Toward Abstractive Summarization Using Semantic Representations

Fei Liu Jeffrey Flanigan Sam Thomson Norman Sadeh Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{feiliu, jflanigan, sthomson, sadeh, nasmith}@cs.cmu.edu

Abstract

We present a novel abstractive summarization framework that draws on the recent development of a treebank for the Abstract Meaning Representation (AMR). In this framework, the source text is parsed to a set of AMR graphs, the graphs are transformed into a summary graph, and then text is generated from the summary graph. We focus on the graph-to-graph transformation that reduces the source semantic graph into a summary graph, making use of an existing AMR parser and assuming the eventual availability of an AMR-to-text generator. The framework is data-driven, trainable, and not specifically designed for a particular domain. Experiments on gold-standard AMR annotations and system parses show promising results. Code is available at: https://github.com/summarization

1 Introduction

Abstractive summarization is an elusive technological capability in which textual summaries of content are generated *de novo*. Demand is on the rise for high-quality summaries not just for lengthy texts (e.g., books; Bamman and Smith, 2013) and texts known to be prohibitively difficult for people to understand (e.g., website privacy policies; Sadeh et al., 2013), but also for non-textual media (e.g., videos and image collections; Kim et al., 2014; Kuznetsova et al., 2014; Zhao and Xing, 2014), where extractive and compressive summarization techniques simply do not suffice. We believe that the challenge of abstractive summarization deserves renewed attention and propose that recent developments in semantic analysis have an important role to play.

We conduct the first study exploring the feasibility of an abstractive summarization system based on transformations of semantic representations such as the Abstract Meaning Representation (AMR; Banarescu et al., 2013). Example sentences and their AMR graphs are shown in Fig. 1. AMR has much in common with earlier formalisms (Kasper, 1989; Dorr et al., 1998); today an annotated corpus comprised of over 20,000 AMR-analyzed English sentences (Knight et al., 2014) and an automatic AMR parser (JAMR; Flanigan et al., 2014) are available.

In our framework, summarization consists of three steps illustrated in Fig. 1: (1) parsing the input sentences to individual AMR graphs, (2) combining and transforming those graphs into a single summary AMR graph, and (3) generating text from the summary graph. This paper focuses on step 2, treating it as a structured prediction problem. We assume text documents as input[1] and use JAMR for step 1. We use a simple method to read a bag of words off the summary graph, allowing evaluation with ROUGE-1, and leave full text generation from AMR (step 3) to future work.

The graph summarizer, described in §4, first merges AMR graphs for each input sentence through a *concept merging* step, in which coreferent nodes of the graphs are merged; a *sentence conjunction* step, which connects the root of each sentence's AMR graph to a dummy "ROOT" node; and an optional

[1]In principle, the framework could be applied to other inputs, such as image collections, if AMR parsers became available for them.

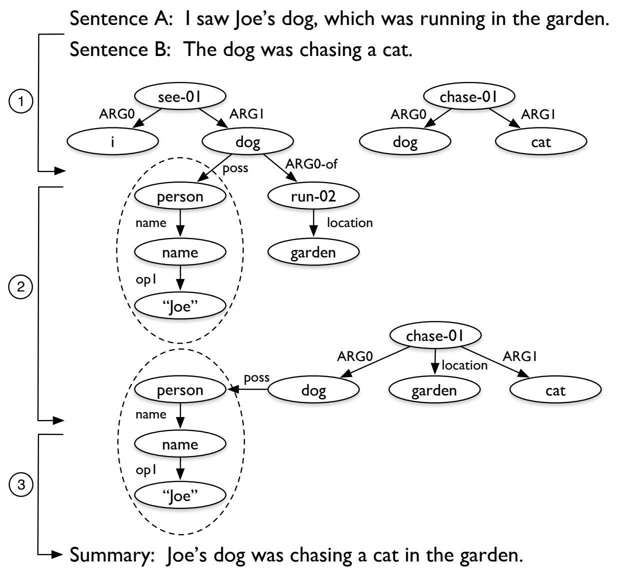


Figure 1: A toy example. Sentences are parsed into individual AMR graphs in step 1; step 2 conducts graph transformation that produces a single summary AMR graph; text is generated from the summary graph in step 3.

graph expansion step, where additional edges are added to create a fully dense graph on the sentence-level. These steps result in a single connected *source graph*. A subset of the nodes and arcs from the source graph are then selected for inclusion in the *summary graph*. Ideally this is a condensed representation of the most salient semantic content from the source.

