T-STAR: <u>T</u>ruthful <u>S</u>tyle <u>T</u>ransfer using <u>A</u>MR Graph as Intermediate <u>R</u>epresentation

Anubhav Jangra[*]   **Preksha Nema**[*]   **Aravindan Raghuveer**
Google Research, India
{anubhavjangra,preksh,araghuveer}@google.com

Abstract

Unavailability of parallel corpora for training text style transfer (TST) models is a very challenging yet common scenario. Also, TST models implicitly need to preserve the content while transforming a source sentence into the target style. To tackle these problems, an intermediate representation is often constructed that is devoid of style while still preserving the meaning of the source sentence. In this work, we study the usefulness of Abstract Meaning Representation (AMR) graph as the intermediate style agnostic representation. We posit that semantic notations like AMR are a natural choice for an intermediate representation. Hence, we propose **T-STAR**: a model comprising of two components, text-to-AMR encoder and a AMR-to-text decoder. We propose several modeling improvements to enhance the style agnosticity of the generated AMR. To the best of our knowledge, T-STAR is the first work that uses AMR as an intermediate representation for TST. With thorough experimental evaluation we show T-STAR significantly outperforms state of the art techniques by achieving on an average 15.2% higher content preservation with negligible loss ($\sim$3%) in style accuracy. Through detailed human evaluation with $90,000$ ratings, we also show that T-STAR has upto 50% lesser hallucinations compared to state of the art TST models.

1   Introduction

A well accepted definition of *style* refers to the manner (via linguistic elements like word choices, syntactic structures, metaphors) in which semantics of a sentence are expressed (McDonald and Pustejovsky, 1985; Jin et al., 2020). Text Style Transfer (TST) is the task of rephrasing an input sentence to contain specific stylistic properties without altering the meaning of the sentence (Prabhumoye et al., 2018). We refer the reader to Jin et al. (2020) for a

---

[*]denotes equal contribution

detailed survey of approaches towards TST problem formulation, metrics and models. In the practical scenario, that we consider in this paper, where a large corpus of parallel data is not available (Niu and Bansal, 2018; Ma et al., 2020; Wu et al., 2020), two family of approaches have been proposed in the literature (Jin et al., 2020). **1. Disentanglement:** Content and Style are disentangled in a latent space and only the style information is varied to transform the sentence. **2. Prototype Editing:** Style bearing words in the source sentence are replaced with those corresponding to the target style. The sentence may be further re-arranged for fluency and naturalness. Both the above approaches have drawbacks in the way the *style agnostic* intermediate representation is constructed, described as follows. First, in the disentangling approaches, it is not easy to verify the efficacy of separation between style and content. Recent approaches (Subramanian et al., 2018; Samanta et al., 2021) even state that as content and style are so subtly entangled in text, it is difficult to disentangle both of them in a latent space. Consequently, this affects the model's interpretability in that it is hard to attribute an effect we see in the output to the latent intermediate vector or the style vector. Second, with prototype editing approaches for lingustic styles (such as author, formality), where the content and style are tightly coupled it is not feasible to segregate style and content carrying words (word examples: *cometh, thou*). Furthermore, style-marker detection is a non-trivial NLP task and needs to be addressed for every new style that is added to the system causing scalabilty concerns (Jin et al., 2020). In this paper, we propose **T-STAR** (**T**ruthful **S**tyle **T**ransfer using **A**MR Graph as Intermediate **R**epresentation) that uses a symbolic semantic graph notation called Abstract Meaning Representation (AMR) as the style agnostic intermediate stage. AMR (Banarescu et al., 2013) is designed to capture semantics of a given sentence in all entirety while abstracting away the

Syntactic Variations for a given sentence

- Over 20 models were produced in the past two years by them.
- They have been producing over 20 models in the past two years.
- They produced over 20 models in the last two years.
- Over 20 models have been produced in the past two years by them.
- They have produced over 20 models in the past two years.

AMR Graph

produce-01
ARG0 ARG1 time
they
model
quant
over
op1
20
before
op1 duration
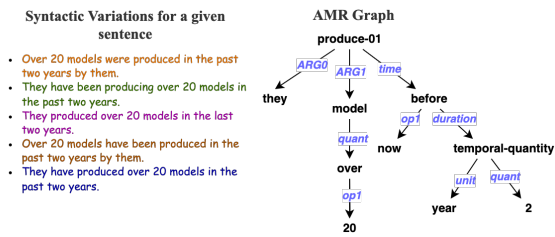now temporal-quantity
unit quant
year 2

Figure 1: Different syntactic variations, leads to the same AMR as they all are similar in meaning.

syntactic variations, inflections and function words. In other words, two sentences with the same meaning but written in very different styles will have a very similar AMR if not exactly the same (See Figure 1).

This addresses the shortcomings with *Disentanglement* and *Prototype Editing* approaches. First, AMR being a representation with well-defined semantics, we can inspect, interpret and measure quality of the intermediate representation and the provenance of knowledge transfer between source and target sentence. Second, AMR being a well known standard has high quality, robust reference implementations, especially for head languages (*e.g.*, English). Our contributions are as follows:

1. We propose a novel TST approach with AMRs, an interpretable, symbolic intermediate representation, to achieve better content preservation. To this end, we enhance AMR parsing techniques to better suit the TST task. To the best of our knowledge, we are the first work to use AMR representations for the style transfer task (Sections 3, 4).

2. Through novel experimentation, we show that an AMR, as a style agnostic intermediate representation, has better content preservation and less style information of the given source sentence compared to competitive baselines.(Sections 5, 6)

3. On multiple datasets we show T-STAR is able to beat competitive baselines by producing sentences with significantly higher content preservation with similar style transfer scores. (Section 7)

4. With thorough human evaluations spanning 90,000 ratings we show T-STAR has $\sim 70\%$ better content preservation compared to state of art baseline with 50% lesser hallucinations. (Section 8)

2 Related Work

2.1 Unsupervised Text Style Transfer

TST systems broadly use two family of approaches: disentanglement (Shen et al., 2017) and prototype-

editing (Jin et al., 2020). Prior works (Hu et al., 2017; John et al., 2019; Fu et al., 2017; Singh and Palod, 2018; Logeswaran et al., 2018) disentangle the content and style information in latent space using style-based classifier or adversarial learning. Prototype-editing based approaches (Li et al., 2018; Madaan et al., 2020; Sudhakar et al., 2019) are used to gain better controllability and interpretability.

Recently, some works propose to jointly optimize for content and style information, to overcome the limitations of explicitly disentangling the style and content information. Yamshchikov et al. (2019b) illustrates that architectures with higher quality of information decomposition perform better on style transfer tasks. Subramanian et al. (2018) argues that it is often easy to fool style discriminators without explicitly disentangling content and style information, which may lead to low content preservation (Xu et al., 2018). Instead, Subramanian et al. (2018); Logeswaran et al. (2018) use back-translation to optimize for content preservation. Luo et al. (2019); Liu et al. (2020) use reinforcement learning framework with explicit rewards designed for content preservation. Wang et al. (2019) pushes the entangled latent representation in the targeted-style direction using style discriminators. Samanta et al. (2019) uses normalizing flow to infuse the content and style information back before passing it to the decoder.

Cheng et al. (2020) proposes a context aware text style transfer framework using two separate encoders for input sentence and the context. Similar to our work, Krishna et al. (2020) also generates an interpretable intermediate representation. The authors first paraphrase the given source to first convert it to a destylized version before passing it to the targeted style-specific decoder. Complementary to our work, (Shi et al., 2021) uses syntactic graphs (dependency tree) as an intermediate representation for attribute transfer to retain the linguistic structure. On the other hand, we focus on retaining the semantics using AMR graphs as intermediate representation and modifying the linguistic structure (authorship style).

2.2 Text ⟺ AMR

In order to improve the parsing performance for AMRs, neural models are receiving increasing attention. Neural AMR parsers can be divided into following categories: i) sequence-to-sequence based AMR-parsers (Xu et al., 2020a), ii) sequence-to-
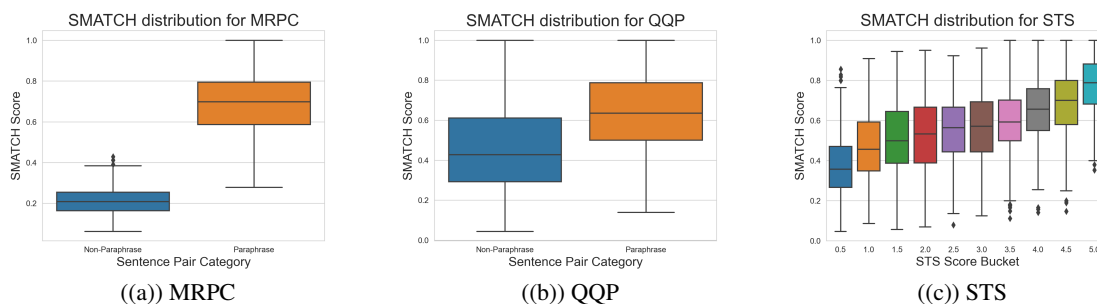
Figure 2: AMR comparison for MRPC, QQP, and STS datasets

graph based AMR parsers (Zhang et al., 2019), where the graph is incrementally built by spanning one node at a time. A more detailed survey on related works can be found in Appendix A.

3 AMR as an Intermediate Representation

Abstract Meaning Representations (AMRs) is a semantic formalism construct that abstracts away the syntactic information and only preserves the semantic meaning, in a rooted, directed and acyclic graph. In Figure 1, we present certain syntactic variations (changing the voice, and tense) for a sentence without altering meaning. All variations result in the same AMR graph. The nodes in the AMR graphs (*"produce-01", "they", "before", etc.*) are concepts (entities/predicates) that are canonicalized and mapped to semantic role annotations present in Propbank framesets[1]. The edges (*"ARG0", "time", "duration", etc.*) are then relations between these concepts. In other words, AMRs aim at decoupling "*what to say*" from "*how to say*" in an interpretrable way. We posit that this could be beneficial for text style transfer, where the goal is to alter the "*how to say*" aspect while preserving "*what to say*".

