Abstract

Syntactically controlled paraphrase generation has become an emerging research direction in recent years. Most existing approaches require annotated paraphrase pairs for training and are thus costly to extend to new domains. Unsupervised approaches, on the other hand, do not need paraphrase pairs but suffer from relatively poor performance in terms of syntactic control and quality of generated paraphrases. In this paper, we demonstrate that leveraging Abstract Meaning Representations (AMR) can greatly improve the performance of unsupervised syntactically controlled paraphrase generation. Our proposed model, AMR-enhanced Paraphrase Generator (AMRPG), separately encodes the AMR graph and the constituency parse of the input sentence into two disentangled semantic and syntactic embeddings. A decoder is then learned to reconstruct the input sentence from the semantic and syntactic embeddings. Our experiments show that AMRPG generates more accurate syntactically controlled paraphrases, both quantitatively and qualitatively, compared to the existing unsupervised approaches. We also demonstrate that the paraphrases generated by AMRPG can be used for data augmentation to improve the robustness of NLP models.

1 Introduction

Syntactically controlled paraphrase generation approaches aim to control the format of generated paraphrases by taking into account additional parse specifications as the inputs, as illustrated by Figure 1. It has attracted increasing attention in recent years since it can diversify the generated paraphrases and benefit a wide range of NLP applications (Iyyer et al., 2018; Huang and Chang, 2021; Sun et al., 2021), including task-oriented dialog generation (Gao et al., 2020), creative generation (Tian et al., 2021), and model robustness (Huang and Chang, 2021).

Recent works have shown success in training syntactically controlled paraphrase generators (Iyyer et al., 2018; Chen et al., 2019; Kumar et al., 2020; Sun et al., 2021). Although their models can generate high-quality paraphrases and achieve good syntactic control ability, the training process needs a large amount of supervised data, e.g., parallel paraphrase pairs. Annotating paraphrase pairs is usually expensive because it requires intensive domain knowledge and high-level semantic understanding. Due to the difficulty in collecting parallel data, the ability of supervised approaches are limited, especially when adapting to new domains.

To reduce the annotation demand, unsupervised approaches can train syntactically controlled paraphrase generators without the need for parallel pairs (Zhang et al., 2019; Bao et al., 2019; Huang and Chang, 2021). Most of them achieve syntactic control by learning disentangled embeddings for semantics and syntax separately (Bao et al., 2019; Huang and Chang, 2021). However, without parallel data, it is challenging to learn a good disentanglement and capture semantics well. As we will show later (Section 4.1), unsupervised approaches can generate bad paraphrases by mistakenly swapping object and subject of a sentence.

In this work, we propose to use Abstract Meaning Representations (AMR) (Banarescu et al., 2013) to learn better disentangled semantic embeddings.
for unsupervised syntactically controlled paraphrase generation. AMR is a semantic graph structure that covers the abstract meaning of a sentence. As shown in Figure 2, two sentences would have the same (or similar) AMR graph as long as they carry the same abstract meaning, even they are expressed with different syntactic structures. This property makes AMRs a good resource to capture sentence semantics.

Based on this, we design an AMR-enhanced Paraphrase Generator (AMRPG), which separately learns (1) semantic embeddings with the AMR graphs extracted from the input sentence and (2) syntactic embeddings from the constituency parse of the input sentence. Then, AMRPG trains a decoder to reconstruct the input sentence from the semantic and syntactic embeddings. The reconstruction objective and the design of the disentanglement of semantics and the syntax makes AMRPG learn to generate syntactically controlled paraphrases without using parallel pairs. Our experiments show that AMRPG performs better syntactic control than existing unsupervised approaches. Additionally, we demonstrate that the generated paraphrases of AMRPG can be used for data augmentation to improve the robustness of NLP models.

