Using Structured Content Plans for Fine-grained Syntactic Control in Pretrained Language Model Generation

Anonymous ACL submission

Abstract

Large pretrained language models offer powerful generation capabilities, but suffer from a lack of interpretability and fine-grained control. We propose an approach to fine-grained control in generating text directly from a semantic representation, Abstract Meaning Representation (AMR) by augmenting the nodes with syntactic tags. We experiment with English-language generation of three modes of syntax relevant to the framing of a sentence - verb voice (active or passive), verb tense, and realization of human entities - and demonstrate that they can be reliably controlled. Controlling how information is framed is important for applications such as summarization, which aim to highlight salient information.

1 Introduction

Language models pretrained on enormous corpora have become a staple in natural language processing because of their power and adaptability (Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020). These models exhibit strong performance across a range of applications, but lack inherent controllability. To place specific constraints on their output, we must modify them in some way, whether by inserting control codes during pretraining (Keskar et al., 2019) or by introducing additional components that provide control (Dathathri et al., 2020). Such methods allow specification of high-level attributes (e.g., topic or sentiment) but leave the specifics of sub-sentential realization to the model.

In contrast, we investigate the setting in which a pretrained language model is used not as an end-to-end generator for some task, but rather to directly generate text from a predefined content plan. We focus on the controllability of BART when it is fine-tuned to generate text from an intermediate Abstract Meaning Representation (AMR) (Banarescu et al., 2013), a form of graphical semantic representation. We choose AMR because it is a relatively widely used semantic representation that already sees use as an intermediate representation in a variety of applications including summarization (Liu et al., 2015) and machine translation (Song et al., 2019). Our work is particularly applicable to summarization, as document-level graphical representations have been shown to be useful intermediate representations in long-document or multidocument summarization settings, where they capture global context more directly than sequential representations of long passages (Wu et al., 2021).

This setup allows us to make use of the powerful generation abilities of a pretrained language model, while also giving us direct access to a graphical representation of the content, allowing us to insert tags at specific nodes (i.e., constraining the text-level realization of specific verbs or entities) in order to impose fine-grained control. This is not possible in an end-to-end approach to some tasks, like summarization: although it is possible to generate summaries that are controlled overall for a high-level attribute, there is no way to insert control codes or tags into the input that directly control the realization of individual verbs or entities in the output, as verbs or entities in the input document(s) may appear in different sentences and contexts in the output summary. In cases where we desire such control - for example, in query-focused summarization, or when highlighting important entities (Nenkova et al., 2005) - it is useful to have a representation that allows us to specify syntactic aspects at the level of individual verbs or entities.

In our experiments, we augment the AMR input to our generator with three modes of syntax: verb voice (i.e., active or passive), verb tense, and syntactic realization of human entities (i.e., using names, pronouns, or descriptors). Controlling voice and entities contributes to the model's ability to use syntax to highlight a specific topic or focus, following centering theory (Grosz et al., 1995). Given that the input document may convey a different focus, having the ability to specifically control focus for the output summary is important, as there is no guarantee that an end-to-end model will learn implicitly how to do this. As plain AMR does not contain information about tense, which is important for maintaining faithful summaries, we additionally consider controlling verb tense to avoid generating hallucinations about when an event took place.

We find that finetuning BART to generate from AMR augmented with syntax tags makes it largely controllable even when evaluated on a test set designed to have a radically different class distribution (e.g., of voice, tense or pronouns) than the training set, without any tradeoff as to the fluency of the generated output. We further find that training the same model with control on multiple syntactic modes improves performance on voice controllability, though not on tense. Our experiments show that our tagged models are far better at controlling voice, tense and entity realization than a model without tags.

In summary, our contributions are:

- A method of labeling relevant AMR nodes with each of three modes of fine-grained syntactic information (verb voice, verb tense, and entity realization);
- Experiments demonstrating controllability in pretrained BART models finetuned for generation from tag augmented AMR in both in-distribution and off-distribution settings;
- Ablation analyses and experiments on interactions between modes of syntax demonstrating which modes work well together.

We will make our code available upon publication.

