Abstract Meaning Representation for Multi-Document Summarization

Kexin Liao **Logan Lebanoff** **Fei Liu**
Computer Science Department
University of Central Florida, Orlando, FL 32816, USA
{ericaryo,loganlebanoff}@knights.ucf.edu feiliu@cs.ucf.edu

Abstract

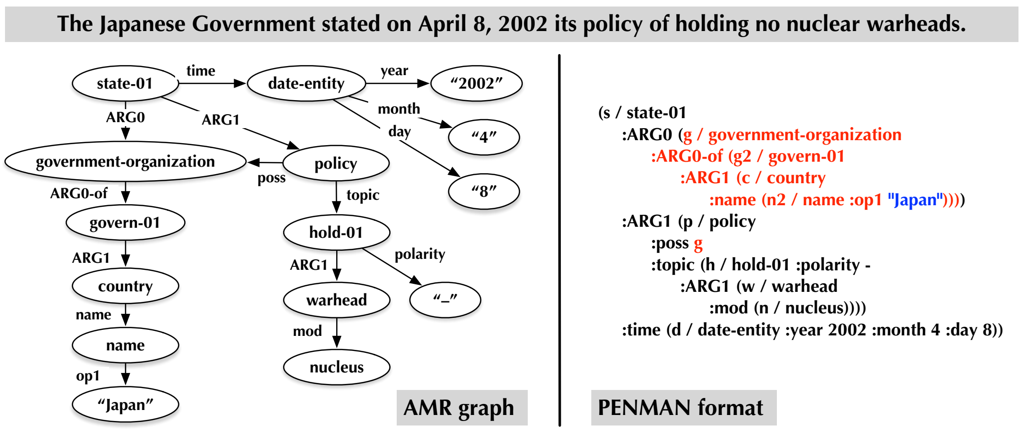
Generating an abstract from a collection of documents is a desirable capability for many real-world applications. However, abstractive approaches to multi-document summarization have not been thoroughly investigated. This paper studies the feasibility of using Abstract Meaning Representation (AMR), a semantic representation of natural language grounded in linguistic theory, as a form of content representation. Our approach condenses source documents to a set of summary graphs following the AMR formalism. The summary graphs are then transformed to a set of summary sentences in a surface realization step. The framework is fully data-driven and flexible. Each component can be optimized independently using small-scale, in-domain training data. We perform experiments on benchmark summarization datasets and report promising results. We also describe opportunities and challenges for advancing this line of research.

1 Introduction

Abstractive summarization seeks to generate concise and grammatical summaries that preserve the meaning of the original; further, they shall abstract away from the source syntactic forms. The task often involves high-level text transformations such as sentence fusion, generalization, and paraphrasing (Jing and McKeown, 1999). Recent neural abstractive summarization studies focus primarily on single-document summarization (Paulus et al., 2017; See et al., 2017). These approaches are limited by the availability of training data, and large datasets for multi-document summarization can be costly to obtain. Generating abstractive summaries for sets of source documents thus remains a challenging task.

Traditional approaches to abstractive summarization often condense the source documents to a set of "semantic units," then reconstruct abstractive summaries from these semantic units. Previous work has investigated various forms of content representation. Examples include noun/verb phrases (Genest and Lapalme, 2011; Bing et al., 2015), word-occurrence graphs (Ganesan et al., 2010), syntactic parse trees (Cheung and Penn, 2014; Gerani et al., 2014), and domain-specific templates (Pighin et al., 2014). Nonetheless, generating summary text from these heuristic forms of representation can be difficult. There is an increasing need to exploit a semantic formalism so that condensing source documents to this form and generating summary sentences from it can both be carried out in a principled way.

This paper explores Abstract Meaning Representation (AMR, Banarescu et al., 2013) as a form of content representation. AMR is a semantic formalism based on propositional logic and the neo-Davidsonian event representation (Parsons, 1990; Schein, 1993). It represents the meaning of a sentence using a rooted, directed, and acyclic graph, where nodes are concepts and edges are semantic relations. Figure 1 shows an example AMR graph. A concept node can be a PropBank frameset ("state-01"), an English word ("warhead"), a special keyword ("date-entity"), or a string literal ("Japan"). A relation can be either a core argument ("ARG0," "ARG1") or a modification relationship ("mod," "time"). The AMR representation abstracts away from surface word strings and syntactic structure, producing a language-neutral representation of meaning. The graph representation is flexible and not specifically designed for a particular domain. It is thus conceptually appealing to explore AMR for abstractive summarization.


The Japanese Government stated on April 8, 2002 its policy of holding no nuclear warheads.

```
(s / state-01
   :ARG0 (g / government-organization
      :ARG0-of (g2 / govern-01
         :ARG1 (c / country
            :name (n2 / name :op1 "Japan"))))
   :ARG1 (p / policy
      :poss g
      :topic (h / hold-01 :polarity -
         :ARG1 (w / warhead
            :mod (n / nucleus))))
   :time (d / date-entity :year 2002 :month 4 :day 8))
```

Figure 1: A example sentence, its goldstandard AMR graph, and the corresponding PENMAN format.

