DocAMR: Multi-Sentence AMR Representation and Evaluation

Tahira Naseem[♣,*] **Austin Blodgett**[♠] **Sadhana Kumaravel**[♣]
Tim O'Gorman[♡] **Young-Suk Lee**[♣] **Jeffrey Flanigan**[◇] **Ramón Fernandez Astudillo**[♣]
Radu Florian[♣] **Salim Roukos**[♣] **Nathan Schneider**[△]
[*]tnaseem@us.ibm.com
[♣]IBM Research AI [♠]U.S. Army Research Laboratory, Adelphi, MD
[♡]Thorn [◇]University of California, Santa Cruz [△]Georgetown University

Abstract

Despite extensive research on parsing of English *sentences* into Abstract Meaning Representation (AMR) graphs, *full-document* parsing into a unified graph representation lacks well-defined representation and evaluation. Taking advantage of a super-sentential level of AMR coreference annotation from previous work, we introduce a simple algorithm for deriving a unified graph representation, avoiding the pitfalls of information loss from over-merging and lack of coherence from under-merging. Next, we describe improvements to the Smatch metric to make it tractable for comparing document-level graphs, and use it to re-evaluate the best published document-level AMR parser. We also present a pipeline approach combining the top performing AMR parser and coreference resolution systems, providing a strong baseline for future research.

1 Introduction

Abstract Meaning Representation (AMR) is a formalism that represents the meaning of a text in the form of a directed graph, where nodes represent concepts and edges are labeled with relations (Banarescu et al., 2013). Until recently, the annotated corpora for AMR only provided sentence-level annotations. As a result, AMR parsing research has been limited to sentence-level parsing. A unified document-level AMR graph representation can be quite useful for applications that require document-level understanding, such as question answering and summarization.

The most recent release of AMR annotations (AMR 3.0) includes a multi-sentence AMR corpus (MS-AMR; O'Gorman et al., 2018) that provides cross-sentential coreference information for AMR graphs. These annotations connect multiple sentence-level graphs via coreference chains, implicit relations and bridging relations. However, an adequate representation of this multi-sentence information in the form of an AMR graph has yet to be decided upon. The initial proposal (O'Gorman et al., 2018), to merge all nodes in a coreference chain into one node, suffers from significant information loss. Without a consistent multi-sentence graph representation, AMR's standard Smatch metric (Cai and Knight, 2013) cannot be used for evaluation and comparison. The Smatch scores differ greatly depending on how the coreference information is added to the AMR graphs. Additionally, the Smatch algorithm is exceedingly slow in its original form when run over multi-sentence graphs. These limitations in multi-sentence AMR graph representation and evaluation get in the way of research efforts in this area. Without a consistent representation and evaluation mechanism, it is impossible to draw meaningful comparisons between different approaches on the task. In this work, we present a simple and non-lossy multi-sentence AMR graph representation as well as a modification of the Smatch algorithm that allows efficient and consistent evaluation.[1]

Our chief contributions are:

- A standard for merging sentence-level AMR graphs into a single document-level graph based on MS-AMR coreference annotations (DocAMR; §3.3)
- A faster implementation of the Smatch metric suitable for evaluating document-level AMR parsers (§4.1)
- A metric, based on Smatch, for specific evaluation of coreference in document-level AMR graphs (§4.3)
- A baseline system for document-level AMR parsing and its evaluation (§5)

[1]The code for producing the proposed representation and efficient Smatch evaluation under Apache License 2.0 is available at: https://github.com/IBM/docAMR

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3496 - 3505

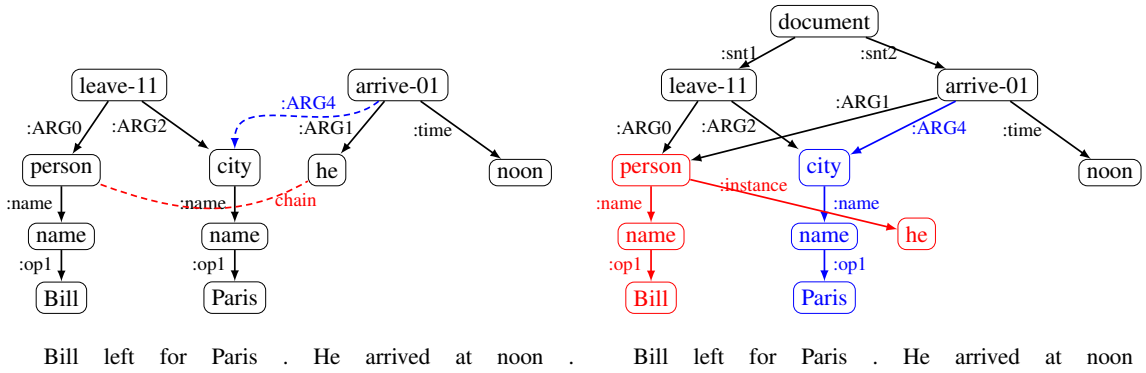

Figure 1: Two example sentences (on the left) annotated with cross sentential identity chain (red) and implicit relation (blue) and the corresponding merged representation (on the right) as per (O'Gorman et al., 2018)