We briefly review AMR and JAMR (§2), then present the dataset used in this paper (§3). The main algorithm is presented in §4, and we discuss our simple generation step in §5. Our experiments (§6) measure the intrinsic quality of the graph transformation algorithm as well as the quality of the terms selected for the summary (using ROUGE-1). We explore variations on the transformation and the learning algorithm, and show oracle upper bounds of various kinds.

2 Background: Abstract Meaning Representation and JAMR

AMR provides a whole-sentence semantic representation, represented as a rooted, directed, acyclic graph (Fig. 1). Nodes of an AMR graph are labeled with *concepts*, and edges are labeled with *relations*. Concepts can be English words ("dog"), PropBank event predicates ("chase-01," "run-02"), or special keywords ("person"). For example, "chase-01" represents a PropBank roleset that corresponds to the first sense of "chase". According to Banarescu et al. (2013), AMR uses approximately 100 relations. The rolesets and core semantic relations (e.g., ARG0 to ARG5) are adopted from the PropBank annotations in OntoNotes (Hovy et al., 2006). Other semantic relations include "location," "mode," "name," "time," and "topic." The AMR guidelines[2] provide more detailed descriptions. Banarescu et al. (2013) describe AMR Bank, a 20,341-sentence corpus annotated with AMR by experts.

Step 1 of our framework converts input document sentences into AMR graphs. We use a statistical semantic parser, JAMR (Flanigan et al., 2014), which was trained on AMR Bank. JAMR's current performance on our test dataset is 63% F-score.[3] We will analyze the effect of AMR parsing errors by comparing JAMR output with gold-standard annotations of input sentences in the experiments (§6).

In addition to predicting AMR graphs for each sentence, JAMR provides alignments between spans of words in the source sentence and fragments of its predicted graph. For example, a graph fragment headed by "date-entity" could be aligned to the tokens "April 8, 2002." We use these alignments in our simple text generation module (step 3; §5).

3 Dataset

To build and evaluate our framework, we require a dataset that includes inputs and summaries, each with gold-standard AMR annotations.[4] This allows us to use a statistical model for step 2 (graph summarization) and to separate its errors from those in step 1 (AMR parsing), which is important in determining whether this approach is worth further investment.

Fortunately, the "proxy report" section of the AMR Bank (Knight et al., 2014) suits our needs. A

[2] `http://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf`

[3] AMR parse quality is evaluated using smatch (Cai and Knight, 2013), which measures the accuracy of concept and relation predictions. JAMR was trained on the in-domain training portion of LDC2014T12 for our experiments.

[4] Traditional multi-document summarization datasets, such as the ones used in DUC and TAC competitions, do not have gold-standard AMR annotations.

	# Docs.	Ave. # Sents.		Source Graph		
		Summ.	Doc.	Nodes	Edges	Expand
Train	298	1.5	17.5	127	188	2,670
Dev.	35	1.4	19.2	143	220	3,203
Test	33	1.4	20.5	162	255	4,002

Table 1: Statistics of our dataset. "Expand" shows the number of edges after performing graph expansion. The numbers are averaged across all documents in the split. We use the official split, dropping one training document for which no summary sentences were annotated.

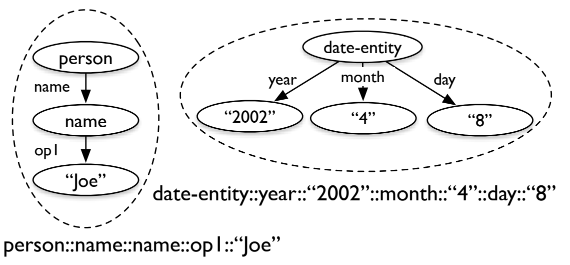

person
name
name
op1
"Joe"
person::name::name::op1::"Joe"
date-entity
year
month
day
"2002"
"4"
"8"
date-entity::year::"2002"::month::"4"::day::"8"

Figure 2: Graph fragments are collapsed into a single concept and assigned a new concept label.

proxy report is created by annotators based on a single newswire article, selected from the English Gigaword corpus. The report header contains metadata about date, country, topic, and a short summary. The report body is generated by editing or rewriting the content of the newswire article to approximate the style of an analyst report. Hence this is a single document summarization task. All sentences are paired with gold-standard AMR annotations. Table 1 provides an overview of our dataset.