2 Related Work

Paraphrase generation. Traditional paraphrase generators are usually based on hand-crafted rules (Barzilay and Lee, 2003) or seq2seq models (Cao et al., 2017; Gupta et al., 2018; Fu et al., 2019). To generate diverse paraphrases, different techniques are proposed, including random pattern embeddings (Kumar et al., 2019), latent space perturbation (Roy and Grangier, 2019; Zhang et al., 2019; Cao and Wan, 2020), multi-round generation (Lin and Wan, 2021), reinforcement learning (Liu et al., 2020), prompt-tuning (Chowdhury et al., 2022), order control (Goyal and Durrett, 2020), and syntactic control (Iyyer et al., 2018; Kumar et al., 2020; Huang and Chang, 2021; Sun et al., 2021).

Abstract meaning representation (AMR). Since AMR (Banarescu et al., 2013) captures high-level semantics, it has been applied for various NLP tasks, including summarization (Sachan and Xing, 2016), dialogue modeling (Bai et al., 2021), information extraction (Zhang et al., 2021). Some works also focus on training high-quality AMR parsers with graph encoders (Cai and Lam, 2020), seq2seq models (Konstas et al., 2017; Zhou et al., 2020), and decoder-only models (Bevilacqua et al., 2021).

3 Unsupervised Syntactically Controlled Paraphrase Generation

3.1 Problem Formulation

We follow previous works (Iyyer et al., 2018; Huang and Chang, 2021) and consider constituency parses (without terminals) as the control signals. Given a source sentence $s$ and a target parse $p$, the goal of the syntactically controlled paraphrase generator is to generate a target sentence $t$ which has similar semantics to the source sentence $s$ and has syntax following the parse $p$. In the unsupervised setting, the paraphrase generator cannot access any target sentences and target parses but only the source sentences and source parses during training.

3.2 Proposed Method: AMRPG

Motivated by previous approaches (Bao et al., 2019; Huang and Chang, 2021), we design AMRPG to learn separate embeddings for semantics and syntax, as illustrated by Figure 3. Then, AMRPG learns a decoder with the objective to reconstruct the source sentence. The challenge here is how to learn embeddings such that the semantic embedding contains only semantic information while the syntactic embedding contains only syntactic information. We introduce the details as follows.

Semantic embedding. Given a source sentence, we first use a pre-trained AMR parser\footnote{https://github.com/bjascob/amrlib-models} to get its AMR graph. Next, we use a semantic encoder to encode the AMR graph into the semantic embedding $e_{sem}$. Specifically, the semantic encoder consists of two parts: a fixed pre-trained AMR encoder (Ribeiro et al., 2021) followed by a learnable Transformer encoder. We additionally perform node masking when training the semantic encoder. Specifically, every node in the AMR graph has a
Figure 3: AMRPG’s framework. It separately encodes the AMR graph and the constituency parse of the input sentence into two disentangled semantic and syntactic embeddings. A decoder is then learned to reconstruct the input sentence from the semantic and syntactic embeddings.

probability to be masked out during training. This can improve the robustness of AMRPG.

As mentioned above, two semantically similar sentences would have similar AMR graphs regardless of their syntax. This property encourages AMRPG to capture only semantic information in semantic embeddings. Compared with previous work (Huang and Chang, 2021), which uses bag-of-words to learn the semantic embeddings, using AMR can capture semantics better and lead to better performance, as shown in Section 4.

Syntactic embedding. Given a source sentence, we use the Stanford CoreNLP toolkit (Manning et al., 2014) to get its constituency parse. Then, we remove all the terminals in the parse and learns a Transformer encoder to encode the parse into the syntactic embedding $e_{\text{syn}}$. Since we remove the terminals, the syntactic embedding contains only the syntactic information of the source sentence.

Decoder. We train a Transformer decoder that takes the semantic embedding $e_{\text{sem}}$ and the syntactic embedding $e_{\text{syn}}$ as the input, and reconstructs the source sentence with a cross-entropy loss. The reconstruction objective makes AMRPG not require parallel paraphrase pairs for training.

Inference. Given a source sentence $s$ and a target parse $p$, we use the semantic encoder to encode the AMR graph of $s$ into the semantic embedding, use the syntactic encoder to encode $p$ into the syntactic embedding, and use the decoder to generate the target sentence $t$.