Recently, semantic meaning representation property of AMRs has been shown to be useful in other generation tasks. In abstractive summarization, Liao et al. (2018) uses AMR as an intermediate representation to first obtain a summary AMR graph from document and then generate a summary from it. Hardy and Vlachos (2018) use AMRs to cover for the lack of explicit semantic modeling in sequence-to-sequence models. In machine translation, Song et al. (2019) adopted AMRs to enforce meaning preservation while translating from English to German. For Paraphrase Generation, Cai

et al. (2021) found that using AMRs as intermediate representation reduces the semantic drift. Moreover, incorporating symbolic representations as an intermediate representation provides a way to effectively understand the reasons behind a model's shortcomings. We utilize this advantage to analyse the weaknesses of T-STAR in Section 8.2.

In order to demonstrate the semantic meaning preservation property of AMRs, we design an experiment using three publicly available paraphrase benchmarks, *i.e.*, MRPC (Dolan and Brockett, 2005), QQP[2], and STS. MRPC and QQP are sentence pair datasets with each pair labeled *yes* if they are paraphrases of each other *no* otherwise. STS dataset assigns a scores from 0 (not similar) to 5 (exactly similar) to a sentence pair. We hypothesize that if AMRs are indeed semantic meaning preserving, two sentences with similar meaning should have highly similar AMRs. To measure the similarity between two AMRs, we use the SMATCH score (Cai and Knight, 2013) that calculates the number of overlapping triplets between two AMRs. We use an off-the-shelf AMR parser [3] to generate AMR given a sentence. We plot the distribution of the SMATCH scores for MRPC, QQP and STS datasets in Figure 2. For MRPC, we infer that the SMATCH scores for paraphrases are significantly higher than the SMATCH scores for non-paraphrases. Similarly, for QQP, quartile distribution of SMATCH scores for paraphrases is higher in comparison to non-paraphrases. For STS dataset, we observe a gradual increase in quartile distribution of SMATCH scores as we move towards more similar sentences.

The experiments above corroborate the claim that AMRs can preserve meaning under lexical variations like paraphrasing, tense and voice changes.

---

[1] https://propbank.github.io/

[2] https://www.quora.com/profile/Ricky-Riche-2/First-Quora-Dataset-Release-Question-Pairs

[3] https://amrlib.readthedocs.io/en/latest/

Vanilla T-STAR Encoder	T-STAR Encoder
Sentence: To make us feel existence, and to shew	

```
(a/and                            (h/have-purpose-91
 :op1(m/make-02                     :ARG2(a/and
  :ARG1(f/feel-01                    :op1(m/make-02
   :ARG0(w/we)                        :ARG1(f/feel-01
   :ARG1(e/exist-01                    :ARG0(w/we)
    :ARG1 w)))                         :ARG1(e/exist-01
 :op2 (s / shew-01                      :ARG1 w)))
  :ARG0 w))                         :op2(s/show-01
                                      :ARG0 w
                                      :ARG1 e)))
```

Sentence: But trust not this; too easy Youth, beware!

```
(m/multi-sentence                 (c/contrast-01
 :snt1(c/contrast-01               :ARG2 (a/and
  :ARG2(t/trust-01                  :op1 (t/trust-02
   :polarity -                       :polarity -
   :mode imperative                  :mode imperative
   :ARG0(y/you)                      :ARG0 (y/you
   :ARG1(t2/this)))                   :mod (y2/youth))
 :snt2(b/beware-01                   :ARG1 (t2/this))
  :mode imperative                  :op2 (h/have-degree-91
  :ARG0(y2/youth                     :ARG1 t2
   :ARG1-of(e/easy-05                :ARG2 (e/easy-05
    :ARG2-of(h/have-degree-91         :ARG1 t2)
     :ARG1 y2                         :ARG3 (t3/too))))
     :ARG3 (t3/too))))))
```

Table 1: Comparison between AMRs from vanilla T-STAR Encoder and T-STAR Encoder. T-STAR Encoder generates better AMRs for stylized sentences.

Recent research discussed earlier, have successfully used this property to show task improvements. Building on the above qualitative, quantitative and prior research evidence, we further explore the applicability of AMR for the TST task.

Figure 3: An overview of T-STAR model architecture. It consists of two modules: T-STAR Encoder, that transforms a given sentence s_i in style i to its AMR representation A_i. To convert the sentence to style j, A_i is passed to T-STAR Decoder specific to style j.

4 Proposed Solution

Our proposed model T-STAR consists of two modules (refer Figure 3). First, T-STAR Encoder generates an AMR given source sentence in style i. Second, T-STAR Decoder generates a sentence in style j with similar meaning as preserved in the generated intermediate AMR. We take T5-Base (Raffel et al., 2020) pre-trained model as our basic seq2seq architecture for both the modules. In order to use AMR as a sequence in T5, we borrow the choice of DFS Traversal from (Bevilacqua et al., 2021), that thoroughly study the effect of various traversals on AMR parsing.

4.1 T-STAR Encoder

We train our simplest encoder, called the vanilla T-STAR Encoder, by fine-tuning T5 base with the
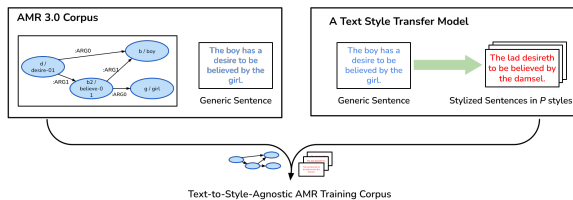

Figure 4: Generic sentences from AMR 3.0 corpus are stylized using a TST model. The corresponding AMR and stylized sentences are mapped together as silver training dataset to finetune T-STAR Encoder.

open source AMR 3.0 dataset (Knight et al., 2021). The AMR 3.0 dataset consists of roughly 59,000 generic English sentences (denoted as s_i for the ith sentence) and their corresponding AMRs (denoted as A_i). In a qualitative analysis, we observe that the vanilla T-STAR encoder under performs in two significant ways as illustrated in Table 1. **First,** style bearing words (such as *"shew"*) become concepts in the AMRs (Sentence-1 in Table 1) as opposed to being canonicalized to their respective propbank role annotations. **Second,** the meaning of the stylized sentences get incorrectly encoded in the AMRs, as shown in the second example in Table 1. To overcome this, we propose a style-agnostic fine-tuning strategy as follows.

Style-agnostic Fine Tuning: We hypothesize that vanilla Text to AMR encoder is unable to effectively transform stylized sentences to AMRs because it has only been trained on generic English sentences. Therefore, we propose a data-augmentation strategy (refer to Figure 4), where we use an off the shelf style transfer model, *e.g.,* STRAP (Krishna et al., 2020), to stylize a generic English sentence s_i in style p ($\hat{s}_i^p$). While converting s_i to $\hat{s}_i^p$, we alter the style of original sentence, keeping the meaning intact. For a high quality synthetic dataset, we filter out samples with low semantic similarity between s_i and $\hat{s}_i^p$. We provide a detailed empirical analysis on this filtering strategy in Appendix C. Since the meaning is preserved, we can now map $\hat{s}_i^p$ to the same AMR A_i. We then fine-tune our T-STAR Encoder on $\bar{\mathbf{S}} = \mathbf{S} \cup \hat{\mathbf{S}}$, where $\hat{\mathbf{S}} = \{\hat{s}_i^p, A_i\}_N \forall p \in P$ and P is the total number of styles in the dataset.

4.2 T-STAR Decoders

Due to the unavailability of parallel style corpora, we are provided P mono-style corpora $\mathbf{R}^p = \{r_i^p\}_{Mp}$ written in style p, where P refers to the total number of styles and r_i^p refers to the ith sen-

Dimension	Metric	Description	Metric used in related works
AMR similarity	SMATCH	SMATCH (Cai and Knight, 2013) measures degree of overlap between AMR graphs of *S,T*. The score is computed based on triplet (edge) overlap by finding a variable node mapping that maximizes the count of matching triplets.	Used extensively to measure similarity between two AMRs, across AMR-parsing literature.
Lexical Diveristy	Self-BLEU ↓	BLEU-4 (Papineni et al., 2002) between *S,T*	(Hu et al., 2017; Li et al., 2018; Logeswaran et al., 2018; Xu et al., 2018)
Content Preservation (C.P.)	WMD ↓	Word Mover Distance (Kusner et al., 2015) measures dissimilarity between *S,T* as the minimum distance between their embedded words.	Yamshchikov et al. (2020) states that WMD correlates best with human-evaluations on semantic similarity
	SIM ↑	SIM (Wieting et al., 2019a) uses an embedding model proposed in (Wieting et al., 2019b) to measure semantic similarity	(Krishna et al., 2020; Luo et al., 2019)
Style Transfer (S.T.)	Style Accuracy ↑	Score of 4-way and 2-way fine-tuned RoBERTa-Large (Liu et al., 2019) model for styles in CDS and Author-Imitation datasets respectively	(Krishna et al., 2020)*,(Hu et al., 2017; Fu et al., 2017; Madaan et al., 2020)*
C.P. & S.T.	**Weighted Style Accuracy ↑**	Style Accuracy weighed by their corresponding semantic similarity scores averaged across all test instances.	(Krishna et al., 2020)*,(Li et al., 2018)*

Table 2: **Evaluation Metrics:** Source and Target sentences denoted as *S,T* respectively. We use all the dimensions except AMR similarity to measure the performance of TST. Note that Weighted Style Accuracy is the only metric that encompasses two crucial dimension : C.P. and S.T. in one metric. AMR similarity is used to select best performing TSTAR-Encoder. * represents the works that use slight variations of the mentioned metrics. Note that across all metrics we compute an average scores across all test instances.

