A Partially Rule-Based Approach to AMR Generation

Emma Manning
Department of Linguistics
Georgetown University
esm76@georgetown.edu

Abstract

This paper presents a new approach to generating English text from Abstract Meaning Representation (AMR). In contrast to the neural and statistical MT approaches used in other AMR generation systems, this one is largely rule-based, supplemented only by a language model and simple statistical linearization models, allowing for more control over the output. We also address the difficulties of automatically evaluating AMR generation systems and the problems with BLEU for this task. We compare automatic metrics to human evaluations and show that while METEOR and TER arguably reflect human judgments better than BLEU, further research into suitable evaluation metrics is needed.

1 Introduction

Abstract Meaning Representation, or AMR, is a representation of a sentence as a rooted, labeled graph. It provides a representation of the sentence's semantics while abstracting away from morphosyntactic details such as tense, number, word order, and part of speech (Banarescu et al., 2013).

Because of these abstractions, it can be very difficult to generate from an AMR back to a fluent English sentence which preserves the original meaning. It is also difficult to accurately evaluate the quality of generation results, since there is typically only one reference sentence available to compare results to, but one of the basic principles of AMR is the fact that the same AMR is used to represent many possible sentences; for example, "he described her as a genius", "his description of her: genius", and "she was a genius, according to his description" would all correspond to the same AMR (Banarescu et al., 2013).

The following represents an AMR graph for the sentence "A key European arms control treaty must be maintained.":

```
(o / obligate-01
  :ARG2 (m / maintain-01
    :ARG1 (t / treaty
      :ARG0-of (c2 / control-01
        :ARG1 (a / arms))
      :ARG1-of (k / key-02)
      :mod (c / continent
        :wiki "Europe"
        :name (n / name
          :op1 "Europe")))))
```

This example demonstrates several of the challenges faced by an AMR generation system. These include properly addressing constructions that do not correspond closely to the words in the reference, such as the use of the frame `obligate-01` to express 'must' and the specific construction used for named entities such as 'Europe', as well as word order and the passive construction 'be maintained'. In fact, this system successfully addresses some but not all of these challenges, producing the output "Must maintain Europe key arms control treaty ."

While most previous work in AMR generation has used statistical and neural techniques, the current work approaches the task with a combination of rules and statistical methods; the rules are intended to constrain possibilities, particularly the possible realizations of concepts and which information from the AMR is expressed. This allows for greater control over the output; even if the overall results do not score as well, on average, as those of other approaches, this approach has the potential to minimize the chances of significant adequacy errors such as omission of key information or addition of information not contained in the AMR, which are possible in machine-learning-based systems. Another advantage of a partially rule-based approach is that it can work without large amounts

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 61–70
Minneapolis, Minnesota, June 3 - 5, 2019.

of AMR-annotated data; it could thus be adapted to a new language or an altered AMR scheme in situations where there is insufficient data for a machine-learning-based system to achieve satisfactory performance.

2 Related Work

2.1 AMR Generation Systems

Flanigan et al. (2016) introduced the first AMR generator (JAMR), which transforms an AMR graph into a tree before using a weighted tree-to-string transducer to generate the string. While most of its rules are automatically extracted, these are supplemented with handwritten rules for some phenomena including dates, conjunctions, and some special concepts. An ablation experiment showed that these handwritten rules contributed significantly to the results.

AMR generation was included as a shared task at SemEval-2017 (May and Priyadarshi, 2017). The winner of the task, as determined by human judgments, was the RIGOTRIO system (Gruzitis et al., 2017), which uses handcrafted rules to convert AMRs to Abstract Syntax Trees, which are then realized as strings using existing Grammatical Framework resources. However, this approach has limited coverage, and is only used for about 12% of sentences, while the system defaults to using JAMR for other sentences.

Other submissions to the shared task included FORGe, which uses graph transducers (Mille et al., 2017); Sheffield, which treats AMR generation as an inverse of transition-based parsing, transforming the AMR graph into a syntactic dependency tree before realizing it as a sentence (Lampouras and Vlachos, 2017); and ISI, which uses phrase-based machine translation (PBMT) methods (Pourdamghani et al., 2016).