4 Graph Summarization

Given AMR graphs for all of the sentences in the input (step 1), graph summarization transforms them into a single summary AMR graph (step 2). This is accomplished in two stages: source graph construction (§4.1); and subgraph prediction (§4.2).

4.1 Source Graph Construction

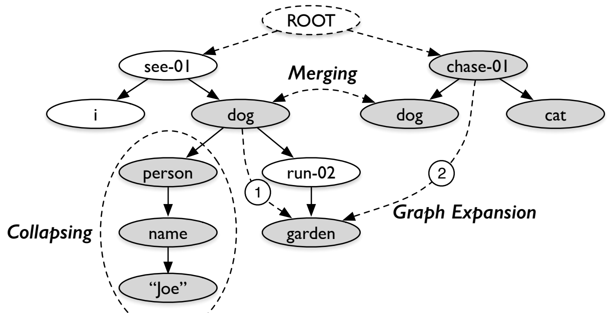
The "source graph" is a single graph constructed using the individual sentences' AMR graphs by merging identical concepts. In the AMR formalism, an entity or event is canonicalized and represented by a single graph fragment, regardless of how many times it is referred to in the sentence. This principle can be extended to multiple sentences, ideally resulting in a source graph with no redundancy. Because repeated mentions of a concept in the input can signal its importance, we will later encode the frequency of mentions as a feature used in subgraph prediction.

Concept merging involves collapsing certain graph fragments into a single concept, then merging all concepts that have the same label. We collapse the graph fragments that are headed by either a date-entity ("date-entity") or a named entity ("name"), if the fragment is a flat structure. A collapsed named entity is further combined with its parent (e.g., "person") into one concept node if it is the only child of the parent. Two such graph fragments are illustrated in Fig. 2. We choose named and date entity concepts since they appear frequently, but most often refer to different entities (e.g., "April 8, 2002" vs. "Nov. 17"). No further collapsing is done. A collapsed graph fragment is assigned a new label by concatenating the consisting concept and edge labels. Each fragment that is collapsed into a new concept node can then only be merged with other identical fragments. This process won't recognize coreferent concepts like "Barack Obama" = "Obama" and "say-01" = "report-01," but future work may incorporate both entity coreference resolution and event coreference resolution, as concept nodes can represent either.

Due to the concept merging step, a pair of concepts may now have multiple labeled edges between them. We merge all such edges between a given pair of concepts into a single unlabeled edge. We remember the two most common labels in such a group, which are used in the edge "Label" feature (Table 3).

To ensure that the source graph is connected, we add a new "ROOT" node and connect it to every concept that was originally the root of a sentence graph (see Fig. 3). When we apply this procedure to the documents in our dataset (§3), source graphs contain 144 nodes and 221 edges on average.

We investigated how well these automatically constructed source graphs cover the gold-standard summary graphs produced by AMR annotators. Ideally, a source graph should cover all of the gold-standard edges, so that summarization can be accomplished by selecting a subgraph of the source



Sentence A: I saw Joe's dog, which was running in the garden.
Sentence B: The dog was chasing a cat.

Figure 3: A source graph formed from two sentence AMR graphs. Concept collapsing, merging, and graph expansion are demonstrated. Edges are unlabeled. A "ROOT" node is added to ensure connectivity. (1) and (2) are among edges added through the optional expansion step, corresponding to sentence- and document-level expansion, respectively. Concept nodes included in the summary graph are shaded.

