Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation

Thiago Castro Ferreira[1] and **Iacer Calixto**[2] and **Sander Wubben**[1] and **Emiel Krahmer**[1]
[1]Tilburg center for Cognition and Communication (TiCC), Tilburg University, The Netherlands
[2]ADAPT Centre, Dublin City University, School of Computing, Ireland
{tcastrof,e.j.krahmer,s.wubben}@tilburguniversity.edu
{iacer.calixto}@adaptcentre.ie

Abstract

In this paper, we study AMR-to-text generation, framing it as a translation task and comparing two different MT approaches (Phrase-based and Neural MT). We systematically study the effects of 3 AMR preprocessing steps (Delexicalisation, Compression, and Linearisation) applied before the MT phase. Our results show that preprocessing indeed helps, although the benefits differ for the two MT models. The implementation of the models are publicly available[1].

1 Introduction

Natural Language Generation (NLG) is the process of generating coherent natural language text from non-linguistic data (Reiter and Dale, 2000). While there is broad consensus among NLG scholars on the output of NLG systems (i.e., text), there is far less agreement on what the input should be; see Gatt and Krahmer (2017) for a recent review. Over the years, NLG systems have taken a wide range of inputs, including for example images (Xu et al., 2015), numeric data (Gkatzia et al., 2014) and semantic representations (Theune et al., 2001).

This study focuses on generating natural language based on Abstract Meaning Representations (AMRs) (Banarescu et al., 2013). AMRs encode the meaning of a sentence as a rooted, directed and acyclic graph, where nodes represent concepts, and labeled directed edges represent relations among these concepts. The formalism strongly relies on the PropBank notation. Figure 1 shows an example.

[1]https://github.com/ThiagoCF05/LinearAMR

AMRs have increased in popularity in recent years, partly because they are relatively easy to produce, to read and to process automatically. In addition, they can be systematically translated into first-order logic, allowing for a well-specified model-theoretic interpretation (Bos, 2016). Most earlier studies on AMRs have focused on text understanding, i.e. processing texts in order to produce AMRs (Flanigan et al., 2014; Artzi et al., 2015). However, recently the reverse process, i.e. the generation of texts from AMRs, has started to receive scholarly attention (Flanigan et al., 2016; Song et al., 2016; Pourdamghani et al., 2016; Song et al., 2017; Konstas et al., 2017).

We assume that in practical applications, conceptualisation models or dialogue managers (models which decide "*what to say*") output AMRs. In this paper we study different ways in which these AMRs can be converted into natural language (deciding "*how to say it*"). We approach this as a translation problem—automatically translating from AMRs into natural language—and the key-contribution of this paper is that we systematically compare different preprocessing strategies for two different MT systems: Phrase-based MT (PBMT) and Neural MT (NMT).

We look at potential benefits of three preprocessing steps on AMRs before feeding them into an MT system: *delexicalisation*, *compression*, and *linearisation*. Delexicalisation decreases the sparsity of an AMR by removing constant values, compression removes nodes and edges which are less likely to be aligned to any word on the textual side and linearisation 'flattens' the AMR in a specific order. Com-

Proceedings of The 10th International Natural Language Generation conference, pages 1–10,
Santiago de Compostela, Spain, September 4-7 2017.

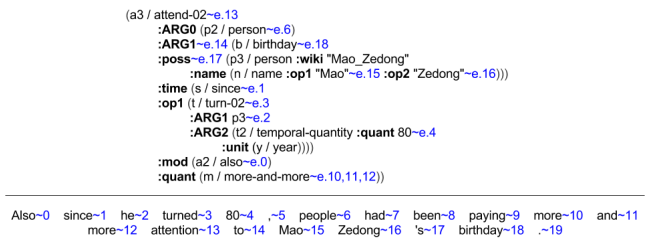

Figure 1: Example of an AMR

bining all possibilities gives rise to $2^3 = 8$ AMR preprocessing strategies, which we evaluate for two different MT systems: PBMT and NMT.

