proc. of ACL*, pages 185–196, Melbourne, Australia.

Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *Proc. of NAACL-HLT*, pages 1537–1547, Denver, Colorado.

Ida Szubert, Marco Damonte, Shay B. Cohen, and Mark Steedman. 2020. The role of reentrancies in Abstract Meaning Representation parsing. In *Proc. of Findings of EMNLP*, pages 2198–2207, Online.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for English-AMR alignment. In *Proc. of NAACL-HLT*, pages 1169–1180, New Orleans, Louisiana.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of AMR Parsing. In *Proc. of EMNLP*, pages 1257–1268, Copenhagen, Denmark.

A Interannotator Agreement

Table 6 illustrates interannotator agreement for each of the four alignment layers.

B Identifying Spans

As a preprocessing step, sentences have their tokens grouped into spans based on three criteria, outlined in detail below:

1. Named entity spans identified by Stanza.
2. Spans matching multiword expressions from a fixed list of ≈1600
   (a) 143 prepositional MWEs from STREUSLE (Schneider and Smith, 2015; Schneider et al., 2018)
   (b) 348 verbal MWEs from STREUSLE
   (c) 1095 MWEs taken from gold AMRs in LDC train data (any concept which is a hyphenated compound of multiple words, e.g., *alma-mater* or *white-collar*) and are not present in the above lists.
   (d) ≈12 hand-added MWEs
3. Any sequence of tokens which is an exact match to a name in the gold AMR (e.g., "United Kingdom" and `(n/name :op1 "United" :op2 "Kingdom"))` is also treated as a span.

C Rule-based Subgraph Alignment Preprocessing

C.1 Token matching

We use three phases of rule-based alignment which attempt to align particular spans to particular AMR subgraphs:

1. Exact token matching: If there is a unique full string correspondence between a span and a name or number in the AMR, they are aligned.
2. Exact lemma matching: If there is a unique correspondence between an AMR concept and the lemma of a span (which in the case of a multiword span is the sequence of lemmas of the tokens joined by hyphens), they are aligned.
3. Prefix token matching: A span with a prefix match of length 6, 5, or 4 is aligned if it uniquely corresponds to an AMR named entity.
4. Prefix lemma matching: A span with a prefix match of length 6, 5, or 4 of its lemma is aligned if it uniquely corresponds to an concept.
5. English rules: Several hand-written rules for matching English strings to specific subgraphs are used to match constructions such as dates, currency, and some frequent AMR concepts with many different ways of being expressed, such as `and` and `-`.
   - Parsing dates and times
   - Numbers written out (e.g., *one*, *two*, *thousand*, etc.)
   - Currencies (e.g., *$*, *€*, etc.)
   - Decades (e.g., *twenties*, *nineties*)
   - `and` (matching *and*, *additionally*, *as well*, etc.)
   - `multi-sentence` (matching punctuation)
   - `:polarity -` (matching *not*, *none*, *never*, etc.)
   - `cause-01` (matching *thus*, *since*, *because*, etc.)
   - `amr-unknown` (matching *?*, *who*, *when*, etc.)
   - `person` (matching *people*)
   - `rate-entity-91` (matching *daily*, *weekly*, etc.)
   - `"United" "States"` (matching *US*, *U.S.*, *American*, etc.)
   - `include-91` (matching *out of*, *include*, etc.)
   - `instead-of-91` (matching *instead*, etc.)
   - `have-03` (matching *have*, *'s*, etc.)
   - `mean-01` (matching *:* and *,*)
   - *how* (matching `:manner thing` or `:degree so`)
   - *as... as* (matching `equal`)

C.2 Graph rules

We also perform preprocessing to expand a subgraph alignment to include some neighboring nodes. These fall into two main categories:

1. Some AMR concepts are primarily notational rather than linguistic and should be aligned together with a neighboring node. For example named entities (e.g., `(country :name (n/name :op1 :United" :op2 "Kingdom")))` are aligned as a unit rather than one node at a time. Likewise, date entities, and subgraphs matching `(x/X-quantity :unit X :quant X)` or `(x/X-entity :value X)` are also aligned as a unit.
2. Neighboring nodes which are associated with morphological information of the aligned span (e.g., *biggest* → `(have-degree-91 :ARG1 big :ARG2 most))` are added to the alignment using a series of rules for identifying comparatives, superlatives, polarity, and suffixes such as *-er* or *-able*, etc.

D Rule-based Relation Alignment Preprocessing

Many of the relations are forced to be aligned in a particular way as a matter of convention. We use a similar approach to that of (Groschwitz et al.,

IAA	Exact Align			Partial Align			Spans
	P	R	F1	P	R	F1	F1
Subgraphs (366)	94.54	94.54	94.54	95.56	95.56	95.56	94.97
Relations (260)	91.09	90.38	90.73	93.38	92.66	93.02	93.75
Reentrancies (65)	76.92	76.92	76.92	90.00	90.00	90.00	90.77
Duplicates (5)	75.00	60.00	66.67	79.17	63.33	70.37	66.67

Table 6: Interannotator Agreement for **subgraph**, **relation**, **reentrancy**, and **duplicate subgraph layers** of alignment scored on a sample of 40 sentences of the gold test data.

2018).

1. `:ARGX` edges are automatically aligned to the same span as the parent (`:ARGX-of` edges are automatically aligned to the child).
2. `:opX` edges are automatically aligned with the parent.
3. `:sntX` edges are automatically aligned with the parent.
4. `:domain` edges are automatically aligned with the parent. (We don't align these edges to copula. Instead, a concept with a `:domain` edge is thought of as a predicate which takes one argument.)
5. `:name`, `:polarity`, and `:li` edges are automatically aligned with the child.

D.1 Token matching

Some relations take the form `:prep-X` or `:conj-X` where `X` is a preposition or conjunction in the sentence. We use exact match to align these relations as a preprocessing step. The relations `:poss` and `:part` may be automatically aligned to *'s* or *of* if the correspondence is unique within a sentence.

E Rule-based Reentrancy Alignment Preprocessing

Primary edges are identified as a preprocessing step before aligning reentrancies with the following rules: Any relation which is aligned to the same span as its token (any incoming edge which is a part of a span's argument structure) is automatically made the primary edge. Otherwise, for each edge pointing to a node, we identify the spans aligned to the parent and child nodes in the subgraph layer. Whichever edge has the shortest distance between the span aligned to the parent and the span aligned to the child is identified as the primary edge. In the event of a tie, the edge whose parent is aligned to the leftmost span is identified as the primary edge. Primary reentrancy edges are always aligned to the same span the edge is aligned to in the relation layer of alignments.