Algorithm 1: Iterative T-STAR

Input: parallel corpora (**S**), mono-lingual corpus ($\mathbf{R}^p, \forall p \in P$)

1   $\hat{\mathbf{S}} = \{\hat{s}_i^p, A_i\}_N \forall p \in P$ using an existing **TST** model, *e.g.,* STRAP

2   $X := \mathbf{S} \cup \hat{\mathbf{S}}$

3   **while** *Convergence* **do**

4   **fine-tune** TSTAR-*Encoder* $f_{amr}(.)$ on X

5      $\hat{\mathbf{S}} := \{\}$

6   **for** $p \in P$ **do**

7   *Use $f_{amr}(.)$ to create* $\hat{\mathbf{R}}^p := \{r_i^p, \tilde{A}_i^p\}_M$

8   **fine-tune** TSTAR-*Decoder* $f_p(.)$ for style p

9   Use TSTAR-*Decoder* to get $\hat{\mathbf{S}}^P = \{\hat{s}_i^p, A_i\}_N$

10         $\hat{\mathbf{S}} = \hat{\mathbf{S}} \cup \hat{\mathbf{S}}^P$

11      $X := \mathbf{S} \cup \hat{\mathbf{S}}$

---

tence of the p style dataset which has M_p sentences. Training a style specific decoder ($f_p(.)$) to generate a sentence in style p from an AMR consists of two steps. **First,** we use our fine-tuned T-STAR Encoder (Section 4.1) to generate AMRs $\hat{A}_i^p$ for every sentence $\mathbf{r}_i^p$ for every style corpora. **Second**, we fine tune a T5 base model to recover the original sentence $\mathbf{r}_i^p$ given the AMR $\hat{A}_i^p$, obtaining style-specific decoders. In other words, we fine tune using M_p pairs of ($\mathbf{r}_i^p$, $\hat{A}_i^p$) constructed from the first step. Note that, we experimented with a data augmentation technique for the decoders as well, however it did not lead to an improvement in the style transfer performance (refer to Appendix E).

Once style specific decoders have been trained for every style in P, we can use the T-STAR Encoder in tandem with the T-STAR Decoders to convert between arbitrary style combinations as in Krishna et al. (2020).

4.3 Iterative T-STAR

The performance of our modules, T-STAR Encoder and T-STAR Decoders depends on the quality of the synthetic datasets ($\hat{\mathbf{S}}$, $\hat{\mathbf{R}}^P$) generated by their complementary modules. We adopt the iterative back-translation technique used in unsupervised machine translation (Hoang et al., 2018). Iteration proceeds in rounds of training an encoder and decoder from mono-style data. In every round, we aim to iteratively improve the quality of the encoder and decoder modules by generating increasingly better synthetic data from the previous round. We briefly describe this process in Algorithm 1.

5 Experimental Setup

In this section we briefly describe the various T-STAR variations that are analyzed in the subsequent sections, baselines, and the implementation details. The models are validated against the metrics summarized in Table 2.

5.1 T-STAR variations

Vanilla T-STAR: The T-STAR Encoder used in this version, is only trained on AMR 3.0 dataset, and not finetuned for stylized sentences.

T-STAR: We train the encoder and decoders using Algorithm 1 for only one iteration.

Iterative T-STAR: We follow two iterations of Algorithm 1 to obtain better quality synthetic dataset.

5.2 Baselines

UNMT: (Subramanian et al., 2018) models style transfer as unsupervised machine translation task. **DLSM** (He et al., 2020) is a deep generative model that unifies back-translation and adversarial loss.

RLPrompt (Deng et al., 2022a) uses a discrete prompt optimization approach using reinforcement learning. It is adopted in a zero-shot setting where we use Distil-BERT (Sanh et al., 2019) and run the optimization for 1k steps.

STRAP (Krishna et al., 2020) first normalizes the style information by paraphrasing the source text into generic English sentence, which is then passed through a style-specific GPT-2 based model to generate the styled output.

5.3 Datasets

We evaluate performance of T-STAR on two English datasets that capture Linguistic Styles (Jin et al., 2020). First, Shakespeare Author Imitation Dataset (Xu et al., 2012) consists of 18K pairs of sentences written in two styles. Original Shakespeare's plays have been written in Early Modern English, a significantly different style. Second, Corpus of Diverse Styles (CDS) (Krishna et al., 2020). This dataset consists of non-parallel sentences written in 11 different styles. We will present our results on a subset of four styles : Bible, Poetry, Shakespeare and Switchboard which consists of 34.8K, 29.8K, 27.5K, 148.8K instances respectively (CDS uses MIT License).

5.4 Implementation Details

We use pre-trained T5-Base model architecture for both the encoder and decoder. Following iterative back translation literature (Kumari et al., 2021), we run Iterative T-STAR for two iterations. The AMRs are preprocessed and postprocessed based on the method mentioned in Appendix B. The best modules are selected based on the performance on validation set. Finer details about the model architecture and hyperparameters can be found in Appendix B.

6 Robustness of AMRs for Text Style Transfer

An ideal style agnostic intermediate representation for TST should a) encode complete semantic information and b) minimal style information. We quantitatively measure AMR's efficacy in these two dimensions. We compare T-STAR with STRAP (Krishna et al., 2020) for it uses a human readable intermediate representation as well.

6.1 Semantic Containment of AMR

If semantics of input sentence is completely preserved in the intermediate representation, we

Style	Model	Content Preservation		S. R. ↑
		WMD ↓	*SIM* ↑	
Bible	STRAP	0.200	0.715	0.979
	T-STAR	**0.170**	**0.792**	0.979
Poetry	STRAP	0.290	0.664	0.969
	T-STAR	**0.215**	**0.760**	0.965
Shakespeare	STRAP	0.328	0.610	0.971
	T-STAR	**0.222**	**0.754**	0.972
Switchboard	STRAP	0.222	0.751	0.999
	T-STAR	**0.163**	**0.848**	1.0

Table 3: Reconstructing the original sentences using the intermediate semantic representation outperforms the baseline with a significant margin with respect to content preservation. **T-STAR** is on par with style retention (S.R.) with the baseline model.

Style	Original	Paraphrase	AMR	
			Vanilla-T-STAR	T-STAR
Bible	97.72	79.97	74.64	**71.90**
Switchboard	99.47	89.79	87.74	**64.66**
Poetry	83.17	81.04	**49.05**	61.61
Shakespeare	**70.07**	78.07	88.8	80.73
Average	87.61	82.22	75.06	**69.73**

Table 4: Accuracy of style classifiers trained on original, paraphrase and AMR inputs. AMR has least performance denoting it encodes least style information.

should be able to reconstruct the input sentence from it. To evaluate the robustness of AMRs across all styles for content preservation, we first generate intermediate AMR given the sentence in style p using our encoder and then reconstruct the same sentence using our decoder $f_p(.)$ in style p. We study how close the generated sentence (from STRAP, T-STAR) is with respect to the original sentence. We can infer from Table 3 that AMRs as an intermediate representation performs significantly better on content preservation as compared to STRAP across all the styles. Specifically, it gives an average of 0.10 and 0.06 absolute improvement on SIM and WMD scores across the four styles respectively. Also, we get comparable performance on retaining the original style. We present an ablation study on AMR parser in Appendix C, and propose a new unsupervised Text-AMR evaluation metric to measure the content preservation of AMR parsing along with its results on the CDS dataset in Appendix D.

6.2 Style Agnosticity of AMR

A style classifier C assigns a style class label to an input sentence S. If S does not encode style information, it should result in poor classifier performance. We use this observation to design an ex-

periment to evaluate the style-agnosticity of AMRs as follows. We train three versions of 4-way style classifier using original sentences, paraphrased sentences (as used in STRAP) and AMRs as the input sequences to the classifier. Two models are used to generate AMRs: Vanilla T-STAR Encoder and T-STAR Encoder. For all versions of the style classifier, we mask content bearing words like entities, numbers, common nouns and AMR tags in the input sequences.

The accuracy of the 4 classifiers is shown in Table 4. First, even Vanilla T-STAR Encoder has lower classifier accuracy for three out of four styles in comparison to original and paraphrases sentences. Second, with the T-STAR Encoder, we observe further reduction in classifier's performance, and obtain an average absolute drop of 15.19% and 7.1% as compared to paraphrase and Vanilla T-STAR Encoder respectively.

We illustrate some examples in Table 1, where the T-STAR Encoder generates better AMRs as compared to the Vanilla T-STAR Encoder. In the first example, the T-STAR Encoder, unlike the vanilla T-STAR Encoder, is able to map style-specific word "*shew*" to the valid concept "*show*" and also associate "*existence*" to it. In the second example, the T-STAR Encoder does not split the AMR into two sentences while parsing. Additionally, it is able to make the association between "*you*" and "*youth*". Through the above quantitative and qualitative analysis we demonstrate that the T-STAR Encoder generates AMRs which are robust in preserving meaning of the source sentence while predominantly losing style information.

7 Performance on Style Transfer Tasks

In this section, we compare our model performance T-STAR and Iterative-T-STAR to the baselines across two datasets: Shakespeare Imitation Dataset and CDS dataset.

7.1 Performance Analysis on Shakespeare Imitation Dataset

In Table 5, we present the results of T-STAR and Iterative-T-STAR in comparison with the baselines for the Shakespeare Author Imitation dataset for both the directions :original to modern style and vice versa. The Weighted Style Accuracy is the primary metric as it is shown to effectively combine both style accuracy and content preservation (Krishna et al., 2020).