2 Multi-Sentence AMR Corpus

Multi-Sentence AMR corpus (henceforth MS-AMR) is part of AMR3.0 release and was introduced by O'Gorman et al. (2018). While a number of works have used document-level AMR graph creation for downstream tasks such as summarization (Liu et al., 2015; Lee et al., 2021), it is the only dataset currently with manual annotations of AMR document graphs. It contains 284 documents in its train split and 9 documents on test split, annotated over 8027 gold AMRs in total. Out of the 284 train documents, 43 are annotated twice; in this work we refer to these double annotation documents as our development set.

Coreference chains were annotated as clusters of coreferent nodes in gold AMR graphs, rather than as coreferent spans of text: this is what allows evaluation with Smatch. Such an annotation also precludes evaluation of system predictions using more traditional coreference evaluations, as there is no notion of mention spans to use when comparing mentions.

The MS-AMR corpus also annotated implicit relations, as shown in the blue ARG4 edge in figure 1, whenever a numbered argument of a predicate was semantically identifiable in another sentence. Similarly, the MS-AMR corpus annotates bridging mentions, such as part-whole or set-member relations. Both implicit relations and bridging relations can be seamlessly added as cross-sentential edges.

Coreference chains form the majority of cross-sentential relations annotated in MS-AMR. Unlike implicit and bridging relations, there is no straight forward way of incorporating coreference chains into a multi-sentence AMR graph. Next section discusses the challenges involved and proposes a solution.

3 Document-level AMR Representation

As discussed in the past section, majority of MS-AMR annotations take the form of coreference chains over sentence-level gold AMR graphs. In this work, we propose a representation DocAMR that incorporates these chains into multi-sentence AMR graphs. The motivation for having such a representation is two-fold. First, representing a multi-sentence document as a single AMR graph will allow downstream applications to treat the whole document as single entity, much like the AMR graphs for sentences. Second, it is crucial to have a consistent representation for the purpose of evaluating MS-AMR predictions. Smatch – the standard evaluation metric for AMR – operates over graphs. We can use this metric for MS-AMR evaluation only if the system outputs and the gold graphs adhere to the same document-level graph representation.

Intrasentential coreferences in *sentence-level* AMR are annotated with what one might call *manual full merge* – i.e., given multiple mentions in the same sentence (such as a named entity and a pronoun), a human annotator would pick a single informative mention, and merge all other information into the root of the subgraph representing that mention. Language being efficient, it is very rare for multiple non-reduced mentions to occur in the same sentence, therefore it was a trivial task for human annotators. MS-AMR human annotations do not perform manual merge – they only indicate the nodes in multiple existing sentence-level graphs that participate in a coreference cluster.

3.1 Why Not Merge All Coreferent Nodes?

The pioneering work on MS-AMR (O'Gorman et al., 2018) that introduced the annotations, also