2 Related Work

Controllable generation. Most prior work on controllable generation focuses on global attributes that apply to the entire output (Hu et al., 2017; Shen et al., 2017; Chawla and Yang, 2020), rather than fine-grained control of the realization of individual units of content, as we do. This includes work on controllable generation with pretrained language models (Keskar et al., 2019; Dathathri et al., 2020).

Work on controllable summary generation also shares this focus on global attributes. Fan et al. (2018) trains a convolutional model to generate summaries controlled by attribute markers for length, entities to focus upon, domain, and subset of the text. He et al. (2020) fine-tunes a BART model to generate output summaries that are controlled using keywords or prompts, allowing the model to focus on specific entities or desired information. This approach addresses a similar problem to our work, but focuses on global rather than fine-grained control, does not necessarily frame a summary around selected relevant content, and is applicable to classic single-document summarization, whereas our approach is generalizable to multi-document and long-document settings due to its use of a content selection model

Pipelines in surface realization. Elder et al. (2019) demonstrate that a symbolic intermediate representation (based on a dependency graph) is input to a neural generator, this can yield improvements on the surface realization task. Castro Ferreira et al. (2019) find that a pipeline of discrete modules can yield improvements over end-to-end neural models for data-to-text generation. However, Farahnak et al. (2020) find that a pretrained language model (BART) can outperform previous state-of-the-art modular pipelines on surface realization. We aim to take the best of both by using BART as our generator, but leaving the choice of content selector free.

AMR-to-text generation. There is an active body of work on AMR-to-text generation (Konstas et al., 2017; Wang et al., 2020; Bai et al., 2020; Zhang et al., 2020), but most of this work uses architectures specialized for the AMR-to-text setting. In contrast, we focus on pretrained language models' controllability when used in this setting. To our knowledge, the most closely related work to ours is that of Ribeiro et al. (2020), who investigate the use of pretrained language models for multiple graph-to-text generation settings, and whose finetuning setup we follow closely.

3 Data and Methodology

The AMR Bank (Knight et al., 2020) is the largest gold standard corpus for AMR, but it contains only around 60,000 annotated sentences in total, which is not enough data to finetune BART. Thus, instead, we use a much larger text corpus which we parse automatically using the published code for the AMR parser of Cai and Lam (2020). As we are interested in summarization as an application, we use the multidocument summarization corpus of

Sentence	**They had received a call to conduct a background check about 6:15 p.m.**
Linearized AMR (voice tags)	(receive **:active** :ARG0 (they) :ARG1 (call :ARG0 they :ARG1 (conduct **:active** :ARG0 they :ARG1 (check :ARG0 they :ARG1 (background))) :ARG1 (rate-entity-91 :ARG2 (temporal-quantity :quant 1 :unit (minute)) :ARG4 (temporal-quantity :quant 1 :unit (hour)))))

Table 1: Example linearized AMR with voice tags inserted. The verbs tagged for voice in the second example are "receive" (active) and "conduct" (active). Tags are bolded for readability.

Gholipour Ghalandari et al. (2020), which consists of 10,200 clusters containing on average 235 documents each. We refer to this as the *reservoir corpus*. Although we do not use the AMR Bank for finetuning, we use its *proxy* subset, which contains news documents and summaries, for additional evaluation.

Split	Sentences	Usage
Train	4.38M	Held in reserve. (10k split for analysis.)
Val	581k	Finetuning data (500k train, 80k validation).
Test	543k	10k sampled for test set.

Table 2: Partitioning of the larger reservoir corpus.

Due to constraints on time and processing power, we do not use the entire reservoir corpus for finetuning, but rather finetune our models primarily on the reservoir validation set, which contains 580,787 sentences in total. We split the reservoir corpus' validation set into a training set of 500,000 sentences and validation set of 80,000 sentences, which we use for finetuning. For evaluation, we sample 10,000 sentences from the reservoir training set to use for model analysis, and sample 10,000 sentences from the reservoir test set for final performance numbers. We reserve the remainder of the reservoir training set for experiments that require filtering out a portion of the data.We give an overview of the reservoir corpus' split sizes and partitioning in Table 2.