Model	Content Preserv.		Lex. Div.	S.T.	WSAcc.
	SIM ↑	*WMD* ↓	*Self-BLEU* ↓		
Original to Modern Style					
UNMT	0.461	0.318	0.118	0.586	0.256
DLSM	0.447	0.369	0.079	0.192	0.115
RLPrompt	0.508	0.387	0.164	0.354	0.292
STRAP	0.647	0.337	0.118	0.886	0.552
Vanilla-TSTAR	0.848	0.182	0.269	0.601	0.497
TSTAR	0.754	0.257	0.175	0.754	0.554
Iterative-TSTAR	0.799	0.227	0.209	0.715	**0.556**
Modern to Original Style					
UNMT	0.373	0.375	0.057	0.414	0.158
DLSM	0.421	0.373	0.086	0.391	0.174
RLPrompt	0.550	0.348	0.261	0.547	0.203
STRAP	0.656	0.332	0.139	0.681	0.433
Vanilla TSTAR	0.897	0.14	0.379	0.47	0.418
TSTAR	0.842	0.185	0.324	0.490	0.402
Iterative-TSTAR	0.853	0.181	0.329	0.540	**0.446**

Table 5: T-STAR models comparison with baseline models on Shakespeare Author Imitation Dataset for both the directions. Iterative T-STAR outperforms all the baselines on Weighted Style Accuracy.

We observe that our model, T-STAR slightly performs better than STRAP model for original to modern style, but has lower performance for modern to original style. However Iterative T-STAR, outperforms all the baselines for both the directions on Weighted Style Accuracy. We observe that STRAP has very high style accuracy that it achieves by compromising on Content Preservation. Through human evaluation (Section 8) we see STRAP employs significantly higher hallucinations to achieve style transfer. Vanilla T-STAR on the other hand achieves high content preservation via significant copying from source as seen by the substantially high self-BLEU score. Iterative T-STAR finds the middle ground of achieving style transfer while not compromising on content preservation. We also found that the length of generated sentence is similar to the input sentence, with on average one word difference. In the subsequent sections, we compare T-STAR to only the best performing baseline, STRAP.

7.2 Performance Analysis on CDS Dataset

In Table 6, we compare the performance of T-STAR and Iterative T-STAR against STRAP and Vanilla T-STAR, across all 12 directions for {*Poetry, Shakespeare, Switchboard, Bible*} styles. We make the following observations: **First**, both our models T-STAR and Iterative T-STAR outperform the state-of-the-art baseline, STRAP, on 11 out of 12 directions, with an average absolute improvement of 7.7% and 9.7% respectively. **Second**, Vanilla T-STAR is observed to be a stronger baseline than STRAP, as it beats STRAP on 8 out of 12 direc-

Direction	Model	L.D. S-BLEU ↓	Cont. Preserv. WMD ↓	SIM ↑	S.R. ↓	S.T. ↑	WSAcc ↑
poetry → bible	STRAP	0.067	0.314	0.571	0.335	0.548	0.289
	Van-TSTAR	0.170	0.183	0.812	0.430	0.289	0.226
	TSTAR	0.084	0.268	0.670	0.202	0.618	**0.391**
	Itr-TSTAR	0.106	0.241	0.721	0.231	0.566	0.383
shak. → bible	STRAP	0.073	0.343	0.535	0.291	0.677	0.346
	Van-TSTAR	0.212	0.194	0.817	0.540	0.422	0.337
	TSTAR	0.123	0.286	0.656	0.277	0.705	0.437
	Itr-TSTAR	0.149	0.260	0.712	0.271	0.701	**0.477**
switch. → bible	STRAP	0.042	0.323	0.476	0.062	0.670	0.307
	Van-TSTAR	0.128	0.196	0.729	0.085	0.456	0.330
	TSTAR	0.758	0.260	0.605	0.007	0.725	0.419
	Itr-TSTAR	0.097	0.241	0.637	0.006	0.745	**0.456**
poetry → shak.	STRAP	0.067	0.327	0.571	0.159	0.810	0.450
	Van-TSTAR	0.207	0.166	0.821	0.398	0.576	0.460
	TSTAR	0.131	0.244	0.717	0.241	0.733	0.509
	Itr-TSTAR	0.164	0.207	0.768	0.278	0.704	**0.522**
switch. → shak.	STRAP	0.045	0.328	0.489	0.012	0.956	0.461
	Van-TSTAR	0.152	0.187	0.740	0.034	0.948	**0.696**
	TSTAR	0.089	0.250	0.651	0.009	0.971	0.628
	Itr-TSTAR	0.122	0.22	0.696	0.010	0.973	0.675
bible → shak.	STRAP	0.110	0.242	0.634	0.231	0.764	0.465
	Van-TSTAR	0.214	0.151	0.842	0.377	0.613	**0.508**
	TSTAR	0.100	0.227	0.713	0.309	0.676	0.472
	Itr-TSTAR	0.126	0.210	0.743	0.299	0.683	0.499
bible → switch.	STRAP	0.088	0.238	0.625	0.080	0.918	0.565
	Van-TSTAR	0.171	0.162	0.810	0.186	0.813	0.649
	TSTAR	0.084	0.227	0.713	0.082	0.916	0.646
	Itr-TSTAR	0.101	0.213	0.73	0.083	0.911	**0.658**
shak. → switch.	STRAP	0.039	0.344	0.534	0.000	0.998	0.533
	Van-TSTAR	0.138	0.212	0.767	0.002	0.988	0.755
	TSTAR	0.089	0.283	0.661	0.000	0.998	0.660
	Itr-TSTAR	0.110	0.249	0.719	0.002	0.991	**0.712**
poetry → switch.	STRAP	0.054	0.312	0.616	0.007	0.993	0.610
	Van-TSTAR	0.164	0.184	0.825	0.028	0.972	**0.801**
	TSTAR	0.113	0.250	0.740	0.015	0.985	0.728
	Itr-TSTAR	0.141	0.213	0.791	0.019	0.981	0.774
bible → poetry	STRAP	0.060	0.275	0.633	0.346	0.621	0.374
	Van-TSTAR	0.094	0.195	0.777	0.533	0.417	0.315
	TSTAR	0.054	0.253	0.684	0.323	0.597	**0.394**
	Itr-TSTAR	0.064	0.237	0.700	0.348	0.559	0.375
shak. → poetry	STRAP	0.071	0.353	0.550	0.160	0.818	0.449
	Van-TSTAR	0.155	0.219	0.753	0.228	0.748	**0.551**
	TSTAR	0.108	0.286	0.641	0.166	0.815	0.512
	Itr-TSTAR	0.128	0.258	0.693	0.190	0.794	0.538
switch. → poetry	STRAP	0.037	0.331	0.524	0.094	0.844	**0.432**
	Van-TSTAR	0.101	0.215	0.715	0.261	0.486	0.334
	TSTAR	0.064	0.268	0.631	0.106	0.701	0.427
	Itr-TSTAR	0.084	0.239	0.673	0.158	0.603	0.388
All Styles Avg.	STRAP	0.063	0.311	0.563	0.148	0.801	0.440
	Van-TSTAR	0.159	0.189	0.784	0.259	0.644	0.497
	TSTAR	0.093	0.256	0.674	0.145	0.787	0.519
	Itr-TSTAR	0.116	0.232	0.715	0.158	0.768	**0.538**

Table 6: Performance comparison of T-STAR models with STRAP on 12 different directions, across four styles. T-STAR and Iterative T-STAR beats STRAP for 11 directions out of 12. S.R. - Style Retention, S.T. - Style Transfer, WSAcc - Weighted Style Accuracy.

tions. **Third**, when we compare iterative T-STAR against vanilla T-STAR, we beat it in 7 out of 12 directions, where the average absolute improvement across these 7 directions is 8%, as compare to the average absolute loss in 5 out of 12 styles is 2%. **Fourth**, Iterative T-STAR model outperforms T-STAR model on 9 out of 12 directions, which shows that the improvement in synthetic dataset quality, is boosting the performance of the downstream task. Consistent with the findings of the previous experiment, we see Iterative T-STAR is able to find the middle ground of transferring style without compromising on content preservation.

7.2.1 Qualitative Analysis

In Table 7, we enumerate few examples with generated stylized sentences using STRAP and T-STAR variations. We can infer that, although STRAP performs well in transforming the sentence to the given style, it alters the meaning (row 1,3, and 4). On the other hand, vanilla T-STAR does not always transform the style (row 1 and 2). However, with T-STAR and Iterative T-STAR are able to transform the sentences while keeping the style intact.

We further quantify these observations through an extensive set of human annotations as described below.

8 Human Evaluations

Automatic metrics are insufficient to thoroughly understand subjective quality measures like content preservation. Therefore, we conduct an extensive case study with human evaluations. Our analysis is two folds, first we compare STRAP with our models T-STAR and Iterative T-STAR on meaning preservation. Second, we further understand the various categories of meaning loss failures.

For both the human evaluation tasks below, the criteria for choosing annotators were i) proficient in English with a minimum education of Diploma. ii) The annotators have to first qualify on two simpler questions, else they are not allowed to continue on the task. Each instance is annotated by three taskers, and the final annotation is a majority vote.

8.1 Comparison on Meaning Preservation

In order to study the faithfulness of the T-STAR models, we do a side-by-side human evaluation. In this task, a source sentence is shown with 2 stylized target sentence (one from T-STAR and another from STRAP). We present the annotators with three options to judge content preservation with respect to source sentence: option on left better than one on right, right better than left and both equal.