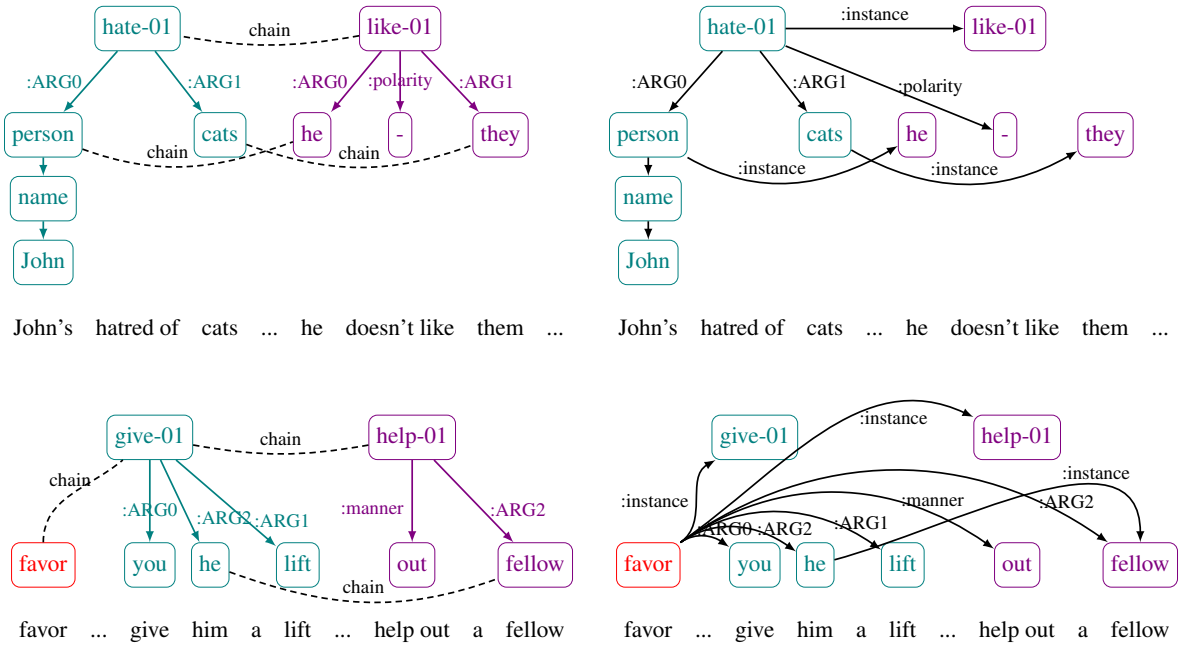

Figure 2: Two examples of merge operations that result in loss or distortion of information. Nodes of the same color belong to the same sentence-level AMR. Dashed lines indicate the identity chain links. In the top example, the merged node representation (right) reverses the meaning of the first sentence because of the negative `:polarity` node migrating from the merged node `like-01`. In the bottom example, the non-predicate node `favor` has received a bunch of `ARG` nodes from the fellow coreference nodes. Moreover, the merged-in predicate nodes `give-01` and `help-01` have lost association with their arguments. For example, the sentence "you help the fellow lift [something]" will be consistent with this graph.

proposed a way to represent MS-AMR in the form of a graph. This approach essentially merges all coreferent nodes from a chain into one node that inherits all the parents and children of the original nodes. Furthermore, if the lexical form of the merged nodes are different, each distinct form is added as an additional instance of the merged node connected via an `:instance` edge. Figure 1 shows a simple example of this merge operation.

This representation has a desirable property in that it treats cross sentential coreference information in a manner similar to sentence-level reentrancies. Another advantage of node merging is that a single node represents each entity/event, thus a document representation can be viewed as a mini Knowledge Base for the document. Moreover, this highly connected representation is consequential for Smatch: if a gold document has a cluster of ten mentions and a system incorrectly splits that cluster into six and four mentions, then the optimal one-to-one Smatch alignment will consider all links to the smaller cluster wrong.

However, merging coreference nodes at the document level is not trivial. Automatic indiscriminate merging of all coreference clusters has serious limitations. Without manual curation of potentially conflicting referential expressions or situations, it can end up merging conflicting information, particularly with event coreference: e.g., "his hatred of cats" and "John doesn't like cats" would be merged into the AMR equivalent of "John+he does not (like + hate) cats". It does not preserve the semantics of the original AMR graphs. Figure 2 shows examples of merge operations that either lose or distort the meaning of the text.

Moreover, the proposed method of (O'Gorman et al., 2018) assumes that a node can have multiple `:instance` edges – however existing AMR tools (including the Smatch implementation) do not allow multiple `:instance` edges on a node. In the hand full of previous works on MS-AMR, edge labels other than `:instance` are used to connect additional forms of a node. One node becomes privileged over others under the proposed approach – as a result, a system that selects the privileged node differently will be unnecessarily penalized.

We wish to avoid the above pitfalls while still retaining the advantages of node merging. The next section discusses the principles considered while formulating the DocAMR representation.

3.2 Considerations

Our primary consideration while designing DocAMR has been to preserve the meaning of the underlying text as encoded by the combination of sentence-level graphs and MS-AMR annotations. Whenever a secondary consideration led towards a lossy representation, we decided in favor of meaning preservation. In addition, we considered the following principles:

1. *Be consistent with AMR's treatment of within-sentence coreference.* In sentence-level AMRs, pronominal concepts are only used in the absence of a contentful antecedent. Pronouns don't contribute enough meaning beyond referential information. Therefore, only those pronouns are kept whose antecedent is unspecified. Furthermore, in sentence-level AMRs all coreferent nodes are merged so they share the same variable. We stick to these principles except in the cases where they can potentially cause information loss.
2. *Do not place more weight on any particular representative member of a coreference chain.* It would be artificial to represent the cluster as pairwise mention-mention links, e.g. by privileging one mention as the primary representative of the cluster. Instead, coref chains should be represented in a way that the non-pronominal ways of referring to the entity/event in different sentences are on equal footing.
3. *Systems should not get too much credit for superficial decisions* (like merely creating a concept for a pronoun, or detecting the name string and entity type on every mention of the name). This can be achieved by merging the nodes whenever it can be done safely without loss of information. The effect is that the scoring places more weight on detection of coreference.
4. *The representation should support comparison via Smatch* (without unnecessary computational overhead) for document-level parsers given the same document as input.[2]
5. *The edges between the nodes in a coref cluster and their sentence-level neighbours should be kept intact*, so that when evaluated with Smatch, at least one mention in a chain of correctly predicted entity mentions will get full credit even if the coreference is missed.
6. Finally, *the representation should not rely on complicated heuristics or new manual annotations*. For example, we considered and ultimately rejected the possibility of merging coreferent non-name nodes, which would have required heuristics for choosing one representative concept from among multiple distinct concepts from different mentions (and also would have violated principle #2).