	Summary Edge Coverage (%)			
			Expand	
	Labeled	Unlabeled	Sent.	Doc.
Train	64.8	67.0	75.5	84.6
Dev.	77.3	78.6	85.4	91.8
Test	63.0	64.7	75.0	83.3

Table 2: Percentage of summary edges that can be covered by an automatically constructed source graph.

graph (§4.2). In Table 2, columns one and two report labeled and unlabeled edge coverage. 'Unlabeled' counts edges as matching if both the source and destination concepts have identical labels, but ignores the edge label.

In order to improve edge coverage, we explore expanding the source graph by adding every possible edge between every pair of concepts within the same sentence. We also explored adding every possible edge between every pair of concepts in the entire source graph. Edges that are newly introduced during expansion receive a default label 'null'. We report unlabeled edge coverage in Table 2, columns three and four, respectively. Subgraph prediction became infeasable with the document-level expansion, so we conducted our experiments using only sentence-level expansion. Sentence-level graph expansion increases the average number of edges by a factor of 15, to 3,292. Fig. 3 illustrates the motivation. Document-level expansion covers the gold-standard summary edge "chase-01" → "garden," yet the expansion is computationally prohibitive; sentence-level expansion adds an edge "dog" → "garden," which enables the prediction of a structure with similar semantic meaning: "Joe's dog was in the garden chasing a cat."

4.2 Subgraph Prediction

We pose the selection of a summary subgraph from the source graph as a structured prediction problem that trades off among including important information without altering its meaning, maintaining brevity, and producing fluent language (Nenkova and McKeown, 2011). We incorporate these concerns in the form of features and constraints in the statistical model for subgraph selection.

Let $G = (V, E)$ denote the merged source graph, where each node $v \in V$ represents a unique concept and each directed edge $e \in E$ connects two concepts. G is a connected, directed, node-labeled graph. Edges in this graph are unlabeled, and edge labels are not predicted during subgraph selection. We seek to maximize a score that factorizes over graph nodes and edges that are included in the summary graph. For subgraph (V', E'):

$$score(V', E'; \boldsymbol{\theta}, \boldsymbol{\psi}) = \sum_{v \in V'} \boldsymbol{\theta}^\top \mathbf{f}(v) + \sum_{e \in E'} \boldsymbol{\psi}^\top \mathbf{g}(e) \quad (1)$$

where $\mathbf{f}(v)$ and $\mathbf{g}(e)$ are the feature representations of node v and edge e, respectively. We describe node and edge features in Table 3. $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are vectors of empirically estimated coefficients in a linear model.

We next formulate the selection of the subgraph using integer linear programming (ILP; §4.2.1) and describe supervised learning for the parameters (coefficients) from a collection of source graphs paired with summary graphs (§4.2.2).

4.2.1 Decoding

We cast decoding as an ILP whose constraints ensure that the output forms a connected subcomponent of the source graph. We index source graph concept nodes by i and j, giving the "ROOT" node

Node Features	Concept	Identity feature for concept label
	Freq	Concept freq in the input sentence set; one binary feature defined for each frequency threshold $\tau = 0/1/2/5/10$
	Depth	Average and smallest depth of node to the root of the sentence graph; binarized using 5 depth thresholds
	Position	Average and foremost position of sentences containing the concept; binarized using 5 position thresholds
	Span	Average and longest word span of concept; binarized using 5 length thresholds; word spans obtained from JAMR
	Entity	Two binary features indicating whether the concept is a named entity/date entity or not
	Bias	Bias term, 1 for any node
Edge Features	Label	First and second most frequent edge labels between concepts; relative freq of each label, binarized by 3 thresholds
	Freq	Edge frequency (w/o label, non-expanded edges) in the document sentences; binarized using 5 frequency thresholds
	Position	Average and foremost position of sentences containing the edge (without label); binarized using 5 position thresholds
	Nodes	Node features extracted from the source and target nodes (all above node features except the bias term)
	IsExpanded	A binary feature indicating the edge is due to graph expansion or not; edge freq (w/o label, all occurrences)
	Bias	Bias term, 1 for any edge

Table 3: Node and edge features (all binarized).

index 0. Let N be the number of nodes in the graph. Let v_i and $e_{i,j}$ be binary variables. v_i is 1 iff source node i is included; $e_{i,j}$ is 1 iff the directed edge from node i to node j is included.

