SAPG: Semantically-Aware Paraphrase Generation with AMR Graphs

Afonso Sousa[a] and Henrique Lopes Cardoso[b]
Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC), Faculdade de Engenharia, Universidade do Porto, Portugal

Keywords: Paraphrase Generation, Abstract Meaning Representation, Semantic Representation, Pretrained Language Model, Graph Neural Networks.

Abstract: Automatically generating paraphrases is crucial for various natural language processing tasks. Current approaches primarily try to control the surface form of generated paraphrases by resorting to syntactic graph structures. However, paraphrase generation is rooted in semantics, but there are almost no works trying to leverage semantic structures as inductive biases for the task of generating paraphrases. We propose SAPG, a semantically-aware paraphrase generation model, which encodes Abstract Meaning Representation (AMR) graphs into a pretrained language model using a graph neural network-based encoder. We demonstrate that SAPG enables the generation of more diverse paraphrases by transforming the input AMR graphs, allowing for control over the output generations' surface forms rooted in semantics. This approach ensures that the semantic meaning is preserved, offering flexibility in paraphrase generation without sacrificing fluency or coherence. Our extensive evaluation on two widely-used paraphrase generation datasets confirms the effectiveness of this method.

1 INTRODUCTION

Humans use natural language to convey information, mapping an abstract idea to a sentence with a specific surface form. A paraphrase is an alternative surface form of the same underlying semantic content. The ability to automatically identify and generate paraphrases is of significant interest, with applications in data augmentation (Iyyer et al., 2018), question answering (Dong et al., 2017), duplicate question detection (Shah et al., 2018) and more recently on studies about detecting AI-generated text (Krishna et al., 2023).

Prior work on paraphrase generation has attempted to include inductive biases to control the generation, most often using syntactic structure to promote alternative surface forms (Iyyer et al., 2018; Chen et al., 2019a; Kumar et al., 2020; Meng et al., 2021; Shu et al., 2019; Hosking and Lapata, 2021). Semantic structures are rarely leveraged for the task of paraphrase generation. The few works that tried to use semantic structures to improve the quality of the generated paraphrases (Wang et al., 2019a; Huang et al., 2022) simply fed semantic information to the model with the assumption it would enhance paraphrase quality. On the other hand, our method actively utilizes semantic structures as control mechanisms, allowing for direct manipulation of the input semantics to influence and shape the generated output. Semantic structures can be very rich and expressive, notably Abstract Meaning Representations (Banarescu et al., 2013) (AMR), consisting of rooted directed acyclic graphs that model semantic concepts.

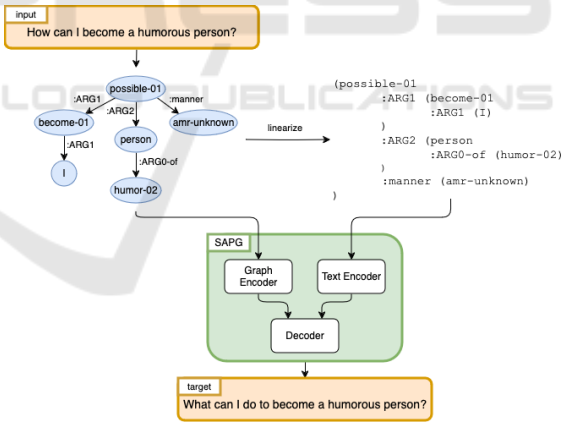

Figure 1: General approach for SAPG. We process pre-extracted AMR graphs by extending a pretrained encoder-decoder model with graph neural networks.

Most of the approaches in the literature use *pre-*

[a] https://orcid.org/0009-0001-1044-3134
[b] https://orcid.org/0000-0003-1252-7515

Sousa, A. and Cardoso, H. L.
SAPG: Semantically-Aware Paraphrase Generation with AMR Graphs.
DOI: 10.5220/0013379700003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 2*, pages 861-871
ISBN: 978-989-758-737-5; ISSN: 2184-433X

trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2020; Radford et al., 2019; Lewis et al., 2020), as these models have demonstrated enhanced performance across various tasks and enable efficient training for downstream applications. However, it is not clear how the above-mentioned graph-like structures can be processed by text-to-text PLM models due to the different structural nature of the expected input data. One solution is to transform the input graph into a text sequence, which can be directly fed into PLMs. Virtually, all recent works on paraphrase generation have converted auxiliary graph structures in sequences by top-down linearization, as it was shown that such linearized graph representation can be used to fine-tune a PLM and improve text generation performance (Kale and Rastogi, 2020). The main drawback of this approach is the weakened graph structural information (i.e., which and how nodes are connected). PLMs must infer how edge connections are specified in the sequence (Song et al., 2018). Some works successfully used Graph Neural Network (GNN) encoders to outperform sequential encoders (Ribeiro et al., 2019, 2021b).