Following earlier work in AMR-to-text generation and the MT literature, we evaluate the system outputs in terms of fluency, adequacy and post-editing effort, using BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007) and TER (Snover et al., 2006) scores, respectively. We show that preprocessing helps, although the extent of the benefits differs for the two MT systems.

2 Related Studies

To the best of our knowledge, Flanigan et al. (2016) was the first study that introduced a model for natural language generation from AMRs. The model consists of two steps. First, the AMR-graph is converted into a spanning tree, and then, in a second step, this tree is converted into a sentence using a tree transducer.

In Song et al. (2016), the generation of a sentence from an AMR is addressed as an asymmetric generalised traveling salesman problem (AGTSP). For sentences shorter than 30 words, the model does not beat the system described by Flanigan et al. (2016). However, Song et al. (2017) treat the AMR-to-text task using a Synchronous Node Replacement Grammar (SNRG) and outperform Flanigan et al. (2016).

Although AMRs do not contain articles and do not represent inflectional morphology for tense and number (Banarescu et al., 2013), the formalism is relatively close to the (English) language. Motivated by this similarity, Pourdamghani et al. (2016) proposed an AMR-to-text method that organises some of these concepts and edges in a flat representation, commonly known as *Linearisation*. Once the linearisation is complete, Pourdamghani et al. (2016) map the flat AMR into an English sentence using a Phrase-Based Machine Translation (PBMT) system. This method yields better results than Flanigan et al. (2016) on development and test set from the LDC2014T12 corpus.

Pourdamghani et al. (2016) train their system using a set of AMR-sentence pairs obtained by the aligner described in Pourdamghani et al. (2014). In order to decrease the sparsity of the AMR formalism caused by the ratio of broad vocabulary and relatively small amount of data, this aligner drops a considerable amount of the AMR structure, such as role edges :ARG0, :ARG1, :mod, etc. However, inspection of the gold-standard alignments provided in the LDC2016E25 corpus revealed that this rule-based compression can be harmful for the generation of sentences, since such role edges can actually be aligned to function words in English sentences. So having these roles available arguably could improve AMR-to-text translation. This indicates that a better comparison of the effects of different preprocessing steps is called for, which we do in this study.

In addition, Pourdamghani et al. (2016) use PBMT, which is devised for translation but also utilised in other NLP tasks, e.g. text simplification (Wubben et al., 2012; Štajner et al., 2015). However, these systems have the disadvantage of having many different feature functions, and finding optimal settings for all of them increases the complexity of the problem from an engineering point of view.

An alternative MT model has been proposed: Neural Machine Translation (NMT). NMT models frame translation as a sequence-to-sequence problem (Bahdanau et al., 2015), and have shown strong results when translating between many different language pairs (Bojar et al., 2015). Recently, Konstas et al. (2017) introduce sequence-to-sequence models for parsing (text-to-AMR) and generation (AMR-to-text). They use a semi-supervised training proce-

dure, incorporating 20M English sentences which do not have a gold-standard AMR, thus overcoming the limited amount of data available. They report state-of-the-art results for the task, which suggests that NMT is a promising alternative for AMR-to-text.

3 Models

We describe our AMR-to-text generation models, which rely on 3 preprocessing steps (*delexicalisation*, *compression*, and/or *linearisation*) followed by a machine translation and realisation steps.

3.1 Delexicalisation

Inspection of the LDC2016E25 corpus reveals that on average 22% of the structure of an AMR are AMR constant values, such as names, quantities, and dates. This information increases the sparsity of the data, and makes it arguably more difficult to map an AMR into a textual format. To address this, Pourdamghani et al. (2016) look for special realisation component for names, dates and numbers in development and test sets and add them on the training set. On the other hand, similar to Konstas et al. (2017), we delexicalised these constants, replacing the original information for tags (e.g., `__name1__`, `__quant1__`). A list of tag-values is kept, aiming to identifying the position and to insert the original information in the sentence after the translation step is completed. Figure 2 shows a delexicalised AMR.

3.2 Compression

Given the alignment between an AMR and a sentence, the nodes and edges in the AMR can either be aligned to words in the sentence or not. So before the linearisation step, we would like to know which elements of an AMR should actually be part of the 'flattened' representation.