The ILP objective to be maximized is Equation 1, rewritten here in the present notation:

$$\sum_{i=1}^{N} v_i \underbrace{\boldsymbol{\theta}^\top \mathbf{f}(i)}_{\text{node score}} + \sum_{(i,j)\in E} e_{i,j} \underbrace{\boldsymbol{\psi}^\top \mathbf{g}(i,j)}_{\text{edge score}} \quad (2)$$

Note that this objective is linear in $\{v_i, e_{i,j}\}_{i,j}$ and that features and coefficients can be folded into node and edge scores and treated as constants during decoding.

Constraints are required to ensure that the selected nodes and edges form a valid graph. In particular, if an edge (i,j) is selected ($e_{i,j}$ takes value of 1), then both its endpoints i, j must be included:

$$v_i - e_{i,j} \geq 0, \quad v_j - e_{i,j} \geq 0, \quad \forall i,j \leq N \quad (3)$$

Connectivity is enforced using a set of single-commodity flow variables $f_{i,j}$, each taking a non-negative integral value, representing the flow from node i to j. The root node sends out up to N units of flow, one to reach each included node (Equation 4). Each included node consumes one unit of flow, reflected as the difference between incoming and outgoing flow (Equation 5). Flow may only be sent over an edge if the edge is included (Equation 6).

$$\sum_i f_{0,i} - \sum_i v_i = 0, \quad (4)$$

$$\sum_i f_{i,j} - \sum_k f_{j,k} - v_j = 0, \quad \forall j \leq N, \quad (5)$$

$$N \cdot e_{i,j} - f_{i,j} \geq 0, \quad \forall i,j \leq N. \quad (6)$$

The AMR representation allows graph reentrancies (concept nodes having multiple parents), yet reentrancies are rare; about 5% of edges are re-entrancies in our dataset. In this preliminary study we force the summary graph to be tree-structured, requiring that there is at most one incoming edge for each node:

$$\sum_j e_{i,j} \leq 1, \quad \forall j \leq N. \quad (7)$$

Interestingly, the formulation so far equates to an ILP for solving the prize-collecting Steiner tree problem (PCST; Segev, 1987), which is known to be NP-complete (Karp, 1972). Our ILP formulation is modified from that of Ljubić et al. (2006). Flow-based constraints for tree structures have also previously been used in NLP for dependency parsing (Martins et al., 2009) and sentence compression (Thadani and McKeown, 2013). In our experiments, we use an exact ILP solver,[5] though many approximate methods are available.

Finally, an optional constraint can be used to fix the size of the summary graph (measured by the number of edges) to L:

$$\sum_i \sum_j e_{i,j} = L \quad (8)$$

The performance of summarization systems depends strongly on their compression rate, so systems are only directly comparable when their compression rates are similar (Napoles et al., 2011). L is supplied to the system to control summary graph size.

[5] `http://www.gurobi.com`

4.2.2 Parameter Estimation

Given a collection of input and output pairs (here, source graphs and summary graphs), a natural starting place for learning the coefficients $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ is the structured perceptron (Collins, 2002), which is easy to implement and often performs well. Alternatively, incorporating factored cost functions through a structured hinge loss leads to a structured support vector machine (SVM; Taskar et al., 2004) which can be learned with a very similar stochastic optimization algorithm. In our scenario, however, the gold-standard summary graph may not actually be a subset of the source graph. In machine translation, *ramp loss* has been found to work well in situations where the gold-standard output may not even be in the hypothesis space of the model (Gimpel and Smith, 2012). The structured perceptron, hinge, and ramp losses are compared in Table 4.

We explore learning by minimizing each of the perceptron, hinge, and ramp losses, each optimized using Adagrad (Duchi et al., 2011), a stochastic optimization procedure. Let β be one model parameter (coefficient from $\boldsymbol{\theta}$ or $\boldsymbol{\psi}$). Let $g^{(t)}$ be the subgradient of the loss on the instance considered on the t^{th} iteration with respect to β. Given an initial step size η, the update for β on iteration t is:

$$\beta^{(t+1)} \leftarrow \beta^{(t)} - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t}(g^{(\tau)})^2}} g^{(t)} \qquad (9)$$

5 Generation

Generation from AMR-like representations has received some attention, e.g., by Langkilde and Knight (1998) who described a statistical method. Though we know of work in progress driven by the goal of machine translation using AMR, there is currently no system available.