Beyond the shared task, other work in AMR generation has approached the task as a Traveling Salesman Problem (Song et al., 2016), with synchronous node replacement grammar (Song et al., 2017), and using a transition-based approach to transform the AMR into a syntactic dependency tree (Schick, 2017). Castro Ferreira et al. (2017) compare the effect of different types of preprocessing on the performance of AMR generation systems based on PBMT and NMT.

The best results to date have been obtained with neural methods, which excel when the small amount of manually-annotated training data is augmented with millions of unlabeled sentences which have been automatically parsed. Konstas et al. (2017) first used this approach to train a sequence-to-sequence model, and and Song et al. (2018) later adapted it to a graph-to-sequence model.

2.2 Evaluation

Most previous work in AMR generation has reported results exclusively using BLEU scores (Papineni et al., 2002), with the original sentence as the only reference. A notable exception is the five systems included in the SemEval-2017 shared task, which were additionally compared by human judgments. The human evaluations were shown not to correlate well with BLEU scores, raising questions about the suitability of the metric for this task (May and Priyadarshi, 2017). In particular, BLEU as used for AMR generation is intuitively inappropriate because it strictly measures similarity to one reference sentence, while by design, a single AMR can correspond to many different English sentences. Thus, BLEU is in practice more of a measure of how closely a system can replicate the exact wording used in the original sentence than of how adequately and fluently it expresses the meaning of the AMR.

Ideally, evaluation of AMR generation would be performed using human judgments or task-based evaluations; unfortunately, however, it is sometimes necessary to rely on the practicality of automatic metrics. We thus follow Castro Ferreira et al. (2017) in reporting two additional automatic metrics alongside BLEU, which may provide slightly more insight into system performance. The first is METEOR, which has been shown to correlate more strongly with human judgments of machine translation quality than BLEU does (Lavie and Agarwal, 2007; Denkowski and Lavie, 2014). It is a particularly appealing alternative to BLEU for AMR generation because, instead of only giving credit to exact word matches, METEOR also allows matching based on stems, synonyms, and paraphrases. This mitigates the issues associated with having a single reference sentence in AMR, because it does not penalize systems as harshly for not correctly guessing the forms of morphological and syntactic variants that are usually not specified within the AMR. The final evaluation metric used is Translation Edit Rate (TER), which has been shown to require only one reference sentence in order to correlate as well with human judgments for ma-

chine translation as BLEU does with four references (Snover et al., 2006). This robustness against lack of extra references makes it, too, likely to be better suited to the AMR generation task than BLEU is.

These metrics were all designed for evaluation of machine translation; it may also be useful in the future to explore evaluating AMR generation with metrics from other NLG-related tasks, such as referenceless measures developed for grammatical error correction (e.g. Napoles et al., 2016; Choshen and Abend, 2018).

3 Methods

The system introduced in this paper uses rules to generate realization hypotheses, which are ranked and combined by statistical methods.

We used the LDC2015E86 version of the AMR corpus for training the linearization models, tuning hyperparameters, and analyzing errors.

3.1 Algorithm

The system uses an algorithm based on cube pruning (Huang and Chiang, 2007). Hypotheses are generated by recursing down the AMR graph, ignoring subsequent mentions of variables that have already been processed (reentrancies) and therefore treating the AMR as a tree. The roles `:wiki` and `:mode` are also ignored: `:wiki` provides extra information about named entities that does not need to be expressed in the English realization. The `:mode` relation could be used in future versions of the system to generate particular sentence types such as questions and imperatives; however, this syntactic manipulation would require more complicated rules than are used in this algorithm, and so the relation is ignored for now.

At each node, a priority queue of scored hypotheses is generated.

In particular, at each leaf node, one or more hypotheses are created for the realization of the node, according to the process described in 3.2, and each of these partial hypotheses is scored by a language model.