This work presents SAPG, a **S**emantically-**A**ware **P**araphrase **G**eneration model that extends PLMs to leverage AMR graphs generated from the input texts. This is done by having a dual encoder architecture which allows embedding the input graph structure into PLMs using a GNN-based graph encoder (see Figure 1). This additional module preserves the graph's explicit connectivity, which would otherwise be lost if the graphs were linearized, by utilizing GNNs instead. The AMR graphs are transformed into their bipartite form, where relations are promoted to nodes and concepts are subword tokenized. The features of the graph are enriched with the PLM's contextual embeddings of the respective subword tokens. This information is fused in the decoder using extra cross-attention layers added to each decoder block. The added parameters from the graph encoder and cross-attention layers amount to less than 2% of the total number of parameters of the PLM. We highlight the sensitivity of SAPG to changes in the input AMR graphs while persisting the semantics, effectively allowing for the editing of these input graphs to change the generations' surface form and produce more varied paraphrases.

In summary, the contributions of our work are:

- We propose SAPG, a paraphrase generation model that uses AMR graphs for explicit control of generated paraphrases.
- We conduct experiments on two popular datasets, analyze in depth the contribution of each component of the approach and analyze the behavior of the model under a case study where we found it to have interesting sensitivity to graph manipulation that can be exploited to produce variations on the generated paraphrases.

Finally, we note that while we work with AMR graphs and for the English language, the approach is neither AMR nor English-specific. Other types of graphs can be used with this approach, and as long as analogous models exist, this approach can be used with minimal changes for other languages. The code to reproduce the experiments is available[1].

2 RELATED WORK

Virtually all recent works leverage neural methods to tackle the task of paraphrase generation, and the latest approaches typically employ some conditioning to control the generations. In controlled text generation, the control conditions are generally divided into three different types: **(1) Syntactic**: represents control over the structure of the generated text. **(2) Lexical**: represents control over the vocabulary and lexicon usage. **(3) Semantic**: generally refers to controlling the meaning of the text. We also cover the strategies employed in the literature for **Graph-to-Text using AMR**.

Syntactic Control. One approach to control the syntax of generations is using exemplar sentences to guide the syntax of the generated paraphrase (Chen et al., 2019b). An alternative is directly employing constituency trees for syntax guidance (Iyyer et al., 2018). Goyal and Durrett (2020) promote syntactic diversity by conditioning over possible syntactic rearrangements of the input.

Lexical Control. To achieve diversity, some works focused on diverse decoding using heuristics such as Hamming distance or distinct n-grams to preserve diverse options during beam search (Vijayakumar et al., 2018). Others generate multiple outputs by perturbing latent representations (Park et al., 2019) or by using distinct generators (Qian et al., 2019). Zeng et al. (2019) use keywords as lexical guidance for the generation process.

Semantic Control. For semantic control, Bandel et al. (2022) used a scalar value computed from popular semantic similarity measurements and tuned on

[1] https://github.com/afonso-sousa/sapg

inference time. Wang et al. (2019b) used PropBank-style semantic annotations to embed semantic information into a Transformer model. More recently, AMR has been used for paraphrase generation as a means to persist the semantics (Huang et al., 2022).

AMR-to-text Generation. Previous studies have demonstrated that explicitly encoding graph structures can improve performance (Song et al., 2018; Ribeiro et al., 2019; Cai and Lam, 2020; Schmitt et al., 2021; Zhang et al., 2020) but fine-tuning PLMs on linearized structured data has been used to great success in data-to-text generation (Nan et al., 2021; Kale and Rastogi, 2020; Ribeiro et al., 2021a). Our work can be seen as integrating the advantage of both graph structure encoding and PLMs, using a dual encoder architecture. Ribeiro et al. (2020a) explored encoder-decoder PLMs for graph-to-text tasks, showing that PLMs improve significantly with AMR structures rather than knowledge graphs, and task-specific pretraining can lead to notable gains. Hoyle et al. (2021) analyzes how PLMs handle graph linearizations, finding that models trained on canonical formats struggle to generalize to alternatives that preserve meaning. Ribeiro et al. (2021b) enhance PLMs by incorporating graph knowledge through added GNN-based modules in each encoder layer, which is most similar to our method, as we also adapt PLMs to process graph structures.

3 PROPOSED APPROACH

We formulate the task of supervised paraphrase generation as Graph-to-Text generation. Specifically, AMR graphs are extracted from the source texts and fed into the model as both textual inputs, linearizing the graph; and graph inputs, processed by a graph encoder attached to a PLM.