Our goal in this work is to generate a text abstract containing multiple sentences from a cluster of news articles discussing a given topic. The system framework includes three major components: *source sentence selection* takes a set of news articles as input and selects sets of similar sentences covering different aspects of the topic; *content planning* consumes a set of similar sentences and derives a summary graph from them; *surface realization* transforms a summary graph to a natural language sentence. This framework allows each component (source sentence selection, content planning, surface realization) to be individually optimized using small-scale, in-domain training data, reducing the need for large-scale parallel training data. Our research contributions are summarized as follows:

- we investigate AMR, a linguistically-grounded semantic formalism, as a new form of content representation for multi-document summarization. Liu et al. (2015) conducted a pilot study using AMR for single-document summarization. This paper exploits the structured prediction framework but presents a full pipeline for generating abstractive summaries from multiple source documents;
- we study to what extent the AMR parser and generator, used for mapping text to and from AMR, can impact the summarization performance. We also compare multiple source sentence selection strategies to group source sentences into clusters covering various aspects of the topic;
- we conduct extensive experiments on benchmark summarization datasets, and contrast our work with state-of-the-art baselines, including the pointer-generator networks (See et al., 2017). Results show that leveraging the AMR representation for summarization is promising. Our framework is flexible, allowing different components to be optimized independently using small-scale, in-domain datasets. We finally describe opportunities and challenges for advancing this line of research.

2 Related Work

Neural abstractive summarization has sparked great interest in recent years. These approaches focus primarily on short text summarization and single-document summarization (Rush et al., 2015; Nallapati et al., 2016). Variants of the neural encoder-decoder architecture have been exploited to reduce out-of-vocabulary tokens and word repetitions (See et al., 2017; Suzuki and Nagata, 2017), improve the attention mechanism (Chen et al., 2016; Zhou et al., 2017; Tan et al., 2017), control the summary length (Kikuchi et al., 2016), improve the learning objective and search (Ranzato et al., 2016; Huang et al., 2017), and generate summaries that are true to the original inputs (Cao et al., 2018; Song et al., 2018). Training neural models requires large amounts of data; they are often acquired by pairing news articles with titles or human-written highlights. Nonetheless, obtaining parallel data for multi-document summarization is often costly. There is thus a need to investigate alternative approaches that are less data-thirsty.

Abstractive summarization via natural language generation (NLG, Reiter and Dale, 2000; Gatt and Krahmer, 2018) is a promising line of work. The approaches often identify salient text units from source documents, arrange them in a compact form, such as domain-specific templates, and subsequently synthesize them into natural language texts (Barzilay et al., 1999; Genest and Lapalme, 2011; Oya et al., 2014; Gerani et al., 2014; Fabbrizio et al., 2014). A challenge faced by these approaches is that there

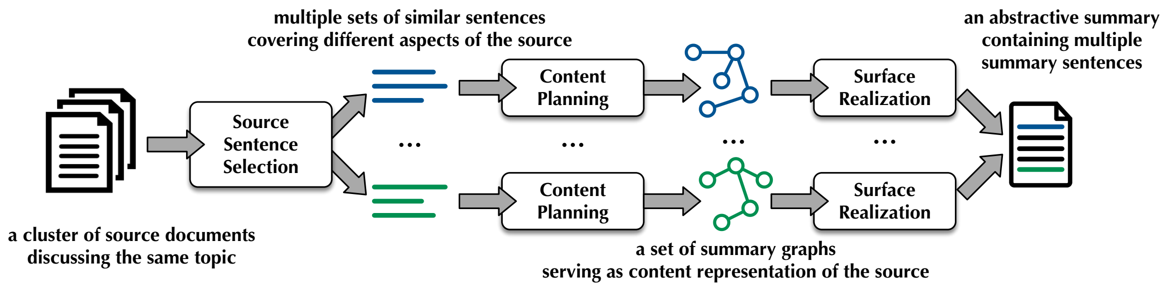

multiple sets of similar sentences
covering different aspects of the source
an abstractive summary
containing multiple
summary sentences
Source
Sentence
Selection
Content
Planning
Content
Planning
Surface
Realization
Surface
Realization
a cluster of source documents
discussing the same topic
a set of summary graphs
serving as content representation of the source

Figure 2: Our system framework, consisting of three major components.

lacks a principled means of content representation. This paper studies the feasibility of using AMR, a semantic formalism grounded in linguistic theory, for content representation. Within this framework, condensing source documents to summary AMR graphs and generating natural language sentences from summary graphs are both data-driven and not specifically designed for any domain.