At each non-terminal node, priority queues are recursively generated containing hypotheses for the realizations of each of the node's children, as well as one for the node itself. Each of these is prioritized by language model score. An additional priority queue represents possible linearizations of the current node and each of its children, scored as discussed in 3.4. Thus, for a node with n children, a total of n+2 priority queues are created, simulating an n+2-dimensional hypercube. k nodes of the cube are then expanded and rescored by the language model.

When the root node of the AMR is reached, a period is added to the end and the k hypotheses are rescored by the language model, this time treated as a complete sentence. The realization associated with the best-scored hypothesis is postprocessed to capitalize the first letter, then returned.

3.2 Realization[1]

For each node, one or more possible strings are generated to realize the node's associated concept or constant and, in some cases, its relation. The system uses specific rules for some special cases, and more generalized rules for most nodes.

Special Cases: Special rules are used for a few constructions. In particular, the constructions for named entities and for people with relationships and roles in organizations (`have-rel-role-91` and `have-org-role-91`) are represented in AMR with some concepts that do not typically align with words in the text; rules prevent these and certain other AMR-specific concepts from being realized.

In addition, a `'-'` representing negative polarity is realized as 'not' or as one of several negative contractions; numbers in ordinal or month-name constructions are realized accordingly, and pronouns are realized as their possessive form in possessive constructions and may be realized as their subjective or objective forms otherwise. Finally, a handful of concepts whose names don't correspond to the English strings they typically represent are mapped to more likely English translations. These represent some conjunctions, modals, and other relationships that are associated with particular concepts in AMR; for example, `contrast-01` is realized as 'but' and `obligate-01` as 'can'.

Frames: Frames not dealt with in the special rules are likely to correspond to verbs, or occasionally adjectives, adverbs, and verb-derived nouns, and are treated as such. These are represented in English as they appear in the concept name, with frame numbers removed and words joined by spaces rather than hyphens. In addition to this base

[1]See the code, available at https://github.com/esmanning/emmAMR, for full details on the realization rules that could not be explained exhaustively due to space constraints.

concept name, several variations are generated, corresponding to different possible verbal, adjectival, and nominal forms. Verbal and adjectival forms are created by combining optional auxiliaries with verb forms such as those ending in '-ing', '-s', and '-ed', (with several variations depending on the base form, such as removing a final 'e' before adding an ending when appropriate). Nominal variants are similarly created with variations on the suffixes '-ment' and '-tion', and adverbs with '-ly'.

In many cases this generates forms that are implausible; this is not a problem because they are ranked by language model score, so forms that are unattested or very infrequent in the language model's training corpus receive poor scores and will be pruned out at a later stage. Nevertheless, this strategy inevitably misses some valid forms, such as those of many irregular verbs; this could be addressed in the future by integrating external resources that can produce inflections and derivations of a given word.

Non-Frame Concepts: Remaining non-frame concepts are given a treatment similar to the frames discussed above, except that they are assumed more likely to be nouns or elements of noun phrases (such as adjectives), and are given corresponding realizations. They may also represent other parts of speech, such as adverbs; in these cases, again, the language model can rule out any implausible forms that are spuriously generated.

The plain concept name is formed by simply replacing any hyphens in the original concept name with spaces, although concepts at this stage are usually already a single word. The hypotheses created are for the plain concept name, as well as variations which append 'the', 'a', or 'an' before the name or plural suffix 's' at its end, and a hypothesis which adds both 'the' and the plural 's'.

Relations: In addition to the variable realizations, a handful of relations are realized by strings attached before or after the realization of their associated variable. These are given in Table 1. Many of these represent prepositions, including the general `:prep-X` relation, which is realized as the given preposition.

3.3 Language Model

A 5-gram language model was created to score hypothesis strings. It was trained on the English Gigaword Fifth Edition Corpus (LDC2011T07) using KenLM (Heafield, 2011; Heafield et al., 2013). For time and space efficiency, the model was pruned: singleton 2-grams were removed, as well as 3-, 4-, and 5-grams that appear 3 or fewer times.

