CALAMR: Component ALignment for Abstract Meaning Representation

Paul Landes, Barbara Di Eugenio
Department of Computer Science, University of Illinois Chicago
{plande2, bdieugen}@uic.edu

Abstract

We present Component ALignment for Abstract Meaning Representation (CALAMR), a novel method for graph alignment that can support summarization and its evaluation. First, our method produces graphs that explain what is summarized through their alignments, which can be used to train graph-based summarization learners. Second, although numerous scoring methods have been proposed for abstract meaning representation (AMR) that evaluate semantic similarity, no AMR based summarization metrics exist despite years of work using AMR for this task. CALAMR provides alignments on which new scores can be based. The contributions of this work include a) a novel approach to aligning AMR graphs, b) a new summarization based scoring methods for similarity of AMR subgraphs composed of one or more sentences, and c) the entire reusable source code to reproduce our results.

Keywords: AMR, alignment, summarization, flow network

1. Introduction

Abstract meaning representation (AMR) is a semantic representation language that captures "who is doing what to whom" in a sentence (Banarescu et al., 2013). AMR graphs represent semantic structure in a syntactic independent way (see Fig 1). For this reason they are balanced in their level of abstraction making their representation conducive to tasks such as summarization (Liu et al., 2015; Liao et al., 2018; Dohare et al., 2018), Question Answering (Kapanipathi et al., 2021), headline generation (Gu et al., 2020), and automatic machine translation (MT) (Blloshmi et al., 2020). In most tasks, text-to-graph models are used to create AMR graphs as an upstream task, which are evaluated by comparing their output to human annotated graphs by similarity (Cai and Knight, 2013; Bonial et al., 2020). Summarization applications often include language generation (Hardy and Vlachos, 2018).

While large language models (LLMs) have seen great promise in recent years, they continue to suffer from hallucination, which precludes faithful and traceable summaries (Maynez et al., 2020) needed by domains such as the clinical medical field. Because our method aligns summaries to every source AMR subgraph, we are able to trace text from the summary via aligned tokens (depicted as arrows in Fig 1) to text of the source text. For this reason, we propose this method for training summarization models, which has the potential of ameliorating these issues.

A measure of overlap between the source and summary is a byproduct of our alignment method, which can be used as a summarization metric for AMR summarization. Despite recent interest in AMR scoring methods (Cai and Lam, 2019; Opitz et al., 2020) to accurately evaluate text-to-graph and graph-to-text models, there still exist no AMR summarization scoring methods.

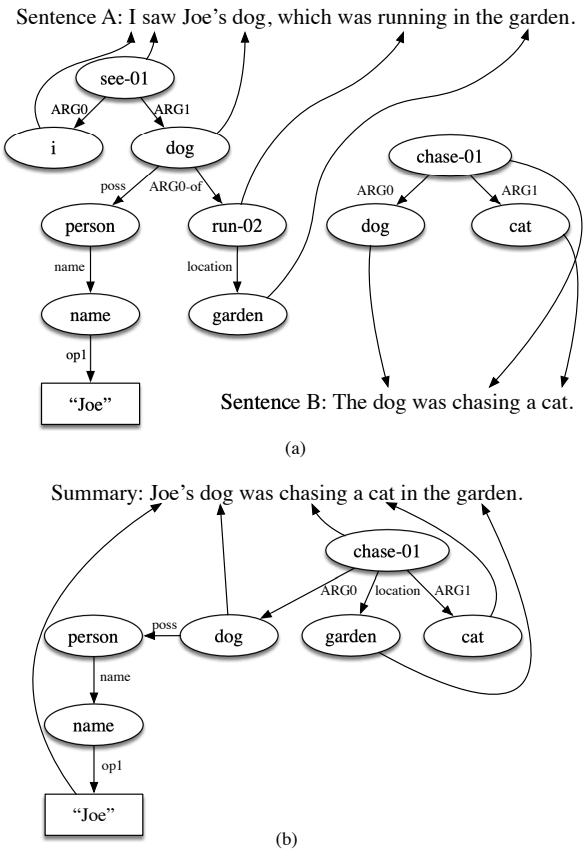

Figure 1: **AMR Graph Components.** Top (a) the source graph. Bottom (b) the summary graph. The aligned tokens shown as arrows. Example from Liu et al. (2015).

To accomplish our goal of alignment and scoring, we propose a model that takes advantage of flow networks (see Sec 3.3), which as far as we know, have rarely been used in NLP, and never on AMR

LREC-COLING 2024, pages 2622–2637

graphs. Our methodology takes sentences annotated with human annotated AMR graphs[1] as input; we then tie them with a root node as components, and then connect them as a bipartite graph. Bipartite alignments' capacities are computed by their respective component node neighborhoods and computed as the cosine similarity of node, edge, and local neighborhood embeddings (see Sec 3).

We use several human annotated corpora to compare with existing similarity AMR scoring methods and our own summarization scores (see Sec 5). In this work, we provide a) a novel method and source code[2] of creating summarization alignments, b) a new AMR summarization metric, which is the first to our knowledge, and c) a freely available[3] PropBank database with precomputed embeddings. The flow network construction and its methods are explained in Sec 3 and Sec 4 describes our scoring method and how the metrics are calculated. Our experimental design is given in Sec 5.2 and results in Sec 6.

2. Related Work

The earliest AMR summarization work of Liu et al. (2015) reduces source AMR graphs into a summary graph using integer linear programming. This was then extended by Liao et al. (2018) to generate text using a rule based method. While this work is most similar to ours, we uses a bipartite flow network (see Sec 3.3) for alignment of the source and summary graphs. Furthermore, their work builds on the graph reduction methods of Thadani and McKeown (2013) for sentence comprehension and re-frames commodity flow (Magnanti and Wolsey, 1995) for edge inclusion to induce a summary graph.

Compared to AMR summarization and language generation efforts (Dohare et al., 2017, 2018; Hardy and Vlachos, 2018), scoring has received as much or more attention. SMATCH (Cai and Knight, 2013) was the first AMR semantic graph similarity score, which measures overlap of graphs and designed for inter-annotator agreement. Several recent replacements of SMATCH have been proposed such as SEMBLEU (Song and Gildea, 2019). Later, Weisfeiler-Leman graph kernel (Shervashidze et al., 2011) using an interesting method of leveraging distributed probability in local network neighborhood for matching. While this work is similar, ours uses a flow network to measure the information transmission across the graph globally while their method employs the word mover algorithm (Kusner et al., 2015) for local subgraphs. To our knowledge, this is the first work that uses the max flow algorithm with AMR graphs.

The study and optimization of a graph as a flow network has been evolving for more than eight decades since the publication of a 1930 article by A. N. Tolstoĭ on Soviet railroad planning (Schrijver, 2002). Several decades later, the max flow problem was formalized in a declassified military report (Harris and Ross, 1955), which became the foundation of the seminal Ford Fulkerson max-cut min-flow algorithm (Ford and Fulkerson, 1962). This resulted in a decades-long track record of the application of flow networks in Computer Science, maximum cardinality matching (Hopcroft and Karp, 1973), baseball elimination (Schwartz, 1966), and airline scheduling (Cormen et al., 2001).

Flow networks have been used in NLP for the task of MT framed as a bipartite flow matching task (Gaussier, 1998), and for information retrieval (IR) to semantically match documents to queries (Guo et al., 2016). However, they have not been used with any language based graph-based, such as AMR, to our knowledge.

3. Methods

An AMR graph represents the abstract meaning of a sentence as a directed acyclic graph (DAG). Each node of an AMR graph represents a concept as an idea grounded by natural language in the form of a verb, noun, or abstract placeholder (i.e. `person` for names). Attribute nodes contain surface text such as cardinals, date tokens, and other named entities, and verbs are modeled as rolesets in PropBank (Kingsbury and Palmer, 2002).

The children of concept verb nodes are other concept or attribute nodes as they "play a role" in a specific way relative to the "sense" of the roleset. Each role is a member of a roleset, semantically labeled with a function tag and an enumerated index used as the edge label that connects it with its parent concept verb node. Some edges have labeled relationships rather than roleset concepts such as `possessive`, `conjunction`, and `location`.

Our method uses AMR sentence graphs as building blocks that are are iteratively constructed into larger graphs that represent any arbitrary language structure such as paragraphs or documents. Each iteration of this process connects one or more graphs from the previous step; the input are *AMR graphs* that represent a sentence from human annotations (Fig 2a):

1. *Source* and *summary components* (see Fig 1) are each one or more sentence AMR graph combined with a root node to form the document's source or summary. Fig 2b

2. *Alignment graph* is a bipartite graph of the

[1] Automatically parsed graphs are available in our source code, but experimentation is left for future work.

[2] https://github.com/uic-nlp-lab/calamr

[3] https://github.com/plandes/propbankdb

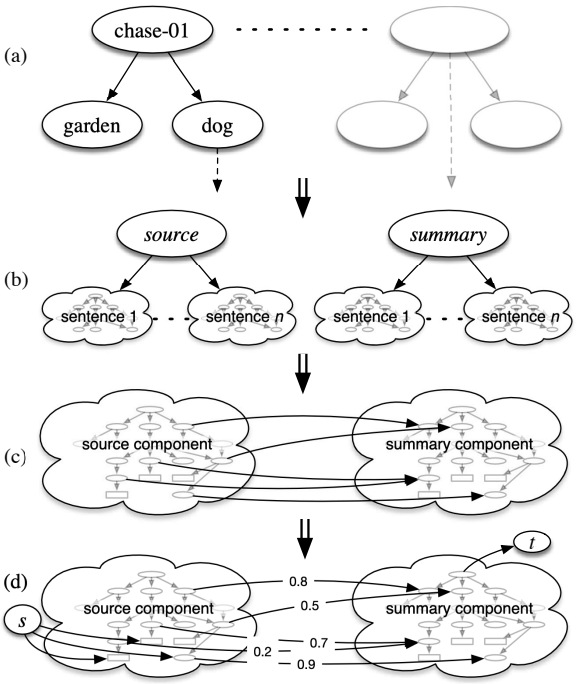

Figure 2: **Graph Construction.** The graph construction process starting with AMR graphs.

source and summary components. Fig 2c

3. *Flow network* models the flow of information through a graph using two weighted edge values: capacities and flow. Fig 2d

In this work, the flow material is information through the connected flow network constructed in Step 3. The direction of the flow goes from the source flow network node[4] to the leaf nodes of the summary component, over the bipartite alignment edges to the source component, and then to the source root. The amount of information flow through each alignment edge provides local scores of the strength of each source to summary node alignment pair and globally for the document.