[2]We do not consider here how to measure similarity between *comparable* documents.

3.3 Proposed Approach (DocAMR)

We propose DocAMR, an extension of AMR to the document level. In this representation, the entity and event coreference annotations from (O'Gorman et al., 2018) are properly assimilated into a single graph representing the meaning of the document. It completely preserves the semantic information while trying to stay close to the sentence-level AMR conventions.

The basic idea is that a new node with concept `coref-entity` is added for each identity chain of nodes – all nodes participating in the chain are then connected to this node via a `:coref` relation. Figure 3 shows an example of this. In addition to the base method of using `coref-entity` nodes, we introduce two exceptions. First, all the **named entities** within an identity chain are merged into one named entity. Second, all **pronominal nodes** participating in a chain are dropped. Both these exceptions are consistent with the treatment of coreferent named entities and pronouns in sentence-level AMR annotations. Further details of these two exceptions are outlined below:

Treatment of Named Entities: All the named entities (that is, AMR nodes with a `:name` relation) participating in an identity chain cluster are merged into one named entity. The structure of named entities is quite consistent and usually all the named entities participating in a chain match exactly. Occasionally, a cluster may contain named entities that differ in their type, wiki link or form of the name. In such cases, all unique forms of the name and all unique wikis are kept under a common variable. If types are different, the most specific type becomes the root and remaining types are connected to it with the new relation `:additional-type`.[3] If nodes to be merged have modifier roles other than `:name` and `:wiki`, they are gathered under the merged entity node.

[3]We use the AMR types ontology to decide which type is the most specific. If a cluster contains types that are not in the AMR ontology, then the ones from the ontology are considered for root position; and if none belong to the ontology, the most frequent type from the within the cluster becomes the root. Frequency ties are broken by picking the first mention.

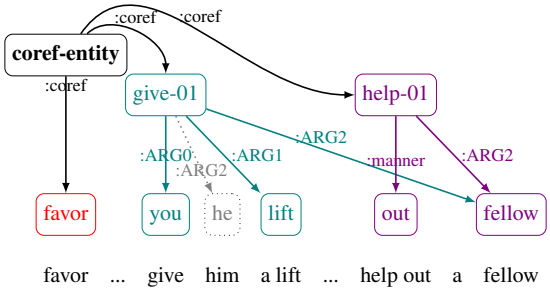

Figure 3: Our proposed DOCAMR representation applied to the bottom example from Figure 2. For the identity chain between `he` and `fellow`, the pronoun is dropped indicated in dotted gray and the links are transferred to the non-pronominal node. For the identity chain between `favor`, `give-01` and `help-01`, merging is deemed potentially lossy – instead a `coref-entity` node is introduced and all nodes in the chain are linked to it via `:coref` edge. This not only preserves the meaning without loss but also avoids preferential treatment of any content node in the chain.

Treatment of Pronominal Nodes: Sentential AMR incorporates pronouns as concepts only as a fallback if there is no contentful antecedent within the same sentence. We extend this philosophy to pronominal nodes whose antecedent can be found in another sentence. That is, all pronominal nodes participating in an identity chain cluster with an antecedent are replaced by the `chain-entity` node, removing the pronominal concepts from the graph. Pronominal concepts are retained only if there is no content antecedent to be found. For an identity chain with exclusively pronominal nodes, no `chain-entity` node is needed; they are simply merged into one node, which is labeled with the most specific pronoun concept in the chain (e.g., "he" is considered more specific than "someone"). As a special case, if a heterogeneous pronoun chain (with multiple pronouns) refers to a participant in a dialogue – indicated by the concept `i` or `you` – a new `interlocutor-entity` node is introduced and all pronouns are merged into that to account for different perspectives taken in different utterances.

A notable consequence of discarding pronoun concepts is that, correct antecedent resolution is required in order for any roles in which the pronominal mention participates, to be counted as correct. Systems could struggle with long chains of pronouns, especially if the first in the chain is resolved to the wrong antecedent and this causes cascading errors. That said, we believe representing the wrong substantive entity as participating in a relation is a serious error, no less so when a system correctly clusters a group of pronouns but fails to resolve their antecedent correctly.