We extensively compare the two models across all 12 directions for four styles. For each direction we randomly sample 500 instances. Each instance is rated by 3 annotators leading to a total of 18,000 ratings. We summarize our findings in Table 8. Both T-STAR ad Iterative T-STAR significantly outperform STRAP in terms of being better at content preservation (The > STRAP column in Figure 8). Further Iterative T-STAR has 7% higher meaning preservation compared to T-STAR. In addition to the quantitative content preservation metrics dis-

Direction	Input Sentence	STRAP	Vanilla T-STAR	T-STAR	Iterative T-STAR
Poetry→Bible	"Dear Lord! It has a fiendish look"–	But they laughed, and said, Oh my God!	Thou lookest fiendish, O LORD	Behold, it beholdeth to be fiendish, saith the LORD.	Dear Lord, it looketh fiendish unto thee.
Bible→Poetry	And he said unto another, Follow me.	And bade the other follow me;	And another: "O follow me!" said he,	And follow me! and I will be be ye gone,	And thou, my love, will follow me.
Shake.→Switch.	Dead art thou, dead!	did you get a uh a uh you have a dead art	you're dead i say you are dead	oh you'd die	you're dead yeah
Shake.→Switch.	Thus with a kiss I die.	so i'm i'm dying to get a kiss	so i die with a kiss	i die with a kiss	so i die with a kiss

Table 7: Example of generated stylized sentences for STRAP, Vanilla T-STAR, T-STAR and Iterative T-STAR models for the given input sentence.

Model	> STRAP	< STRAP	= STRAP
T-STAR	70.8%	26.4%	2.8%
Itr. T-STAR	77.5%	20.5%	2.0%

Table 8: Comparison of T-STAR and Iterative T-STAR models against STRAP for content preservation.

Model	Type of Error			
	None↑	*Hal.*	*Sem. Drift.*	*Incomp.*
STRAP	10.48%	39.38%	35.38%	14.75%
T-STAR	19.36%	24.46%	33.71%	22.45%
Itr. T-STAR	24.83%	22.6%	36.98%	15.58%

Table 9: Aggregate Error Analysis for error types: Hallucinations, Semantic Drift and Incomplete for 6,000 samples. With Iterative T-STAR, the number of samples with no errors and less hallucinations increase significantly.

cussed in Section 7, this analysis gathers additional qualitative evidence towards AMRs as an effective intermediate representation for content preserving style transfer. The complete statistics per direction are available in Appendix F.

8.2 Error Analysis

In the next study, we further aim to study the nature of meaning loss errors made by style tranfer models. We categorize these errors into three categories i) *Hallucinations*: new information not present in the source sentence is added to target ii) *Semantic Drift:* the target sentence has a different meaning to source sentence iii) *Incomplete:* some important content information is missed in the target. The taskers also have the option to select "No Error" if the meaning is preserved in the generated target. As in the previous experiment, we collect 18,000 ratings and the results are summarized in Table 9.

We observe that our models T-STAR and Iterative T-STAR consistently beat STRAP in the *"No Error"* category. Furthermore, the amount of hallucinations significantly drops to 24.46% and further 22.6% with Iterative T-STAR across all styles from

39.3% for STRAP. Reduction in hallucination can be clearly seen as a benefit of encoding critical information in the source sentence using a semantic parse representation like AMR. As a sign of improving the AMR parsing quality, we see that iterative T-STAR further reduce the Incomplete to 15.5%. For further details refer to Appendix F.

8.2.1 Usefulness of an interpretable intermediate representation

With intermediate AMRs being interpretable, it is possible to broadly understand if such errors are emerging from either encoder or decoder module. To intuitvely understand the reason for high number in *Incomplete* and *Semantic Drift* errors, we qualitatively analyzed some instances along with the generated intermediate AMRs. We have listed down these examples in Appendix F. We observed that for *Incomplete* errors, the generated AMRs were not encoding complete information, and thus this error percolated from the T-STAR encoder. For the majority of the instances, either some entities were missing, and if the clause was separated using ":,;", it was not parsed in the intermediate graph. *Semantic Drift* errors indicate shortcomings in both the modules, for some instances the encoder is not abstracting out the meaning efficiently and for others the decoder is not able to generate sentences with the meaning encoded in the intermediate AMR.

9 Conclusion

We explored the use of AMR graphs as an intermediate representation for the TST task. We see that the performance of the proposed method T-STAR surpasses state of the art techniques in content preservation with comparable style accuracy. Through qualitative analysis we show that obtaining very high style accuracy scores without altering meaning is indeed a challenging problem.

10 Limitations

Some of the limitations for T-STAR based models are the following. **First**, although our proposed models are performing better in the joint objective of content preservation and style transfer, but they are not able to outperform vanilla T-STAR (overall best performing model for CP) and STRAP (overall best performing model for ST). This is a promising future direction, to keep boosting the performance on both the directions without comprising on the other dimension. **Second**, we are not incorporating graph structure in our models, and thus there could be some information loss while interpreting and generating the AMRs. **Third,** based on our error analysis, although our T-STAR encoder is able to generate better AMRs for stylized sentences as compared to vanilla T-STAR model, we are generating significant incomplete AMRs that are missing out on important entities and relations to preserve meaning of source sentence. **Fourth**, similar to prior research to generate synthetic dataset, initial iteration of our model are dependant on an existing off the shelf TST model, however the quality of the generated AMRs improves significantly using the described data augmentation strategy. **Fifth**, our work is dependant on a robust AMR parsing approach, which makes it challenging to adopt our approach for other languages. However, with the recent advancements in multilingual AMR parsing, it will be feasible in upcoming future works.

References

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for amr parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015.

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2020. Online back-parsing for amr-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1206–1219.

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. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Yitao Cai, Yue Cao, and Xiaojun Wan. 2021. Revisiting pivot-based paraphrase generation: Language is not the only optional pivot. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4255–4268, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yu Cheng, Zhe Gan, Yizhe Zhang, Oussama Elachqar, Dianqi Li, and Jingjing Liu. 2020. Contextual text style transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2915–2924.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022a. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Zhenyun Deng, Yonghua Zhu, Yang Chen, Michael Witbrock, and Patricia Riddle. 2022b. Interpretable amr-based question decomposition for multi-hop question answering. *arXiv preprint arXiv:2206.08486*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Wenchao Du and Jeffrey Flanigan. 2021. Avoiding overlap in data augmentation for amr-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1043–1048.

Angela Fan and Claire Gardent. 2020. Multilingual amr-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2889–2901.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. 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. Association for Computational Linguistics.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. *CoRR*, abs/1711.06861.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. *CoRR*, abs/2002.03912.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2020. Deep learning for text style transfer: A survey. *CoRR*, abs/2011.00416.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, et al. 2021. Abstract meaning representation (amr) annotation release 3.0.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. *CoRR*, abs/2010.05700.

Surabhi Kumari, Nikhil Jaiswal, Mayur Patidar, Manasi Patwardhan, Shirish Karande, Puneet Agarwal, and Lovekesh Vig. 2021. Domain adaptation for nmt via filtered iterative back-translation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 263–271.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 957–966. JMLR.org.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *CoRR*, abs/1804.06437.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yixin Liu, Graham Neubig, and John Wieting. 2020. On learning text style transfer with direct rewards. *CoRR*, abs/2010.12771.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5108–5118.

Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. A dual reinforcement learning framework for unsupervised text style transfer. *CoRR*, abs/1905.10060.

Xinyao Ma, Maarten Sap, Hannah Rashkin, and Yejin Choi. 2020. Powertransformer: Unsupervised controllable revision for biased language correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7426–7441. Association for Computational Linguistics.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabás Póczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W. Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. *CoRR*, abs/2004.14257.

David D McDonald and James Pustejovsky. 1985. A computational theory of prose style for natural language generation. In *Second Conference of the European Chapter of the Association for Computational Linguistics*.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Trans. Assoc. Comput. Linguistics*, 6:373–389.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Bidisha Samanta, Mohit Agrawal, and NIloy Ganguly. 2021. A hierarchical VAE for calibrating attributes while generating text using normalizing flow. 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 2405–2415, Online. Association for Computational Linguistics.

Bidisha Samanta, Niloy Ganguly, and Soumen Chakrabarti. 2019. Improved sentiment detection via label transfer from monolingual to synthetic code-switched text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3528–3537, Florence, Italy. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30.

Yukai Shi, Sen Zhang, Chenxing Zhou, Xiaodan Liang, Xiaojun Yang, and Liang Lin. 2021. Gtae: Graph transformer–based auto-encoders for linguistic-constrained text style transfer. 12(3).

Ayush Singh and Ritu Palod. 2018. Sentiment transfer using seq2seq adversarial autoencoders. *CoRR*, abs/1804.04003.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *CoRR*, abs/1902.07282.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2018. Multiple-attribute text style transfer. *CoRR*, abs/1811.00552.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "transforming" delete, retrieve, generate approach for controlled text style transfer. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3267–3277. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Alexey Tikhonov, Viacheslav Shibaev, Aleksander Nagaev, Aigul Nugmanova, and Ivan P Yamshchikov. 2019. Style transfer for texts: Retrain, report errors, compare with rewrites. 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 3936–3945.

Ke Wang, Hang Hua, and Xiaojun Wan. 2019. Controllable unsupervised text attribute transfer via editing entangled latent representation. *CoRR*, abs/1905.12926.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Tianming Wang, Xiaojun Wan, and Shaowei Yao. 2021. Better amr-to-text generation with graph structure reconstruction. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3919–3925.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019a. Beyond BLEU: training neural machine translation with semantic similarity. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4344–4355. Association for Computational Linguistics.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019b. Simple and effective paraphrastic similarity from parallel translations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4602–4608, Florence, Italy. Association for Computational Linguistics.

Yu Wu, Yunli Wang, and Shujie Liu. 2020. A dataset for low-resource stylized sequence-to-sequence generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9290–9297. AAAI Press.

Qingrong Xia, Zhenghua Li, Rui Wang, and Min Zhang. 2021. Stacked amr parsing with silver data. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4729–4738.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020a. Improving amr parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511.

Jin Xu, Yinuo Guo, and Junfeng Hu. 2020b. Incorporate semantic structures into machine translation evaluation via UCCA. *CoRR*, abs/2010.08728.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914, Mumbai, India. The COLING 2012 Organizing Committee.