3.1 Architecture

SAPG re-purposes an encoder-decoder PLM using additional graph information. It uses GNNs to encode AMR graphs extracted from the input texts. Figure 2 illustrates the proposed architecture. The center and right part of the Figure represent the encoder and decoder of the PLM, respectively (from here on we may refer to the PLM encoder as the text encoder, contrasting with the graph encoder). On the left side, we have a graph-processing block that starts by extracting an AMR graph from the inputs using an AMR parser[2]. This is a third-party AMR parser that is only inferred on, not trained. This AMR graph is processed by a randomly initialized graph encoder based on GNNs (in Section 3.2 we provide more details on the graph features). The representation generated by the graph encoder is fused in the PLM's decoder. This is done by adding extra randomly initialized multi-head cross-attention layers[3] to each decoder block, k. In each decoder block, the added cross-attention layer attends to both the representation coming from the k^{th} decoder cross-attention layer as well as the representation generated by the graph encoder. The rationale is that the graph encoder computes the refined structural node representation based on the local node context while the new cross-attention layers incorporate this information in the PLM without substantially impacting the knowledge learned during pretraining. To reduce the computational expense, we set the dimensionality of the projection matrices of the added cross-attention layers to 8 for all experiments.

3.2 Model Inputs

Our model processes two types of inputs in its dual encoder architecture: the text encoder takes **a linearized graph** textual representation, and the graph encoder takes **a bipartite graph with subword embedding features**.

Text as a Linearized Graph. Referring again to Figure 2, notice the input embeddings are fed with the output of the AMR parser. To convert the graph into a representation that can be processed by the text encoder, we linearize it. This linearization process is done by traversing the graph using depth-first search and appending the nodes as they are visited – using text makes the model converge into a standard text-to-text model, discarding the added graph information.

Graph as a Set of Features and Edges. To better model the graph connectivity, we use GNNs. These structures allow for information flows between adjacent nodes and the exploration of distinct strategies for neighborhood aggregation to model the structural information of the AMR graph. We first transform the multi-relational graph into a bipartite graph (i.e., a transformed version of a labeled graph where the relations are promoted to new nodes). Formally, we can represent an AMR graph, which is a rooted,

[2]We use a pretrained AMR parser https://github.com/bjascob/amrlib.

[3]As we use T5 as PLM, the cross-attention layers follow the original T5 cross-attention.

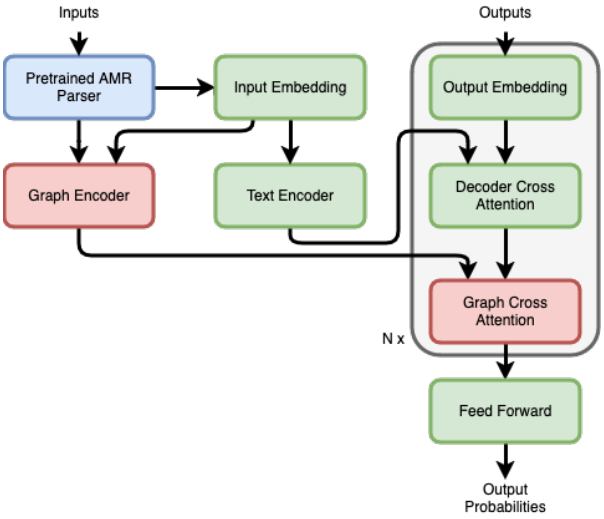

Figure 2: The architecture of our model. Green boxes are parts of a PLM, whereas red boxes are added parts with new weights. The blue box is a frozen part just to infer from. *Inputs* and *outputs* refer to source sentences and target sentences, respectively.

directed graph where nodes represent concepts and edges correspond to relations between concepts, as: $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{R})$ with concept nodes $w \in \mathcal{V}_w$ and labeled edges $(u_w, r, v_w) \in \mathcal{E}_w$, where $r \in \mathcal{R}$ denotes the relation existing between concepts u_w and v_w. Similar to Beck et al. (2018), to build the bipartite graph $\mathcal{G}_b = (\mathcal{V}_b, \mathcal{E}_b)$ from $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, \mathcal{R})$, we replace each labeled edge $(u_w, r, v_w) \in \mathcal{E}_w$ with two unlabeled edges $e_1 = (u_w, r)$ and $e_2 = (r, v_w)$, where relation r is converted into a new node in $\mathcal{V}_b$, and e_1, e_2 in $\mathcal{E}_b$. As node features, we use subword embeddings. Analogous to PLMs, which typically use vocabularies with subword units Sennrich et al. (2016) to represent words, we also represent our concept nodes as subword tokens. Inspired by Ribeiro et al. (2020b, 2021b), we transform each $\mathcal{G}_b$ into a new token graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, where each subword token from a concept in $\mathcal{V}_b$ becomes a node $t \in \mathcal{V}_t$. We convert each edge $e_1 \in \mathcal{E}_b$ into a new set of edges where every resulting subword token from u_w is connected to every subword token from r (the same process applies to e_2).

3.3 Graph Encoder

The architecture of the graph encoder closely follows the form originally proposed for T5 Raffel et al. (2019) feed-forward networks, but instead of dense layers, we use GNNs. Specifically, a GNN layer followed by a nonlinearity and another GNN layer. Dropout Srivastava et al. (2014) is used in between the GNN layers. We do not use layer normalization Ba et al. (2016) or residual skip connections He et al. (2016), as these techniques were originally employed to improve gradient flow in deep networks, while our graph encoder is shallow and only fused later on in the network. We perform a global mean pooling (average node features across the node dimension) and end up with another dropout layer.