Concept	Realization	Type
:prep-X	X	prefix (+space)
:accompanier	with	prefix (+ space)
:destination	to	prefix (+ space)
:purpose	to	prefix (+ space)
:condition	if	prefix (+ space)
:compared-to	than	prefix (+ space)
:poss	's of	suffix prefix (+space)
:domain	is	suffix (+ space)
:location	in at by	prefix (+ space) prefix (+ space) prefix (+ space)

Table 1: Realizations for relations.

3.4 Linearization

The linearization model contains two components, the pair-order model and the coreness model. These may be used in combination, in which case the scores they assign are averaged, or either one may be used alone. There is also a simpler baseline, described below, which is used by default when both models are disabled. Each of the models is trained on the alignments provided with the training data.

When non-baseline linearization scoring is used, all permutations of a node and its children are scored, assuming the node has no more than 5 children (producing at most $6! = 720$ combinations to score). This covers the vast majority of cases, but on the occasion that a node has more children, only the three orderings generated by the baseline linearization are considered. Unlike baseline linearization, these three candidates are still scored by the model(s) rather than being assigned identical scores. This limiting of possibilities serves a practical purpose in limiting the maximum number of permutations that must be calculated and scored; however, it is likely that it does not hurt performance, since nodes with a large number of children often represent lists, where preserving the original order of the children is likely to match the original sentence.

Pair-Order Model: The pair-order model is designed to capture the intuition that particular relations are likely to be realized to either the left or the right of other particular relations, or of their parent, represented here as the special relation `ROOT`. For example, `:ARG0`, which usually represents an agent, occurs before its `ROOT` 77% of the time, while `:ARG1`, usually a patient, precedes its `ROOT`

in only 25% of cases.

The model is trained by counting ordered pairs of relations when it can be determined that one is realized before the other based on the provided alignments. Counts are then normalized into the probability of a particular order for any pair of relations, using add-1 smoothing to avoid assigning probabilities of 0 to unattested orderings. The model scores a hypothesized ordering by combining the scores of each pair of relations in the ordering.

Focusing only on relations allows for a small, simple model and avoids data sparseness; however, there are situations where a lexicalized version of this model may provide useful information that is lost here, such as the fact that some frames are more likely than others to have `:ARG1` realized to their left. A version of this model that could capture such information is left for future work.

Coreness Model: The second component to the linearization model is the coreness model, which attempts to capture the intuition that in addition to the left-to-right ordering represented by the pair-order model, some relations are more 'core' and are more likely to be realized closer to their parent than others. For example, whether `:ARG1` appears before or after its parent, it is likely to be close to it, while a relation like `:time` or `:purpose` will usually appear farther away, either very early or very late in a sentence. Thus, the model stores a single score for each relation representing its average absolute distance from the root, as a proportional distance relative to other children. A hypothesis's score is penalized based on the difference between each child's observed and expected distance from the root.

Baseline Linearization: When neither of the statistical linearization models are available, only three orderings are considered. The children are kept in the order they appear in the penman representation of the AMR, with the parent inserted either initially, penultimately, or finally. The penultimate option is considered because this is typically the appropriate place when the parent is a conjunction like 'and'. This is particularly useful in cases where the baseline is used as a default due to a node having a large number of children, since these often represent a long list of conjuncts. These hypotheses are given equal scores, meaning that all will be combined with realization hypotheses to create new hypotheses, and only the language model determines which are best.

4 Experiments

4.1 System Evaluation

First, several variations on this system with different hyperparameters were tested on the 1368 AMRs in the dev data. As discussed in 3.4, the linearization model contains two separate components, each of which is optional, resulting in four different linearization configurations. The pruning parameter k was tested at values of 5, 10, and 100. In total, 12 different versions of the system were tested on dev data. Based on these results, an optimal version of the system was chosen, which was then evaluated quantitatively on test data with automatic metrics. This system's output on dev data was also evaluated qualitatively by reviewing a small sample of its sentences, and quantitatively by comparing the number of tokens and frequency of parts of speech to those of the references.