The AMR formalism has demonstrated great potential on a number of downstream applications, including machine translation (Tamchyna et al., 2015), entity linking (Pan et al., 2015), summarization (Liu et al., 2015; Takase et al., 2016), question answering (Jurczyk and Choi, 2015), and machine comprehension (Sachan and Xing, 2016). Moreover, significant research efforts are dedicated to map English sentences to AMR graphs (Flanigan et al., 2014; Wang et al., 2015a; Wang et al., 2015b; Ballesteros and Al-Onaizan, 2017; Buys and Blunsom, 2017; Damonte et al., 2017; Szubert et al., 2018), and generating sentences from AMR (Flanigan et al., 2016; Song et al., 2016; Pourdamghani et al., 2016; Song et al., 2017; Konstas et al., 2017). These studies pave the way for further research exploiting AMR for multi-document summarization.

3 Our Approach

We describe our major system components in this section. In particular, content planning (§3.1) takes as input a set of similar sentences. It maps each sentence to an AMR graph, merges all AMR graphs to a connected *source graph*, then extracts a *summary graph* from the source graph via structured prediction. Surface realization (§3.2) converts a summary graph to its PENMAN representation (Banarescu et al., 2013) and generates a natural language sentence from it. Source sentence extraction (§3.3) selects sets of similar sentences from source documents discussing different aspects of the topic. The three components form a pipeline to generate an abstractive summary from a collection of documents. Figure 2 illustrates the system framework. In the following sections we describe details of the components.

3.1 Content Planning

The meaning of a source sentence is represented by a rooted, directed, and acyclic AMR graph (Banarescu et al., 2013), where nodes are concepts and edges are semantic relations. A sentence AMR graph can be obtained by applying an AMR parser to a natural language sentence. In this work we investigate two AMR parsers to understand to what extent the performance of AMR parsing may impact the summarization task. JAMR (Flanigan et al., 2014) presents the first open-source AMR parser. It introduces a two-part algorithm that first identifies concepts from the sentence and then determines the relations between them by searching for the maximum spanning connected subgraph (MSCG) from a complete graph representing all possible relations between the identified concepts. CAMR (Wang et al., 2015b) approaches AMR parsing from a different perspective. It describes a transition-based AMR parsing algorithm that transforms from a dependency parse tree to an AMR graph. We choose JAMR and CAMR because these parsers have been made open-source and both of them reported encouraging results in the recent SemEval evaluations (May, 2016; May and Priyadarshi, 2017).

Source Graph Construction. Given a set of source sentences and their AMR graphs, source graph construction attempts to consolidate all sentence AMR graphs to a connected *source graph*. This is accomplished by performing *concept merging*. Graph nodes representing the same concept, determined by the surface word form, are merged to a single node in the source graph. Importantly, we perform coreference resolution on the source documents to identify clusters of mentions of the same entity or event. Graph nodes representing these mentions are also merged. A special treatment to date entity (see "date-entity" in Figure 1) and named entity ("country") includes collapsing the subtree to a "mega-node" whose surface form is the concatenation of the consisting concepts and relations (e.g., "date-entity_:year_2002_:month_1_:day_5"). These mega-nodes can then only be merged with other identical fragments. Finally, a 'ROOT' node is introduced; it is connected to the root of each sentence AMR graph, yielding a connected source graph.

Summary Graph Extraction. We hypothesize that a summary graph, containing the salient content of source texts, can be identified from the source graph via a trainable, feature-rich structured prediction framework. The framework iteratively performs *graph decoding* and *parameter update*. The former identifies an optimal summary graph using integer linear programming, while the latter performs parameter update by minimizing a loss function that measures the difference between the system-decoded summary graph and the goldstandard summary graph.

We use $G = (V, E)$ to represent the source graph. Let v_i and $e_{i,j}$ be a set of binary variables where $v_i = 1$ (or $e_{i,j} = 1$) indicates the corresponding source graph node (or edge) is selected to be included in the summary graph. The node and edge saliency are characterized by a set of features, represented using $\mathbf{f}(i)$ and $\mathbf{g}(i, j)$, respectively. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are the feature weights. Eq. (1) presents a scoring function for any graph G. It can be factorized into a sum of scores for selected nodes and edges. In particular, $[\boldsymbol{\theta}^\top \mathbf{f}(i)]_{v_i=1}$ denotes the node score (if v_i is selected) and $[\boldsymbol{\phi}^\top \mathbf{g}(i, j)]_{e_{i,j}=1}$ denotes the edge score.

$$score(G; \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{N} v_i \underbrace{[\boldsymbol{\theta}^\top \mathbf{f}(i)]_{v_i}}_{\text{node score}} + \sum_{(i,j)\in E} e_{i,j} \underbrace{[\boldsymbol{\phi}^\top \mathbf{g}(i, j)]_{e_{i,j}}}_{\text{edge score}} \tag{1}$$

Features characterizing the graph nodes and edges are adopted from (Liu et al., 2015). They include concept/relation labels and their frequencies in the documents, average depth of the concept in sentence AMR graphs, position of sentences containing the concept/relation, whether the concept is a named entity/date entity, and the average length of concept word spans. We additionally include the concept TF-IDF score and if the concept occurs in a major news event (Wiki, 2018). All features are binarized.