Two kinds of alignments are described are a) text-to-graph aligned tokens (TATs) associate AMR nodes to tokens, and b) graph alignment edges connect nodes across the flow network. Embeddings generated from the source text of the TATs are attached to AMR graph nodes at inference time. However, static PropBank roleset embeddings (see Sec 3.1) are computed before the method begins. The end-to-end pipeline includes:

1. Preprocessing PropBank embeddings (3.1).
2. Constructing the flow network (3.3).
3. Attaching graph embeddings from TATs and PropBank to concept verb nodes (3.3).
4. Computing alignment edge capacities (3.3.1).
5. Reducing the alignment (3.4).
   (a) Run max flow algorithm (3.4.1).
   (b) Normalize flow-per-node.
   (c) Remove low flow alignment edges (3.4).
   (d) Go to step 5a until convergence.

[4]The terminology difference between the source node s (flow network) and the source component (AMR graph) is explicitly differentiated since the source node can be connected to the source component.

3.1. AMR Graph Embeddings

When the alignment graph (Fig 2c) is created (see Sec 3.3), we compute and attach embeddings to nodes and edges. These embeddings are taken from the text of PropBank (Kingsbury and Palmer, 2002) entries related to concept nodes, and then combined to compose local network neighborhood embeddings (see Sec 3.2).

PropBank is a lexicon consisting of verb senses as "frames" defined in frame files (Loper et al., 2007), which we use to generate graph embeddings. The embeddings, for PropBank and all graph nodes and edges, are generated from Sentence-BERT (SBERT)[5], a siamese network model that captures sentinel semantic similarity (Reimers and Gurevych, 2019). The model embeddings were taken from the public checkpoint of the pretrained model and used without further fine-tuning.

Fig 3 shows an example of PropBank labels used as the model input on the top left for concept node `chase-01`. This concept node $\boldsymbol{n}$ has embedding $\mathrm{emb}_n(\boldsymbol{n})$ calculated from SBERT model $\mathbf{S}$ using the PropBank roleset label "follow, pursue". The TAT embeddings of single tokens are aggregated with concept node embeddings for text-to-graph aligned tokens as depicted with the `garden` node on the bottom of the same figure. The concept node embeddings are then added by surrounding nodes' embeddings to represent the local context of the graph, such as with the `dog` node.

The AMR graph root nodes are attached to non-AMR sentence nodes with their own embedding $\mathrm{emb}_s(\boldsymbol{n})$. The example shows sentence s *"Joe's dog was chasing a cat in the garden."* in sentence node $\boldsymbol{n}$ on the top right of Fig 3.

The edge embeddings are computed from additional PropBank labels, including role descriptions (i.e. "follower"), function tags semantic tags of the role(i.e. "PAG" for "prototypical agent"), and role edge AMR role descriptions (i.e. "argument frame" for `:ARG0`) or "possessive" for `:poss`).

[5]The large model (all-mpnet-base-v2) was used for all experiments.

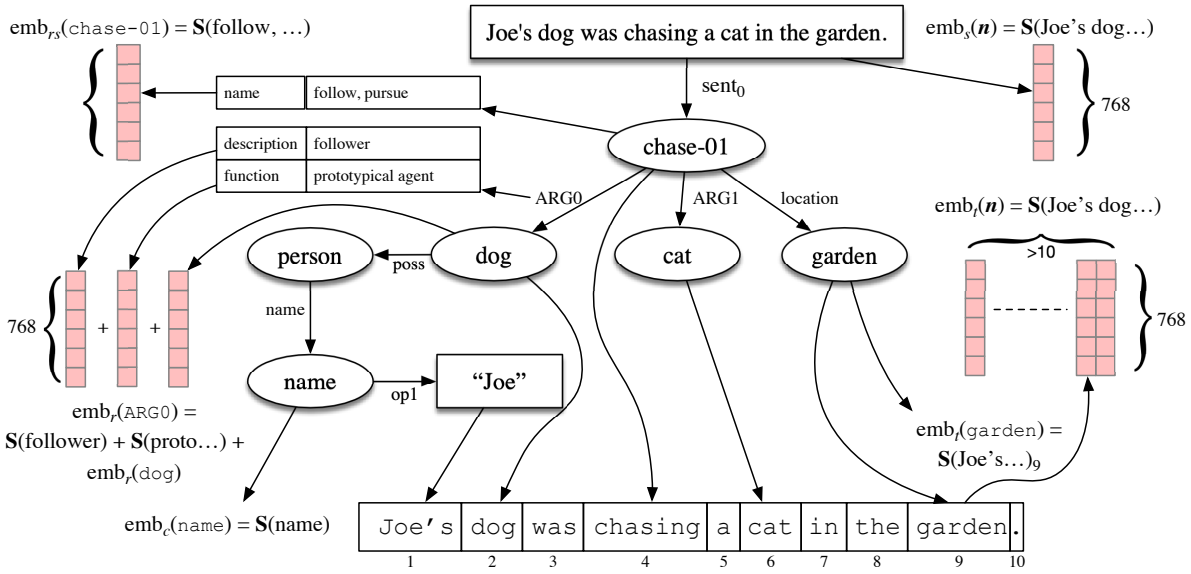

Figure 3: **Graph Embeddings**. An AMR graph of the sentence, *"Joe's dog was chasing a cat in the garden."*. The embedding node definitions with the model output are given for TAT `garden` at index 9, the roleset (`chase-01`), the role (`ARG0` with description "follower" and the function tag "prototypical agent"). Each token is represented as one word piece for simplicity.

See Appendix A for detailed formulation and Appendix C for additional example figures.

3.2. Network Neighborhood Embeddings

The nodes adjacent to an alignment target node are traversed to create a local network neighborhood embedding. This local embedding better contextualizes node alignments. Fig 4 shows the network neighborhood around the target node `run-02`.

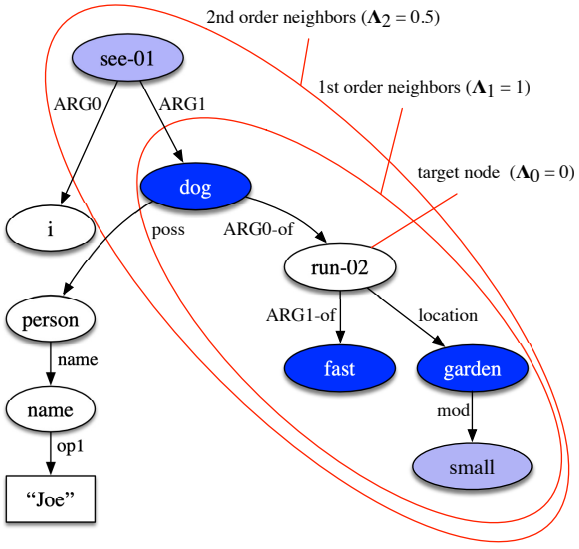

Figure 4: **Network Neighborhood Weights**. The AMR graph for the sentence, *"I saw Joe's dog, which was running fast in the small garden."*, with the network neighborhoods of verb concept node `run-02` and their weights. The $\mathbf{\Lambda}$ weights are shown with default hyperparameters.

Embeddings are aggregated by adding and/or averaging similar to previous work (Opitz et al., 2021). The dark blue nodes are the k^{th} order neighbor set that are at exactly k hops away from the target node. They are colored according to how much they influence the embeddings have scaled by the hyperparameter $\mathbf{\Lambda}$, which is an array of real values for each concentric k^{th} order neighbor set. The higher the k the lower the weight, and influence their embeddings have on the target node. See Appendix A.5 for the network neighborhood formulation.

3.3. Flow Network Construction

A flow network is a graph with two values associated with each edge: a capacity and a flow. It can be conceptualized as a network of pipes where the capacity is the maximum amount of material allowed to flow, and the flow is the amount of material traversing a pipe. Every flow network has a source s that produces an infinite flow, and a sink t that accepts an infinite flow.