Ivan Yamshchikov, Viascheslav Shibaev, and Alexey Tikhonov. 2019a. Dyr bul shchyl. proxying sound symbolism with word embeddings. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 90–94, Minneapolis, USA. Association for Computational Linguistics.

Ivan P. Yamshchikov, Viacheslav Shibaev, Nikolay Khlebnikov, and Alexey Tikhonov. 2020. Style-transfer and paraphrase: Looking for a sensible semantic similarity metric. *CoRR*, abs/2004.05001.

Ivan P Yamshchikov, Viacheslav Shibaev, Aleksander Nagaev, J ürgen Jost, and Alexey Tikhonov. 2019b. Decomposing textual information for style transfer. *EMNLP-IJCNLP 2019*, page 128.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Amr parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. Amr parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319.

A Extended Related Works

A.1 TST Metrics

Table 2 summarizes different metrics that have been used to measure content preservation and style transfer efficacy. Yamshchikov et al. (2020) presents a comprehensive analysis and categorization of several such metrics with respect to human evaluations. Tikhonov et al. (2019) also points out some flaws in traditional evaluation techniques and insists on using human written reformulations for better evaluation[4].

A.2 Text to AMR

Recent works (Bevilacqua et al., 2021; Cai and Lam, 2020; Zhou et al., 2020) for text-to-AMR task have pushed the SOTA, that makes it feasible to automatically construct AMR given a sentence. As a consequence, semantic-preserving NLG tasks, such as Neural Machine Translation (Song et al., 2019; Xu et al., 2020b), Abstractive Summarization (Takase et al., 2016), and Question Decomposition for multi-hop question answering (Deng et al., 2022b) use AMRs as intermediate representations. However, AMRs have not been explored for style transfer tasks before our work.

The increase in AMRs being adopted for several seq2seq tasks is due to the boost in the quality of AMR parsers. Earlier works, relied on statistical approaches (Peng et al., 2017; Flanigan et al., 2014, 2016) to generate AMRs for a given piece of text. With the emergence of deep learning, various AMR parsers are being proposed, which can be divided into following categories: i) sequence-to-sequence based AMR-parsers (Xu et al., 2020a), ii) sequence-to-graph based AMR parsers (Zhang et al., 2019), where the graph is incrementally built by spanning one node at a time. More recently, several works have adopted pretrained models for AMR parsers, and have observed a boost in performance. Bai et al. (2022) uses BART model and posit the AMR parsing task as a seq2seq task, and generates a traversal of the AMR parser as the output. Bai et al. (2022) incorporates a pretraining strategy to better encode graph information in the BART architecture. Xu et al. (2020a) uses sentence encoding generated from BERT model. In this work, we adopt the pretrained technique based AMR parser, to generate high quality AMRs for the given stylized sentences.

Although off-the-shelf AMR parsers work well for some problems (Fan and Gardent, 2020), they often need to be modified to be useful in the downstream tasks. For instance, Deng et al. (2022b) proposed graph segmentation strategy to perform question decomposition on a multi-hop query.

Xia et al. (2021) and Du and Flanigan (2021) illustrated that using silver data augmentation can help improve in the task of AMR parsing. In this work, we also illustrate the benefit of using silver data towards improving the style agnosticity of AMR graphs as an intermediate representation.

A.3 AMR to Text

Similar to text-to-AMR models, AMR-to-text frameworks can also be categorised into two types - i) sequence-to-sequence generation frameworks (), ii) graph-encoder based frameworks (Song et al., 2018; Wang et al., 2021, 2020). Bai et al. (2020) propose a decoder that back-predicts projected AMR graphs to better preserve the input meaning than standard decoders. Bai et al. (2022) argues that PLMs are pretrained on textual data, making is suboptimal for modeling structural knowledge, and hence propose self-supervised graph-based training objectives to improve the quality of AMR-to-text generation.

B Implementation Details

Offensive language We used the "List of Dirty, Naughty, Obscene or Otherwise Bad Words"[5] to validate that the source and the generated target text do not contain any offensive text.

Model Architecture: We use a standard `t5-base` encoder-decoder model as described in (2020). The pre-trained `HuggingFace`[6] T5 transformer is used for both text-to-AMR and AMR-to-text parts of the proposed architecture. The model is pre-trained on the Colossal Clean Crawled Corpus (C4) [7] comprising of ∼750 GBs worth of text articles. The model comprises of 220 billion parameters, and is pre-trained for $2^1 9$ steps before fine-tuning. For pre-training, AdaFactor optimizer (Shazeer and Stern, 2018) is used with "inverse square root" learning rate schedule.

AMR Graph Construction: We use the SPRING model (Bevilacqua et al., 2021) to generate AMR graphs from source style text. We use

---

[4]Due to the difficulty of our task, we instead restrict to human evaluations instead of obtaining human gold-standard benchmarks.

[5]https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words
[6]https://github.com/huggingface/transformers
[7]https://www.tensorflow.org/datasets/catalog/c4

amrlib[8] package to generate the AMR graphs. This implementation uses `T5-base` (Raffel et al., 2020) as its underlying model, as opposed to the `BART model` (Lewis et al., 2019) in the SPRING architecture. It is trained on **AMR 3.0** (LDC2020T02) dataset (Knight et al., 2021) that consists of 59K manually created sentence-AMR pairs. The model is trained for 8 epochs using a learning rate of 10^{-4}. The source and target sequence lengths are restricted to 100 and 512 tokens respectively. Note that t5-based SPRING model achieves an SMATCH score of 83.5, which guarantees the quality of obtained AMR representations z_i.

AMR-Based Style Transfer: We use the `T5wtense` (T5 with tense) architecture from the `amrlib` package. The `T5wtense` architecture encodes part-of-speech (POS) tags to the concepts in the AMR graph, which helps the generation model to predict the tense of the output sentence since AMR graphs do not retain any tense information from their corresponding sentence. This model outperforms the standard T5-based model by 10 BLEU points on the **AMR 3.0** (LDC2020T02) dataset (Knight et al., 2021). To keep the training steps comparable for the subsets of the CDS dataset (Krishna et al., 2020), we train this `t5-base` model for 20 epochs for the Bible, Romantic Poetry, and Shakespeare datasets, and 5 epochs for the Switchboard dataset. The model was trained for 20 epochs for the Shakespeare Author Imitation Dataset (Xu et al., 2012) as well. We used a learning rate of 10^{-4} for both datasets, and restricted source and target sequence lengths to 512 and 90 throughout, respectively. Everything else was kept same as the `amrlib` implementation to keep the results consistent.

STRAP baseline: We train the model keeping the same hyperparameter configuration as reported in Krishna et al. (2020). We train each style-specific decoder for 3 epochs with learning rate of 5×10^{-5} with Adam optimizer (Kingma and Ba, 2014). During inference we set the *p*-value for nucleus sampling (Holtzman et al., 2019) to 0.7 to have an appropriate balance between content preservation and style accuracy scores.

Style classifiers: Similar to Krishna et al. (2020), we fine-tune a `RoBERTa-large` model (Liu et al., 2019) using the official implementa-

tion in the `fairseq` package[9] to train the style classifiers mentioned in Table 3. For all classifier variants, learning rate of 10^{-5} and a mini-batch size of 32 was used. The models were trained for 10 epochs using Adam optimizer (Kingma and Ba, 2014). We also masked out named entities, nouns and numbers from the input before training the classifier. We used `spacy` package[10] to obtain the named entities and POS tags. To obtain the named-entities and POS for AMR graphs, information extracted from original sentences was used.

Train-Validation-Test splits: The data splits were kept the same as the baseline model, STRAP (Krishna et al., 2020) and can be found in Table 10.

Dataset		Train split	Validation split	Test Split
CDS	Bible	31,404	1,714	1,714
	Switchboard	145,823	1,487	1,488
	Poetry	26,880	1,464	1,470
	Shakespeare	24,852	1,313	1,293
SAID	Original	18,395	1,128	1,462
	Modern	18,395	1,218	1,462

Table 10: Size of train, validation and test sets for CDS dataset (Krishna et al., 2020) and Shakespeare Author Imitation Dataset (SAID) (Xu et al., 2012).

Computational time and device setup: All experiments were done on a 16 GB NVIDIA V100 GPU system with 120 GB n1-standard-32 Intel Broadwell CPU. It took ∼16 hrs to train the AMR-to-Text models for Shakespeare Author Imitation dataset (Xu et al., 2012) and ∼25 hrs for the CDS dataset.

Evaluation Metrics: We used the `gensim`[11] package to compute the Word Mover Distance (WMD) (Kusner et al., 2015), `nltk`[12] package to compute the BLEU scores (Papineni et al., 2002), `smatch` package[13] to compute the SMATCH score (Cai and Knight, 2013) and implementation by Krishna et al. (2020)[14] for the cosine similarity using SIM embeddings (Wieting et al., 2019a).

License of the packages used: The following packages use the MIT License[15] - `amrlib`, `spacy`, `fairseq`, `smatch`, and `STRAP`. The following packages use the Apache License 2.0[16] - `nltk`, and `huggingface's transformer`.

The following packages use the GNU LGPL license 3.0[17] - `gensim`.

C T-STAR-*Encoder* Ablation Study

The performance of our T-STAR-Encoder heavily depends on the quality of synthetic dataset generated while stylizing the sentences present in AMR 3.0 dataset. Therefore, we conducted thorough empirical analysis to identify a filtering strategy to boost the performance of the encoder.

To obtain the initial set of stylized sentences, we use the state-of-the-art model available to transform generic English sentences to relevant styles {Poetry, Shakespeare, Bible, Switchboard}, i.e., seq2seq inverse paraphrase module (Krishna et al., 2020).