The *graph decoding* process searches for the summary graph that maximizes the scoring function: $\hat{G} = \arg\max_G score(G)$. We can formulate graph decoding as an integer linear programming problem. Each summary graph corresponds to a set of values assigned to the binary variables v_i and $e_{i,j}$. We implement a set of linear constraints to ensure the decoded graph is connected, forms a tree structure, and limits to L graph nodes (Liu et al., 2015).

The *parameter update* process adjusts $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to minimize a loss function capturing the difference between the system-decoded summary graph ($\hat{G}$) and the goldstandard summary graph (G^*). Eq. (2) presents the structured perceptron loss. Minimizing this loss function with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is straightforward. However, the structured perceptron loss has problems when the goldstandard summary graph G^* is unreachable via the graph decoding process. In that case, there remains a gap between $score(\hat{G})$ and $score(G^*)$ and the loss cannot be further minimized. The structured ramp loss (Eq. (3)) addresses this problem by performing cost-augmented decoding. It introduces a cost function $cost(G; G^*)$ that can also be factored over graph nodes and edges. A cost of 1 is incurred if the system graph and the goldstandard graph disagree on whether a node (or edge) should be included. As a result, the first component of the loss function $\max_G(score(G) + cost(G; G^*))$ yields a decoded system graph that is slightly *worse* than $\hat{G} = \arg\max_G score(G)$; and the second component $\max_G(score(G) - cost(G; G^*))$ yields a graph that is slightly *better* than $\hat{G}$. The scoring difference between the two system graphs becomes the structured ramp loss we wish to minimize. This formulation is similar to the objective of the structured support vector machines (SSVMs, Tsochantaridis et al., 2005). Becase decent results have been reported by (Liu et al., 2015), we adopt structured ramp loss in all experiments.

$$\mathcal{L}_{\text{perc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \max_G score(G) - score(G^*) = score(\hat{G}) - score(G^*) \tag{2}$$

$$\mathcal{L}_{\text{ramp}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \max_G(score(G) + cost(G; G^*)) - \max_G(score(G) - cost(G; G^*)) \tag{3}$$

3.2 Surface Realization

The surface realization component converts each summary graph to a natural language sentence. This is a nontrivial task, because AMR abstracts away from the source syntactic forms and an AMR graph may correspond to a number of valid sentence realizations. In this section we perform two subtasks that have not been investigated in previous studies. We first convert the summary AMR graphs to the PENMAN format (Figure 1), which is a representation that can be understood by humans. It is also the required input form for an AMR-to-text generator. We then leverage the AMR-to-text generator to generate English sentences from summary graphs.

Algorithm 1 presents our algorithm for transforming a summary graph to the PENMAN format. Because the summary graph is rooted and acyclic, we can extract all paths from the ROOT node to all leaves. These paths are sorted by the concept indices on the path, and the paths sharing the same ancestors will be processed in order. The core of Algorithm 1 is the *while* loop (line 9–35); it writes out one concept per line. An AMR concept can have three forms: a regular form ("c / country"), string literal ("Japan"), or a re-entrance ("g"). The last two are treated as "special forms." In these cases, a relation is first written out, followed by the special concept ($\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$). "::" denotes a whitespace. A regular concept will be wrapped by a left bracket ($\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$, line 23/25) and a right bracket ($(k - k')*$")"::EOS, line 41). Finally, a proper number of closing brackets is postpended to each path (line 37–42).

To transform the PENMAN string to a summary sentence we employ the JAMR AMR-to-text generator (Flanigan et al., 2016). JAMR is the first full-fledged AMR-to-text generator. It is trained on approximately 10K sentences and achieves about 22% BLEU score on the test set. The system first transforms the input graph to a spanning tree, and then decodes it into a string using a tree-to-string transducer and a language model. The final output sentence is the highest-scoring sentence according to a feature-rich discriminatively trained linear model. We choose the JAMR AMR-to-text generator because of its competitive performance in the recent SemiEval evaluations (May and Priyadarshi, 2017).