More formally, a flow network is a graph of vertices and edges $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

$$\forall v, \in \mathcal{V} \sum_{\text{e into v}} f(e) = \sum_{\text{e leaving v}} f(e) \quad (1)$$

$$\forall e, \in \mathcal{E}, 0 \leqslant f(e) \leqslant c_e,\ v(f) \triangleq f^{\text{out}}(s) \quad (2)$$

where $\mathcal{V}$ is the set of vertexes; $\mathcal{E}$ the set of edges; $f(e)$ is the amount of material flowing through edge e; c_e is a capacity for edge e, and $v(f)$ the value of flow. Conceptually, the edge weights of the flow

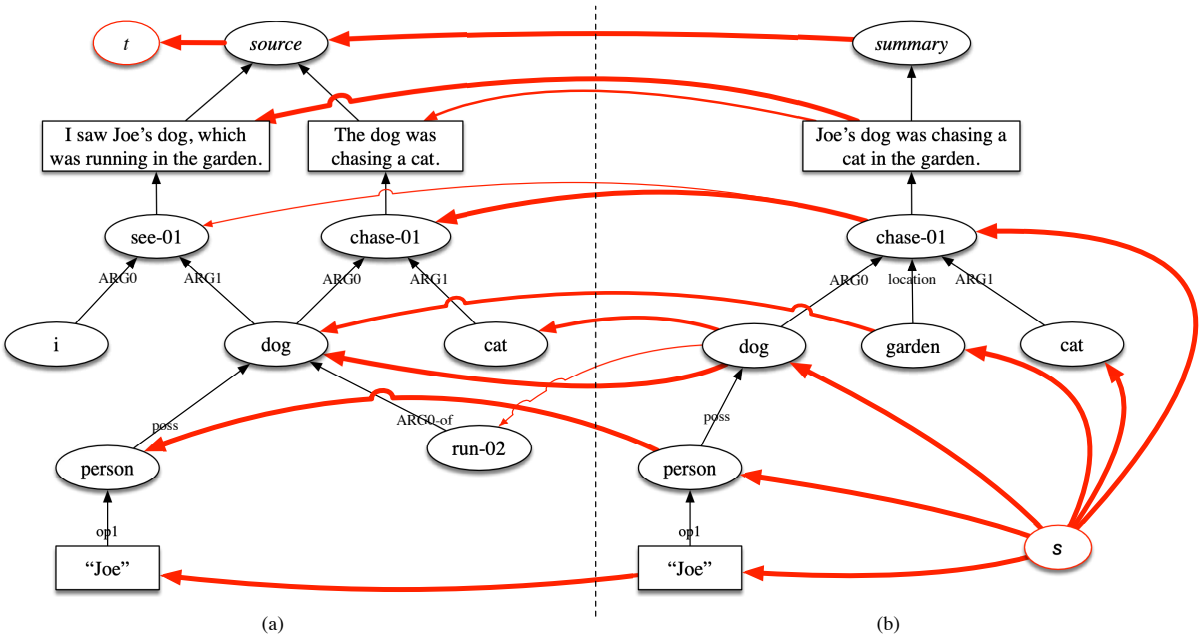

Figure 5: **Flow Network Construction**. The terminal flow nodes and alignment edges added to the bipartite source and summary components with the changes shown in red. The width of the edge represents the value of its capacity as a function of its similarity. Some nodes and edges are omitted for simplicity; see Appendix C for example figures.

network are capacities, which represent the limit of water pumped through a pipe before it breaks from the pressure. The amount of flow, like the amount of water in a pipe, is the information gain between the source and summary.

Equation 1 states that all material (i.e. water) going into a node must have the same amount of material leaving that node. The flow into the graph is the sum leaving the source s constrained by the capacities (c_e) of edges connected to it. The flow capable of flowing through the graph $\mathcal{G}$ from source s to sink t is defined in Equation 2. Finding the maximum possible flow on all edges is solved by the max flow algorithm (Gao et al., 2022). See Appendix B for the capacity formulation.

3.3.1. Graph Component Connection

The first step to creating the source and summary components is to add sentence nodes for each AMR graph, which are then tied together with a root node. During construction, parsed sentences are attached to sentence nodes, and TATs are attached to concept and attribute nodes (see Fig 1). Concept nodes are coupled with PropBank rolesets, and outgoing role edges are coupled with connected concepts.

The source and summary components are then connected with the alignment edges to create the bipartite alignment graph. They are created as the Cartesian product of the concept nodes across the components with their capacity values set to the semantic similarity for each bipartite node pair. Capacities for all alignment edges are assigned (as described in Appendix B) and edges discarded for those that fall under the similarity threshold (τ_δ) hyperparameter. The edges are then reversed to allow the flow network configuration necessary for alignment assignments as described in Sec 3.4.

3.3.2. Attach Terminal Nodes

Finally, the source node s is connected to the summary's AMR graph leaf nodes and the sink t is connected to the source component root node. The role edges of the AMR graph that relate the roles of concept nodes also become capacitated edges with values set to infinity. The resulting flow network graph, shown in Fig 5, is described and constrained by Equation 1 and used with the max flow algorithm to compute the flow through all edges.

3.4. Alignment Graph Algorithm

After the flow network is created, the alignment capacities between the concept nodes of source and summary components are modified. A byproduct of the algorithm is a score assigned to each role edge of how well the respective branch of the component is aligned across the alignment graph. The initial capacities that are set on the constructed alignment edges (see Sec 3.3.1) provide an initial estimate of concept node pair alignments. However, these capacity values only take into account the local node neighborhood at the sentence, or document level. Instead, the flow values assigned by the max flow

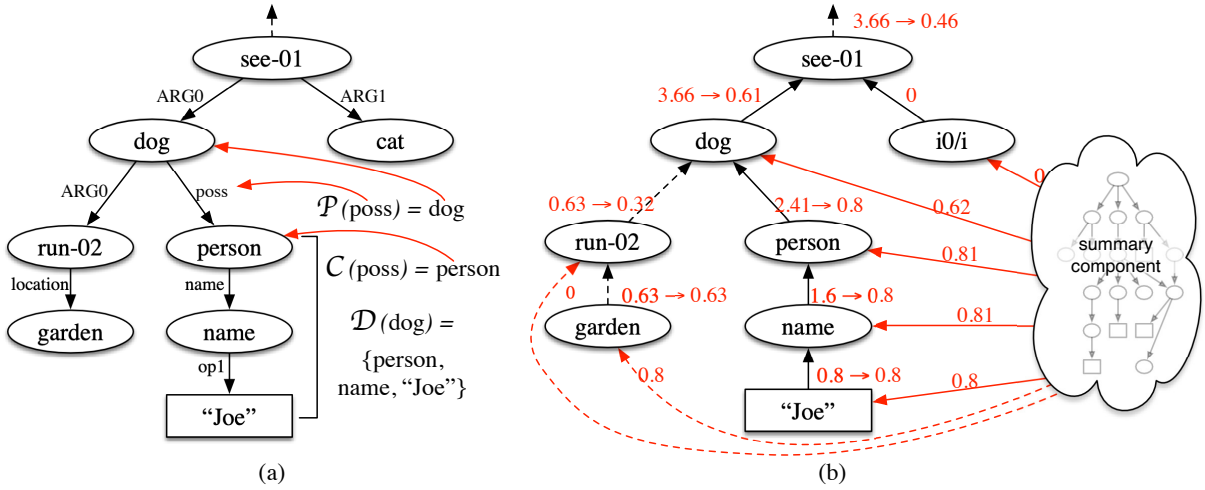

Figure 6: **Network Flow**. Left (a): the edge node parent of edge `poss` is `dog` and its edge node child is `person`. Right (b): example flow from the summary to the source component with flows (left of arrow) and normalized flow per node values (right of arrow) with constricted edges as dotted lines.

algorithm provide a global representation with a better estimate of the alignment strength. The algorithm goes a step further and iteratively updates capacities informed by flow changes by re-running the max flow algorithm until convergence.

3.4.1. Max Flow

The algorithm (Goldberg and Tarjan, 1988) computes the max flow by pushing flow from the flow network's source s node to the AMR summary component. It then flows through the alignment edges from the summary to the source component, up toward the root of the source component, and out to the flow network's sink node t. Since all role edges are set to infinity, flow goes through to every node and edges of the graph[6] as shown in Fig 5.

3.4.2. Flow Normalization

The flow network at this point has flow values on each edge, which are then normalized for each node based on its descendants. After the maxflow algorithm has run, flow values are normalized for each subgraph. First, graph terminology is to be defined. Let:

- The edge node parent $\mathcal{P}(e)$ be the node at the source end of a directed edge e in the alignment graph (the edge node parent of `poss` is `dog` in Fig 6a).
- The edge node child $\mathcal{C}(e)$ be the node at the target end of a directed edge e.
- The node descendants $\mathcal{D}(v)$ of v be all paths to the terminal leaf node grandchildren ($\mathcal{D}$(dog) = {`person`, `name`, `Joe`} in Fig 6a). For example, in Fig 6a, the $\mathcal{D}$(dog) are the nodes `person`, `name`, and attribute `Joe`.
- The flow per node $\tilde{f}(e)$ be the flow at an edge divided by all its source node's descendants:

$$\tilde{f}(e) = \frac{f(e)}{|\mathcal{D}(\mathcal{P}(e))|} \quad (3)$$

where $f(e)$ is the flow value through edge e. Fig 6b shows the values from a run of the algorithm with flows (left of arrow) and flow per node values (right of arrow). For example, a query of the flow that leaves `person`, which is one of the role edges labeled `:poss`, starts with traversing node $\mathcal{P}$(poss) = dog with $\mathcal{D}(\mathcal{P}(\text{dog})) = \{\text{dog}, \text{name}, \text{Joe}\}$. The flow on `:poss` is 2.41 so our flow per node = $\frac{2.41}{3} = 0.8$.

3.4.3. Capacity Constriction

After flow normalization, we "squeeze" capacities of the alignment edges, which affects the next iteration (see Sec 3.4.4). The edge capacities are set to zero if the edge's flow falls under the alignment edge minimum capacity cutoff (τ_α). The same applies for the parent's descendant's role edge for flows less than role edge minimum capacity cutoff (τ_ρ):

$$\{\forall e \in \mathcal{E} : e < \tau_\rho\}\ \mathcal{D}(\mathcal{P}(e)) \leftarrow 0$$

where e is the component role edge; $\mathcal{P}(e)$ is the edge node parent, and $\mathcal{D}(\mathcal{P}(e))$ are the descendant nodes of the edge node parent.