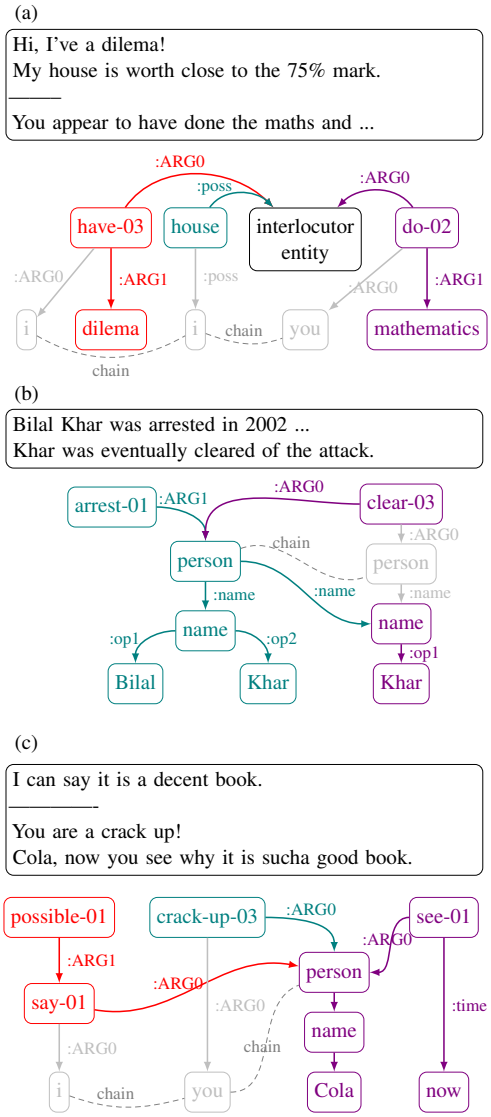

Figure 4: Various scenarios of merge operations in DOCAMR: (a) Merging first and second person pronouns in a dialogue into `interlocutor-entity` (b) Merging named entities with multiple forms, keeping all distinct forms under separate `name` nodes (c) Dropping pronominal nodes and replacing them with the named entity in the chain Nodes of the same color belong to the same sentence-level AMR. Dotted gray indicates dropped nodes and edges.

Discard Single Node Clusters: After creating a document-level graph per the rules outlined above, we perform a final step of removing any `coref-entity` nodes with one `:coref` edge, and replace references to that node with its only member node. This can happen if there is a cluster with one non-pronominal concept and one or more pronouns

	Train	Dev1	Dev2
Nodes in chains	12513	1241	1141
Pronouns in chains	4456	433	423
↪ Pronoun	1029	141	121
↪ Interlocutor-entity	773	66	90
↪ Other node	1494	140	118
↪ Coref-entity	1160	86	94
NEs in chains	2917	224	205
NEs merged	903	85	84

Table 1: Statistics on DOCAMR merge operations. Top row shows the total number of un-merged nodes in all identity-chains. Middle section shows the total number of pronominal nodes in identity-chain and the numbers of different target node types they are merged into. Most pronouns are dropped only about one fourth – in pronoun-only chains – are merged into one of the pronouns in the chain. Bottom sections shows number of Named Entities in chains before and after merging.

that got deleted. It can also happen if a cluster had multiple names that were merged. This simplification will discard superfluous `coref-entity` nodes. It would make alignment faster for Smatch that uses surface form matching during initialization.

3.4 The Roads not Taken

In an attempt to adhere to the sentence-level AMR conventions (principle 1 in §3.2) to a greater extent, we considered merging all non-predicate nodes participating in a cluster – keeping the most specific one. This required heuristics and even manual input on deciding the most specific instances, we therefore did not take that route.

A less complicated version of this would have been merging only those nodes that have the same concept. However, even if a concept recurs, the different mentions may have different modifiers/arguments that may collide and make merging problematic. Merging only concepts with the same modifiers could place outsize importance on the attachment of modifiers by the parser. We opted for the simplest approach of not performing any merging of non-name, non-pronominal nodes.

We also considered an alternative representation where, similar to DOCAMR, all nodes participating in a coreference cluster are connected to a `coref-entity` – but unlike DOCAMR, the parents of all participating nodes now point to the `coref-entity`. In other words, all mention nodes are clustered (via `:coref` edge without any merge) under `coref-entity` along with their descendent sub-graphs, but the `coref-entity` nodes sit between the mention nodes and their sentence-level parents. This flipped version of DOCAMR makes dense connections at document level making Smatch more sensitive to coreference errors. However, this often results in multiple similar or identical sub-graphs collected under `coref-entity` – with no connection to the corresponding sentences. Merging these sub-graphs based on the similarity of their structure will make the final representation highly dependent on small modifier level differences. Another significant side effect of this approach is that due to lack of connection with original sentence level graphs, the Smatch algorithm (as given in §4.1) cannot benefit from sentence alignments and becomes prohibitively inefficient.