We then fine-tune our T-STAR-Encoder on synthetic datasets obtained from different filtering strategies, and compare the performan on test split of AMR 3.0, to ensure that the quality of the generated AMRs do not drop significantly. We present our findings in Table 11. We observe that using the whole set of generated samples, leads to a significant drop in the performance (row-1).

Therefore, we filter out the augmented stylized sentences with SIM similarity score (Wieting et al., 2019a) below the threshold δ. This filtering strategy was able to significantly improve over the T-STAR-Encoder performance, giving competitive results to the non-augmented Vanilla T-STAR-Encoder. We select the best performing δ based on the performance on the test-split of AMR 3.0. Note that we have used the best performing threshold, $\delta = 0.7$

Datasets	Precition	Recall	F-Score
Vanilla T-STAR Encoder	0.829	0.794	0.811
T-STAR-Encoder	0.671	0.390	0.493
T-STAR Encoder-Flt (δ=0.5)	0.830	0.790	0.810
T-STAR Encoder-Flt (δ=0.6)	0.807	0.753	0.779
T-STAR Encoder-Flt (δ=0.7)	0.836	0.798	0.816
T-STAR Encoder-Flt (δ=0.8)	0.828	0.793	0.810
Iterative T-STAR-Encoder(δ=0.7)	0.829	0.794	0.811

Table 11: SMATCH scores on various versions of T-STAR-Encoder on AMR 3.0 dataset's test split.

D Unsupervised Evaluation of AMR parsing

Since we use AMR graphs as the intermediate representation for the TST task, it is important to validate the generation quality of generated AMRs in terms of content preservation with respect to the input sentence. However, there does not exist an

unsupervised metric to evaluate content overlap between an AMR graph and a sentence. Hence we propose to use a slight variation of the Word Mover Distance (Kusner et al., 2015) for this purpose. We choose WMD over other content preservation metrics for the following reasons -

- Yamshchikov et al. (2019a) illustrate the efficacy of WMD to evaluate text style transfer over other metrics based on correlation with human evaluations. Which means that it is more robust to the domain difference in the input and the output sentence, making it an ideal candidate for text-AMR similarity measurement.

- Syntactic metrics like BLEU would not be able to compute the content overlap between an AMR graph and a sentence because word representation in an AMR graph discards noun forms and tense information, and some verb tokens are mapped to a different Prop-Bank verb. These modifications along with the disparity in sequential-graphical representation makes syntactic metrics infeasible for the task.

- Semantic representations like SIM are fragile to the input sequence order, and affected by non-content bearing words as well. However, WMD adopts on a bag-of-words paradigm, making it more suitable for the task.

We propose the following two variants of WMD -

- **WMD Overall** - In this variant, we aimed to keep the content bearing tokens from the sentence and AMR graphs. For sentence, we removed the stopwords (after doing a detailed corruption study on sentences, refer to Table 13), while for AMR Graphs we removed AMR notation specific tokens (like ":op?", "ARG?"), punctuations (like "(", " ' "), assigned variables and propbank code for verbs (eg. changing "s / say-01" to "say").

- **WMD Verb Overall** - In this variant we specifically want to compute the similarity of verbs in the parsed AMR graphs and input sentence. For this, use `nltk` POS tagging tool to extract out verbs from the input sentence, and directly extract out propbank based verbs from the AMR graph.

Refer to Table 12 for an example of the preprocessing strategy adopted.

Input Sentence -
Malaysian vice-prime minister Anwar ended a visit to China this afternoon , and left Shanghai for Tokyo.
Extracted content from Input Sentence -
malaysian vice-prime minister anwar ended visit china afternoon , left shanghai tokyo.
Extracted Verbs from Input Sentence -
ended left
Input AMR-
(a2 / and :op1 (e2 / end-01 :ARG0 (p / person :name (n / name :op1 "Anwar") :ARG0-of (h / have-org-role- 91 :ARG1 (c7 / country :name (n3 / name :op1 "Malaysia")) :ARG2 (m / minister :mod (p2 / prime) :mod (v / vice)))) :ARG1 (v2 / visit-01 :ARG0 p :ARG1 (c6 / country :name (n2 / name :op1 "China"))) :time (d / date-entity :dayperiod (a3 / afternoon) :mod (t / today))) :op2 (l / leave-11 :ARG0 p :ARG1 (c8 / city :name (n4 / name :op1 "Shanghai")) :ARG2 (c9 / city :name (n5 / name :op1 "Tokyo"))))
Extracted sequence-
and end person name Anwar have-org-role country name Malaysia minister prime vice visit country name China afternoon today leave city name Shanghai city name Tokyo
Corresponding Verb extraction (AMR) -
end visit leave

Table 12: Illustrative example of the preprocessing done in proposed text-AMR unsupervised WMD Overall and WMD Verb Overall metrics.

Model	WMD Mean / Std dev.	SIM Mean / Std dev.
Original	0.0 / 0.0	1.0 / 0.0
Stop Words	0.1049 / 0.0761	0.9373 / 0.0734
Lowercase	0.0754 / 0.1319	0.9946 / 0.0321
Stop Words + Lowercase	0.1663 / 0.1347	0.9373 / 0.0734
POS	0.1081 / 0.0731	0.8680 / 0.1929
Stop Words + Lowercase + POS	0.1966 / 0.1438	0.8492 / 0.1957
Synonym Replacement	0.1864 / 0.1266	0.6946 / 0.1832
Synonym + Stop Words	0.2228 / 0.1335	0.6545 / 0.1979
Synonym + Lowercase	0.2133 / 0.1511	0.6946 / 0.1832
Synonym + Stop Words + Lowercase	0.2476 / 0.1534	0.6545 / 0.1979
Synonym + POS	0.2320 / 0.1306	0.5948 / 0.2209
Synonym + Stop Words + Lowercase + POS	0.2686 / 0.1525	0.5817 / 0.2243

Table 13: WMD scores on various corruption strategies against original sentence on test set of AMR 3.0 dataset. POS refers to removing tags other than nouns, verbs and adjectives from the sentence.

We also study the effect on text-AMR WMD score juxtaposed to text-text WMD scores on varying degree of similarity between the compared sequences. For this we use the diverse paraphraser trained by Krishna et al. (2020) on the test set of AMR 3.0 dataset, and generate paraphrases with varying nucleus sampling p-values (Holtzman et al., 2019) from 0.0 to 1.0 with step size of 0.1. We notice that the WMD scores for text-text WMD (blue line in Fig. 5) and text-AMR WMD (red line in Fig. 5) are similar to each other throughout.

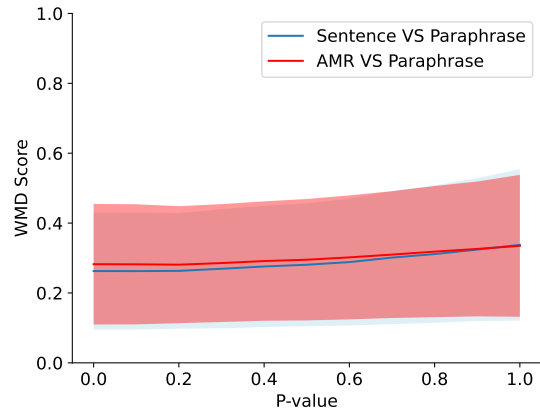We present the WMD Overall scores and WMD Verb Overall scores for the CDS dataset in Table

Figure 5: WMD Scores on AMR 3.0 dataset for sentence-paraphrase pairs (blue) and amr-paraphrase pairs (red). The shaded region denotes standard deviations from the mean.

Models	Styles	WMD Overall	Verb WMD Overall
Vanilla T-STAR Encoder	Bible	0.243	0.409
	Poetry	0.272	0.532
	Shakespeare	0.322	0.484
	Switchboard	0.292	0.467
T-STAR Encoder (Unfiltered)	Bible	0.339	0.511
	Poetry	0.357	0.599
	Shakespeare	0.403	0.552
	Switchboard	0.341	0.537
T-STAR Encoder Flt (δ=0.7)	Bible	0.290	0.461
	Poetry	0.322	0.568
	Shakespeare	0.365	0.522
	Switchboard	0.323	0.512
Iterative T-STAR Encoder Flt (δ=0.7)	Bible	0.281	0.439
	Poetry	0.300	0.550
	Shakespeare	0.344	0.500
	Switchboard	0.301	0.491

Table 14: WMD Overall and WMD Verb Overall scores for evaluation of unsupervised content preservation of different models across different styles in the CDS dataset (Krishna et al., 2020).

14; validating the content retention in parsed AMR graphs for different strategies. We notice that Iterative T-STAR Encoder outperforms the T-STAR Encoder-Flt (δ=0.7) and T-STAR Encoder (unfiltered) baselines in both WMD overall and Verb WMD overall. Even though comparable, it is still however lesser than the Vanilla T-STAR Encoder numbers. However, we believe we can credit that to more style information retention in Vanilla T-STAR Encoder (refer to Section 6.2), leading to poorer performance in downstream text style transfer task.

E Data Augmentation for T-STAR Decoder

We hypothesize that the T-STAR decoder performances will improve if the underlying model, is better at generating text given an AMR graph. To this end, we create synthetic dataset using sentences from Wikipedia corpus. We sample 10 million

Dataset	BLEU Score
GoldAMR (Baseline)	44.36
Wikipedia 1M (Unfiltered)	48.14
Wikipedia 120K- Filter(<0.15)	44.56
Wikipedia 1M -Filter (<0.15)	48.58
Wikipedia 2.3M - Filter (<0.15)	**49.14**

Table 15: Fine-tuning model for AMR-To-Text generation task using data augmentation technique using Wikipedia Sentences

sentences from it and generate the corresponding AMRs using our vanilla T-STAR model. We further filter out the samples for which WMD Overall, mentioned in Appendix D and keep samples with a WMD score below 0.15, which results in 2.3M instances.