If the τ_ρ is set to 0.4, the role edge that connects `run-02` and `dog` in Fig 6b would meet the criteria for its edge node parent and descendants to be set to zero. The path of role edges affected in the figure are represented with dotted lines. Since $\mathcal{P}(e)$ = `dog`, and $\mathcal{D}$(`dog`) = {`run-02`, `garden`}

[6] Reentrancies are exceptions to this rule (see Sec 7).

then the capacities of the alignment edges between the summary component and `run-02` (already set to 0) and `garden` (from 0.8) will be set to 0 (represented with dotted lines). Note that even though there are capacity edges well over the value of the τ_ρ, we still set them to zero since each edge takes into account the flow per node value of all flows of descendants at that level.

3.4.4. Summary Max Flow

The non-zero alignment edges remaining after the steps described in Sec 3.4.1 to Sec 3.4.3 explain how much, and what part of the source is included in the summary. Similarly, these steps are repeated on a second flow network with its output explaining the extent of the summarized source. This second flow network shares capacities, and alignment edges are reversed so it flows from the source to the summary. The source flow node s is connected to the leaf nodes of the source component and the sink t node is moved to the summary root.

Before the max flow algorithm is run on the second reversed flow graph, the flow values on all role edges are tracked, and if any changes, the steps described in Sec 3.4.1 through Sec 3.4.3 are run again. The method's execution is alternated across the first and second flow network until convergence indicative of static flow values is reached.

3.5. Final Alignment Graph

In summary, the alignment graph is made up of the source and summary components composed of AMR graphs, which are then connected to create the flow network. Alignment edges are then deleted by clamping shared capacities across two flow network instances (each with a reversed flow). Low flow edges result in minimized or deleted alignment edges for iteration of the algorithm.

The final alignment graph looks similar to Fig 5 before the alignment edges are deleted. For example, `chase-01` to `run-02` nodes for $\tau_\delta = 0.3$.

4. Scoring Method

Our methods measure the semantic similarity similar to SMATCH, but includes metrics for summarization overlap. We define the value of flow exiting the source node to the sink as the *source root flow* using Equation 2 with $\mathbf{C}_{f\mathbf{c}} \triangleq f^{\text{out}}(s_{\text{source}})$.

This metric applies for every subgraph or globally as the source component's root node connected edge. For example, this value is 0.46 in Fig 6b for the subgraph from the `see-01` node to the leaf nodes. Likewise, the value of flow exiting the summary node to the sink is defined as the *summary root flow* with $\mathbf{C}_{f\mathbf{y}} \triangleq f^{\text{out}}(s_{\text{summary}})$. Additional CALAMR scoring methods include the portion of nodes in the source component that have at least one alignment with the summary, defined as the *source aligned portion* ($\widetilde{\mathbf{C}}_\mathbf{c}$) and the portion of nodes in the summary component that have at least one alignment with the source defined as the *summary aligned portion* ($\widetilde{\mathbf{C}}_\mathbf{y}$).

Given the symmetry of the algorithm as detailed in Sec 3.4.4, we can treat the source as any sentence (and the likewise the summary) if we wish to score them in like fashion to previous baseline scoring methods (Cai and Knight, 2013; Opitz et al., 2021); meaning we can score the similarity of two AMRs.

Unlike previous methods, we can also score AMR graphs as multi-sentence graphs by the summarization overlap as a non-negative real value called the *aggregate flow*:

$$\mathbf{C}_f = 2\frac{\mathbf{C}_{f\mathbf{c}} \cdot \mathbf{C}_{f\mathbf{y}}}{\mathbf{C}_{f\mathbf{c}} + \mathbf{C}_{f\mathbf{y}}}, \tag{4}$$

which results in higher values for graphs with multiple node alignments. This score is useful for subgraphs but not globally, so we define the *aggregate alignment portion* score as the harmonic mean of the aligned node portion across components:

$$\widetilde{\mathbf{C}} = 2\frac{\widetilde{\mathbf{C}}_\mathbf{c} \cdot \widetilde{\mathbf{C}}_\mathbf{y}}{\widetilde{\mathbf{C}}_\mathbf{c} + \widetilde{\mathbf{C}}_\mathbf{y}}, \tag{5}$$

which is also a real value but has range $[0, 1]$, and is advantageous as it has the same range as established AMR scores such as SMATCH.

5. Experiment Design and Setup

We report two kinds of experiments with the first concerning summarization (see Sec 5.1) and the second similarity scoring (see Sec 5.2) between human annotated AMRs and the output of three popular parsers. We run these evaluations to assess how CALAMR compares to other methods of finding summarized content and judging alignment as a similarity measure. Given the results reported in Sec 6, we believe CALAMR is a more perspicuous score as it judges semantic similarity both locally and globally. The second set of experiments use the overlap of two multi-sentence AMR graphs as a measure of summarization.

5.1. Summarization

The Proxy Report AMR corpus contains news articles with sentences tagged as a date, country, topic, summary and body. Only sentences tagged with summary or body are used in our experiments. The corpus development set has 35 articles and 826 sentences, and its test set has 33 articles and 823 sentences. The alignment method was first used

to find summarized text through text-to-graph alignments. It was then used to score summarizations in development and test sets.

After scoring, sentences of the source and summary for 33 articles of the two data sets were swapped and scored a second time (we call this the "Mismatch set"). This was accomplished by switching the source sentences from the test set with the summary sentences in the development set. Source sentences in the Proxy Report corpus resulted in in 14,108 role edges, 660 concept nodes and the summary totaled 1,153 role edges and 802 concept nodes.

5.2. Similarity Scoring

Three popular and peer-reviewed parsers were used to generate AMR graphs from natural language sentences scored with SMATCH (Cai and Knight, 2013), WLK (Opitz et al., 2021)[7] and our scoring method. The JAMR (Flanigan et al., 2014) parser is also included for a baseline comparison.

The parsers' output was scored against the first 20 sentences of the human annotated "Proxy Report" LDC2020T02 AMR Annotation Release 3.0 (Knight et al., 2021) corpus and two Information Science Institute[8] corpora, which include *Little Prince* (1,562 sentences) and *Biomedical* (6,952 sentences). The amrlib[9] library was used with the SPRING (Bevilacqua et al., 2021) and Gsii (Cai and Lam, 2020) parsers as it has been shown to produce good results on text-to-graph tasks (Heinecke and Shimorina, 2022; Opitz and Frank, 2022; Opitz et al., 2021).

5.3. Baselines

Our primary baseline uses the Liu et al. (2015) method of comparing the text-to-graph aligned words from the source to the summary as a bag of words. ROUGE (Lin, 2004) is used on the unigrams to evaluate the aligned overlap to gauge how well CALAMR "finds" summarized content.

We also compare the resulting CALAMR edge coverage of alignment outputs with the document-level AMR graph heuristic method that links based on exact match of Liu et al. (2015) on the LDC2014T12 AMR Annotation Release 1.0 corpus as a baseline[10]. We cannot directly compare coverage as their method is a graph reduction into the summary and ours is an alignment method. Instead, we compare CALAMR positive value flow role edges to their sentence level graph expansion since both methods' goal is to identify AMR subgraphs used for alignment. Both of our metrics are taken as a quotient of the total number of summary component role edges.

[7] All experiments used WLK settings of $K = 2$ iterations with "all directional communication".

[8] https://amr.isi.edu/download.html

[9] https://github.com/bjascob/amrlib

[10] The AMR 3.0 corpus was used for other experiments.

6. Results

Sec 6.1 pertains to CALAMR as a summarization method with respect to scoring, Sec 6.2 explains the measure of summarization between two AMR graphs, and our results are given in Sec 6.3.

6.1. Summarization

The stark contrast of the alignment portion across the corpora is a strong indicator of the ability of the method. The scoring method captures the rate of summarization as evidenced with 86.6% average aligned AMR nodes in the unaltered document set (Proxy Report) compared to the 35.1% in the development and test switched set (Mismatch) shown in Table 1. The source component aligned portion also reveals the difference between the unaltered and Mismatch document set as the portion of the summary is represented in the source (43.2% vs. 14.6%). The high summary root flow ($\mathbf{C}_{f\mathbf{y}}$) of 7.21 (see Table 1) in the unaltered set indicates the source graph is strongly aligned compared to the low summary root flow in the Mismatch set of 2.61 ($\mathbf{C}_{fc}$). These flow metrics represent the degree to which each graph is aligned with the other.

Corpus	$\widetilde{\mathbf{C}}_{\mathbf{y}}$	$\widetilde{\mathbf{C}}_{\mathbf{c}}$	$\mathbf{C}_{f\mathbf{y}}$	$\mathbf{C}_{fc}$
Proxy report	86.6%	43.2%	721.4%	67.1%
Mismatch	35.1%	14.6%	261.1%	19.6%

Table 1: **Document Summarization Scores.** Scoring matched vs. mismatch corpus. See Sec 4 for CALAMR scoring notation.

6.2. Alignment

As explained in Sec 4, the particularity of the source and summary becomes moot given the symmetry of the algorithm across the components, which motivates the semantic similarity scoring method. We hypothesize that the aggregate alignment portion CALAMR $\widetilde{\mathbf{C}}$ score reflects the similarity between any two AMR graphs, and thus, is an indicator of the effectiveness of the score for summarization.

To estimate this effectiveness we compared the text-to-graph parser output of AMR sentence metrics with previous methods as a reference point. This was done by comparing the scoring methods with the aggregate alignment portion ($\widetilde{\mathbf{C}}$ from Equation 5) using Pearson (ρ) correlations. Table 2 lists the scores between the human annotated AMR

sentences and parser output. We find highly correlated scores between WLK (69.2) and SMATCH (67.7) using JAMR on the Little Prince corpus.