Algorithm 1 An algorithm for transforming a summary graph to the PENMAN format.

```
Input: Triples: (src_concept, relation, tgt_concept).
 1: $r \leftarrow$ index of the ROOT concept
 2: $\mathcal{P} \leftarrow$ all paths from ROOT to leaves, sorted by
    the concept indices on the paths
 3: ▶ Set all concepts and relations as unvisited
 4: $visited[\mathcal{C}_m] \leftarrow$ FALSE, $\forall m$
 5: $visited[\mathcal{R}_{m,n}] \leftarrow$ FALSE, $\forall m, n$
 6: for $i = 1, \dots, |\mathcal{P}|$ do
 7:   $flag\_special\_concept \leftarrow$ FALSE.
 8:   $k \leftarrow 0$
 9:   while $k < |p_i|$ do
10:     $k \leftarrow k + 1$
11:     $n' \leftarrow$ index of the $k$-th concept on the path
12:     $m' \leftarrow$ index of the previous concept
13:     if $visited[\mathcal{C}_{n'}]$ = FALSE then
14:       ▶ Concept unvisited
15:       $visited[\mathcal{C}_{n'}] \leftarrow$ TRUE
16:       $visited[\mathcal{R}_{m',n'}] \leftarrow$ TRUE
17:       $output$ += $(k-1)*$TAB
18:       if $\mathcal{C}_{n'}$ is a string literal then
19:         $output$ += $\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$
20:         $flag\_special\_concept \leftarrow$ TRUE
21:         BREAK
22:       else if $k < |p_i|$ then
23:         $output$ += $\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$::EOS
24:       else
25:         $output$ += $\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$
26:       end if
27:     else if $visited[\mathcal{R}_{m',n'}]$ = FALSE then
28:       ▶ Concept reentrance
29:       $visited[\mathcal{R}_{m',n'}] \leftarrow$ TRUE
30:       $output$ += $(k-1)*$TAB
31:       $output$ += $\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$
32:       $flag\_special\_concept \leftarrow$ TRUE
33:       BREAK
34:     end if
35:   end while
36:   ▶ Output path ending brackets and EOS.
37:   $k' \leftarrow$ tracing the path backwards to find position of the closest ancestor who has an unvisited child; if none exists, $k' \leftarrow 0$
38:   if $flag\_special\_concept$ = TRUE then
39:     $output$ += $(k - k' - 1)*$")"::EOS
40:   else
41:     $output$ += $(k - k')*$")"::EOS
42:   end if
43: end for
```

3.3 Source Sentence Selection

We seek to generate an abstractive summary containing multiple sentences from a cluster of documents discussing a single topic (e.g., health and safety). Each summary sentence will cover a topic aspect; it is generated by fusing a set of relevant source sentences. We thus perform clustering on all source sentences to find salient topic aspects and their corresponding sets of similar sentences. Spectral clustering has been shown to perform strongly on different clustering problems (Ng et al., 2002; Yogatama and Tanaka-Ishii, 2009). The approach constructs an affinity matrix by applying a pairwise similarity function to all source sentences. It then calculates the eigenvalues of the matrix and performs clustering in the low-dimensional space spanned by the largest eigenvectors. A large cluster indicates a salient topic aspect. We focus on the M largest clusters and extract N sentences from each cluster.[1] These sentences have the highest similarity scores with other sentences in the cluster. The selected sets of relevant sentences are later fed to the content planning component to generate summary AMR graphs.

Training the content planning component, however, requires sets of source sentences paired with their goldstandard summary graphs. Manually selecting sets of sentences and annotating summary graphs is costly and time-consuming. Instead, we leverage human reference summaries to create training instances. We obtain summary graphs by AMR-parsing sentences of human reference summaries. For every reference sentence, we further extract a set of source sentences. They are judged similar to the reference sentence via a similarity metric. The summary AMR graphs and sets of source sentences thus form the training data for content planning. We gauge how best to select source sentences by exploring different similarity metrics. In particular, (i) **LCS** calculates the longest common subsequence between a candidate source sentence and the reference sentence; (ii) **VSM** represents sentences using the vector space model and calculates the cosine similarity between the two sentence vectors; (iii) **Smatch** (Cai and Knight, 2013) calculates the F-score of AMR concepts between the candidate and reference sentences; (iv) **Concept Coverage** selects source sentences to maximize the coverage of AMR concepts of the reference sentence. We experiment with these source sentence selection strategies and compare their effectiveness in Section §5.1.

4 Datasets and Baselines

We perform experiments on standard multi-document summarization datasets[2], prepared by the NIST researchers for DUC/TAC competitions and later exploited by various summarization studies (Nenkova and McKeown, 2011; Hong et al., 2014; Yogatama et al., 2015). A summarization instance includes generating a text summary containing 100 words or less from a cluster of 10 source documents discussing a single topic. 4 human reference summaries are provided for each cluster of documents; they are created by NIST assessors. We use the datasets from DUC-03, DUC-04, TAC-09, TAC-10, and TAC-11 in this study, containing 30/50/44/46/44 clusters of documents respectively.

We compare our AMR summarization framework with a number of extractive (*ext*-$*$) and abstractive (*abs*-$*$) summarization systems, including the most recent neural encoder-decoder architecture (See et al., 2017). They are described as follows.

- *ext*-**LexRank** (Erkan and Radev, 2004) is a graph-based approach that computes sentence importance based on the concept of eigenvector centrality in a graph representation of source sentences;
- *ext*-**SumBasic** (Vanderwende et al., 2007) is an extractive approach that assumes words occurring frequently in a document cluster have a higher chance of being included in the summary;
- *ext*-**KL-Sum** (Haghighi and Vanderwende, 2009) describes a method that greedily adds sentences to the summary so long as it decreases the KL divergence;
- *abs*-**Opinosis** (Ganesan et al., 2010) generates abstractive summaries by searching for salient paths on a word co-occurrence graph created from source documents;

[1]We use N=M=5 in our experiments. This setting fuses 5 source sentences to a summary sentence. It then produces 5 summary sentences for each topic, corresponding to the average number of sentences in human summaries.