Following the aligner of Pourdamghani et al. (2014), Pourdamghani et al. (2016) clean an AMR by removing some nodes and edges independent of the context. Instead, we are using alignments that may relate a given node or edge to an English word according to the context. In Figure 1 for instance, the first edge :ARG1 is aligned to the preposition *to* from the sentence, whereas the second edge with a similar value is not aligned to any word in the sentence. Therefore, we need to train a classifier to decide which parts of an AMR should be in the flattened representation according to the context.

To solve the problem, we train a Conditional Random Field (CRF) which determines whether a node or an edge of an AMR should be included in the flattened representation. The classification process is sequential over a flattened representation of an AMR obtained by depth first search through the graph. Each element is represented by their name and parent name. We use *CRFSuite* (Okazaki, 2007) to implement our model.

3.3 Linearisation

After Compression, we flatten the AMR to serve as input to the translation step, similarly as proposed in Pourdamghani et al. (2016). We perform a depth-first search through the AMR, printing the elements according to their visiting order. In a second step, also following Pourdamghani et al. (2016), we implemented a version of the 2-Step Classifier from Lerner and Petrov (2013) to preorder the elements from an AMR according to the target side.

2-Step Classifier We implement the preordering method proposed by Lerner and Petrov (2013) in the following way. We define the order among a head node and its subtrees in two steps. In the first, we use a trained maximum entropy classifier to predict for each subtree whether it should occur before or after the head node. As features, we represent the head node by its frameset, whereas the subtree is represented by its head node frameset and parent edge.

Once we divide the subtrees into the ones which should occur before and after the head node, we use a maximum entropy classifier for the size of the subtree group to predict their order. For instance, for a group of 2 subtrees, a maximum entropy classifier specific for groups of 2 subtrees would be used to predict the permutation order of them (0-1 or 1-0). As features, the head node is also represented by its PropBank frameset, whereas the subtrees of the groups are represented by their parent edges, their head node framesets and by which side of the head node they are (before or after). We train classifiers for groups of sizes between 2 and 4 subtrees. For bigger groups, we used the depth first search order.

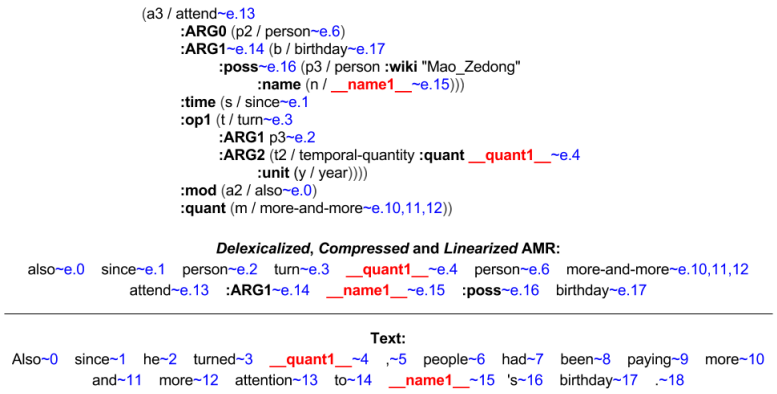

Figure 2: Example of a *Delexicalised*, *Compressed* and *Linearised* AMR

3.4 Translation models

To map a flat AMR representation into an English sentence, we use phrase-based (Koehn et al., 2003) and neural machine translation (Bahdanau et al., 2015) models.

3.4.1 Phrase-Based Machine Translation

These models use Bayes rule to formalise the problem of translating a text from a source language f to a target language e. In our case, we want to translate a flat amr into an English sentence e as Equation 1 shows.

$$P(e \mid amr) = argmax\ P(amr \mid e)P(e) \quad (1)$$

The *a priori* function $P(e)$ usually is represented by a language model trained on the target language. The *a posteriori* equation is calculated by the log-linear model described at Equation 2.

$$P(amr \mid e) = argmax \sum_{j=1}^{J} \lambda_j h_j(amr, e) \quad (2)$$

Each $h_j(amr, e)$ is an arbitrary feature function over AMR-sentence pairs. To calculate it, the flat amr is segmented into I phrases $\bar{amr}_1^I$, such that each phrase $\bar{amr}_i$ is translated into a target phrase $\bar{e}_i$ as described by Equation 3.

$$h_j(amr, e) = argmax\ h_j(\bar{amr}_i^I, \bar{e}_i^I) \quad (3)$$

As feature functions, we used direct and inverse phrase translation probabilities and lexical weighting; word, unknown word and phrase penalties.