4 Experiments

4.1 Syntactically Controlled Paraphrase Generation

Datasets. We consider ParaNMT (Wieting and Gimpel, 2018) for training and testing. We use only the source sentences in ParaNMT to train AMRPG and other unsupervised baselines, and use both the source sentences and target sentences to train supervised baselines. To further test the model’s ability to generalize to new domains, we directly use the models trained with ParaNMT to test on Quora (Iyyer et al., 2017), MRPC (Dolan et al., 2004), and PAN (Madnani et al., 2012)

Evaluation metrics. Following the previous work (Huang and Chang, 2021), we consider the BLEU score to measure the similarity between the gold target sentences and the predicted target sentences, and consider the template matching accuracy$^2$ (TMA) to evaluate the goodness of syntactic control. More details about the evaluation can be found in Appendix B.2.

Baselines. We consider the following unsupervised models: SIVAE (Zhang et al., 2019), SynPG (Huang and Chang, 2021), AMRPG, and T5-Baseline, which replaces the AMR encoder with a T5-encoder. We also consider SCPN (Iyyer et al., 2018) as the supervised baseline.

Results. Table 1 shows the results of syntactically controlled paraphrase generation. AMRPG performs the best among the unsupervised approaches. Specifically, AMRPG outperforms SynPG, the

---

$^2$Template matching accuracy is defined as the exact matching accuracy of top-2 levels of parse trees.
Table 1: Results of syntactically controlled paraphrase generation. AMRPG performs the best among all unsupervised approaches and can outperform supervised approaches when considering the target domain source sentences.

<table>
<thead>
<tr>
<th>Model</th>
<th>ParaNMT TMA</th>
<th>ParaNMT BLEU</th>
<th>Quora TMA</th>
<th>Quora BLEU</th>
<th>PAN TMA</th>
<th>PAN BLEU</th>
<th>MRPC TMA</th>
<th>MRPC BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsupervised Approaches (without using parallel pairs)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIVAE (Zhang et al., 2019)</td>
<td>30.0</td>
<td>12.8</td>
<td>48.3</td>
<td>13.1</td>
<td>26.6</td>
<td>11.8</td>
<td>21.5</td>
<td>5.1</td>
</tr>
<tr>
<td>SynPG (Huang and Chang, 2021)</td>
<td>71.0</td>
<td>32.2</td>
<td>82.6</td>
<td>33.2</td>
<td><strong>66.3</strong></td>
<td>26.4</td>
<td><strong>74.0</strong></td>
<td>26.2</td>
</tr>
<tr>
<td>T5-Baseline</td>
<td>57.1</td>
<td>22.8</td>
<td>66.1</td>
<td>22.2</td>
<td>55.3</td>
<td>21.0</td>
<td>66.2</td>
<td>18.8</td>
</tr>
<tr>
<td>AMRPG</td>
<td><strong>74.3</strong></td>
<td><strong>39.1</strong></td>
<td><strong>84.8</strong></td>
<td><strong>33.9</strong></td>
<td>65.6</td>
<td><strong>31.0</strong></td>
<td>71.9</td>
<td><strong>34.8</strong></td>
</tr>
<tr>
<td><strong>Unsupervised Approaches (using target domain source sentences)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SynPG (Huang and Chang, 2021)</td>
<td>-</td>
<td>-</td>
<td>86.3</td>
<td>44.4</td>
<td>66.4</td>
<td>34.2</td>
<td><strong>80.7</strong></td>
<td>44.6</td>
</tr>
<tr>
<td>AMRPG</td>
<td>-</td>
<td>-</td>
<td><strong>86.5</strong></td>
<td><strong>45.4</strong></td>
<td><strong>67.5</strong></td>
<td><strong>37.6</strong></td>
<td>76.8</td>
<td><strong>45.9</strong></td>
</tr>
<tr>
<td><strong>Supervised Approaches (using additional parallel pairs in ParaNMT; not comparable to ours)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCPN (Iyyer et al., 2018)</td>
<td>83.9</td>
<td>58.3</td>
<td>87.1</td>
<td>41.0</td>
<td>72.3</td>
<td>37.6</td>
<td>80.1</td>
<td>41.8</td>
</tr>
</tbody>
</table>

Table 2: Paraphrase examples generated by SynPG and AMRPG. AMRPG captures semantics better and generates higher quality of paraphrases than SynPG.