[2]`https://duc.nist.gov/data.html` `https://tac.nist.gov/data/index.html`

	Nodes			**(Oracle) Nodes**			**Edges**			**(Oracle) Edges**		
Approach	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
LCS	16.7	26.7	19.9	31.5	49.5	37.6	6.7	8.0	6.9	16.1	18.7	16.6
Smatch	20.9	33.2	24.9	33.2	52.0	39.6	9.3	10.7	9.4	17.2	20.1	17.8
Concept Cov.	**25.0**	**40.3**	**30.1**	**48.8**	**77.5**	**58.7**	7.3	10.0	8.0	18.9	**25.3**	20.8
VSM	24.0	38.6	28.8	40.8	64.3	48.9	**9.6**	**11.3**	**9.8**	**21.1**	25.1	**22.1**

Table 1: Summary graph prediction results on the DUC-04 dataset. The scores measure how well the predicted summary graphs match reference summary graphs on nodes and edges. Reference summary graphs are created by parsing reference summary sentences using the CAMR parser. "Oracle" results are obtained by performing only cost-based decoding. They establish an upper bound for the respective approaches.

- *abs*-**Pointer-Generator** (See et al., 2017) describes a neural encoder-decoder architecture. It encourages the system to copy words from the source text via pointing, while retaining the ability to produce novel words through the generator. It also includes a coverage mechanism to keep track of what has been summarized, thus reducing word repetition. The pointer-generator networks have not been tested for multi-document summarization. In this study we evaluate their performance on the DUC/TAC datasets.

5 Experimental Results

In this section we evaluate our AMR summarization framework. We are interested in knowing how well the system performs on predicting summary graphs from sets of relevant source sentences (§5.1). We also investigate the system's performance on generating abstractive summaries and its comparison with various baselines (§5.2). Finally, we provide an analysis on system summaries and outline challenges and opportunities for advancing this line of work (§5.3).

5.1 Results on Summary Graph Prediction

Graph prediction results on the DUC-04 dataset (trained on DUC-03) are presented in Table 1. We report how well the decoded summary graphs match goldstandard summary graphs on nodes (concepts) and edges (relations). We compare several strategies to select sets of source sentences. The goldstandard summary graphs are created by parsing the reference summary sentences via the CAMR parser (Wang et al., 2015b). Note that we cannot obtain goldstandard summary graphs for sets of source sentences selected by spectral clustering. This approach therefore is not evaluated for graph prediction. The system-decoded summary graphs are limited to 15 graph nodes, corresponding to the average number of words in reference summary sentences (stopwords excluded). We additionally report "Oracle" decoding results, obtained by performing only cost-based decoding $\hat{G} = \arg\max_G(-cost(G; G^*))$ on the source graph, where G^* is the goldstandard summary graph. The oracle results establish an upper bound for the respective approaches.

We observe that node prediction generates better results than edge prediction. Using 'Concept Cov,' the system-decoded summary graphs successfully preserve 40.3% of the goldstandard summary concepts, and this number increases to 77.5% when using oracle decoding, indicating the content planning component is effective at identifying important source concepts and preserving them in summary graphs. 'VSM' performs best on edge prediction. It achieves an F-score of 9.8% and the oracle decoding further boosts the performance to 22.1%. We observe that only 42% of goldstandard summary bigrams appear in the source documents, serving as a cap for edge prediction. The results suggest that 'VSM' is effective at selecting sets of source sentences containing salient source relations. The high performance on summary node prediction but low on edge prediction suggests that future work may consider increasing the source graph connectivity by introducing edges between concepts so that salient summary edges can be effectively preserved.

5.2 Results on Summarization

In Table 2 we report the summarization results evaluated by ROUGE (Lin, 2004). In particular, R-1, R-2, and R-SU4 respectively measure the overlap of unigrams, bigrams, and skip bigrams (up to 4 words)

		ROUGE-1			ROUGE-2			ROUGE-SU4		
	System	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
DUC 2004	*ext*-SumBasic	37.5	24.9	29.5	5.3	3.6	4.3	11.1	7.3	8.6
	ext-KL-Sum	31.1	31.1	31.0	6.0	6.1	6.0	10.2	10.3	10.2
	ext-LexRank	34.3	34.6	34.4	7.1	7.2	7.1	11.1	11.2	11.2
	abs-Opinosis	36.5	23.7	27.5	7.2	4.3	5.1	11.7	7.4	8.6
	abs-Pointer-Gen-all	37.5	20.9	26.5	8.0	4.4	**5.6**	12.3	6.7	8.5
	abs-Pointer-Gen	33.2	21.5	25.6	5.8	3.8	4.5	10.3	6.6	7.9
	abs-AMRSumm-Clst	29.9	30.5	**30.2**	4.1	4.2	4.1	8.7	8.9	**8.8**
	abs-AMRSumm-VSM	36.7	39.0	**37.8**	6.5	6.9	**6.6**	11.4	12.2	**11.8**
TAC 2011	*ext*-SumBasic	37.3	28.2	31.6	6.9	5.5	6.1	11.8	9.0	10.1
	ext-KL-Sum	31.2	31.4	31.2	7.1	7.1	7.1	10.5	10.6	10.6
	ext-LexRank	32.9	33.3	33.1	7.4	7.6	7.5	11.1	11.2	11.1
	abs-Opinosis	38.0	20.4	25.2	8.6	4.0	5.1	12.9	6.5	8.1
	abs-Pointer-Gen-all	37.3	22.2	27.6	7.8	4.6	**5.8**	12.2	7.1	8.9
	abs-Pointer-Gen	34.4	21.6	26.2	6.9	4.4	5.3	10.9	6.8	8.2
	abs-AMRSumm-Clst	32.2	31.7	**31.9**	4.7	4.7	4.7	9.8	9.7	**9.7**
	abs-AMRSumm-VSM	40.1	42.3	**41.1**	8.1	**8.5**	**8.3**	13.1	13.9	**13.5**