To that end, JAMR produces the highest correlations for the other corpora as well. In fact, this trend extends to SPRING (second most correlated) and Gsii (least correlated). Because the parser is a consistent indicator of highly positively correlated results, regardless of corpus, we conclude that CALAMR is an effective scoring method.

Corpus	Parser	Sent	$\rho\,\widetilde{\mathbf{C}}, \mathbf{S}$	$\rho\,\widetilde{\mathbf{C}}, \mathbf{W}$
Biomedical	Gsii	6,644	41.2	31.8
Biomedical	Jamr	5,612	66.2	65.2
Biomedical	Spring	6,617	50.1	41.3
Little prince	Gsii	1,464	38.8	35.7
Little prince	Jamr	1,501	**67.7**	**69.2**
Little prince	Spring	1,497	41.3	47.1
Proxy report	Gsii	8,042	22.9	30.8
Proxy report	Jamr	7,781	53.2	56.2
Proxy report	Spring	8,120	37.3	48.2

Table 2: **Parser Alignment Scoring.** AMR Sentence Pearson correlations (ρ) between aggregate alignment portion ($\widetilde{\mathbf{C}}$)ALAMR (see Equation 5) and previous scoring methods (S)MATCH and (W)LK. Metrics are reported only for successfully parsed (Sent)ences.

6.3. Previous Methods

Even though our method is not a summarization model, we compare to other methods (summarization and non-summarization) as a reference point. Table 3 shows the alignment results on the LDC2014T12 corpus using the same methodology as Liu et al. (2015). The high ROUGE1 score (F1 of 17.5 over the baseline), shows that CALAMR is effective at finding summarized content.

Method	Precision	Recall	F1
Liu et al. (2015)	51.9%	39.0%	44.3%
Dohare et al. (2017)	52.4%	55.7%	51.3%
Fu et al. (2021)	-	-	49.1%
CALAMR	69.0%	68.6%	68.8%

Table 3: **Aligned Node Text Comparison.** Unigram (bag of words) aligned source to summary overlap of text-to-graph tokens.

The Liu et al. (2015) method is compared with role and alignment edge coverage in Table 4 as explained in Sec 5.3. The results show a significantly higher coverage of edges from the expanded sentence-level edges to CALAMR component role edges. CALAMR also has a much higher coverage of summary component alignment with the source compared to the previous method's document-level expanded edges. This shows our method is a better method comparatively.

Split	Sent	Doc	$\widetilde{\mathbf{C}}$ Role	$\widetilde{\mathbf{C}}$ Alignment
Train	75.5%	84.6%	84.3%	94.8%
Dev.	85.4%	91.8%	83.1%	92.5%
Test	75.0%	83.3%	84.3%	96.3%

Table 4: **Summary Alignment Coverage.** Edge coverage as a portion within (Sent)ences and (Doc)uments (Liu et al., 2015), which are analogous to the portion of non-zero flow Role and Alignment edges using the ($\widetilde{\mathbf{C}}$)ALAMR method.

7. Reentrancies

AMR nodes with multiple parents (reentrancies) lead to catastrophic alignment failure. Because the role edge's capacities are set to infinity, the max flow algorithm redirects all flow through only one of the parent's incoming edges starving the other paths from the reentrancy to the root. We observed 2.6% of the source graphs nodes and 2.18% of the summary graphs nodes to be reentrancies. To address this issue we clamped the capacities of all incoming edges of reentrancy nodes to the normalized sum of the incoming flow so that:

$$\forall e \in \mathcal{P}(n), c_e \leftarrow \frac{1}{|\mathcal{P}(n)|} \sum_{u \in \mathcal{P}(n)} f(u) \qquad (6)$$

where c_e is the capacity value; $\mathcal{P}(n)$ are incoming edges to n, and $f(u)$ is the flow through e into n. This resulted in all of the summary graphs being repaired (all reentrancy positive edge flows) and 80.3% repaired in the source graphs. This method fixed three catastrophic alignment failures of the 366 Proxy Report documents, but one (0.27%) remained unchanged.

8. Conclusions and Future Work

We have presented CALAMR, an alignment method that supports scoring metrics that provide a semantic similarity metric for multi-sentence AMR graphs and subgraph level summarization metrics. The method is suitable as both a scoring method capable of determining the portion of overlapping content through both alignment and flow metrics. More importantly, we believe it has the potential to a solution toward traceable summaries (an area needed for assisting in ground truth), which we leave as a future work.

9. Acknowledgments

This work was partially supported by award R01 CA225446 from the National Institutes of Health (NIH). We thank Professor Cornelia Caragea at the University of Illinois Chicago for her guidance in this work.

References

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

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12564–12573.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling Cross-Lingual AMR Parsing with Transfer Learning Techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500. Association for Computational Linguistics.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020. InfoForager: Leveraging Semantic Search with AMR for COVID-19 Research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2019. Core Semantic First: A Top-down Approach for AMR Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3799–3809. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. AMR Parsing via Graph-Sequence Iterative Inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301. 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. Association for Computational Linguistics.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*, second edition edition. MIT Press and McGraw-Hill.

Shibhansh Dohare, Vivek Gupta, and Harish Karnick. 2018. Unsupervised Semantic Abstractive Summarization. In *Proceedings of ACL 2018, Student Research Workshop*, pages 74–83. Association for Computational Linguistics.

Shibhansh Dohare, Harish Karnick, and Vivek Gupta. 2017. Text summarization using abstract meaning representation.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics.

Lester Randolph Ford and Delbert Ray Fulkerson. 1962. Flows in networks. In *Flows in Networks*, Princeton Landmarks in Mathematics and Physics, page 212. Princeton University Press.

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. Association for Computational Linguistics.

Yu Gao, Yang P. Liu, and Richard Peng. 2022. Fully Dynamic Electrical Flows: Sparse Maxflow Faster Than Goldberg-Rao. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 516–527.

Eric Gaussier. 1998. Flow Network Models for Word Alignment and Terminology Extraction from Bilingual Corpora. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 444–450. Association for Computational Linguistics.

Andrew V. Goldberg and Robert E. Tarjan. 1988. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940.

Xiaotao Gu, Yuning Mao, Jiawei Han, Jialu Liu, You Wu, Cong Yu, Daniel Finnie, Hongkun Yu, Jiaqi Zhai, and Nicholas Zukoski. 2020. Generating Representative Headlines for News Stories. In *Proceedings of The Web Conference 2020*, WWW '20, pages 1773–1784. Association for Computing Machinery.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. Semantic Matching by Non-Linear Word Transportation for Information Retrieval. In

Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16, CIKM '16, pages 701–710. ACM Press.

Hardy Hardy and Andreas Vlachos. 2018. Guided Neural Language Generation for Abstractive Summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773. Association for Computational Linguistics.

TE Harris and FS Ross. 1955. Fundamentals of a method for evaluating rail net capacities. *Rand Corp, Santa Monica, CA*.

Johannes Heinecke and Anastasia Shimorina. 2022. Multilingual Abstract Meaning Representation for Celtic Languages. In *Proceedings of the 4th Celtic Language Technology Workshop within LREC2022*, pages 1–6. European Language Resources Association.

John E. Hopcroft and Richard M. Karp. 1973. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for Knowledge Base Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA).

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2021. Abstract Meaning Representation (AMR) Annotation Release 3.0.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings to Document Distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 957–966. JMLR.org.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for Multi-Document Summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81. 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. Association for Computational Linguistics.

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining Lexical Resources: Mapping Between PropBank and VerbNet. In *Proceedings of the 7th International Workshop on Computational Linguistics*.

Thomas L. Magnanti and Laurence A. Wolsey. 1995. Optimal trees. In *Handbooks in Operations Research and Management Science*, volume 7 of *Network Models*, pages 503–615. Elsevier.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919. Association for Computational Linguistics.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-Leman in the Bamboo: Novel AMR Graph Metrics and a Benchmark for AMR Graph Similarity. *Transactions of the Association for Computational Linguistics*, 9:1425–1441.

Juri Opitz and Anette Frank. 2022. Better Smatch = Better Parser? AMR evaluation is not so simple anymore. In *Proceedings of the 3rd Workshop on Evaluation and Comparison of NLP Systems*, pages 32–43. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. AMR Similarity Metrics from Principles.

Transactions of the Association for Computational Linguistics, 8:522–538.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Alexander Schrijver. 2002. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445.

Benjamin L. Schwartz. 1966. Possible winners in partially completed tournaments. *SIAM Review*, 8(3):302–308.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *The Journal of Machine Learning Research*, 12:2539–2561.

Linfeng Song and Daniel Gildea. 2019. SemBleu: A Robust Metric for AMR Parsing Evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4552. Association for Computational Linguistics.

Kapil Thadani and Kathleen McKeown. 2013. Sentence Compression with Joint Structural Inference. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 65–74. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv: 1609.08144.

A. AMR Graph Embedding Calculation

This appendix explains the graph and TAT embeddings formulation.