<table>
<thead>
<tr>
<th>Input</th>
<th>Parse template</th>
<th>Target</th>
<th>SynPG</th>
<th>AMRPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>The dog chased the cat on the street.</td>
<td>$(S \ (NP \ (DT) \ (NN)) \ (VP \ (VBN) \ (PP)) \ (.))$</td>
<td>The cat was chased by the dog on the street.</td>
<td>The dog was chased by the cat on the street.</td>
<td>The cat was chased by a dog in the street.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Parse template</th>
<th>Target</th>
<th>SynPG</th>
<th>AMRPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>John will send a gift to Tom when Christmas comes.</td>
<td>$(S \ ((SBAR \ (WHADVP) \ (S)) \ (,) \ (NP \ (NNP)) \ (VP \ (MD) \ (VP)) \ (.))$</td>
<td>When Christmas comes, John will send a gift to Tom.</td>
<td>When Tom comes, John will send a gift to Christmas.</td>
<td>When Christmas comes, John will send a gift to Tom.</td>
</tr>
</tbody>
</table>

We observe that there is indeed a performance gap between AMRPG and SCPN (supervised baseline). However, since AMRPG is an unsupervised model, it is possible to use the source sentences from the target domains to further fine-tune AMRPG without additional annotation cost. As shown in the table, AMRPG with further fine-tuning can achieve even better performance than SCPN when considering domain adaptation (Quora, MRPC, and PAN). This demonstrates the flexibility and the potential of unsupervised paraphrase models.

**Qualitative examples.** Table 2 lists some paraphrases generated by SynPG and AMRPG. As we mentioned in Section 3, SynPG uses bag-of-words to learn semantic embeddings and therefore SynPG is easy to get confused about the relations between entities or mistake the subject for the object. In contrast, AMRPG can preserve more semantics.

4.2 Improving Robustness of NLP Models

We demonstrate that the paraphrases generated by AMRPG can improve the robustness of NLP models by data augmentation. Following the setting of previous work (Huang and Chang, 2021), we consider three classification tasks in GLUE (Wang et al., 2019): MRPC, RTE, and SST-2. We compare three baselines: (1) the classifier trained with original training data, (2) the classifier trained with original training data and augmented data generated by SynPG, and (3) the classifier trained with original training data and augmented data generated by AMRPG. Specifically, for every instance in the original training data, we generate four paraphrases as the augmented examples by considering four common syntactic templates. More details can be found in Appendix C.1.

Table 3 shows the clean accuracy and the broken rate (the percentage of examples being attacked) after attacked by the syntactically adversarial examples generated with SCPN (Iyyer et al., 2018). Although the classifiers trained with data augmentation...
Table 3: Augmenting paraphrases generated by AMRPG improves the robustness of NLP models. Acc denotes the clean accuracy (the higher is the better). Brok denotes the percentage of examples being successfully attacked (the lower is the better).

<table>
<thead>
<tr>
<th>Model</th>
<th>MRPC</th>
<th>RTE</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>83.3</td>
<td>52.9</td>
<td>62.1</td>
</tr>
<tr>
<td>+ SynPG</td>
<td>80.6</td>
<td>42.2</td>
<td>61.7</td>
</tr>
<tr>
<td>+ AMRPG</td>
<td>80.6</td>
<td>38.3</td>
<td>58.8</td>
</tr>
</tbody>
</table>

5 Conclusion

We propose AMRPG that utilizes AMR to learn a better disentanglement of semantics and syntax without using any parallel data. This enables AMRPG to capture semantics better and generate more accurate syntactically controlled paraphrases than existing unsupervised approaches. We also demonstrate that how to apply AMRPG to improve the robustness of NLP models.