4.2 Evaluation Metrics for AMR Generation

As discussed in 2.2, AMR Generation is usually evaluated only with BLEU, but one shared task obtained human judgments of five systems which were shown not to correlate well with BLEU scores (May and Priyadarshi, 2017). We tested the system outputs from this task with BLEU[2] as well as METEOR[3] and TER[4] to determine whether these metrics would correspond more closely to human judgments; results are presented and discussed in 5.5.

5 Results

5.1 Intra-System Variation

Table 2 shows the performance of each variation of the system on the dev data.

Systems using the pair-order model for linearization always perform better than their counterparts without it. While the coreness model's results are more mixed, it seems overall to hurt more than it helps. Increasing the stack size k always improved BLEU and METEOR scores, and doesn't hurt TER scores except in the +P+C condition, where the

[2]Using the multi-bleu-detok.perl script provided with the MOSES toolkit (Koehn et al., 2007)

[3]Using version 1.5, downloaded from `http://www.cs.cmu.edu/~alavie/METEOR/`

[4]Downloaded from `http://www.cs.umd.edu/~snover/tercom/`

Model	BLEU	METEOR	TER
+P+C,k=5	7.51	27.9	76.1
+P+C,k=10	8.12	28.1	**75.9**
+P+C,k=100	8.8	28.3	76.1
+P-C,k=5	7.76	28.0	76.0
+P-C,k=10	8.13	28.1	76.0
+P-C,k=100	**9.03**	**28.4**	76.0
-P+C,k=5	6.18	26.9	81.4
-P+C,k=10	6.48	27.0	81.4
-P+C,k=100	7.70	27.7	80.4
-P-C,k=5	5.99	26.6	79.1
-P-C,k=10	6.87	27.0	78.3
-P-C,k=100	7.71	27.4	77.9

Table 2: System performance on dev data with various hyperparameter combinations. '+P' is used to designate systems using the pair-order model and '-P' to designate those without it; similarly, '+C' and '-C' are used to designate whether or not the coreness model was used. In the '-P-C' configuration, baseline linearization is used. The best score for each metric is shown in bold. The shaded row represents the configuration of the system selected for further evaluation.

k=10 condition achieves a slightly better TER score than either k=5 or k=100.

Because it performs best according to BLEU and METEOR, and achieves barely short of its best TER score, the system using only the pair-order linearization model and k=100 was selected as the best version. These are the hyperparameter values that are used for the following quantitative and qualitative analyses.

5.2 Comparison to Other Systems

Table 3 summarizes the best BLEU and (when available) METEOR and TER scores reported for this system and various others. This table excludes the results of participants in the SemEval shared task, which were evaluated on a separate test set and whose scores are given below in Table 6.

The best BLEU score of this system is substantially lower than that of others. However, the METEOR score does appear a little more competitive, outperforming Castro Ferreira et al.'s NMT system by a point, and it is currently unknown how most other system's METEOR and TER scores compare with this one.

While automatic metrics are not ideal for comparing systems, they do seem to indicate that this system is not currently competitive with state-of-the-art systems. While it could certainly be improved in many ways given more time, this may indicate that rules augmented only with limited statistical models are not as well-suited to this task as other approaches, particularly the state-of-the-art neural models (Konstas et al., 2017; Song et al., 2018). Still, as discussed in Section 1, the partially rule-based approach has potential to be useful in situations where greater control over output is necessary or where training data is particularly limited.

5.3 Error Analysis

The system's output on a random sample of 25 sentences from the dev data was analyzed. Sentences were subjectively coded for quality into four categories: 'excellent', 'good', 'fair', and 'poor'. Descriptions, counts, and examples of each of these categories are given in Table 4. Crucially, unlike the automatic evaluations, this evaluation was based on how fluently and adequately the system output expresses the meaning of the AMR, not on how closely it matches the reference string. For example, a reference-based automatic metric would count 'Kinkel' in the third example as incorrect, because it does not match the misspelling 'Kinkerl' in the reference set; this evaluation, based on the AMR itself, recognizes the system's output of 'Kinkel' as correct.