3.1 AMR Linearization

As we use finetuned BART models as our AMR-to-text generators, the input we give them must be sequential rather than arbitrary directed graphs. Thus, following the methodology of Ribeiro et al. (2020), we linearize AMR graphs into a modified version of PENMAN format (Kasper, 1989) that omits identifying handles for each node: for each AMR graph, we start at the root node and perform a depth-first traversal of the graph, adding node and edge labels in order to the linearized sequence, as well as parentheses indicating depth levels (see Table 1).

3.2 Syntax labeling

We describe the labeling procedure for each of our three modes of syntax in this section. In overview, our process is as follows: (1) extract syntactic labels from the raw text using a parser or part-of-speech tagger; (2) use the extracted labels to augment the linearized AMR we use to finetune our AMR-to-text models; (3) extract syntactic labels a second time from our models' output to evaluate against the original tags. We use spaCy (Honnibal et al., 2020) for dependency parsing and part-of-speech tagging. When augmenting linearized AMR, we insert syntactic tags as a modifier directly following the relevant node (see Table 1).

We provide class distributions for each mode of syntax in Appendix B. We note that the voice and entity classes are highly imbalanced, while tense is relatively more evenly distributed across classes.

Voice. To extract passive/active labels from a sentence, we examine the automatically extracted dependency parse for the sentence and individually label each verb as appearing in passive or active tense by checking whether it has an active subject (i.e., a child whose edge has the dependency label `nsubj` or `csubj`) or a passive subject (i.e., a child whose edge is labeled `nsubjpass` or `csubjpass`). We evaluate the reliability of this method in §6.4.

Given these labels, we identify corresponding nodes in the AMR graph by performing exact string matching between the lemmas of each labeled verb and the concept labels of all AMR nodes representing a verb. Verb nodes that are not matched to any labeled verb lemma are not assigned a label. In linearization, the label appears as an additional modifier after the concept string of the verb: either :active or :passive.

Tense. We use the fine-grained part-of-speech tag under the Penn Treebank labeling scheme (Marcus et al., 1993) as the tense tag for each verb (i.e., one of {VB, VBD, VBG, VBN, VBP, VBZ}). We obtain these tags directly from the part-of-speech tagger.

Entity realization. In order to ensure that we can reliably distinguish which pronouns refer to which entities without introducing the potential for additional error from automatic coreference, we filter the data in our entity realization experiments to consider only sentences that fulfill two requirements. First, there must be at most one person node in the AMR. Second, the sentence must contain at most one of the pronouns {she, he, they} or their associated forms (i.e., a sentence containing "he" and "himself" would be acceptable; a sentence containing "her" and "their" would be discarded).

We assume that if one of these pronouns occurs in such a sentence, it is associated with the single person node in the AMR, and give that node the appropriate tag among :pronoun-{he/she/they}. If the person node is named in the AMR, we give it the tag :named.

Otherwise, we assume that the person in question is described in some other way, such as by profession (e.g. "scientist") or by an action they perform (e.g. "visitor"). The description class, however, contains not only cases where the entity described is a specific reference (i.e., a particular identifiable person), but also generic references (i.e., terms describing classes of people, such as "visitors to the location"). As our focus is on realization of specific entities and not generics, we omit the entire :desc class from our experiments. This has the additional benefit of leaving us with much more balanced data, as descriptions made up the majority class originally (approximately 80% of all person instances). Since this drops a substantial portion of our data, in our entity realization experiments, we filter out sentences with :desc tags or multiple person nodes from the reservoir training set until we have an equivalent amount of data to that used in the other experiments, giving us an alternate training and validation set of equal size. To obtain our entity test set, we filter out sentences with :desc tags or multiple person nodes, as well as sentences with no person nodes, until we have 10,000 sentences.

3.3 Finetuning

We closely follow the methodology of Ribeiro et al. (2020) for finetuning. We fine-tune a pretrained BART-large model on AMR graph-sentence pairs to produce the sentence text given the linearized AMR graph. Once we have the linearized AMR, we insert our syntactic tags into the AMR as metadata tags for the appropriate nodes, and train a family of models to generate text with each of these types of augmentations. For each mode of syntax, we additionally train a baseline "untagged" model without control tags for comparison.