We refer generally to the GNN layers and have tried different ones (refer to Section 5.2), but ultimately SAPG uses Graph Convolutional Networks (GCNs) Kipf and Welling (2017).

4 EXPERIMENTAL SETUP

This section reviews the datasets, models, and respective training schemes and metrics employed to evaluate our approach. HuggingFace Wolf et al. (2020) is the main developing tool we use.

4.1 Datasets

We evaluate our model on two datasets:

ParaNMT. ParaNMT-50M (Wieting and Gimpel, 2018) is a dataset collected via back-translation referring to English sentences. We use a subset proposed by Chen et al. (2019b) and strip sequences longer than 20 tokens. The dataset contains roughly 450,000 training pairs of $\langle sentence, paraphrase \rangle$, 500 pairs for validation, and 800 pairs for testing. From here on, we call this subset ParaNMT. It is worth noting that our approach needs AMR graphs, so we only use entries from which we were able to extract a correct

AMR graph using the pretrained AMR parser. However, the number of entries which produced erroneous graphs is negligible, with 1470 for the training set and 1 for the test set – erroneous graphs can have syntactic problems, inconsistent/missing roles, undefined nodes, dangling edges, etc.

QQP. The original Quora Question Pairs (QQP)[4] dataset is built for paraphrase identification with about 400K sentence pairs. The dataset includes about 150,000 positive pairs (pairs of paraphrases) and 250,000 negative pairs (pairs that are not paraphrases). We keep the positive pairs with a maximum token length of 30, leaving us with ∼140,000 pairs, from which 3,000 samples are used for evaluation and 3,000 more for testing. This is the subset used by Kumar et al. (2020). Throughout this document, we call this subset QQP. The number of erroneous AMR graphs for this dataset is 12 for the training set and 1 for the validation set.

4.2 Metrics

To probe the semantic retention of the generations, we measure semantic similarity using SBERT (Reimers and Gurevych, 2019). Bandel et al. (2022) conducted a human-judged study that found this metric to have the lowest coupling between semantic similarity and linguistic diversity[5].

Following recent work (Hosking et al., 2022), we also use the **iBLEU** score (Sun and Zhou, 2012) to get a measure of quality as a trade-off between semantic preservation and lexicon diversity:

$$\begin{aligned} &\text{iBLEU} = \alpha \cdot \text{BLEU} - (1-\alpha) \cdot \text{self-BLEU}, \\ &\text{BLEU} = \text{BLEU-4}(hypothesis, reference), \\ &\text{self-BLEU} = \text{BLEU-4}(hypothesis, input) \end{aligned} \quad (1)$$

This metric measures the fidelity of generated outputs to reference paraphrases (using BLEU (Papineni et al., 2002)) as well as the level of diversity introduced (using self-BLEU, that is, BLEU between hypothesis and input sources). We set α to 0.7, following the original paper (Sun and Zhou, 2012).

The iBLEU score is computed corpus-wise. For SBERT, we used one of the official pre-trained models[6]. Because SBERT computes scores per pair, the reported results are averaged, and we also report the standard deviation of the mean.

4.3 Models

For our experiments, unless explicitly mentioned otherwise, we use as PLM encoder-decoder the large-version[7] of the T5 model Raffel et al. (2019).

4.4 Implementation Details

For training, we followed the setup from Bandel et al. (2022), training all models with a batch size of 32 and a learning rate of 1e-4 for 6 epochs and keeping the model that achieves better BLEU score in the validation set. The optimizer was AdamW Loshchilov and Hutter (2019) with no weight decay and a linear scheduler with 100 steps for warmup. No model hyperparameter tuning was performed. At inference time, for all experiments, we use beam search with a beam size of 4.

5 RESULTS

In this section, we report on the results achieved for our experiments and discuss their impact.

5.1 Benchmark Results

Using semantic knowledge sources as inductive biases for paraphrase generation is not a much-explored work (for details, refer to Section 2). As such, there are few works with publicly available source code to compare. We select two works from the literature to compare, one for the architectural side and another for the control strategy. For architecture, we compare with StructAdapt (Ribeiro et al., 2021b), which also proposes to encode an AMR graph into a PLM using GNNs, however, for the task of AMR parsing. As for the control strategy, we compare it with QCPG (Bandel et al., 2022), which uses semantic, syntactic, and lexical control codes appended as text to the input sequence. For a fairer comparison, we reimplement these works, training and evaluating the models with our training and evaluation setup. Following are some notable differences from the original works:

- For StructAdapt, the main difference is that we fully fine-tune the model instead of training just the adapter weights – also note that this architecture was originally proposed for AMR parsing.

[4] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[5] The authors compared SBERT, BLEURT (Sellam et al., 2020), and BERTScore (Zhang et al., 2019).