We also used models to reorder a flat amr according to the target sentence e at decoding time. They work on the word-level (Koehn et al., 2003), at the level of adjacent phrases (Koehn et al., 2005) and beyond adjacent phrases (hierarchical-level) (Galley and Manning, 2008). Phrase- and hierarchical level models are also known as lexicalized reordering models.

As Koehn et al. (2003), given s_i the start position of the source phrase $\bar{amr}_i$ translated into the English phrase $\bar{e}_i$, and f_{i-1} the end position of the source phrase $\bar{amr}_{i-1}$ translated into the English phrase $\bar{e}_{i-1}$, a distortion model $\alpha^{|s_i - f_{i-1} - 1|}$ is defined as a distance-based reordering model. α is chosen by tunning the model.

Lexicalised models are more complex than distance-based ones, but usually help the system to obtain better results (Koehn et al., 2005; Galley and Manning, 2008). Given a possible set of target phrases $e = (\bar{e}_1, \ldots, \bar{e}_n)$ based on a source amr, and a set of alignments $a = (a_1, \ldots, a_n)$ that maps a source phrase $\bar{amr}_{a_i}$ into a target phrase $\bar{e}_i$, a lexicalised model aims to predict a set of orientations $o = (o_1, \ldots, o_n)$ as Equation 4 shows.

$$P(o \mid e, amr) = \prod_{i=1}^{n} P(o_i \mid \bar{e}_i, \bar{amr}_{a_i}, a_{i-1}, a_i) \quad (4)$$

Each orientation o_i, attached to the hypothesised target phrase e_i, can be a monotone (M), swap (S) or discontinuous (D) operation according to Equation 5.

$$o_i = \begin{cases} M, & \text{if } a_i - a_{i-1} = 1 \\ S, & \text{if } a_i - a_{i-1} = -1 \\ D, & \text{if } |a_i - a_{i-1}| \neq 1 \end{cases} \quad (5)$$

In the hierarchical model, we distinguished the discontinuous operation by the direction: discontinuous right ($a_i - a_{i-1} < 1$) and discontinuous left ($a_i - a_{i-1} > 1$). These models are important for our task, since the preordering method used in the Linearisation step can be insufficient to adequate it to the target sentence order.

3.4.2 Neural Machine Translation

Following the attention-based Neural Machine Translation (NMT) model introduced by Bahdanau et al. (2015), given a flat $amr = (amr_1, amr_2, \cdots, amr_N)$ and its English sentence translation $e = (e_1, e_2, \cdots, e_M)$, a single neural network is trained to translate amr into e by directly learning to model $p(e \mid amr)$. The network consists of one *encoder*, one *decoder*, and one *attention mechanism*.

The encoder is a bi-directional RNN with gated recurrent units (GRU) (Cho et al., 2014), where one forward RNN $\overrightarrow{\Phi}_{\text{enc}}$ reads the amr from left to right and generates a sequence of *forward annotation vectors* $(\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \cdots, \overrightarrow{\boldsymbol{h}}_N)$ at each encoder time step $i \in [1, N]$, and a backward RNN $\overleftarrow{\Phi}_{\text{enc}}$ reads the amr from right to left and generates a sequence of *backward annotation vectors* $(\overleftarrow{\boldsymbol{h}}_N, \overleftarrow{\boldsymbol{h}}_{N-1}, \cdots, \overleftarrow{\boldsymbol{h}}_1)$. The final annotation vector is the concatenation of forward and backward vectors $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}\right]$, and $C = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N)$ is the set of source annotation vectors.