4 Experiments

In our experiments, we compare three types of models: first, a baseline BART model finetuned on pure AMR-to-text generation with no control tags; second, BART models finetuned to produce text from linearized AMR with syntax tags for each mode of syntax individually; and finally, a set of models finetuned with control tags for multiple modes of syntax. For both voice and tense, we report results both on our own test set (10k sentences) as well as the test set of the proxy subset of the AMR Bank (approximately 800 sentences) for comparison on gold AMR parses; we do not report results on the proxy set for entity, as after filtering out sentences with :desc tags and sentences with greater or fewer than one person node, only 16 sentences remained.

4.1 Hyperparameter settings

We use Fairseq (Ott et al., 2019) for finetuning. Based on preliminary experimental results, we evaluate all models after four epochs. We train all models using the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 3×10^{-5} and polynomial learning rate decay. (For full hyperparameter settings, see Appendix A.)

4.2 Syntactic control

To investigate the effect of finetuning with our syntax tags, we automatically extract syntactic labels from each verb in the output from each model using our automatic labeling procedure. For each mode of syntax, we measure performance on generation with the corresponding labels as a classification task using macro F1. As person nodes may have both a :named tag and a pronoun tag attached, we measure entity realization performance on two separate tasks: whether our model generated a name or not, and whether our model used the right pronoun (if it used a pronoun at all). For tense and pronouns, we additionally provide breakdowns of F1 per class on our test set.

As we only measure F1 over the set of labeled nodes from the original sentence that are reproduced in the generated text, we additionally record the percentage of such nodes, i.e. the *node retention*, which effectively indicates the proportion of labels that are dropped from our evaluation because the corresponding nodes are not realized as the same lemma in the generated output. Across all models for all settings, node retention is upwards of .8 on our test set. We thus omit retention from our results tables.

Evaluating directly on the test set measures how well our finetuned models can reproduce the desired syntax in a setting where syntactic labels follow a similar distribution to the training set, as they are both drawn from the same base corpus. In order to isolate the effect that using control tags gives us, we also report performance in an off-distribution setting. For each mode of syntax, we create a “flipped" evaluation set by perturbing the control tags inserted into the evaluation inputs, which directs BART to generate less distributionally plausible voices. For voice, this simply entails flipping between active and passive. For tense, we flip between past and present, i.e., VBG and VBN are swapped, and VBD is swapped with VBP and VBZ. (VB is left unchanged.) For entities, we flip as follows: if the node had only a pronoun tag, we replace it with a random pronoun tag that differs from the original, and with 0.5 probability we add a dummy name node (drawn from a list of the top 100 most common unisex names in the United States[1]) and a `:named` tag. If the node originally had a `:named` tag, with 0.5 probability we remove it and add a random pronoun tag that differs from the one it had, if any.

4.3 Sentence quality

To measure fluency, we compute BLEU score (Papineni et al., 2002) of the generated text against the original sentence. While smatch (Cai and Knight, 2013) could be used to compare automatic AMR parses of the generated text against the input AMR, that would also inherently evaluate the AMR parser used, which is not our focus.

[1]https://github.com/fivethirtyeight/data/tree/master/unisex-names

5 Results

Model	Original F1	Original Flip	Proxy F1	Proxy Flip
Untagged	0.833	0.048	0.630	0.086
Voice	0.965	0.498	0.909	0.516
v + t	0.957	**0.500**	0.934	0.546
v + e	**0.972**	0.463	0.941	0.541
v + t + e	0.965	0.499	**0.979**	**0.581**

Table 3: F1 of finetuned BART variants for verb voice, in original and flipped settings, on our test set and the proxy test set. Components of combined models are abbreviated: voice (v), tense (t), entity (e).

5.1 Voice

We report performance of each model on the active/passive reproduction task in Table 3. The ‘flipped’ statistics (shown on the right) are on the flipped evaluation set, i.e. with all voice tags flipped, which forces the model to generate against the regular voice distribution.

The model finetuned with control tags performs noticeably better than the untagged model on the regular evaluation set, but its controllability truly shows through on the flipped set, where it outperforms the uncontrolled model by an order of magnitude. Though there is a sharp drop in performance from the original setting, it still manages to reproduce a nontrivial proportion of verbs in the specified voice even when the tags are flipped, indicating that it is indeed able to some extent to disregard whatever signal may be present in the raw AMR. We note that, although we naïvely flip all tags in the flipped setting, some verbs actually cannot appear in the passive (e.g., intransitive verbs such as “sleep"), which lowers the maximum possible score.