[6] https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2

[7] https://huggingface.co/t5-large

Table 1: Comparing top-1 results for automatic evaluation on the test sets of ParaNMT and QQP with works in the literature. The best results are in **bold**.

	QQP		ParaNMT	
	iBLEU↑	SBERT↑	iBLEU↑	SBERT↑
Sources as generations	-9.3	94.41 ± 4.11	-17.0	91.85 ± 5.14
StructAdapt	8.3	93.07 ± 5.76	6.1	89.47 ± 6.8
QCPG	6.2	**94.37** ± **4.62**	6.8	**90.99** ± **6.31**
T5-XL	5.6	94.2 ± 4.65	3.5	90.04 ± 6.26
SAPG	**8.8**	93.49 ± 5.26	6.4	89.24 ± 6.91

- For QCPG, we train a T5 model (instead of a BART) using our training settings and discard the syntactic control code. Because the codes have to be set manually at inference time, we use for all datasets the semantic and lexicon diversity codes of $\langle 50, 30 \rangle$, as suggested in the original paper.

Since our focus is on the supervised setting, we also report a baseline using the inputs as generations, serving as fluency guidance for the model Bandel et al. (2022); Hosking et al. (2022). Additionally, we compare with a Large Language Model (LLM) version of the T5 model, T5 XL[8]. This model is fine-tuned using LoRA Hu et al. (2021), with a rank of 16, a scaling factor of 32, dropout of 5% and the target modules being the query and value projection matrices.

5.1.1 Top-1 Automatic Evaluation

In Table 1, we show the results for the different aforementioned models. One of the main takeaways is that SAPG shows better performance than StructAdapt except for SBERT on ParaNMT, where it got a slight decrease. Nevertheless, this shows that our simple approach for modeling graph connectivity into a PLM can achieve comparable/better results than a complex encoder that requires careful mapping between linearized input and graph connectivity – for example, StructAdapt breaks if the input has nodes without edge connections as it maps the text encoded features as graph features. Another important takeaway is that if conforming to the gold standard is the main focus, our solution does not perform better than PLM-only approaches that take straight-up text as input. This is somewhat expected as our solution does not use plain text as input. Especially for supervised learning paraphrase generation, it is easier for models to map text-to-text, as these are often similar, and PLMs already have a good understanding of semantics. It is worth noting that the LLM – T5 XL – did not perform particularly well for this task, having the produced generations very similar to the input texts.

[8]https://huggingface.co/google/t5-v1_1-xl

Table 2: Ablation study. Top-1 results for automatic evaluation on the test sets of ParaNMT and QQP. *With encoder attention* denotes architectures with cross-attention layers on the encoder, and *with decoder attention* architectures with cross-attention layers on the decoder. The colored differences are reported between the variant and SAPG (green improves on SAPG, red decreases performance when compared to SAPG).

	QQP		ParaNMT	
	iBLEU↑	SBERT↑	iBLEU↑	SBERT↑
SAPG (decoder attention + GCN)	8.8	93.49	6.4	89.24
w/o graph encoder	9.6 (+0.8)	93.42 (-0.07)	6.6 (+0.2)	89.55 (+0.31)
w/ text input	8.2 (-0.6)	94.35 (+0.86)	8.9 (+2.5)	90.9 (+1.66)
GAT-based encoder	8.7 (-0.01)	93.29 (-0.20)	6.5 (+0.1)	89.30 (+0.06)
RGCN-based encoder	9.9 (+0.9)	93.51 (+0.02)	6.1 (-0.3)	88.54 (-0.70)
w/ encoder attention	9.2 (+0.4)	93.49 (0.00)	6.2 (-0.2)	89.22 (-0.02)
w/ encoder attention + GAT	9.2 (+0.4)	93.49 (0.00)	5.6 (-0.8)	88.97 (-0.27)
w/ encoder attention + RGCN	9.6 (+0.8)	93.53 (+0.04)	5.8 (-0.6)	88.82 (-0.42)

5.2 Ablation Study

To investigate the effectiveness of the different components in SAPG, we further conduct experiments to compare our model with the following variants (whose results are shown in Table 2, and some depicted in Figure 3):

1. w/o graph encoder. This variant consists just of the PLM to which we input the linearized graph representation. There are no added parameters in this solution. This variant outperforms our solution in regards to top-1 score (in Section 5.3 we emphasise some of the advantages of our solution over this variant).
2. w/ text input. This variant gets as textual input the raw source text – instead of the linearized graph in SAPG– but maintains the graph encoder. This variant is the most performant and, as previously mentioned, performs the best because of the text input instead of the graph. However, after some empirical testing, it seems to completely discard the graph connectivity fed through the graph encoder and converge into a standard text-to-text fine-tuned PLM. We attribute this behavior to the PLM being fine-tuned with supervised learning, making it easier to minimize cross entropy using only texts. Additionally, the results for ParaNMT suggest the noisiness of the data to be hindering the AMR graph quality and, thus, such a high difference in performance when compared to the linearized graph input.
3. w/ different graph encoders. We test changing GCN for GAT Veličković et al. (2018) and RGCN Schlichtkrull et al. (2018). Note that the latter requires a different graph compilation, where relations are not nodes nor present in the linearized graph. These variants perform similarly to GCN, but are more computationally expensive.