The decoder is a neural LM conditioned on the previously emitted words and the source sentence via an attention mechanism over C. A multilayer perceptron is used to initialise the decoder's hidden state $\boldsymbol{s}_0$, where the input to this network is the concatenation of the last forward and backward vectors $\left[\overrightarrow{\boldsymbol{h}_N}; \overleftarrow{\boldsymbol{h}_1}\right]$.

At each time step t of the decoder, we compute a *time-dependent* context vector $\boldsymbol{c}_t$ based on the annotation vectors C, the decoder's previous hidden state $\boldsymbol{s}_{t-1}$ and the target English word $\tilde{e}_{t-1}$ emitted by the decoder in the previous time step. A single-layer feed-forward network computes an *expected alignment* $a_{t,i}$ between each source annotation vector $\boldsymbol{h}_i$ and the target word to be emitted at the current time step t, as in (6):

$$a_{t,i} = {\boldsymbol{v}_a}^T \tanh(\boldsymbol{U}_a \boldsymbol{s}_{t-1} + \boldsymbol{W}_a \boldsymbol{h}_i). \quad (6)$$

In Equation (7), these expected alignments are normalised and converted into probabilities:

$$\alpha_{t,i} = \frac{\exp(a_{t,i})}{\sum_{j=1}^{N} \exp(a_{t,j})}, \quad (7)$$

where $\alpha_{t,i}$ are called the model's *attention weights*, which are in turn used in computing the time-dependent context vector $\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha_{t,i} \boldsymbol{h}_i$. Finally, the context vector $\boldsymbol{c}_t$ is used in computing the decoder's hidden state $\boldsymbol{s}_t$ for the current time step t, as shown in Equation (8):

$$\boldsymbol{s}_t = \Phi_{\text{dec}}(\boldsymbol{s}_{t-1}, \boldsymbol{W_e}[\tilde{e}_{t-1}], \boldsymbol{c}_t), \quad (8)$$

where $\boldsymbol{s}_{t-1}$ is the decoder's previous hidden state, $\boldsymbol{W_e}[\tilde{e}_{t-1}]$ is the embedding of the word emitted in the previous time step, and $\boldsymbol{c}_t$ is the updated time-dependent context vector. Given a hidden state $\boldsymbol{s}_t$, the probabilities for the next target word are computed using one projection layer followed by a softmax, as illustrated in eq. (9), where the matrices $\boldsymbol{L}_o$, $\boldsymbol{L}_s$, $\boldsymbol{L}_w$ and $\boldsymbol{L}_c$ are transformation matrices and $\boldsymbol{c}_t$ is the time-dependent context vector.

3.5 Realisation

Since we delexicalise names, dates, quantities and values from AMRs, we need to textually realise this information once we obtain the results from the translation step. As we kept all the original information and their relation with the tags, we just need to replace one for the other.

We implement some rules to adequate our generated texts to the ones we saw in the training set. Different from the AMRs, we represent months nominally, and not numerically - month *5* will be *May* for example. Values and quantities bigger than a thousand are also part realised nominally. The value *8500000000* would be realised as *8.5 billion* for instance. On the other hand, names are realised as they are.

$$p(e_t = k \mid \boldsymbol{e_{<t}}, \boldsymbol{c_t}) \propto \exp(\boldsymbol{L_o} \tanh(\boldsymbol{L_s s}_t + \boldsymbol{L_w E_e}[\hat{e}_{t-1}] + \boldsymbol{L_c c_t})). \quad (9)$$

4 Evaluation

4.1 Data

We used the corpus LDC2016E25 provided by the SemEval 2017 Task 9 in our evaluation. This corpus consists of aligned AMR-sentence pairs, mostly newswire. We considered the train/dev/test sets splitting proposed in the original setting, totaling 36,521, 1,368 and 1,371 AMR-sentence pairs, respectively. Compression and Linearisation methods, as well as Phrase-based Machine Translation models were trained over the gold-standard alignments between AMRs and sentences on the training set of the corpus.