Interestingly, even the untagged model achieves impressive performance on the evaluation set, suggesting that the model does receive some signal as to verb voice. We hypothesize that it is picking up on the highly skewed verb distribution (see Appendix B), as the active voice is about an order of magnitude more prevalent in the data than the passive. However, its performance on the flipped evaluation set is trivially far worse, as the untagged

Model	Flipping	Macro F1	Proxy F1	VB	VBD	VBG	VBN	VBP	VBZ
Untagged	None	0.691	0.448	0.830	0.747	0.687	0.757	0.562	0.564
Tense	None	**0.979**	**0.961**	**0.990**	**0.981**	**0.994**	**0.979**	0.949	0.982
v + t	None	0.978	0.953	0.988	0.976	0.992	0.974	**0.953**	**0.986**
v + t + e	None	0.971	0.941	0.986	0.968	0.991	0.967	0.933	0.983
Untagged	Flipped	0.185	0.196	0.826	0.114	0.035	0.075	0.006	0.055
Tense	Flipped	0.784	0.806	**0.986**	**0.909**	**0.871**	**0.830**	0.143	0.964
v + t	Flipped	**0.786**	**0.899**	0.983	0.902	0.859	0.800	**0.207**	**0.966**
v + t + e	Flipped	0.760	0.736	0.981	0.891	0.818	0.736	0.173	0.959

Table 4: Performance of finetuned BART models for verb tense. F1 is reported on both our test set and the proxy test set; individual class F1 scores are on our test set.

Model	Flipping	Name F1	Pronoun F1	she	he	they
Untagged	None	0.399	0.720	0.473	0.782	0.906
Entity	None	**0.407**	0.996	0.993	**0.998**	**0.998**
v + e	None	0.395	0.995	0.992	0.997	0.996
v + t + e	None	0.403	**0.999**	**1.000**	**0.998**	**0.998**
Untagged	Flipped	0.401	0.204	0.083	0.283	0.245
Entity	Flipped	0.399	**0.980**	**0.986**	0.985	**0.970**
v + e	Flipped	0.426	0.976	0.978	**0.988**	0.961
v + t + e	Flipped	**0.433**	0.967	0.974	0.975	0.953

Table 5: Performance of finetuned BART models for entity realization. F1 is reported for the binary named - not named task as well as for the pronoun generation task. Numbers here are only on our test set, as there were only 16 sentences remaining in the proxy set after filtering.

model does not see the control tags at test time and produces exactly the same output either way.

We note that results on the proxy set are slightly lower in the original setting, but slightly higher in the flipped setting as compared to our original test set.

5.2 Tense

We report performance on tense in Table 4. We have a much more dramatic improvement for tense than for voice when comparing the model fine-tuned with tags to the model fine-tuned without; the untagged model does poorly even on the in-distribution evaluation set, whereas the tagged model does quite well on both. This may indicate that the distinctions between the multiple types of verb tense are more difficult for the model to learn without supervision than the active/passive distinction. One note is that the VBP class seems to be more difficult to accurately reproduce than the others, perhaps due to its relatively small size (approximately 5% of all verbs).

5.3 Entity realization

Finally, we report performance on entity realization in Table 5. Interestingly, it seems that names are quite difficult to learn - our scores simply measure whether the model generated a name or not, regardless of whether it was the correct name, and even in that case, F1 is quite low. The other interesting observation is that flipping does not seem to have a large effect either on names or on pronouns.

In the case of pronouns, this suggests that the model has learned to generalize across the different types of pronouns quite well - there is not a noticeable difference between performance across pronoun classes, even though there is a moderate imbalance in the distribution.

In the case of names, scores for some models actually go up slightly in the flipped setting. This may indicate that the model actually has a tendency to guess something closer to the randomized distribution of name information in the flipped evaluation sets.

Model	Syntax mode	BLEU
Untagged	Voice	0.679
Voice	Voice	0.688
Untagged	Tense	0.679
Tense	Tense	0.703
Untagged	Entity	0.670
Entity	Entity	0.707

Table 6: BLEU scores for generated outputs from base tagged and untagged models against original sentences.