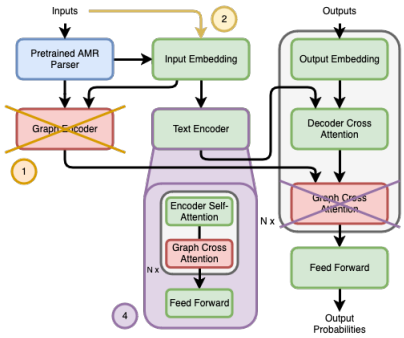

Figure 3: Illustration of some ablations.

4. w/ encoder attention. This variant has added cross-attention layers at each encoder layer instead of at each decoder layer. The performance is similar, suggesting it has a low impact whether we fuse representations earlier or later in the model.

5.3 Case Study

To test in-depth some properties that we deem of most interest in our solution, SAPG, we use the first entry of the QQP test set and assume the findings to hold for the other texts. The source is *"What are the prospect of mechanical engineers in future?"*, and the respective AMR graph is:

```
( prospect-02
  :ARG0 ( person
    :ARG0-of ( engineer-01
      :ARG1 ( mechanics )
    )
  )
  :ARG1 ( amr-unknown )
  :time ( future )
)
```

5.3.1 Behaviour Under Focus Change

Huang et al. (2023) showed that changing the focus of an AMR can lead PLMs fed with linearized AMR graphs as input to produce different surface forms. We perform a similar experiment to assess the sensitivity of SAPG to different linearization strategies. To test this feature, for the entry we picked, we compiled a different AMR graph for each concept in the source text where that concept is the *focus*. Note that this transformation actually might change the connectivity of the graph (e.g., transforming a `:ARG0` relation into a `ARG0-of` in the opposite direction).

Table 3 shows the results of this experiment. SAPG shows much more diversity when the focus is changed when compared to a PLM fine-tuned with linearized AMR graphs as input. We associate this behavior to the actual connectivity change

Table 3: Focus change experiment on case study sentence. *lin. graph* stands for PLM fine-tuned with linearized AMR graphs as input.

	Focus Concept	*What are the prospect of mechanical engineers in future?*
lin. graph	*amr-unknown*	What is the future of mechanical engineering?
	prospect-02	What is the future of mechanical engineering?
	mechanics	What is the future of mechanical engineering?
	engineer-01	What is the future of mechanical engineering?
	person	What is the future of mechanical engineering?
	future	What is the future of mechanical engineering?
SAPG	*amr-unknown*	What are the prospects for mechanical engineering?
	prospect-02	What is the scope of mechanical engineering in the future?
	mechanics	What are the prospects for mechanical engineering in the future?
	engineer-01	What are the prospects for mechanical engineering?
	person	What are the prospects for mechanical engineering?
	future	What is the future of mechanical engineering?

with the aforementioned transformation like `:ARG0` to `ARG0-of` when *mechanics* is the focus or `:time` to `time-of` when *future* is the focus. Our main takeaway is that SAPG is aware of the graph connectivity, so the added graph encoder is useful – it could otherwise be said that while training, the model converged to a solution like the PLM with linearized graph inputs, where the graph encoder representations have minimal contribution towards the final generations.

5.3.2 Using Semantics to Control the Surface Form

Virtually all approaches in the literature (refer to Section 2) that try to promote different surface forms in the generated paraphrases induce a bias in the form of syntactic structures. Here, we want to test how our solution responds to concept changes and how this affects the generated output. We perform (1) a branch addition, (2) a branch editing and (3) a branch removal.

For branch addition, we expanded the concept *mechanics* with the new modifier: *vehicle*. This entails adding the branch `:mod (v / vehicle)` to the AMR graph. SAPG generated "What are the prospects for a mechanical engineer in the field of vehicle engineering?" while the linearized graph model generated "What is the scope of mechanical engineering in vehicle engineering?".

For branch editing, we changed the temporal anchor from *future* to *yesterday* and the concept *mechanics* to *eletronics*. SAPG generated "What is the prospectus of electronic engineering on wednesday?" and the linearized graph model generated "What is the prospectus for an electronic engineer?".

For branch removal, we removed the branch that specifies the person's role (concepts *engineer-01* and *mechanics*). SAPG generated "What is the prospect of people in the future?" and the linearized graph model generated "What is the prospect for the future?".