4 DOCAMR Evaluation

DOCAMR represents MS-AMR annotations of multiple sentences in the form of one AMR graph. Ideally, the quality of this graph should be assessed as single unified entity. Traditional measures of coreference, such as MUC, CEAF etc., try to align the gold coreference chains with the predicted ones based on the shared mentions. In the case of text based coreference resolution, identifying shared mentions is trivial since the mentions are anchored in the input text. AMR, by contrast, is not anchored in the text – the nodes of AMR graphs are not aligned to words. As a result, traditional coreference measures can be applied to AMR only if the graphs are either identical to the gold graphs or have been aligned at the node level.

The Smatch algorithm used for evaluation of AMR parsers is a randomly initialized, greedy, hill-climbing algorithm. Due to the greedy nature of the algorithm, multiple random restarts are needed to get a stable matching score. Moreover, the problem itself is NP-complete and even its greedy implementation becomes prohibitively slow as the size of graph increases. As a result, Smatch in its original form is quite inefficient for document-level AMR evaluation.

4.1 Smatch Evaluation

Smatch (Cai and Knight, 2013) views an AMR graph as a set of triples, where each triple comprises of either a pair of nodes with a relation label (i.e. edges in the graph) or one node with an attribute and its value (i.e. concepts and attributes).

Given the set of triples for a source and a target graph and an *optimal* alignment between their nodes, the metric computes the precision, recall and F-score over triples between the two graphs. An *optimal* alignment is the one that will maximize the F-score. Cai and Knight (2013) provide a greedy implementation of Smatch that uses a combination of a 'smart initialization' and greedy hill-climbing steps to get closer to an optimal mapping. Each greedy step needs to evaluate $m(n-1)$ options where m and n are the numbers of nodes in the two graphs being matched. Therefore, Using this implementation of Smatch at the document-level is very slow.

4.2 Document-level Smatch

The sentence-level Smatch implementation[4] relies on a pool of candidate node-mappings to efficiently select the next greedy step. The pool includes all the node mappings that can possibly add to the triple F-score. We can restrict this candidate pool further for document-level Smatch if the source and target documents are aligned at the sentence level. More specifically, we can forbid any source to target node alignments where the sentence(s) of the source node is(are) not aligned with the sentence(s) of the target node. Note, that in DocAMR graphs, certain nodes can belong to multiple sentences such as `coref-entity` or a merged node.

We propose an implementation of Smatch (DocSmatch) that assumes alignment between the roots of sentence-level subgraphs of a pair of document-level AMR graphs. Nodes are first categorised by sentences with each node possibly assigned to multiple sentences. Next, the candidate mappings pool is constructed respecting the sentence-level alignments. In particular, a node in one AMR cannot be mapped to a node in the other AMR if none of their assigned sentences are aligned (see appendix A for details). For instance, consider the example in figure 4(c) – the node for the concept **`now`** must be aligned to a node in the third sentence in a target graph, whereas the **`person`** node, merged from mentions in all 3 original sentences, is not constrained.[5]

DocSmatch allows us to evaluate the DocAMR development set comprising of 42 documents in roughly 4 minutes with the default four random-restarts. This is a manageable time frame for the purpose of parser comparisons. The original Smatch evaluation for the same setup ran out-of-memory (with up-to 200GB allocation) without a result. Table 3 compares the Smatch scores and the runtimes of our implementation with those of the original Smatch with 1 random restart.

4.3 Coreference Subscore

A side effect of representing and evaluating the document AMR as a single unified graph is that we can not analyze the coreference performance of the parser separately. To mitigate this, we propose and implement a breakdown of Smatch that provides a separate coreference subscore. For the purpose of coreference subscore, all nodes connected to multiple sentences are considered coreferent nodes. Incoming edges for each coreferent node are counted as a part of the coreference subscore, as well as bridging relations and nodes with the labels `coref-entity` or `interloculor-entity`. Note, in the case of merged nodes, their incoming edges count towards the coreference scores, however the node themselves (i.e. their instance triples) are not counted towards the coreference score.

Since what is considered a coreference depends on the graph structure, the edges and nodes that are considered coreferent in the gold graph might be different from those in the predicted graph. Therefore, to calculate F1 for the coreference subscore, we consider an edge or node to be a correct match if (1) it has a matching node or edge according to the standard Smatch score, and (2) the node or edge is part of coreference in both the gold and predicted graph. Recall is calculated as a percentage of gold coreference nodes/edges, precision is a percentage of predicted coreference nodes/edges, and F1 is taken as the harmonic average.