5.4 Sentence quality

We report BLEU scores in the original (not-flipped) setting in Table 6. Adding tags in finetuning slightly improves BLEU score, suggesting that the additional signal is helpful to the model. At minimum, this indicates that we can finetune BART to use syntax control tags without having to worry about interfering with the content of its generated output.

6 Analysis

6.1 Syntax interactions

In order to investigate the interaction between the different modes of syntax, we additionally train a set of models that incorporate multiple types of tags. These are reported in the results tables as the "voice+tense", "voice+entity", and "voice+tense+entity" models.

Interestingly, adding tense seems to improve performance on voice, whereas the converse does not hold, while entity realization seems to be an orthogonal task: the "voice+tense" model achieves better performance on voice but not on tense in the more difficult flipped setting, whereas combining entity realization with other types of syntax leads to a drop in performance.

6.2 Qualitative analysis

We provide examples of generated output with voice tags in Table 7. The first example is a fairly straightforward case from the original evaluation set where the tagged voice model correctly generates the main verb ("lead") in the passive voice, whereas the untagged model incorrectly guesses that it should be active. In cases like these, where a verb is generated in an unusual voice, the untagged model seems to make its best guess based on what voice it usually sees the verb in, whereas the tagged model is still able to adjust its output based on the control tag.

The second example illustrates a different interesting phenomenon that we observed in some cases - here, the input is taken from the flipped evaluation set, and the untagged model generates the voice that would have been correct in the original setting (but is incorrect here). In this case, the tagged model correctly generates the main verb ("seen") in the passive tense, but it actually makes a semantic error in doing so, changing the shrine to the object rather than the subject of the seeing. In a sense, the model seems to be overcorrecting for the change in voice.

6.3 Structural ablation

We have now seen that we can successfully use tagged AMR as input to give us fine-grained controllability. However, it is still unclear exactly how much information from the AMR the model is using, or how much it is able to fill in on its own. In order to investigate precisely which parts of the AMR are necessary, we additionally train a series of ablation models for comparison on the voice control task by gradually removing components of the input AMR.

We train three ablated AMR-to-text models: a model where we remove relation tags (i.e., edge labels); a model where we remove relations and graph structure (i.e., parentheses); and a model where we remove relations, structure, and the syntax control tags themselves.

We present results on ablated models alongside the original tagged model in Table 8, and include the ablated models' content metrics in Table 9. Somewhat surprisingly, our first two ablations (removing edge labels and removing parentheses) both yield slight improvements in voice controllability. However, this comes at the expense of BLEU score, which has a drop of 2 points when edges are removed and a drop of 7 points when both edges and structure are removed, suggesting that removing edge and structural information somehow makes it easier for the models to focus on the correspondence between tags and syntax in the training data, but at the cost of information about content.

6.4 Voice tagging accuracy

As our voice labels are derived from automatic dependency parses, we check that our tagging method is giving us reasonable labels by evaluating it separately. We compare the voice tags from our tagging

Generator	Example sentence
Input	(lead *:passive* :ARG0 (this) :ARG0 (chief :ARG0-of (act *:active*)) :ARG1 (organization :name (Pentagon)) :frequency 3 :time (history))
Untagged	This is the third time in history that an acting chief has led the Pentagon.
Voice	This is the third time in history that the Pentagon **has been led** by an acting chief.
Input	(see *:passive* :ARG0 (shrine) :ARG1 (person :ARG0-of (visit) :source (religion :mod (all))) :duration (multiple :op1 (temporal-quantity :quant 1 :unit (century))))
Untagged	For centuries, the shrine has seen visitors from all religions.
Voice	The shrine **has been seen by** visitors from all religions for centuries.

Table 7: Example inputs and outputs from tagged and untagged models, with correct syntactic realizations in **bold** and incorrect underlined. Control tags in the input are *italicized*; these tags are not present in the version of the input passed to the untagged model.

Components removed	F1	Flipped F1
None (untagged)	0.838	0.047
None (voice tags)	0.953	0.431
Edges	0.964	0.438
Edges + structure	0.964	0.462
Edges + structure + tags	0.796	0.052

Table 8: Performance of full and ablated BART models on the analysis set for verb voice.