We therefore use a heuristic approach to generate a bag of words. Given a predicted subgraph, a system summary is created by finding the most frequently aligned word span for each concept node. (Recall that the JAMR parser provides these alignments; §2). The words in the resulting spans are generated in no particular order. While this is not a natural language summary, it is suitable for unigram-based summarization evaluation methods like ROUGE-1.

6 Experiments

In Table 5, we report the performance of subgraph prediction and end-to-end summarization on the test set, using gold-standard and automatic AMR parses for the input. Gold-standard AMR annotations are used for model training in all conditions. During testing, we apply the trained model to source graphs constructed using either gold-standard or JAMR parses. In all of these experiments, we use the number of edges in the gold-standard summary graph to fix the number of edges in the predicted subgraph, allowing direct comparison across conditions.

Subgraph prediction is evaluated against the gold-standard AMR graphs on summaries. We report precision, recall, and F_1 for nodes, and F_1 for edges.[6]

Oracle results for the subgraph prediction stage are obtained using the ILP decoder to minimize the cost of the output graph, given the gold-standard. We assign wrong nodes and edges a score of -1, correct nodes and edges a score of 0, then decode with the same structural constraints as in subgraph prediction. The resulting graph is the best summary graph in the hypothesis space of our model, and provides an upper bound on performance achievable within our framework. Oracle performance on node prediction is in the range of 80% when using gold-standard AMR annotations, and 70% when using JAMR output. Edge prediction has lower performance, yielding 52.2% for gold-standard and 31.1% for JAMR parses. When graph expansion was applied, the numbers increased to 64% and 46.7%, respectively. The uncovered summary edge (i.e., those not covered by source graph) is a major source for low recall values on edge prediction (see Table 2); graph expansion slightly alleviates this issue.

Summarization is evaluated by comparing system summaries against reference summaries, using ROUGE-1 scores (Lin, 2004)[7]. System summaries are generated using the heuristic approach presented in §5: given a predicted subgraph, the approach finds the most frequently aligned word span for each concept node, and then puts them together as a bag of words. ROUGE-1 is particularly usefully for eval-

[6]Precision, recall, and F_1 are equal since the number of edges is fixed.

[7]ROUGE version 1.5.5 with options '-e data -n 4 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -a -x'

$$\begin{array}{lrcl}
\text{Structured perceptron loss:} & -score(G^*) & + & \max_G score(G) \\
\text{Structured hinge loss:} & -score(G^*) & + & \max_G (score(G) + cost(G; G^*)) \\
\text{Structured ramp loss:} & -\max_G (score(G) - cost(G; G^*)) & + & \max_G (score(G) + cost(G; G^*))
\end{array}$$

Table 4: Loss functions minimized in parameter estimation. G^* denotes the gold-standard summary graph. $score(\cdot)$ is as defined in Equation 1. $cost(G; G^*)$ penalizes each vertex or edge in $G \cup G^* \setminus (G \cap G^*)$. Since $cost$ factors just like the scoring function, each $\max$ operation can be accomplished using a variant of ILP decoding (§4.2.1) in which the cost is incorporated into the linear objective while the constraints remain the same.

		Subgraph Prediction				**Summarization**		
		Nodes			**Edges**	**ROUGE-1**		
		P (%)	R (%)	F (%)	F (%)	P (%)	R (%)	F (%)
gold-standard parses	Perceptron	39.6	46.1	42.6	24.7	41.4	27.1	32.3
	Hinge	41.2	47.9	44.2	26.4	42.6	28.3	33.5
	Ramp	**54.7**	**63.5**	**58.7**	**39.0**	**51.9**	**39.0**	**44.3**
	Ramp + Expand	53.0	61.3	56.8	36.1	50.4	37.4	42.8
	Oracle	75.8	86.4	80.7	52.2	89.1	52.8	65.8
	Oracle + Expand	78.9	90.1	83.9	64.0			
JAMR parses	Perceptron	42.2	48.9	45.2	14.5	46.1	35.0	39.5
	Hinge	41.7	48.3	44.7	15.8	44.9	33.6	38.2
	Ramp	**48.1**	**55.6**	**51.5**	**20.0**	50.6	40.0	44.4
	Ramp + Expand	47.5	54.6	50.7	19.0	**51.2**	**40.0**	**44.7**
	Oracle	64.1	74.8	68.9	31.1	87.5	43.7	57.8
	Oracle + Expand	66.9	76.4	71.2	46.7			