5 Experiments and Results

We use DocAMR along with our efficient implementation of Smatch to assess the quality of two document-level AMR parsing systems. First, we develop a pipeline system combining a top-performing AMR parser (Zhou et al., 2021) and a state-of-the-art coreference resolution system

[4] https://amr.isi.edu/evaluation.html

[5] Note that there could be a case where the predicted parse of sentence 2 resembles the correct parse of sentence 1 and vice versa without any coreference link between the two—then the proposed constraint would prevent finding the optimal alignment. However, this happens quite infrequently. Moreover, we argue that accidental mapping of triples between graphs of entirely unrelated sentences should not be rewarded. Note also that for the border case where all nodes are connected with all sentences, the constrained version will be same as the original Smatch.

MS-AMR Splits		Double Anno. (Dev1)			Double Anno. (Dev2)			Test		
AMR	Coref	Smatch	Coref	Reent	Smatch	Coref	Reent	Smatch	Coref	Reent
Gold	None	87.6	0	72	88.6	0	73	86.4	0	72
Gold	CoreNLP	89.7	34	76	90.6	35	78	90.6	47	80
Gold	AllenNLP	90.5	40	78	91.0	41	79	91.3	53	82
(Anikina et al., 2020)		-	-	-	-	-	-	44.3	17	21
S-BART	None	67.1	0	53	67.8	0	53	67.5	0	55
S-BART	CoreNLP	68.7	28	57	69.3	29	57	71.3	43	63
S-BART	AllenNLP	**69.4**	**33**	**59**	**69.8**	**34**	**59**	**72.0**	**50**	**65**

Table 2: Document-level Smatch, coreference sub-scores (Coref) and reentrancy scores (Reent) on MS-AMR double annotations (Dev1 and Dev2) and test splits – using various combinations of gold and predicted AMR graphs with predicted coreferences from CoreNLP (Clark and Manning, 2016) and AllenNLP (Joshi et al., 2020).

Impl.		Original		DocSmatch	
Split	R	Time	Smatch	Time	Smatch
Dev1	4	-	-	244	69.4
Dev2	4	-	-	136	69.8
Test	4	-	-	417	72.0
Dev1	1	927	69.5	66	69.3
Dev2	1	945	69.8	41	69.7
Test	1	1314	71.3	104	72.0

Table 3: Comparison of Smatch scores and runtimes (in seconds) between the original Smatch implementation (Original) and our proposed implementation (DocSmatch). All results are on our best performing pipeline system. R is the number of random restarts. 'Original' Smatch runs out of memory for R>1.

(Joshi et al., 2020). We also provide the pipeline results with CoreNLP's neural coreference resolution system (Clark and Manning, 2016) v4.3.2 for additional point of comparison. Second, we reevaluate the past best system of Anikina et al. (2020) – this is also a pipeline approach combining AMR parser of (Lindemann et al., 2019) with AllenNLP coreference resolution system.[6]

5.1 Our Pipeline Approach

We use the BART-based structured transformer model of Zhou et al. (2021) to produce sentence-level AMR graphs. In particular, we use the StructBART-S version of their system referred to as S-BART in table 2. This parser produces node-to-token alignments as part of its output – we use these alignments to match coreference systems' outputs with AMR graphs. Text-based coreferences are obtained using the systems of Joshi et al. (2020) and Clark and Manning (2016). Coreference chains are computed from these prediction files.[7]

In order to incorporate this coreference information into the AMR graphs, we first convert node-to-token alignments into node-to-span alignments. The span of a node is defined as the smallest text span containing all the tokens aligned to any of its descendants (to avoid loops, re-entrant edges are removed keeping only the first). With node-to-span alignments, a predicted mention is assigned to the node with the shortest span containing the mention. If there is more than one candidate node, the one with the greatest height, subsuming the other candidates is selected. Instances of coreference within a sentence are ignored assuming that the sentence-level parser has already taken care of them.

5.2 Results

Table 2 shows the results on MS-AMR double annotation documents (used here as development set) and its test split. Both gold and predicted AMR graphs are converted to DocAMR before running Smatch evaluation. All numbers are produced using document-level Smatch with 4 random restarts.

Our pipeline approach outperforms the previous best system by a large margin, providing a strong baseline for future research on this task. This is due mainly to difference in quality of the underlying sentence-level parsers.

[6] We obtained the document-level graphs before merge operations from Anikina et al. (2020) for the purpose of re-evaluation in DocAMR format.

[7] Using the package https://github.com/boberle/corefconversion

Dev1	Smatch	Time (s)
No-Merge	93.3	66
Merge-NE	92.8	66
DocAMR	87.6	90
Merge-All	82.4	153

Table 4: Impact of representation on evaluation scores and runtimes. Comparing double-annotation gold documents (Dev1) with and without coreference links in different document-level representations.