4.2 Evaluated Models

We test models with and without the Delexicalisation/Realisation (-Delex and +Delex) and Compression (-Compress and +Compress) steps. In models without the Compression step, we include all the elements from an AMR in the flattened representation. For the Linearisation step, we flatten the AMR structure based on a depth-first search (-Preorder) or preorder it with our 2-step classifier (+Preorder). Finally, we translate a flattened AMR into text using a Phrase-based (PBMT) and a Neural Machine Translation model (NMT). In total, we evaluated 16 models.

Phrase-based Machine Translation We used a standard PBMT system built using Moses toolkit (Koehn et al., 2007). At training time, we extract and score phrase sentences up to the size of 9 tokens. All the feature functions were trained using the gold-standard alignments from the training set and their weights were tuned on the development data using k-batch MIRA with $k = 60$ (Cherry and Foster, 2012) with BLEU as the evaluation metric. A distortion limit of 6 was used for the reordering models. Lexicalised reordering models were bidirectional. At decoding time, we use a stack size of 1000.

Our language model $P(e)$ is a 5-gram LM trained on the Gigaword Third Edition corpus using KenLM (Heafield et al., 2013). For the models with the Delexicalisation step, we trained the language model with a delexicalised version of Gigaword by parsing the corpus using the Stanford Named Entity Recognition tool (Finkel et al., 2005). All the entities labeled as `LOCATION`, `PERSON`, `ORGANISATION` or `MISC` were replaced by the tag `__nameX__`. Entities labeled as `NUMBER` or `MONEY` were replaced by the tag `__quantX__`. Finally, entities labeled as `PERCENT` or `ORDINAL` were replaced by `__valueX__`. In the tags, `X` is replaced by the ordinal position of the entity in the sentence.

Neural Machine Translation The encoder is a bidirectional RNN with GRU, each with a 1024D hidden unit. Source and target word embeddings are 620D each and are both trained jointly with the model. All non-recurrent matrices are initialised by sampling from a Gaussian ($\mu = 0, \sigma = 0.01$), recurrent matrices are random orthogonal and bias vectors are all initialised to zero. The decoder RNN also uses GRU and is a neural LM conditioned on its previous emissions and the source sentence by means of the source attention mechanism.

We apply dropout with a probability of 0.3 in both source and target word embeddings, in the encoder and decoder RNNs inputs and recurrent connections, and before the readout operation in the decoder RNN. We follow Gal and Ghahramani (2016) and apply dropout to the encoder and decoder RNNs using the same mask in all time steps.

Models are trained using stochastic gradient descent with Adadelta (Zeiler, 2012) and minibatches of size 40. We apply early stopping for model selection based on BLEU scores, so that if a model does not improve on the validation set for more than 20 epochs, training is halted.

4.3 Models for Comparison

We compare BLEU scores for some of the AMR-to-text systems described in the literature (Flanigan et al., 2016; Song et al., 2016; Pourdamghani et al., 2016; Song et al., 2017; Konstas et al., 2017). Since the models of Flanigan et al. (2016) and Pourdamghani et al. (2016) are publicly available, we also use them with the same training data as our models. For Flanigan et al. (2016), we specifically

use the version available on GitHub[2].

For Pourdamghani et al. (2016), we use the version available at the first author's website[3]. The rules used for the preordering model and the feature functions from the PBMT system are trained using alignments over AMR–sentence pairs from the training set obtained with the aligner described by Pourdamghani et al. (2014). We do not use lexicalised reordering models as Pourdamghani et al. (2016). Moreover, we tune the weights of the feature functions with MERT (Och, 2003).

Both models make use of a 5-gram language model trained on Gigaword Third Edition corpus with KenLM.

4.4 Metrics

To evaluate fluency, adequacy and post-editing effort of the models, we use BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007) and TER (Snover et al., 2006), respectively.

5 Results

Table 1 depicts the scores of the different models by the size of the data they were trained on. For illustration, we depicted the BLEU scores of all the AMR-to-text systems described in the literature. The models of Flanigan et al. (2016) and Pourdamghani et al. (2016) were officially trained with 10,313 AMR-sentence pairs from the LDC2014T12 corpus, and with 36,521 AMR-sentence pairs from the LDC2016E25 in our study (as our models). The ones of Song et al. (2016) and Song et al. (2017) were trained with 16,833 pairs from the LDC2015E86 corpus. Konstas et al. (2017), which presents the highest quantitative result in the task so far, also used the LDC2015E86 corpus plus 20 million English sentences from the Gigaword corpus with a semi-supervised approach. We report the results when their model were trained only with AMR-sentence pairs from the corpus, and when improved with more 20 million sentences.