Most sentences, especially those classified as fair and poor, include both linearization and realization errors. An example of a linearization error is in the fourth example in the table, where the linearization in the system output incorrectly implies that jcboy was the speaker, rather than the addressee. However, linearization that differs from the reference is not always considered incorrect: in the second example, the system places the 'even if...' clause before the main clause of the sentence, which is a valid ordering even though it differs from that of the reference.

Further insight into the limitations of this system can be gained by looking at the lengths of output sentences and the frequency of different parts of speech. To analyze these discrepancies, the system output and references were both automatically tagged, using the NLTK tagger (Bird et al., 2009) and part of speech tags from the Penn Treebank (Santorini, 1990). Table 5 summarizes the counts of each part-of-speech tag found in system output and in the references.

One striking finding is that the system output has noticeably fewer tokens in total than the reference. Part of this is due to punctuation: while the system output contains more periods than the references due to adding a period to every sentence, it has no rules to output other punctuation such as commas, colons, parentheses, and quotation marks.

LDC2014T12				LDC2015E86				LDC2016E25			
System	BLEU	METEOR	TER	System	BLEU	METEOR	TER	System	BLEU	METEOR	TER
Pourdamghani (2016)	26.9	-	-	Flanigan (2016)	22.1	-	-	Castro Ferreira (2017)[5]			
Schick (2017)	28.9	-	-	Song (2016)	22.4	-	-	-NMT	18.9	26.6	66.2
				Song (2017)	25.2	-	-	-PBMT	26.8	34.7	59.4
				Konstas (2017)	33.8	-	-				
				Konstas (2017)[6]	31.1	37.5	46.9				
				Song (2018)	33.0	-	-				
				This System	8.7	28.2	76.0	This System	8.6	28.0	76.2

Table 3: Comparison of test scores achieved by this system and others.

Further investigation into adding punctuation might not only improve scores from automatic metrics, but also increase readability of more complicated sentences if done correctly.

Beyond punctuation, the system under-produces many types of function words, particularly determiners (DT), existentials (EX), prepositions and subordinating conjunctions (IN), possessives (POS, PRP$), the word 'to' (TO), and all types of WH-words (WDT, WP, WP$, WRB). In the nominal domain, the system realizes too many nouns as singular rather than plural for both common and proper nouns, which is probably due to a combination of some irregular plurals being missed by the realization rules, and the language model perhaps preferring singular forms more than is appropriate for these reference sentences. In the verbal domain, the system outputs more '-ing' forms (VBG) than are found in the references, and under-produces all other verb types, especially base forms (VB). This last is somewhat surprising, since the base form should always be accurately generated as an option by the realization rules; it is likely that its probability is underestimated by the language model, perhaps because the infinitival 'to' is not usually generated in hypotheses.

5.4 Future Work

The prevalence of linearization errors in the output sampled indicates that improving the linearization models would substantially improve system performance. As mentioned in 3.4, a lexicalized version of the linearization model would likely improve performance, especially if the data sparseness is mitigated by using augmented training data, similar to the approaches used by Konstas et al. (2017) and Song et al. (2018). Additionally, it may help to base the linearization models on the alignments of Szubert et al. (2018), which provide more complete coverage of the AMRs.

Realization errors mostly arise from the fact that the restrictive realization rules do not allow for all valid possibilities; for example, there is no rule to allow relative clauses such as the 'the ones who...' construction used in the reference of the second example sentence. The analysis of part of speech and overall token frequency also shows the need for more realization options involving function words, such as prepositions, which are currently not produced in many situations where they should be. These problems can be improved by the addition of more realization options to the handcrafted rules, although this is a time-consuming process.

5.5 Evaluation Metrics

In the SemEval shared task, only BLEU scores were originally reported alongside measures of human rankings. To explore how well each of the three automatic metrics used in this paper correlate with human judgments, we tested the output of all the systems that participated in the task with each of these metrics. Table 6 shows these new results, alongside the results of the human rankings.