Note that significant improvements in coreference quality result in only small improvements in overall Smatch score – showing that a separate coreference subscore is essential for assessing a system's performance on cross-sentential relations. We also report reentrancy scores (Damonte et al., 2017) for comparison. While the performance gap is more pronounced in reentrancy scores compared to overall Smatch, the gains are best highlighted in coreference subscore. For instance, reentrancy scores improves by up to 2 points from CoreNLP to AllenNLP in all settings – coref subscores, on the other hand, shows up to 7 points improvement giving a finer range for coreference evaluation.

Impact of Representation on Smatch: To highlight how document-level representation can affect Smatch scores and efficiency, we compare the gold double-annotation development set with and without coreference links. In addition to DocAMR we consider three representations: 1) No-Merge: where all coreference nodes are linked via coref-entity without any merging 2) Merge-NE: where only Named Entities are merged and 3) Merge-All: where all coreference nodes in a chain are merged (O'Gorman et al., 2018). One of our aims for DocAMR was to ensure that the lack of coreference links is visible in the overall Smatch score. Table 4 shows that DocAMR makes this gap bigger without losing efficiency or semantic information.

6 Related Work

MS-AMR Annotations MS-AMR annotations by O'Gorman et al. (2018) include coreference chains, implicit roles and bridging relations. In the context of AMR-based summarization, Lee et al. (2021) present a novel dataset consisting of human-annotated alignments between the nodes of paired documents and summaries to evaluate merge strategies for merging individual AMR graphs into a document graphs. However, they sought out the merge operations that can serve as cross sentential coreference in the absence of any annotations – they only merge nodes with same surface forms, except for 'person' nodes. Our work, on the other hand, outlines a representation for already available gold annotations, where nodes' surface forms don't match in a large number of cases.

MS-AMR Evaluations and Models O'Gorman et al. (2018) proposes Smatch as primary method for scoring MS-AMRs. They also report CoNLL-F1 relying on Smatch alignments. Adopting the methods from O'Gorman et al. (2018), Anikina et al. (2020) presented a comparative evaluation of various coreference resolution systems over MS-AMR test sets and document-level Smatch evaluations of machine generated sentence-level AMRs augmented with coreference predictions from various systems. The best approach from their study is incorporated as a baseline in §5. Fu et al. (2021) introduce an AMR coreference resolution system that uses graph neural network to model gold sentence-level AMR graphs for coreference predictions. This system assumes gold graphs and is not comparable with document-level parsing systems. Use of gold graphs also alleviates the need for alignments between gold and predicted graphs for the purpose of evaluation. Bai et al. (2021) constructed dialogue-level AMR graphs from multiple utterance level AMRs by incorporating inter-sentence coreference, speaker and identical concept information into sentence-level AMRs.

7 Conclusion

We have presented DocAMR, a graph representation for document-level AMR graphs based on coreference annotations. Relative to the original sentence-level graphs, DocAMR removes redundancy without information loss. We modified the implementation of Smatch to take advantage of sentence provenance to efficiently search for node alignments when comparing two document-level graphs. Finally, we reported results for a document-level parsing pipeline that can serve as a strong baseline for future work on this task.

Acknowledgment

Jeffrey Flanigan was supported by the NSF National AI Institute for Student-AI Teaming (iSAT) under grant DRL 2019805. The opinions expressed are those of the authors and do not represent views of the NSF.

References

Tatiana Anikina, Alexander Koller, and Michael Roth. 2020. Predicting coreference in Abstract Meaning Representations. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 33–38, Barcelona, Spain (online). Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. 2021. Semantic representation for dialogue modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4430–4445, Online. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Berlin, Germany. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR coreference resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Fei-Tzin Lee, Chris Kedzie, Nakul Verma, and Kathleen McKeown. 2021. An analysis of document graph construction method for AMR summarization. *arXiv preprint: arXiv:2111.13993v1*.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. 2021. Structure-aware fine-tuning of sequence-to-sequence transformers for transition-based AMR parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Constrained Candidate Node Mappings for DocSmatch

Algorithm 1 Constrained candidate node mappings for efficient document-level Smatch computation.

inputs:
$dAMR_1, dAMR_2$ pair of AMRs for a document
$sRoots_1, sRoots_2$ aligned sentence roots

$N \leftarrow$ *Number of sentences* {in the document} {collect descendant of *sRoots*}
for $i \leftarrow 1..N$ **do**
  $Desc_1[i] \leftarrow \text{GETDESC}(dAMR_1, sRoots_1[i])$
  $Desc_2[i] \leftarrow \text{GETDESC}(dAMR_2, sRoots_2[i])$
end for
$CandMap \leftarrow \{\}$ {Candidate Node Mappings}
for $i \leftarrow 1..N$ **do**
  for $node \in Desc_1[i]$ **do**
    $CandMap[node] \mathrel{+}= Desc_2[i]$
  end for
end for
return *CandMap*