Limitations

Our goal is to demonstrate the potential of AMR for syntactically controlled paraphrase generation. The current experimental setting follows previous works (Iyyer et al., 2018; Huang and Chang, 2021), which considers the full constituency parses as the control signals. In real applications, getting full constituency parses before the paraphrase generation process might take additional efforts. One potential solution is to consider relatively noisy or simplified parse specifications (Sun et al., 2021). In addition, some parse specifications can be inappropriate for certain source sentences (e.g., the source sentence is long but the target parse is short). How to score and reject some of the given parse specifications is still an open research question. Finally, although training AMRPG does not require any parallel paraphrase pairs, it does require a pre-trained AMR parser, which can be a potential cost for training AMRPG.

Broader Impacts

Our proposed method focuses on improving syntactically controlled paraphrase generation. It is intended to be used to improve the robustness of models and facilitate language generation for applications with positive social impacts. All the experiments are conducted on open benchmark datasets. However, it is known that the models trained with a large text corpus could capture the bias reflecting the training data. It is possible for our model to potentially generate offensive or biased content learned from the data. We suggest to carefully examining the potential bias before deploying models in any real-world applications.

References


Laura Banerescu, 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 (LAW-ID@ACL).


A Implementation Details

We use around 20 millions of examples\(^4\) in ParaNMT (Wieting and Gimpel, 2018) to train AM-RPG and all baselines. The semantic encoder and the syntactic decoder are trained from scratch, with the default architecture and the default parameters of torch.nn.Transformer. The max length for input sentences, the linearized constituency parses, and the linearized AMR graph are set to 40, 160, and 250, respectively. The word dropout rate is 0.4 while the node masking rate is 0.6. We consider Adam optimizer with the learning rate being \(10^{-4}\) and the weight decay being \(10^{-5}\). The total number of epochs is set to 10. When generating the outputs, we use random sampling with temperature being 0.5. The model is trained with 4 NVIDIA V100 GPUs with 16 GB memory each. It takes around 7 days to finish the training process.

B Experimental Settings of Syntactically Controlled Paraphrase Generation

B.1 Datasets

Following previous work (Huang and Chang, 2021), our test data is: (1) 6,400 examples of ParaNMT (Wieting and Gimpel, 2018), (2) 6,400 examples of Quora (Iyer et al., 2017), (3) 2,048 examples of PAN (Madnani et al., 2012), and (4) 1,920 examples of MRPC (Dolan et al., 2004).

B.2 Evaluation

Following previous work (Huang and Chang, 2021), we consider paraphrase pairs to evaluate the performance. Given a paraphrase pairs \((s_1, s_2)\), we use the Standford CoreNLP constituency parser (Manning et al., 2014) to get their parses \((p_1, p_2)\). The input of all baselines would be \((s_1, p_2)\) and the ground truth would be \(s_2\).

Assuming the generated paraphrase is \(g\). We use BLEU score to measure the similarity between the generated paraphrase \(g\) and the ground truth \(s_2\). We also calculate the template matching accuracy (TMA) by computing the exact matching accuracy of the top-2 levels of \(p_g\) and \(p_2\) (\(p_g\) is the constituency parse of \(g\)).

C Experimental Settings of Model Robustness

C.1 Training Details

We use the pre-trained SynPG parse generator to generate the full parse for each instance with the following parse templates: “(S (NP) (VP) (.))”, “(S (VP) (.))”, “(NP (NP) (.))”, and “(FRAG (SBAR) (.))”. Then, we use the generated full parses as the parse specifications to generate paraphrases for data augmentation. When training classifiers with data augmentation, the original instances have four times of weights as the augmented instances when computing the loss. We use the scripts from Huggingface\(^5\) with default values to train the classifiers.

C.2 Generating Adversarial Examples

We use the official script\(^6\) of SCPN (Iyyer et al., 2018) to generate syntactically adversarial examples. Specifically, we consider the first five parse templates for RTE and SST-2 and first three parse templates for MRPC to generate the adversarial examples. As long as one of the adversarial examples makes the classifier change the prediction, we count it as a successful attack on this instance.

[^4]: https://github.com/uclanlp/synpg
[^6]: https://github.com/miyyer/scpn