A.1. Document Nodes

In future work, the embeddings for document nodes will be computed as the mean of their constituent subgraph children node embeddings:

$$\mathrm{emb}_d(\boldsymbol{n}) \triangleq \frac{1}{|\mathcal{C}(\boldsymbol{n})|} \sum_{\boldsymbol{u} \in \mathcal{C}(\boldsymbol{n})} \mathrm{emb}(\boldsymbol{u})$$

For paragraph nodes will be the mean of the sentence nodes (children nodes $\mathcal{C}(\boldsymbol{n})$ are defined in Sec 3.4.2) that compose that paragraph. The $\mathrm{emb}(\boldsymbol{u})$ notation is used to get the node embeddings of node $\boldsymbol{u}$. Each node has their own embedding definition including sentence node embeddings defined in Sec A.2

A.2. Sentence Nodes

Sentence nodes use SBERT's `[CLS]` output:

$$\mathrm{emb}_s(\boldsymbol{n}) \triangleq \mathbf{S}(\boldsymbol{s_n}) \in \mathbb{R}^d$$

where $\boldsymbol{n}$ is a sentence node in the graph, and $\mathbf{S}(\boldsymbol{s_n})$ is the SBERT sentinel embedding for sentence $\boldsymbol{s_n}$. The $\mathrm{emb}_s(\boldsymbol{n})$ notation denotes a sentinel embedding with the s subscript. The other nodal embeddings include concept nodes ($\mathrm{emb}_c(\boldsymbol{n})$), and attribute nodes ($\mathrm{emb}_a(\boldsymbol{n})$).

A.3. Concept Nodes

Concept nodes have a rich set of information, such as PropBank entries and graph aligned text. The PropBank rolesets are preprocessed with embeddings as described in Sec 3.1 and the TAT embeddings are generated at graph create time.

Instead of the sentence `[CLS]` SBERT token, the SBERT last output layer is used for tokens:

$$\mathrm{emb}_t(\boldsymbol{n}) \triangleq \begin{cases} \sum_{\boldsymbol{w}}^{\boldsymbol{s_n}} \sum_{i}^{|\boldsymbol{w}|} \mathbf{S}(\boldsymbol{s_n})_i & \text{if } \boldsymbol{n} \text{ has alignments} \\ \mathbf{1}_d & \text{otherwise} \end{cases} \quad (7)$$

where $\boldsymbol{n}$ is the concept node, and d is the SBERT embedding dimension (768). This output has a separate embedding for each aligned token $\boldsymbol{w}$, which maps to one or more wordpieces (Wu et al., 2016). The aligned token $\boldsymbol{w}$ embedding is the sum of each ith wordpiece $\mathbf{S}(\boldsymbol{s_n})_i$ from the sentence text $\boldsymbol{s_n}$. A single model embedding space across the sentence node's `[CLS]` and wordpiece vectors provides continuity when aggregating embeddings. Fig 3 illustrates an AMR graph with TAT `garden` at index 9 in the sentence, which gives the 9th index in to the SBERT embedding output. This embedding has $d \in \mathbb{R}^{10 \times 768}$ for the model's output layer (assuming each word token is mapped to a single wordpiece).

Role A roleset is a grouping of PropBank roles and is attached to a concept node defined as:

$$\mathbf{h} = [\mathbf{S}(\boldsymbol{s_r}) + \mathbf{S}(\boldsymbol{s_f})] \cdot \rho_{\mathbf{r}}$$
$$\mathrm{emb}_r(\boldsymbol{e}) \triangleq \mathrm{emb}(\mathcal{C}(e)) \cdot \rho_{\mathbf{c}} + \mathbf{h} \quad (8)$$

where $\boldsymbol{e}$ is the role edge; $\rho_{\mathbf{c}}$ and $\rho_{\mathbf{r}}$ are role and role child node hyperparameters, and $\mathrm{emb}(\mathcal{C}(e))$ is the embedding of role edge node child. $\mathbf{S}(\boldsymbol{s_r})$ and $\mathbf{S}(\boldsymbol{s_f})$ are the role name and role semantic tag embeddings. The `ARG0` role embedding of `chase-01` is shown in Fig 3 as the sum of the components, the edge, and the child node `dog`.

Node In addition to aligned tokens and roles, concept node embeddings are aggregated with PropBank roleset embeddings. The roleset can be thought of as metadata attached to verb concept nodes and static across all nodes of the same verb roleset. For example, the `chase-01` found in two sentences in Fig 5 have the same roleset and embedding. Most concept nodes have TATs that are also aggregated into the node's embedding. The embedding for a concept verb node $\boldsymbol{n}$ is defined as the weighted mean of the TATs, the roleset, and the role edges:

$$\mathbf{h_{rs}} = \mathbb{1}_v[\boldsymbol{n}]\ \mathbf{S}(\boldsymbol{s_{rs}}) + (1 - \mathbb{1}_v[\boldsymbol{n}])\,\mathbf{S}(\boldsymbol{s_n})$$
$$\mathbf{h_r} = \mathbb{1}_v[\boldsymbol{n}] \left[\sum_{\boldsymbol{e} \in \mathcal{N}(\boldsymbol{n})} \mathrm{emb}_r(\boldsymbol{e})\,\omega_{\mathbf{r}} \right]$$
$$\mathrm{emb}_c(\boldsymbol{n}) \triangleq \mathrm{emb}_t(\boldsymbol{n})\,\omega_{\mathbf{t}} + \mathbf{h_{rs}}\,\omega_{\mathbf{rs}} + \mathbf{h_r} \quad (9)$$

where $\mathrm{emb}_t(\boldsymbol{n})$ is the token alignment embedding produced; $\mathbf{S}(\boldsymbol{s_{rs}})$ is the SBERT roleset embedding; $\mathbf{S}(\boldsymbol{s_n})$ is the non-verb node text embedding; $\mathcal{N}(\boldsymbol{n})$ are the outgoing role edges of node $\boldsymbol{n}$ that connect to its children; $\omega_{\mathbf{t}}$, $\omega_{\mathbf{rs}}$ and $\omega_{\mathbf{r}}$ are hyperparameters, and $\mathbf{h_r}$ is the role embedding defined by Equation 8. The indicator function $\mathbb{1}_v[\boldsymbol{n}]$ yields 1 when $\boldsymbol{n}$ is a verb node and 0 for noun and other abstract meaning nodes (see Fig 3). This "toggles" the use of the roleset embedding for verb nodes. Otherwise, the text of the node itself is used as the input.

Fig 3 shows the constituent parts of the $\mathrm{emb}_c(\boldsymbol{n})$ embeddings and how PropBank and TAT embeddings are aggregated.

A.4. Attribute Nodes

If models that assigned TATs were completely accurate, attribute nodes would always have at least one token alignment. Instead, they leave some attributes with only the surface text given to the node by the text-to-graph model without the alignment. Similar to Equation 9, we use the attribute node's text when TATs are not available. Alignments are "toggled" by defining the attribute embedding as:

$$\mathbf{h_t} = \min(1, |\mathcal{T}|)\,\mathrm{emb}_t(\boldsymbol{n})$$
$$\mathrm{emb}_a(\boldsymbol{n}) = \mathbf{h_t} + (1 - \min(1, |\mathcal{T}|))\,\mathbf{S}(\boldsymbol{s_n}) \quad (10)$$

where $\mathcal{T}$ is the set of TATs; $\mathrm{emb}_t(\boldsymbol{n})$ is the TAT embedding (see Equation 7), and $\mathbf{S}(\boldsymbol{s_n})$ is the attribute's text embedding.

A.5. Network Neighborhood

Sec 3.2 describes network neighborhood embeddings. Let $\mathcal{U}(\boldsymbol{n}, k)$ be the k^{th} order neighbor set nodes, concentric "rings" around a target node at exactly k hops distance from node $\boldsymbol{n}$, then the network neighborhood embedding is defined as:

$$\mathrm{emb}_n(\boldsymbol{n}) = \sum_{i}^{k} \sum_{\boldsymbol{u} \in \mathcal{U}(\boldsymbol{n}, i)} \mathrm{emb}(\boldsymbol{u}) \cdot \boldsymbol{\Lambda}_i \quad (11)$$
$$\forall x \in \boldsymbol{\Lambda} : x > 0$$

where k is the maximum order k^{th} order neighbor set.; $\mathcal{U}(\boldsymbol{n}, i)$ is the k^{th} order neighbor set i hops from $\boldsymbol{n}$; $\mathrm{emb}(\boldsymbol{u})$ node $\boldsymbol{n}$'s embedding, and $\boldsymbol{\Lambda}_i$ is the i^{it}'s weight hyperparameter that dampens the node $\boldsymbol{n}$'s embedding. The hyperparameter weights are set so that the farther a node is from the target node, the less influence it has on the neighborhood embedding with the exception of $\boldsymbol{\Lambda}_0 = 0$ so that the target node's embedding is dropped.

B. Capacity Calculation

The alignment edge capacities using the node pairings (see Sec 3.1) are defined as:

$$\mathrm{cossim}(\boldsymbol{n_1}, \boldsymbol{n_2}) = \left[\frac{\mathrm{emb}(\boldsymbol{n_1}) \cdot \mathrm{emb}(\boldsymbol{n_2})}{||\mathrm{emb}(\boldsymbol{n_1})||\,||\mathrm{emb}(\boldsymbol{n_2})||} \right]$$
$$\mathrm{sim}(\boldsymbol{n_1}, \boldsymbol{n_2}, \mu) = \mathrm{cossim}(\boldsymbol{n_1}, \boldsymbol{n_2})^{\mu}, \mu > 0 \quad (12)$$

where $\mathrm{emb}_{s_1}(\boldsymbol{n})$ and $\mathrm{emb}_{s_2}(\boldsymbol{n})$ are the embeddings of the node pair, and μ is the hyperparameter that non-linearly adjusts the similarity to penalize for over-zealous similarities with settings for each node type. Some nodes or local network neighborhoods might be semantically similar, but for the wrong reasons. This is addressed globally by the max flow algorithm (see Sec 3.4) and locally with combined network neighborhood embeddings (see Sec 3.2). The μ hyperparameter is used to penalize certain node pairs for over-zealous similarities with a settings for each node type.