Among the PBMT models, the Delexicalisation step (+Delex) does not seem to play a role in obtaining better sentences from AMRs. All the models with the preordering method in Linearisation (+Preorder) introduce better results than Flanigan et al. (2016) and Song et al. (2016), whereas only the lexicalised models with the preordering method (PBMT+Delex[+|-]Compress+Preorder) outperform Song et al. (2017) and introduce competitive results with Pourdamghani et al. (2016).

In our NMT models, apparently the Compression step is harmful to the task, whereas Delexicalisation and preordering in Linearisation lead to better results. However, none of the NMT models outperform neither the PBMT models nor the baselines.

	Data Size	BLEU	METEOR	TER
(Flanigan et al., 2016)	∼10K	22.1	–	–
(Pourdamghani et al., 2016)	∼10K	26.9	–	–
(Konstas et al., 2017)	∼17K	22.0	–	–
(Song et al., 2016)	∼17K	22.4	–	–
(Song et al., 2017)	∼17K	25.6	–	–
(Flanigan et al., 2016)	∼36K	19.6	–	–
(Pourdamghani et al., 2016)	∼36K	24.3	–	–
(Konstas et al., 2017)	∼20M	33.8	–	–
NMT				
+Delex-Compress-Preorder	∼36K	18.9	26.6	66.2
+Delex+Compress-Preorder		14.6	23.6	77.0
+Delex-Compress+Preorder		19.3	26.3	69.3
+Delex+Compress+Preorder		15.2	23.8	77.8
-Delex-Compress-Preorder		18.2	24.8	67.7
-Delex+Compress-Preorder		15.2	22.4	72.8
-Delex-Compress+Preorder		19.0	25.5	66.6
-Delex+Compress+Preorder		15.9	22.6	71.4
PBMT				
+Delex-Compress-Preorder	∼36K	20.6	32.8	64.5
+Delex+Compress-Preorder		22.2	33.0	63.3
+Delex-Compress+Preorder		24.6	34.3	60.4
+Delex+Compress+Preorder		23.9	33.7	60.5
-Delex-Compress-Preorder		21.0	32.7	65.5
-Delex+Compress-Preorder		25.6	34.1	60.9
-Delex-Compress+Preorder		26.5	34.9	59.9
-Delex+Compress+Preorder		26.8	34.7	59.4

Table 1: MT scores for the evaluated models by the size of the training data. Best baseline, PBMT and NMT results were underlined.

6 Discussion

In this paper, we studied models for AMR-to-text generation using machine translation. We systematically analysed the effects of 3 processing strategies on AMRs before feeding them either to a Phrase-based or a Neural MT system. The evaluation was performed on the LDC2016E25 corpus, provided by SemEval 2017 Task 9. All the models had the fluency, adequacy and post-editing effort of their produced sentences measured by BLEU, METEOR and TER, respectively. In general, we found that pro-

[2] http://github.com/jflanigan/jamr/tree/Generator

[3] http://isi.edu/~damghani/papers/amr2eng.zip

cessing AMRs helps, although the effects differ for the different systems.

Phrase-based MT Delexicalisation (+Delex) does not seem to play a role in obtaining better sentences from AMRs using PBMT. Our best model (PBMT-Delex+Compress+Preorder) presents competitive results to Pourdamghani et al. (2016) with the advantage that no technique is necessary to overcome data sparsity.

Compressing an AMR graph with a classifier shows improvements over a comparable model without compression, but not as strong as preordering the elements in the Linearisation step. In fact, preordering seems to be the most important preprocessing step across all three MT preprocessing metrics. We note that the preordering success was expected, based on previous results (Pourdamghani et al., 2016).