Table 5: Subgraph prediction and summarization (to bag of words) results on test set. Gold-standard AMR annotations are used for model training in all conditions. "+ Expand" means the result is obtained using source graph with expansion; edge performance is measured ignoring labels.

uating such less well-formed summaries, such as those generated from speech transcripts (Liu and Liu, 2013).

Oracle summaries are produced by taking the gold-standard AMR parses of the reference summary, obtaining the most frequently aligned word span for each unique concept node using the JAMR aligner (§2), and then generating a bag of words summary. Evaluation of oracle summaries is performed in the same manner as for system summaries. The above process does not involve graph expansion, so summarization performance is the same for the two conditions "Oracle" and "Oracle + Expand."

We find that JAMR parses are a large source of degradation of edge prediction performance, and a smaller but still significant source of degradation for concept prediction. Surprisingly, using JAMR parses leads to slightly improved ROUGE-1 scores. Keep in mind, though, that under our bag-of-words generator, ROUGE-1 scores only depend on concept prediction and are unaffected by edge prediction. The oracle summarization results, 65.8% and 57.8% F_1 scores for gold-standard and JAMR parses, respectively, further suggest that improved graph summarization models (step 2) might benefit from future improvements in AMR parsing (step 1).

Across all conditions and both evaluations, we find that incorporating a cost-aware loss function (hinge vs. perceptron) has little effect, but that using ramp loss leads to substantial gains.

In Table 5, we show detailed results with and without graph expansion. "+ Expand" means the results are obtained using the expanded source graph. We find that graph expansion only marginally affects system performance. Graph expansion slightly hurts the system performance on edge prediction. For example, using ramp loss with JAMR parser as input, we obtained 50.7% and 19.0% for node and edge prediction with graph expansion; 51.5% and 20.0%

without edge expansion. On the other hand, it increases the oracle performance by a large margin. This suggests that with more training data, or a more sophisticated model that is able to better discriminate among the enlarged output space, graph expansion still has promise to be helpful.

7 Related and Future Work

According to Dang and Owczarzak (2008), the majority of competitive summarization systems are extractive, selecting representative sentences from input documents and concatenating them to form a summary. This is often combined with sentence compression, allowing more sentences to be included within a budget. ILPs and approximations have been used to encode compression and extraction (McDonald, 2007; Martins and Smith, 2009; Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013; Li et al., 2014). Other decoding approaches have included a greedy method exploiting submodularity (Lin and Bilmes, 2010), document reconstruction (He et al., 2012), and graph cuts (Qian and Liu, 2013), among others.

Previous work on abstractive summarization has explored user studies that compare extractive with NLG-based abstractive summarization (Carenini and Cheung, 2008). Ganesan et al. (2010) propose to construct summary sentences by repeatedly searching the highest scored graph paths. (Gerani et al., 2014) generate abstractive summaries by modifying discourse parse trees. Our work is similar in spirit to Cheung and Penn (2014), which splices and recombines dependency parse trees to produce abstractive summaries. In contrast, our work operates on semantic graphs, taking advantage of the recently developed AMR Bank.

Also related to our work are graph-based summarization methods (Vanderwende et al., 2004; Erkan and Radev, 2004; Mihalcea and Tarau, 2004). Vanderwende et al. (2004) transform input to logical forms, score nodes using PageRank, and grow the graph from high-value nodes using heuristics. In Erkan and Radev (2004) and Mihalcea and Tarau (2004), the graph connects surface terms that co-occur. In both cases, the graphs are constructed based on surface text; it is not a representation of propositional semantics like AMR. However, future work might explore similar graph-based calculations to contribute features for subgraph selection in our framework.