The capacity value $\mathrm{C}(\boldsymbol{n_1}, \boldsymbol{n_2})$ assigned to alignment edges is tuned by a translated sigmoid and clamped to $[0, 1]$:

$$\sigma(x) = \left(1 + \exp(0.5 - x)\right)^{-1} - 0.5$$
$$c = \mathrm{sim}(\boldsymbol{n_1}, \boldsymbol{n_2}, \mu_n)$$
$$\mathrm{C}_n(\boldsymbol{n_1}, \boldsymbol{n_2}) \triangleq \min(1, \max(0, c + \sigma(c))) \quad (13)$$

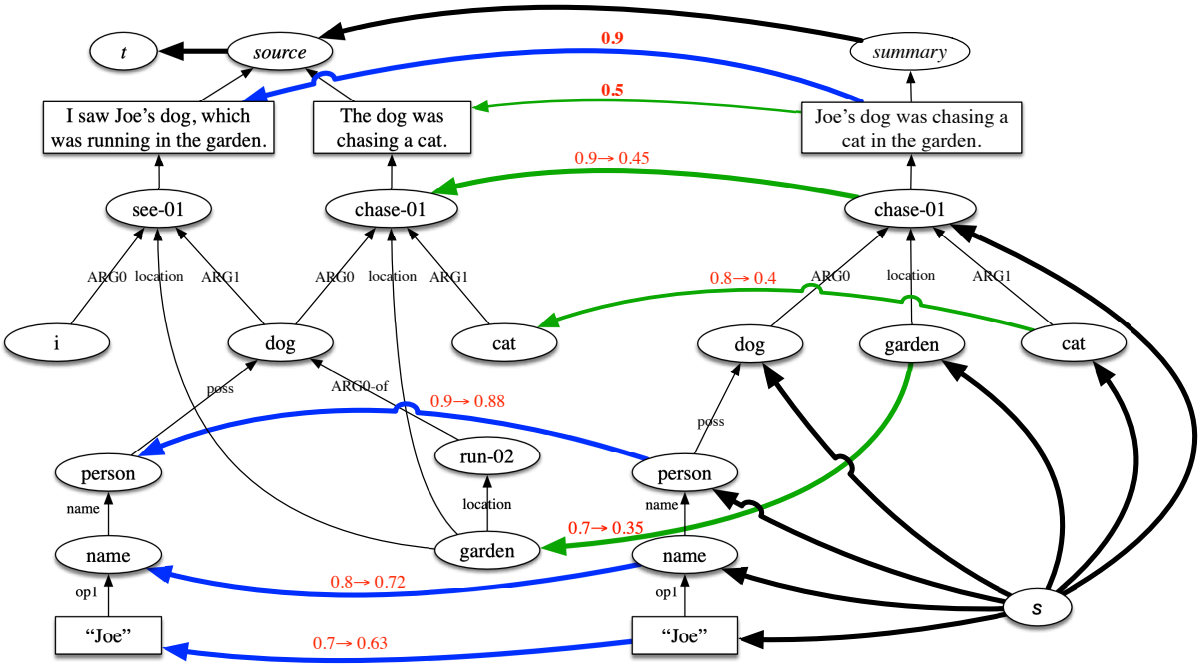

Figure 7: **Sentence Scaled Capacities**. The scaled capacities with alignment edges colored by related sentence with bold red capacities. The concept and attribute node capacity updates are in red with the initial capacity on the left and the sentence skew updated on the right.

B.1. Document Node Capacities

The capacities of the document node alignment edges are computed using the document node embeddings (see Sec A.1) as $\mathrm{C}_d(\boldsymbol{n_1}, \boldsymbol{n_2})$d defined in Equation 13.

B.2. Sentence Node Capacities

Sentence edge alignment capacities computed from sentence node pairs (see Equation 13). A sentence skew is also computed to linearly dampen concept and attribute capacities per sentence cosine similarity, which is scaled by the hyperparameter $\gamma \in [0, 1]$:

$$\mathrm{ssk}(\boldsymbol{s_1}, \boldsymbol{s_2}) \triangleq \mathrm{C}_s(\boldsymbol{n_1}, \boldsymbol{n_2})\, \gamma + (1 - \gamma) \quad (14)$$

B.3. Concept Node Capacities

Concepts alignment capacities are computed like sentences, but are scaled by their alignment capacity sentence skew:

$$\{(\boldsymbol{n_1}, \boldsymbol{n_2}) \in \mathcal{E}_a, (\boldsymbol{s_1}, \boldsymbol{s_2}) \in \mathcal{E}_a \mid \boldsymbol{n_1} \in \mathcal{D}(\boldsymbol{s_1}) \wedge \boldsymbol{n_2} \in \mathcal{D}(\boldsymbol{s_2})\},$$
$$\mathrm{C}_c(\boldsymbol{n_1}, \boldsymbol{n_2}) \triangleq \mathrm{C}(\boldsymbol{n_1}, \boldsymbol{n_2}) \cdot \mathrm{ssk}(\boldsymbol{s_1}, \boldsymbol{s_2}) \quad (15)$$

where $\mathcal{E}_a$ is the set of alignment edges, and $(\boldsymbol{n_1}, \boldsymbol{n_s})$ is a bipartite aligned node descendant pair of their respective sentence nodes. The capacity in Equation 15 is set to a capacity maximum value of 1 when the variable name is the same for the node pair, which happens for reentrancies or co-referenced AMR nodes between the source component and summary component.

The capacity in Equation 15 is set to the max value 1 when the variable name is the same for both concept nodes (i.e. "c" in "`c / chase-01`"). This is an example of an AMR reference between the source component and summary component.

Otherwise, it is scaled as a function of the sentence embedding to which it belongs using the sentence skew defined in Equation 13. Fig 7 shows a strongly semantic similarity with capacity 0.9, and weakly semantically similar sentence with capacity 0.5, between the bipartite components. The strongly similar sentence uses blue to denote the sentence alignment and that sentence alignment's affect on the concept and attribute alignment edges. For example, the strongly similar sentence's `person` only loses a flow value of 0.02 after applying the sentence skew, while the weakly similar sentence's `cat` loses half its capacity (0.4).

B.4. Attribute Node Capacities

Attribute node capacities are calculated in the same way as concept nodes defined in Equation 15. However, they have no variable, so the capacity definition given in Equation 13 is used directly with the attribute's sentence skew scaled embeddings:

$$\mathrm{C}_a(\boldsymbol{n_1}, \boldsymbol{n_2}) \triangleq \mathrm{C}(\boldsymbol{n_1}, \boldsymbol{n_2}) \cdot \mathrm{ssk}(\boldsymbol{s_1}, \boldsymbol{s_2}) \quad (16)$$

C. Alignment Graph Examples

This appendix contains graphs were generated by our source for several steps of the alignment method.

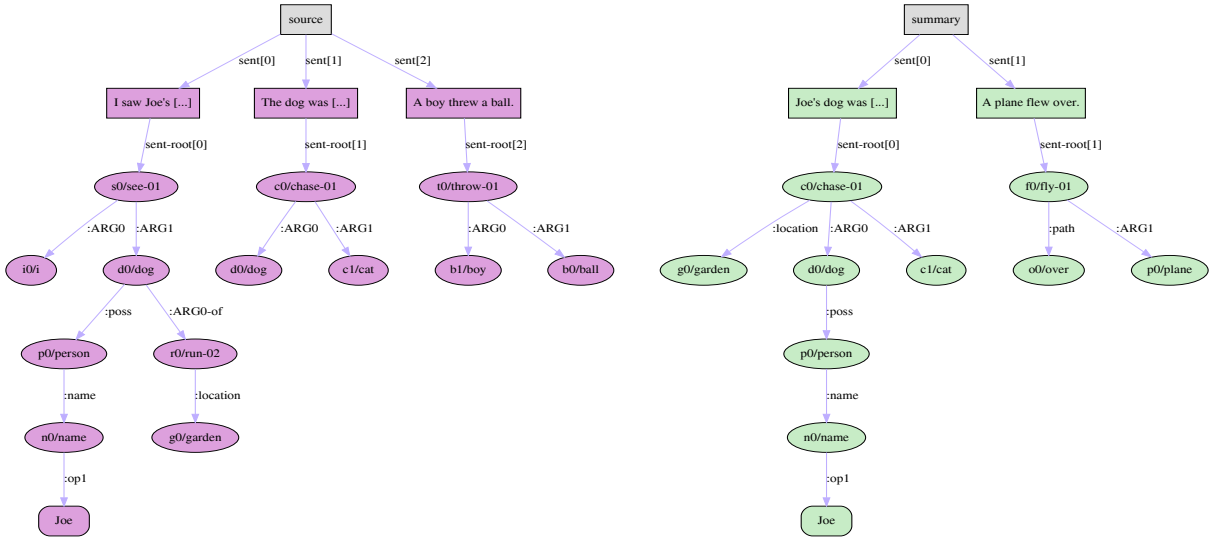

Figure 8: **Disconnected Source and Summary AMR Graphs**. Left: the source component (purple nodes) including the sentences: a) *"I saw Joe's dog, which was running in the garden."*, b) *"The dog was chasing a cat."*, c) *"A boy threw a ball."*, and d) *"A plane flew over."*. Right: the summary component (green nodes) including the sentence *"Joe's dog was chasing a cat in the garden."* Each AMR is tied with sentence, then root notes (see Sec 3.3).