SAPG generated examples more consistent to the contents of the AMR graph and better semantically preserving than the linearized graph model for the

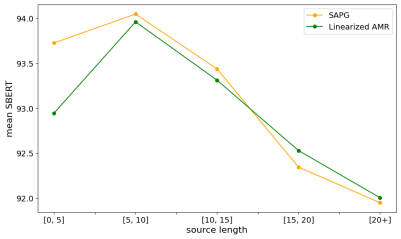

Figure 4: SBERT scores per source length bin on QQP test set.

three transformations. As far as we know, SAPG is the first solution for paraphrase generation that allows for controlling the output surface form rooted in semantic knowledge.

5.4 Performance Based on Sentence Length

We try to understand how is the performance of SAPG bound to the texts length and AMR graphs complexity. To do this, we split the source texts in bins with steps of 5.

Figure 4 shows the mean SBERT scores on the QQP test set for each bin. Although we have shown that the model with just the linearized graph input outperforms SAPG (refer to Section 5.2), the plot seems to show that this might be because of the quality of the AMR. As the complexity of the AMR increases, so does the potential for noisy inputs, which shows in the degradation of the performance for source texts of higher length. Actually, for very small to small-sized texts (between 0 and 15 tokens), SAPG outperforms the linearized graph version on semantic preservation.

5.5 Computational Considerations

As SAPG builds on top of a PLM, it is worth considering the added computational expense of a solution like this. SAPG only adds 13M parameters to the T5 large model, adding less than 2% of its total number of parameters (from 738M to 751M). In the datasets we experimented with, it converged fast, requiring only one or two epochs to train (under the stopping criteria we specified in Section 4.4). SAPG's pipeline starts by extracting AMR graphs. This extraction takes 424 milliseconds per entry on an NVIDIA Quadro RTX 8000 (averaged over 100 iterations). The batch building with graphs takes 17 ms, while a text-only batch build takes <1ms. SAPG takes roughly the same time to infer as a plain PLM, averaging 554ms on the previously mentioned GPU. In conclusion, the computational overhead of our solution is somewhat negligible when compared to the standard PLM backbone.

6 CONCLUSION

In this paper, we presented SAPG, a novel semantically-aware paraphrase generation model that explicitly models the graph structure of AMR graphs into PLMs. Results on two popular paraphrase generation datasets showed that SAPG is competitive when compared to other baselines and state-of-the-art solutions. Most notably, SAPG allows for changing the surface form of generated paraphrases by transforming the input AMR graphs, allowing for edits rooted in semantics, contrary to most current approaches, which rely on syntactic structures. SAPG also shows better adherence to the meaning encapsulated in the AMR graphs, which is arguably not so good for a model relying solely on the linearized graph as input.

Limitations. Throughout this document, we have already acknowledged some of the limitations of SAPG. The main limitation of this work is that it relies on the proper extraction of AMR graphs, which can affect the overall performance if the AMR parsers fail to produce accurate or complete representations. Inaccurate or incomplete AMR graphs may lead to paraphrases that do not fully preserve the original meaning or exhibit semantic inconsistencies. Moreover, the computational cost associated with graph-based methods can be higher compared to traditional sequence-based approaches. Finally, our approach, due to its reliance on the AMR parser, may not be easily employed for specialized domains or languages other than English, for which the appropriate AMR parsers may not be available.

Future Work. The limitations we mentioned above mainly concern the source of additional semantic knowledge. Trying to overcome these limitations, we see a future direction of this work as switching the semantic knowledge structure, in this case, the AMR graphs, to a more efficient one. For example, the *pseudo*-semantic graphs proposed by Sousa and Lopes Cardoso (2024) are built from dependency parsing (DP) trees, which parsers are generally more accurate and efficient when compared with AMR parsers due to DP trees being more simple in structure. Additionally, DP parsers are more readily available in a variety of languages, contrary to AMR, which is almost English-exclusive.

ACKNOWLEDGEMENTS

The first author is supported by a PhD studentship with reference 2022.13409.BD from Fun-

dação para a Ciência e a Tecnologia (FCT). This work was financially supported by: Base Funding - UIDB/00027/2020 and Programmatic Funding - UIDP/00027/2020 of the Artificial Intelligence and Computer Science Laboratory - LIACC - funded by national funds through the FCT/MCTES (PIDDAC).

REFERENCES

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Bandel, E., Aharonov, R., Shmueli-Scheuer, M., Shnayderman, I., Slonim, N., and Ein-Dor, L. (2022). Quality controlled paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 596–609, Dublin, Ireland. Association for Computational Linguistics.

Beck, D., Haffari, G., and Cohn, T. (2018). Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Cai, D. and Lam, W. (2020). Graph transformer for graph-to-sequence learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7464–7471.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. (2019a). Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. (2019b). Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dong, L., Mallinson, J., Reddy, S., and Lapata, M. (2017). Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

Goyal, T. and Durrett, G. (2020). Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hosking, T. and Lapata, M. (2021). Factorising meaning and form for intent-preserving paraphrasing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1418, Online. Association for Computational Linguistics.

Hosking, T., Tang, H., and Lapata, M. (2022). Hierarchical sketch induction for paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2489–2501, Dublin, Ireland. Association for Computational Linguistics.