Table 2: Summarization results on DUC-04 and TAC-11 datasets. We compare the AMR summarization framework (AMRSumm-*) with both extractive (*ext*-*) and abstractive (*abs*-*) summarization systems.

between system and reference summaries. Our AMR summarization framework outperforms all other abstractive systems with respect to R-1 and R-SU4 F-scores on both DUC-04 (trained on DUC-03) and TAC-11 (trained on TAC-09,10) datasets. "AMRSumm-VSM" further produces the highest R-2 F-scores. We conjecture that the R-2 scores of AMRSumm-* are related to the performance of the AMR-to-text generator (Flanigan et al., 2016). When it transforms a summary graph to text, there can be multiple acceptable realizations (e.g., "I hit my hand on the table" or "My hand hit the table") and the one chosen by the AMR generator may not always be the same as the source text. Because abstractive systems are expected to produce novel words, they may yield slightly inferior results to the best extractive system (LexRank). Similar findings are also reported by Nallapati et al. (2017) and See et al. (See et al., 2017).

We experiment with two variants of the pointer-generator networks: "Pointer-Generator-all" uses all source sentences of the document set and "Pointer-Generator" uses the source sentences selected by spectral clustering, hence the same input as "AMRSumm-Clst." We observe that "AMRSumm-Clst" performs stronger than "Pointer-Generator" at preserving salient summary concepts, yielding R-1 F-scores of 30.2% vs. 25.6% and 31.9% vs. 26.2% on DUC-04 and TAC-11 datasets. Further, we found that the summaries produced by the pointer-generator networks are more extractive than abstractive. We report the percentages of summary n-grams contained in the source documents in Figure 3. "Pointer-Generator-all" has 99.6% of unigrams, 95.2% bigrams, and 87.2% trigrams contained in the source documents (DUC-04). In contrast, the ratios for human summaries are 85.2%, 41.6% and 17.1%, and for "AMRSumm-Clst" the ratios are 84.6%, 31.3% and 8.4% respectively. Both human summaries and "AMRSumm-Clst" summaries tend to be more abstractive, with fewer bigrams/trigrams appeared in the source. These results suggest that future abstractive systems for multi-document summarization may need to carefully balance between copying words from the source text with producing new words/phrases in order to generate summaries that resemble human abstracts.

5.3 Result Analysis

Table 3 shows results of the AMR-Summ framework with different system configurations. We use CAMR (Wang et al., 2015b) to parse source sentences into AMR graphs during training, and apply either JAMR (Flanigan et al., 2014) or CAMR to parse sentences at test time. We observe that the quality of AMR parsers has an impact on summarization performance. In particular, JAMR reports 58% F-score and CAMR reports 63% F-score for parsing sentences. The inter-annotator agreement places

		JAMR			CAMR			(Oracle) CAMR		
	Approach	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
ROUGE-1	AMRSumm-Clst	29.0	29.8	29.4	29.9	30.5	30.2	36.4	37.8	37.1
	AMRSumm-Concept Cov	36.3	37.8	**36.9**	36.9	39.3	**38.1**	46.9	49.8	**48.3**
	AMRSumm-VSM	35.9	37.2	36.5	36.7	39.0	37.8	43.3	46.1	44.6
ROUGE-2	AMRSumm-Clst	3.2	3.3	3.2	4.1	4.2	4.1	6.0	6.3	6.1
	AMRSumm-Concept Cov	4.8	5.0	**4.9**	5.7	6.0	5.8	9.8	10.4	**10.1**
	AMRSumm-VSM	4.6	4.8	4.7	6.5	6.9	**6.6**	9.7	10.3	10.0

Table 3: Summarization results of the AMR-Summ framework on the DUC-04 dataset. "JAMR" uses the JAMR parser (Flanigan et al., 2014) to produce AMR graphs for source sentences, while "CAMR" uses the CAMR parser (Wang et al., 2015b). "Oracle" results are obtained by performing only cost-based decoding.