Our constructed source graph can easily reach ten times or more of the size of a sentence dependency graph. Thus more efficient graph decoding algorithms, e.g., based on Lagrangian relaxation or approximate algorithms, may be explored in future work. Other future directions may include jointly performing subgraph and edge label prediction; exploring a full-fledged pipeline that consists of an automatic AMR parser, a graph-to-graph summarizer, and a AMR-to-text generator; and devising an evaluation metric that is better suited to abstractive summarization.

Many domains stand to eventually benefit from summarization. These include books, audio/video segments, and legal texts.

8 Conclusion

We have introduced a statistical abstractive summarization framework driven by the Abstract Meaning Representation. The centerpiece of the approach is a structured prediction algorithm that transforms semantic graphs of the input into a single summary semantic graph. Experiments show the approach to be promising and suggest directions for future research.

Acknowledgments

The authors thank three anonymous reviewers for their insightful input. We are grateful to Nathan Schneider, Kevin Gimpel, Sasha Rush, and the ARK group for valuable discussions. The research was supported by NSF grant SaTC-1330596, DARPA grant FA8750-12-2-0342 funded under the DEFT program, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, and by IARPA via DoI/NBC contract number D12PC00337. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsors.

References

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual de-composition and multi-task learning. In *Proceedings of ACL*.

David Bamman and Noah A. Smith. 2013. New alignment methods for discriminative book summarization. In *arXiv:1305.1319*.

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 Linguistic Annotation Workshop*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of ACL*.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. NLG-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG)*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of EMNLP*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Proceedings of Text Analysis Conference (TAC)*.

Bonnie Dorr, Nizar Habash, and David Traum. 1998. A thematic hierarchy for efficient generation from lexical-conceptual structure. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup: Proceedings of the Third Conference of the Association for Machine Translation in the Americas*, Lecture Notes in Computer Science. Springer.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

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 ACL*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of COLING*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of EMNLP*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the NAACL Workshop on Integer Linear Programming for Natural Langauge Processing*.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL-HLT*.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *Proceedings of AAAI*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of NAACL*.

Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Springer US.

Robert T. Kasper. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Proceedings of the DARPA Speech and Natural Language Workshop*.

Gunhee Kim, Leonid Sigal, and Eric P. Xing. 2014. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Proceedings of CVPR*.

Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. Abstract meaning representation (AMR) annotation release 1.0 LDC2014T12. Web Download. Philadelphia: Linguistic Data Consortium.

Polina Kuznetsova, Vicente Ordonez, Tamara L. Berg, and Yejin Choi. 2014. TREETALK: Composition and compression of trees for image descriptions. *Transactions of ACL*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of COLING*.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse tree. In *Proceedings of EMNLP*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of ACL Workshop on Text Summarization Branches Out*.

Fei Liu and Yang Liu. 2013. Towards abstractive speech summarization: Exploring unsupervised and supervised approaches for spoken utterance compression. *IEEE Transactions on Audio, Speech, and Language Processing*.

Ivana Ljubić, René Weiskircher, Ulrich Pferschy, Gunnar W. Klau, Petra Mutzel, and Matteo Fischetti. 2006. An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem. In *Mathematical Progamming, Series B*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Andre F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of ECIR*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP*.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating Sentence Compression: Pitfalls and Suggested Remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 91–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*.

Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. McDonald, Joel R. Reidenberg, Noah A. Smith, Fei Liu, N. Cameron Russell, Florian Schaub, and Shomir Wilson. 2013. The usable privacy policy project. *Technical Report, CMU-ISR-13-119, Carnegie Mellon University*.

Arie Segev. 1987. The Node-Weighted Steiner Tree Problem. *Networks*, 17(1):1–17.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Lucy Vanderwende, Michele Banko, , and Arul Menezes. 2004. Event-centric summary generation. In *Proceedings of DUC*.

Bin Zhao and Eric P. Xing. 2014. Quasi real-time summarization for consumer videos. In *Proceedings of CVPR*.