Hoyle, A. M., Marasović, A., and Smith, N. A. (2021). Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, K.-H., Iyer, V., Hsu, I.-H., Kumar, A., Chang, K.-W., and Galstyan, A. (2023). ParaAMR: A large-scale syntactically diverse paraphrase dataset by AMR back-translation. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8047–8061, Toronto, Canada. Association for Computational Linguistics.

Huang, K.-H., Iyer, V., Kumar, A., Venkatapathy, S., Chang, K.-W., and Galstyan, A. (2022). Unsupervised syntactically controlled paraphrase generation with Abstract Meaning Representations. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1547–1554, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Kale, M. and Rastogi, A. (2020). Text-to-text pre-training

for data-to-text tasks. In Davis, B., Graham, Y., Kelleher, J., and Sripada, Y., editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Krishna, K., Song, Y., Karpinska, M., Wieting, J. F., and Iyyer, M. (2023). Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kumar, A., Ahuja, K., Vadapalli, R., and Talukdar, P. (2020). Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:329–345.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2020). Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Meng, Y., Ao, X., He, Q., Sun, X., Han, Q., Wu, F., fan, C., and Li, J. (2021). Conrpg: Paraphrase generation using contexts as regularizer.

Nan, L., Radev, D., Zhang, R., Rau, A., Sivaprasad, A., Hsieh, C., Tang, X., Vyas, A., Verma, N., Krishna, P., Liu, Y., Irwanto, N., Pan, J., Rahman, F., Zaidi, A., Mutuma, M., Tarabar, Y., Gupta, A., Yu, T., Tan, Y. C., Lin, X. V., Xiong, C., Socher, R., and Rajani, N. F. (2021). DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (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, USA. Association for Computational Linguistics.

Park, S., Hwang, S.-w., Chen, F., Choo, J., Ha, J.-W., Kim, S., and Yim, J. (2019). Paraphrase diversification using counterfactual debiasing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6883–6891.

Qian, L., Qiu, L., Zhang, W., Jiang, X., and Yu, Y. (2019). Exploring diverse expressions for paraphrase generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3173–3182, Hong Kong, China. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. In *Technical report, OpenAI*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ribeiro, L. F. R., Gardent, C., and Gurevych, I. (2019). Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Ribeiro, L. F. R., Pfeiffer, J., Zhang, Y., and Gurevych, I. (2021a). Smelting gold and silver for improved multilingual amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

Ribeiro, L. F. R., Schmitt, M., Schütze, H., and Gurevych, I. (2020a). Investigating pretrained language models for graph-to-text generation. *arXiv e-prints*.

Ribeiro, L. F. R., Zhang, Y., Gardent, C., and Gurevych, I. (2020b). Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Ribeiro, L. F. R., Zhang, Y., and Gurevych, I. (2021b). Structural adapters in pretrained language models for AMR-to-Text generation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Schlichtkrull, M. S., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607.

Schmitt, M., Ribeiro, L. F. R., Dufter, P., Gurevych, I., and Schütze, H. (2021). Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based*

Methods for Natural Language Processing (TextGraphs-15), pages 10–21, Mexico City, Mexico. Association for Computational Linguistics.

Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shah, D., Lei, T., Moschitti, A., Romeo, S., and Nakov, P. (2018). Adversarial domain adaptation for duplicate question detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1056–1063, Brussels, Belgium. Association for Computational Linguistics.

Shu, R., Nakayama, H., and Cho, K. (2019). Generating diverse translations with sentence codes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1823–1827, Florence, Italy. Association for Computational Linguistics.

Song, L., Zhang, Y., Wang, Z., and Gildea, D. (2018). A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Sousa, A. and Lopes Cardoso, H. (2024). Pseudo-semantic graphs for generating paraphrases. In *Progress in Artificial Intelligence: 23rd EPIA Conference on Artificial Intelligence, EPIA 2024, Viana Do Castelo, Portugal, September 3–6, 2024, Proceedings, Part III*, page 215–227, Berlin, Heidelberg. Springer-Verlag.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sun, H. and Zhou, M. (2012). Joint learning of a dual SMT system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42, Jeju Island, Korea. Association for Computational Linguistics.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2018). Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wang, S., Gupta, R., Chang, N., and Baldridge, J. (2019a). A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

Wang, S., Gupta, R., Chang, N., and Baldridge, J. (2019b). A task in a suit and a tie: Paraphrase generation with semantic augmentation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Wieting, J. and Gimpel, K. (2018). ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zeng, D., Zhang, H., Xiang, L., Wang, J., and Ji, G. (2019). User-oriented paraphrase generation with keywords controlled network. *IEEE Access*, 7:80542–80551.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhang, Y., Guo, Z., Teng, Z., Lu, W., Cohen, S. B., Liu, Z., and Bing, L. (2020). Lightweight, dynamic graph convolutional networks for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2162–2172, Online. Association for Computational Linguistics.