All four measures agree that RIGOTRIO and CMU are the best two systems out of the task participants, but METEOR and TER both agree with humans in rating RIGOTRIO highest, while CMU obtains a higher BLEU score. This provides some evidence for the claim that METEOR and TER may be better suited than BLEU to evaluating AMR generation, especially when it comes to distinguishing among stronger systems. However, none of the metrics fully match the ranking given by humans—in particular, while humans considered FORGe the third-best system, all of the automatic metrics rank it lower. Thus, while these alternatives may be an improvement over BLEU, more research is necessary to determine a more accurate way to auto-

[5]Castro Ferreira et al. report results for 8 versions of each of system (NMT and PBMT), using different combinations of preprocessing steps. For each system, we select the version that performs best according to two of the three metrics to represent here.

[6]Data provided to me by Konstas in personal communication; scores determined by the same tests used for my system's data.

Excellent (1): Human-like quality; fluently presents all meaning

```
(t / tool
  :mod (m / machine)
  :purpose (s / set-up-03
    :ARG1 (f2 / facility)
    :location-of (f / fabricate-01)))
```

REF: Machine Tools for setting up a fabrication facility.
SYS: Machine tools to set up fabrication facility .

Good (5): Adequately and intelligibly expresses all or nearly all meaning with minor disfluencies

```
(m / multi-sentence
  :snt1 (s / suffer-01
    :ARG0 (p / person
      :mod (o2 / ordinary)))
  :snt2 (f / fish
    :ARG1-of (s2 / salt-01)
    :mod (s3 / still)
    :domain f2
    :concession (e / even-if
      :op1 (t / turn-01
        :ARG1 (b / body
          :poss (f2 / fish
            :ARG1-of (s4 / salt-01)))
        :direction (o3 / over)))))
```

REF: the ones who are suffering are the ordinary people: even if the body of a salted fish is turned over, it is still a salted fish ...
SYS: An ordinary person suffer . salted fish is still salted fish even if the body turns over .

Fair (6): Meaning is partially intelligible

```
(s / say-01
  :ARG0 (p / person :wiki "Klaus_Kinkel"
    :name (n2 / name :op1 "Kinkel"))
  :ARG1 (a / and
    :op1 (c / correct-02
      :ARG1 (t / thing
        :ARG1-of (d / decide-01
          :ARG0 (m / military
            :wiki "NATO"
            :name (n3 / name
              :op1 "NATO")))))
    :op2 (n4 / need-01
      :ARG1 t)))
```

REF: Kinkerl said that NATO 's decision was " correct and necessary " .
SYS: Said Kinkel NATO decided things correctly and needs .

Poor (13): Meaning is barely or not at all intelligible

```
(s / say-01
  :ARG1 (c / correct-02
  :ARG1 (y / you))
    :ARG2 (p / person :wiki -
      :name (n / name :op1 "jcboy")))
```

REF: @jcboy, You are correct.
SYS: You correct jcboy said .

Table 4: Description and example for each of the four quality categories.

matically evaluate AMR generation. In particular, due to the limitations of reference-based evaluation, we plan to focus on developing referenceless automatic metrics in future work.

Tag	System	Ref	Tag	System	Ref
$	2	22	NNS	1847	2079
“	0	57	PDT	4	5
”	0	42	POS	253	71
(	0	106	PRP	345	541
)	0	105	PRP$	22	129
,	1	823	RB	887	876
.	1431	1230	RBR	53	28
:	0	255	RBS	25	2
CC	859	856	RP	81	63
CD	853	777	SYM	0	1
DT	820	2663	TO	155	663
EX	0	50	UH	3	9
FW	7	5	VB	451	910
IN	1425	3215	VBD	887	1029
JJ	1806	2144	VBG	838	521
JJR	73	84	VBN	427	656
JJS	38	55	VBP	431	508
LS	0	1	VBZ	430	569
MD	200	312	WDT	10	110
NN	4399	3883	WP	2	86
NNP	3131	2931	WP$	0	12
NNPS	65	83	WRB	30	80

Table 5: Comparison of counts of part-of-speech tags in system output vs. references.