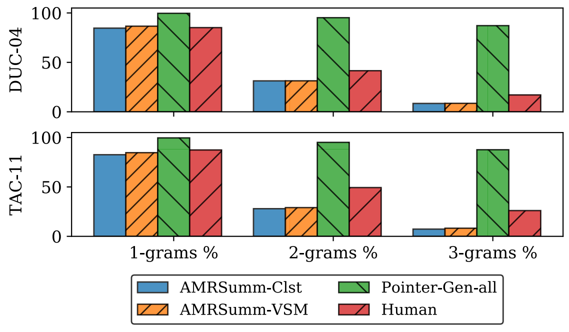


Figure 3: Percentages of summary n-grams contained in the source documents. Both human summaries and AMRSumm summaries are highly abstractive, with few bigrams and trigrams contained in the source. Pointer-Generator summaries appear be more extractive than abstractive.

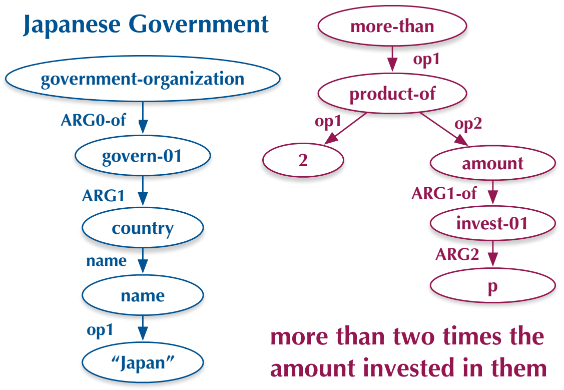


Figure 4: Example AMR for text segments.

an upper bound of 83% F-score on expected parser performance (Wang et al., 2015b). There remains a significant gap between the current parser performance and the best it can achieve. Consequently, we notice that there is a gap of 1∼2 points in terms of ROUGE scores when comparing summaries produced using the two parsers. We notice that source sentence selection strategies making use of reference summary sentences produces better results than 'AMRSumm-Clst.' Using oracle decoding further boosts the summarization performance by 7-10% for R-1 F-score and 2-5% for R-2 F-score.

When examining the source and summary graphs, we notice that a simplified AMR representation could be helpful to summarization. As a meaning representation grounded in linguistic theory, AMR strives to be comprehensive and accurate. However, a summarization system may benefit from a reduced graph representation to increase the algorithm robustness. For example, the large 'semantic content units' could be collapsed to "mega-nodes" in some cases. Figure 4 shows two examples. In the first example ("Japanese Government"), the human annotator chooses to decompose derivational morphology given that a relative clause paraphrase is possible (Schneider et al., 2015). It produces 5 concept nodes, representing "government organization that governs the country of Japan." In the second example, "more than two times the amount invested in them" also has fine-grained annotation. For the purpose of summarization, these graph fragments could potentially be collapsed to "mega-nodes" and future AMR parsers may consider working on reduced AMR graphs.

6 Conclusion

In this paper we investigated the feasibility of utilizing the AMR formalism for multi-document summarization. We described a full-fledged approach for generating abstractive summaries from multiple source documents. We further conducted experiments on benchmark summarization datasets and showed that the AMR summarization framework performs competitively with state-of-the-art abstractive approaches. Our findings suggest that the abstract meaning representation is a powerful semantic formalism that holds potential for the task of abstractive summarization.

Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments. The authors thank Chuan Wang, Jeffrey Flanigan, and Nathan Schneider for useful discussions.

References

Miguel Ballesteros and Yaser Al-Onaizan. 2017. Amr parsing using stack-LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

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.*

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of ACL.*

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).*

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI).*

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL.*

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

Giuseppe Di Fabbrizio, Amanda J. Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. *Proceedings of the 8th International Natural Language Generation Conference (INLG).*

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 Annual Meeting of the Association for Computational Linguistics (ACL).*

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL).*

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the International Conference on Computational Linguistics (COLING).*

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of ACL Workshop on Monolingual Text-To-Text Generation.*

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? Optimal beam search for neural text generation (modulo beam size). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Tomasz Jurczyk and Jinho D. Choi. 2015. Semantics-based graph approach to complex question-answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of EMNLP*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of SemEval*.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

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

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Terence Parsons. 1990. *Events in the semantics of English*. MIT Press.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference (INLG)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press New York, NY, USA.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mrinmaya Sachan and Eric P. Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Barry Schein. 1993. *Plurals and Events*. MIT Press.

Nathan Schneider, Jeffrey Flanigan, and Tim O'Gorman. 2015. The logic of amr: Practical, unified, graph-based sentence semantics for nlp. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts (NAACL)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for english-amr alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ales Tamchyna, Chris Quirk, and Michel Galley. 2015. A discriminative model for semantics-to-string translation. In *Proceedings of the ACL Workshop on Semantics-Driven Statistical Machine Translation (S2MT)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Wiki. 2018. Portal:Current events. https://en.wikipedia.org/wiki/Portal:Current_events.

Dani Yogatama and Kumiko Tanaka-Ishii. 2009. Multilingual spectral clustering using document similarity propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.