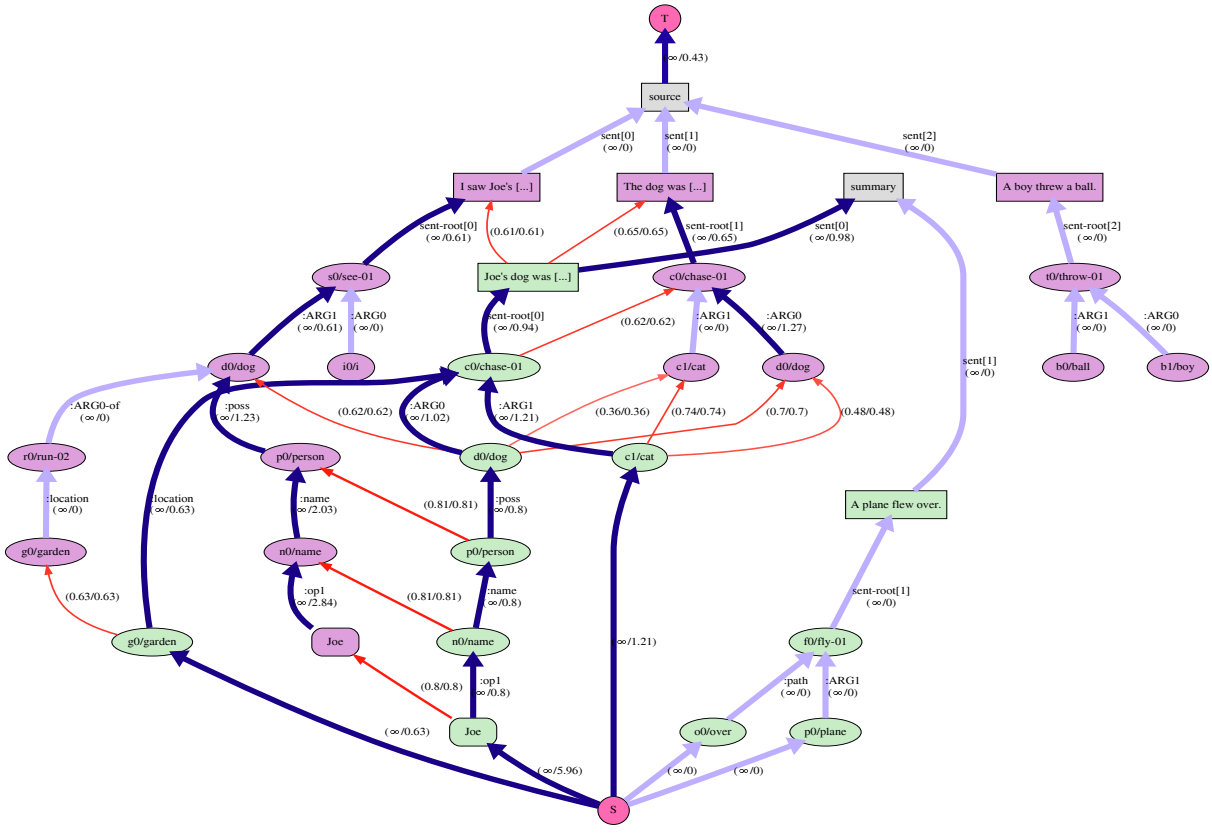

Figure 9: **Source Component Alignment**. The flow from the summary component (green nodes) to the source component (purple nodes) after capacity constriction (see Sec 3.4.3). The role edges are in blue and the alignment edges are in red with all edges' width representing the capacity. The flow is represented by the darkness of the edges' color. The capacity and flow is shown in parenthesis.

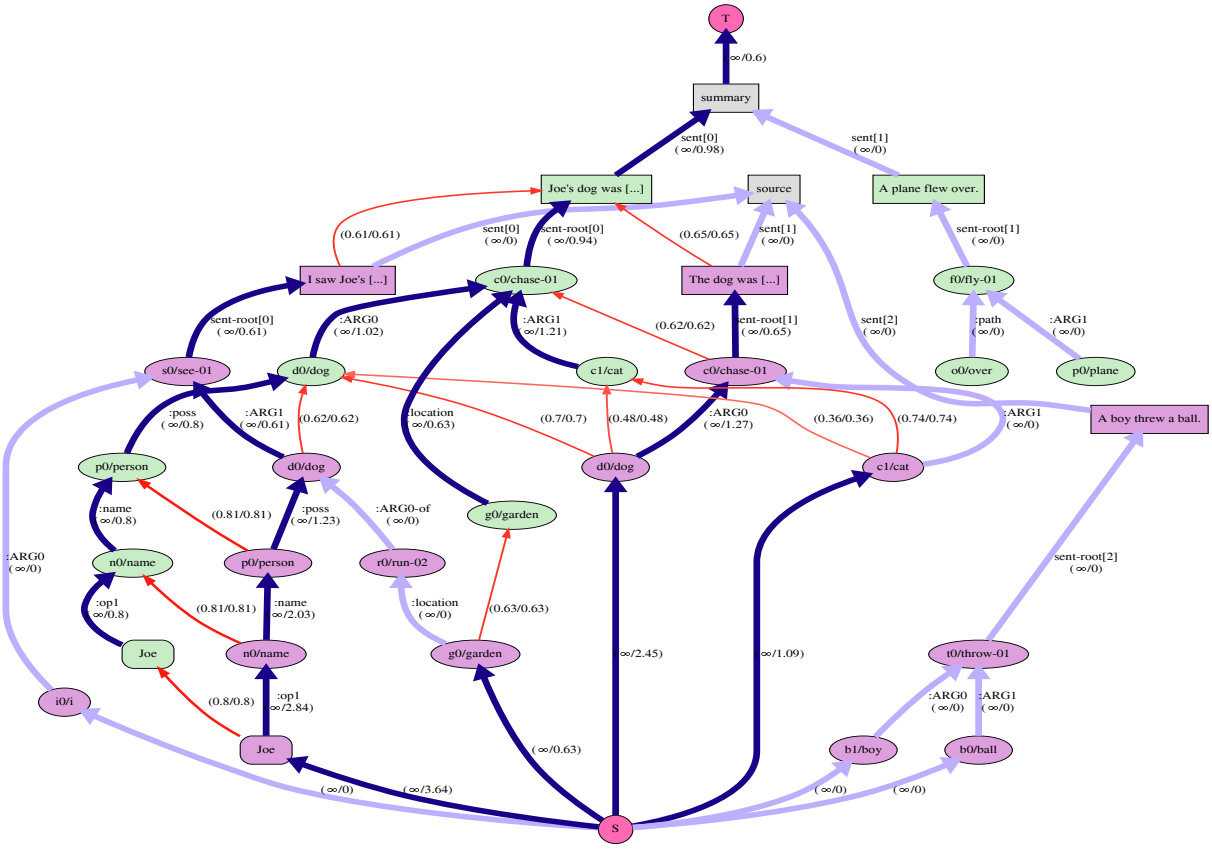

Figure 10: **Source Component Alignment**. The alternated graph with flow from the summary to the source. Like Fig 11, this is taken after the capacity constriction step in Sec 3.4.3.

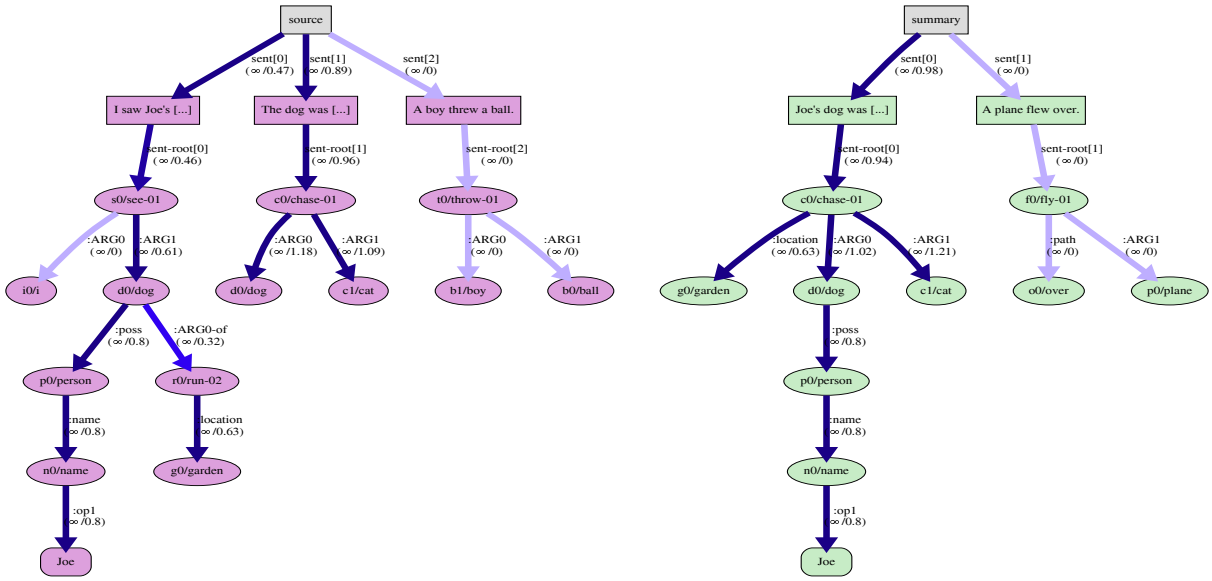

Figure 11: **Final Flow**. The final flow of both components rendered without the capacity edges in the same format as the disconnected components in Fig 8. The sentence *"A boy threw a ball."* has zero flow, and thus indicates this entire sentence is not summarized. Similarly, the `i0/i` node has no flow so it will also not be summarized. The light blue role edge incoming to `r0/run-02` node has less flow, so its partial summarization follows from the "garden" mention in the summary, but "running" is missing. The solid blue in all edges of the summary component indicates the entire summarization is represented in the source. The fact that the `c1/cat` has a higher flow value than one means it has higher presentation in the source and has more than one alignment.