System	Trueskill	BLEU[7]	METEOR	TER
RIGOTRIO	1.03	18.6	32.3	80.1
CMU	0.819	19.0	31.4	82.4
FORGe	0.458	2.8	20.0	92.0
ISI	-1.172	10.9	28.9	98.7
Sheffield	-2.132	1.2	20.0	87.5
This System	-	6.7	28.7	79.4

Table 6: Comparison of evaluation metrics to Trueskill (measure of human rankings) for shared task data and systems. This system's performance on the same data according to automatic metrics is provided for comparison.

6 Conclusion

Given that the relatively small amount of available data and the difficulty of the task have made it difficult for statistical and neural approaches to achieve truly satisfactory results in AMR generation, we hypothesized that a partially rule-based system, combined with some simple statistical methods, might be able to effectively harness human linguistic knowledge to achieve comparable results.

Judging by automatic metrics, this method does not seem able to compete well with state-of-the-art systems—although this system's scores could doubtless be improved somewhat with further development, especially if the primary goal were to optimize toward metrics like BLEU by providing more realization candidates that might better match the reference sentence's n-grams. However, we also argue that BLEU scores are a poor metric for

[7]These numbers differ slightly from the previously reported BLEU scores, presumably due to differences in BLEU configuration, but the ranking of systems is the same.

the AMR generation task. While METEOR and TER appear to do slightly better than BLEU at least at distinguishing among better-performing systems, none of these metrics fully reflect human rankings, making it difficult to fully determine how this system compares to others without human evaluation.

As research moves forward in AMR generation, it is essential to ensure that we are truly moving in a direction that will help us generate English realizations that both adequately and fluently express the meaning represented in an AMR. It is clear that the automatic metrics that have been used for this task fail to achieve these goals. More research is necessary to develop new metrics that are better suited to this task.

Acknowledgments

I would like to thank Nathan Schneider for his guidance, Ioannis Konstas and Jonathan May for sharing their data, and the anonymous reviewers for their comments and suggestions.

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 & Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation.*, pages 1–10, Santiago de Compostela, Spain. Association for Computational Linguistics.

Leshem Choshen and Omri Abend. 2018. Reference-less Measure of Faithfulness for Grammatical Error Correction. In *Proceedings of NAACL-HLT 2018*, pages 124–129, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using Tree Transducers. In *Proceedings of NAACL-HLT 2016*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Normunds Gruzitis, Didzis Gosko, and Guntis Barzdins. 2017. RIGOTRIO at SemEval-2017 Task 9: Combining Machine Learning and Grammar Engineering for AMR Parsing and Generation. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 924–928, Vancouver, Canada. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Liang Huang and David Chiang. 2007. Forest Rescoring: Faster Decoding with Integrated Language Models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi Itc-Irst, Brooke Cowan, Wade Shen, Christine Moran Mit, Richard Zens, Rwth Aachen, Chris Dyer, Alexandra Constantin, Williams College, and Evan Herbst Cornell. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 177–180, Prague, Czech Republic.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, Luke Zettlemoyer, and Paul G Allen. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 146–157, Vancouver, Canada.

Gerasimos Lampouras and Andreas Vlachos. 2017. Sheffield at SemEval-2017 Task 9: Transition-based language generation from AMR. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 586–591, Vancouver, Canada. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech

Republic. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 Task 9: Abstract Meaning Representation Parsing and Generation. In *Proceedings of the 11th International Workshop on Semantic Evaluations*, pages 536–545, Vancouver, Canada.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. FORGe at SemEval-2017 Task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 920–923, Vancouver, Canada. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2109–2115, Austin, TX. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of The 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK.

Beatrice Santorini. 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision). Technical report, University of Pennsylvania.

Timo Schick. 2017. *Transition-Based Generation from Abstract Meaning Representations*. Ph.D. thesis, Technische Universität Dresden.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge.

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 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 7–13, Vancouver, Canada. Association for Computational Linguistics.

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 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2084–2089, Austin, TX. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A Structured Syntax-Semantics Interface for English-AMR Alignment. In *Proceedings of NAACL-HLT 2018*, pages 1169–1180, New Orleans, Louisiana. Association for Computational Linguistics.