We first fine-tune the T5-Base model for AMR to Text task on this filter dataset. We obtain a BLEU score of 49.13 on Gold AMR test set. Note that this performance is very close to the state-of-the-art result 49.2 BLEU for this task (Bai et al., 2022). Table 15 lists down the different filtering strategy and dataset sizes we experimented to identify the best strategy to improve the performance.

We then compare the performance of the best performing model on the style transfer task again STRAP. We observe that this model is not beating the STRAP performance across various style directions. Therefore, we conclude that a vanilla fine-tuning of model for AMR to text task, does not necessarily boost the performance in the downstream tasks.

F   Error Analysis

F.1   Comparison on Meaning Preservation

We present the results across all the 12 directions for content preservation comparitive analysis in Table 16 and 17 respectively. We can observe that for every direction our models are consistently better in content preservation with respect to STRAP model.

F.2   Error Analysis per direction

In this section, we present the error analysis for each direction in Table 18. We observe that T-STAR and Iterative T-STAR models are consistently better on No-Error and Hallucinations across all the directions. Moreover, we observe that for 7 out of 12 direction, Iterative T-STAR model is better than STRAP for Incompleteness error. Note that our T-STAR model was under-performing, however another iteration of model improvements increases

Direction	TSTAR >STRAP	STRAP <TSTAR	TSTAR=STRAP
bible → poetry	60.8%	37.6%	1.6%
bible → shak.	66.6%	32.2%	1.2%
bible→switch.	67%	30.4%	2.6%
poetry→bible	68.6%	29.4%	2%
poetry → shak.	78.6%	19.6%	1.8%
poetry→switch.	69.4%	27.4%	3.2%
shak.→bible	73%	24.8%	2.2%
shak.→poetry	66.2%	27.4%	3.2%
shak.→switch	73.6%	24%	2.4%
switch.→bible	73%	24%	3%
switch.→poetry	73.6%	23.4%	3%
switch.→shak.	79.4%	18.6%	4%

Table 16: Comparitive analysis of TSTAR and STRAP model to understand which model generates more meaning preserving outputs

Direction	TSTAR=STRAP	TSTAR < STRAP	TSTAR > STRAP
bible→switch.	1.8	31.6	66.6
poetry→switch.	2	15.6	82.4
shak.→switch.	4	20.6	75.4
switch.→bible	1.4	19.4	79.2
switch.→poetry	2.2	16.2	81.6
switch.→shak.	0.8	13	86.2
bible→poetry	1.8	27.8	70.4
bible→shak.	0.8	28.8	70.4
poetry→bible	2	20.2	77.8
poetry→shak.	2.8	12.2	85
shak.→bible	2.4	19.6	78

Table 17: Comparison on content preservation using human evaluations on Iterative T-STAR and STRAP across all 12 direction

that number significantly. Note that all the models are giving high error in Semantic Drift, and improving the model for this type of errors can be explored in future works.

F.3   Qualitative Analysis for Incompleteness and Semantic Drift

As we are using interpretable intermediate representations, it is easily possible to understand the intuition behind these errors, and broadly understand which modules (encoder or decoder) needs to be improved further. Therefore, we study few instances and analyze the generated Intermediate AMRs to understand the reason for high number in *Semantic Drift* and *Incomplete* errors. We list down some intstances in Table 20 and Table 19

Across the various instances that we analyzed, we observe that the generated AMRs were not encoding the complete information themselves. For instance, either missing some entities (example 1, 4, 5 and 6 in Table 19), if the clauses were separated using ":", ";", only one of those were parsed in the intermediate graph (example 2 and 3 in Table 19. For semantic drift, we observed that the errors was arising due to the shortcomings in both the modules, i.e., it was leading to meaning change if the Encoder didn't generate an efficient AMR

Direction	STRAP	TSTAR	Itr-TSTAR	STRAP	TSTAR	Itr-TSTAR	STRAP	TSTAR	Itr-TSTAR	STRAP	TSTAR	Itr-TSTAR
	No Error ↑			Hallucination ↓			Incomplete ↓			Semantic Drift		
bible→poetry	42	43	62	98	68	70	239	270	228	121	119	140
bible→shakespeare	45	72	93	224	76	87	55	190	149	176	162	171
bible→switchboard	54	98	106	203	65	87	59	189	132	184	148	175
poetry→bible	46	85	110	257	232	175	22	27	15	175	156	200
poetry→shakespeare	43	134	166	224	89	79	50	53	47	183	224	208
poetry→switchboard	80	142	160	171	63	39	63	82	65	186	213	236
shakespeare→bible	44	69	120	253	203	186	19	43	14	184	185	180
shakespeare→poetry	44	55	85	187	190	161	77	89	69	192	166	187
shakespeare→switchboard	70	132	161	151	69	62	42	98	47	237	201	230
switchboard→bible	50	85	129	241	213	221	42	54	14	167	148	135
switchboard→poetry	49	98	114	165	113	97	140	165	123	146	124	166
switchboard→shakespeare	62	149	186	189	87	92	77	87	31	172	177	191

Table 18: Type of Error Analysis across three models STRAP, TSTAR, Iterative-TSTAR across all the four styles.

graph, or if the decoder was not able to interpret AMR correctly. We have listed the modules that could be the potential reason for the error in the last column in Table 20.

It is important to note that, the source of errors is very easy to identify now because we are using robust, interpretable and symbolic representation as pivot to transfer from style A to style B. We have also provided a case study of performance of various baselines and proposed model in Table 7 on the CDS dataset.

S.No	Input Stylized Sentence	Output Stylized Sentence	TSTAR-Encoder AMR
1.	And she answered and said unto him, Yes, Lord: yet the dogs under the table eat of the children's crumbs.	Yet the dogs under the table Eat children's crumbs.	(e / eat-01 :ARG0 (d / dog :location (u / under :op1 (t / table))) :ARG1 (c / crumb :poss (c2 / child)) :mod (s / still))
2.	And the priest said unto them, Go in peace: before the LORD is your way wherein ye go.	Before the Lord go in peace on high;	(g / go-02 :mode imperative :ARG0 (y / you) :manner (p / peace) :time (b / before :op1 (p2 / person :name (n / name :op1 "Lord"))))
3.	And there were made on them, on the doors of the temple, cherubims and palm trees, like as were made upon the walls; and there were thick planks upon the face of the porch without.	The cherubims and palm-trees are at the temple doors, And the thick planks on the porch face.	(a / and :op1 (b / be-located-at-91 :ARG1 (a2 / and :op1 (c / cherubim) :op2 (t / tree :mod (p / palm))) :ARG2 (d / door :part-of (t2 / temple))) :op2 (b2 / be-located-at-91 :ARG1 (p2 / plank :ARG1-of (t3 / thick-03)) :ARG2 (f / face :part-of (p3 / porch))))
4.	and uh they've had catalytic con- you know catalytic converters on those on uh i think they're required in California and Oregon and and Washington	In Caledonia, Oregon, and Washington catalytic conversion required.	(r / require-01 :ARG1 (t / thing :ARG0-of (c / convert-01 :mod (c2 / catalytic))) :location (a / and :op1 (s / state :name (n / name :op1 "California")) :op2 (s2 / state :name (n2 / name :op1 "Oregon")) :op3 (s3 / state :name (n3 / name :op1 "Washington"))))
5.	That crowns a lofty clift, which proudly towers	crowns the lofty towers	(c / crown-01 :ARG0 (ii / it) :ARG1 (t / tower :mod (l / lofty)))
6.	And the rough goat is the king of Grecia: and the great horn that is between his eyes is the first king.	The rough goat o'er the king of Greece	(h / have-org-role-91 :ARG0 (g / goat :ARG1-of (r / rough-04)) :ARG1 (c / country :name (n / name :op1 "Greece")) :ARG2 (k / king))

Table 19: Samples from Human Evaluations where the T-STAR output was marked incomplete. We observe that entities and relations missing in the sentence, were not present in their corresponding intermediate AMRs.

Input Stylized Sentence	Output Stylized Sentence	Input generated AMR	Potential Error

Ay, mine own fortune in my misery.	my fortune is my misery	(m / misery
		  :poss (ii / i)
		  :domain (f / fortune
		    :poss ii))
			T-STAR Decoder

yeah it makes you wonder um had the jury been a different group of people	It maketh you to wonder at the diversities of the jury.	(m / make-02
		  :ARG0 (ii / it)
		  :ARG1 (w / wonder-01
		    :ARG0 (y / you)
		    :ARG1 (d / differ-02
		      :ARG1 (j / jury))))
			T-STAR Decoder

Tis not Maria's whispering call;	It is not whispered by Maria.	(w / whisper-01
		  :polarity -
		  :ARG0 (p / person
		    :name (n / name
		      :op1 "Maria"))
		  :ARG1 (ii / it))
			T-STAR Encoder

Yea," quoth he, "Dost thou fall upon thy face?	Quoth he, "Dost thou fall thy face,	(q / quote-01
		  :ARG1 (f / fall-01
		    :ARG1 (y / you
		      :part (f2 / face))
		    :ARG4 f2
		    :polarity (a / amr-unknown))
		  :ARG2 (h / he))
			T-STAR Encoder

Around my grave they'll wither.	They will wither about my grave.	(w / wither-01
		  :ARG1 (t / they)
		  :location (a / around
		    :op1 (g / grave
		      :poss (ii / i))))
			T-STAR Decoder

Justice is sworn 'gainst tears, and hers would crave	uh Justice has sworn to cried and cried	(s / swear-01
		  :ARG0 (p / person
		    :name (n / name
		      :op1 "Justice"))
		  :ARG1 (a / and
		    :op1 (c / cry-02
		      :ARG0 p)
		    :op2 (c2 / cry-02
		      :ARG0 p)))
			T-STAR Encoder

Table 20: Various Instances that had Semantic Drift as a type of error. We manually analyze the same, and hypothesize that the potential error in the listed module.