Neural MT The first impression from our NMT experiments is that using Compression consistently deteriorates translations according to all metrics evaluated. Delexicalisation seems to improve results, corroborating the findings from Konstas et al. (2017). While Delexicalisation is harmful and Compression is beneficial for PBMT, we see the opposite in NMT models. Besides the differences between these two MT architectures, applying preordering in the Linearisation step improves results in both cases. This seems to contradict the finding in Konstas et al. (2017) regarding neural models. We conjecture that the additional training data used by Konstas et al. (2017) may have decreased the gap between using and not using preordering (see also below). More research is necessary to settle this point.

PBMT vs. NMT PBMT models generate much better sentences from AMRs than NMT models in terms of fluency, adequacy and post-editing effort. We believe that the lower performance of NMT models is due to the small size of the training set (36,521 AMR-sentence pairs). Neural models are known to perform well when trained on much larger data sets, e.g. in the order of millions of entries, as exemplified by Konstas et al. (2017). PBMT models trained on small data sets clearly outperform NMT ones, e.g. Konstas et al. (2017) reported 22.0 BLEU, whereas Pourdamghani et al. (2016)'s best model achieved 26.9 BLEU, and our best model performs comparably (26.8 BLEU).

Model comparison While the best PBMT models are comparable to the state-of-the-art AMR-to-text systems, the current best results are reported by Konstas et al. (2017), showing the potential of applying deep learning onto large amounts of training data with a 33.8 BLEU-score. However, this result crucially relies on the existence of a very large dataset. Interestingly, when applied in a situation with limited amounts of data, Konstas et al. (2017) report substantially lower performance scores. In such situations, our PBMT models, like Pourdamghani et al. (2016), look appear to be a good alternative option.

7 Conclusion

In this work, we systematically studied different MT models to *translate* AMRs into natural language. We observed that the Delexicalisation, Compression, and Linearisation steps have different impacts on AMR-to-text generation depending on the MT architecture used. We observed that delexicalising AMRs yields the best results in NMT models, in contrast to PBMT models. On the other hand, for both PBMT models and NMT models, preordering the AMR in Linearisation introduces better results.

Among our models, PBMT generally outperforms NMT. Finally, the literature suggests that the improvements obtained by having more data are larger than those obtained with improved preprocessing strategies. Nonetheless, combining the right preprocessing strategy with large volumes of training data should lead to further improvements.

Acknowledgments

This work has been supported by the National Council of Scientific and Technological Development from Brazil (CNPq). Iacer Calixto has received funding from Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund and the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21).

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations, ICLR 2015*, San Diego, California.

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

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Johan Bos. 2016. Expressive power of abstract meaning representations. *Comput. Linguist.*, 42(3):527–535, September.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California, June. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems, NIPS*, pages 1019–1027, Barcelona, Spain.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October. Association for Computational Linguistics.

A. Gatt and E. Krahmer. 2017. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *ArXiv e-prints*, March.

Dimitra Gkatzia, Helen Hastie, and Oliver Lemon. 2014. Comparing multi-label classification with reinforcement learning for summarisation of time-series data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1231–1240, Baltimore, Maryland, June. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneserney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria, August. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005

IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. *ArXiv e-prints*, April.

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*, StatMT '07, pages 228–231, Prague, Czech Republic.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Seattle, Washington, USA, October. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October. Association for Computational Linguistics.

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, September 5-8. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

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 Association for Machine Translation in the Americas, AMTA*, pages 223–231, Cambridge, MA, USA.

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, Texas, November. Association for Computational Linguistics.

L. Song, X. Peng, Y. Zhang, Z. Wang, and D. Gildea. 2017. AMR-to-text Generation with Synchronous Node Replacement Grammar. *ArXiv e-prints*, February.

M. Theune, E. Klabbers, J. R. De Pijper, E. Krahmer, and J. Odijk. 2001. From data to speech: A general approach. *Nat. Lang. Eng.*, 7(1):47–86, March.

Sanja Štajner, Iacer Calixto, and Horacio Saggion. 2015. Automatic text simplification for spanish: Comparative evaluation of various simplification strategies. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 618–626, Hissar, Bulgaria, September.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 1015–1024, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.