Ablation	BLEU
None (untagged)	0.717
None (voice tags)	0.731
Edges	0.713
Edges + structure	0.666
Edges + structure + tags	0.642

Table 9: BLEU scores for base and ablated models on the analysis set for voice.

method against the gold voice labels from two Universal Dependencies treebanks in English (GUM and LinEs) and present the results in Table 10. Both treebanks present a similar active/passive skew to our data. On both treebanks, performance on the majority active class is very high, whereas performance on passive verbs differs between the two: on GUM, our tagging method still picks up passive verbs quite well, whereas on LinEs, recall on the passive class is much lower. Given the GUM results, in our experiments, we assume that our tagging method is reliable enough to use as gold standard, but in future research, further work on picking up the missing passive instances from LinEs-like sentences may thus prove valuable.

Treebank	True voice	Prec.	Recall	F1
GUM	Passive	0.982	0.885	0.931
GUM	Active	0.993	0.936	0.964
LinEs	Passive	0.937	0.468	0.625
LinEs	Active	0.941	0.918	0.930

Table 10: Our automatic voice tagging on the development sets of Universal Dependencies treebanks. Precision, recall and F1 are evaluated against gold labels.

7 Conclusion and future directions

In this paper, we have investigated the controllability of three modes of syntax - verb voice, verb tense, and syntactic entity realization - when generating from augmented AMR inputs with BART. We find that all three modes of syntax can be more reliably reproduced when the model is given these augmentations, yielding more accurate and more faithful outputs. We find that even when we artificially engineer the distribution of tags to be as far from training as possible, the models with tags still far outperform the model without, with no drop in fluency. Ultimately, using a content plan augmented with syntactic tags allows us to control syntactic realization at a fine-grained level in the output. This is particularly useful in tasks such as summarization, where there is no one-to-one mapping between input and output content, and thus no appropriate place to insert fine-grained tags in the original input.

A natural future direction for this research would be an expansion of the setting from individual sentence AMRs to a collection of AMRs forming a contiguous passage or document, as is the ultimate goal of this work.

References

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, Online. Association for Computational Linguistics.

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.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 552–562, Hong Kong, China. Association for Computational Linguistics.

Kunal Chawla and Diyi Yang. 2020. Semi-supervised formality style transfer using language model discriminator and mutual information maximization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2340–2354, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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.

Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. Designing a symbolic intermediate representation for neural surface realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Farhood Farahnak, Laya Rafiee, Leila Kosseim, and Thomas Fevens. 2020. Surface realization using pretrained language models. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 57–63, Barcelona, Spain (Online). Association for Computational Linguistics.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Junxian He, Wojciech Kryściński, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2020. Ctrlsum: Towards generic controllable text summarization. *arXiv preprint arXiv:2012.04281*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Robert T. Kasper. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

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, and Nathan Schneider. 2020. Abstract meaning representation (AMR) annotation release 3.0. Web Download.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ani Nenkova, Advaith Siddharthan, and Kathleen McKeown. 2005. Automatically learning cognitive status for multi-document summarization of newswire. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 241–248, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. 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(140):1–67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

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.

Wenhao Wu, Wei Li, Xinyan Xiao, Jiachen Liu, Ziqiang Cao, Sujian Li, Hua Wu, and Haifeng Wang. 2021. BASS: Boosting abstractive summarization with unified semantic graph. 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 6052–6067, Online. Association for Computational Linguistics.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 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.

A Finetuning details

In our experiments, we use Fairseq to finetune from a pretrained bart-large model for four epochs using Adam; we use a learning rate of 3e-05, dropout of 0.1, and polynomial learning rate decay with 500 warmup updates and 2,000,000 total updates.

B Data Imbalance

Mode	Classes
Voice	Active (0.926) Passive (0.074)
Tense	VB (0.216) VBD (0.259) VBG (0.165) VBN (0.230) VBP (0.053) VBZ (0.078)
Entity	desc (0.770) named (0.109) pronoun-she (0.010) pronoun-he (0.041) pronoun-they (0.070)

Table 11: Class distributions for each syntax mode.