Controllable Active-Passive Voice Generation using Prefix Tuning

Valentin Knappich[1,2], **Timo Pierre Schrader**[1,2,3]
[1]Bosch Center for Artificial Intelligence, Renningen, Germany
[2]University of Stuttgart, Germany [3]University of Augsburg, Germany
valentin.knappich|timo.schrader@de.bosch.com

Abstract

The prompting paradigm is an uprising trend in the field of Natural Language Processing (NLP) that aims to learn tasks by finding appropriate prompts rather than fine-tuning the model weights. Such prompts can express an intention, e.g., they can instruct a language model to generate a summary of a given event. In this paper, we study how to influence ("control") the language generation process such that the outcome fulfills a requested linguistic property. More specifically, we look at controllable active-passive (AP) voice generation, i.e., we require the model to generate a sentence in the requested voice. We build upon the prefix tuning approach and introduce control tokens that are trained on controllable AP generation. We create an AP subset of the WebNLG dataset to fine-tune these control tokens. Among four different models, the one trained with a contrastive learning approach yields the best results in terms of AP accuracy ($\approx 95\%$) but at the cost of decreased performance on the original WebNLG task.

1 Introduction

Prompt-based learning is an uprising trend in Natural Language Processing. In contrast to the pre-train and fine-tune paradigm, prompt-based methods aim to adapt to new downstream tasks by finding or using new prompts that maximize the task performance. In that way, arbitrary tasks can be wrapped as language modeling tasks (Radford et al., 2019; Brown et al., 2020). This approach has the advantage that the pre-trained model parameters remain intact, and a single model can be used to solve many tasks by using the appropriate prompt without having to readjust all model parameters. However, it poses the challenge of finding a prompt with satisfactory performance. Manually finding such prompts can be tedious, and the results can sometimes be unintuitive (Gao et al., 2020). To

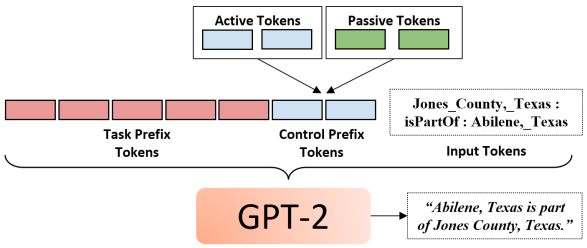

Figure 1: Our proposed method inputs task-specific prefix tokens and control tokens into GPT-2 in order to create a sentence from an input triple in the requested voice (active or passive).

that end, many techniques have been proposed to automatically find good discrete (Shin et al., 2020) or continuous prompts (Lester et al., 2021; Qin and Eisner, 2021; Li and Liang, 2021).

Our work investigates the use of continuous prompts, as proposed by Li and Liang (2021), for controllable generation. More specifically, we attempt to guide a generative model to create sentences that are formulated in either active or passive voice. For instance, the fact that "J.V. Jones" wrote the book "A Fortress of Grey Ice" can be formulated in either active voice (*The author of A Fortress of Grey Ice is J.V. Jones.*) or passive voice (*A Fortress of Grey Ice was written by J. V. Jones.*). We use the data-to-text dataset WebNLG (Gardent et al., 2017) as a benchmark and measure both task performance and the model's ability to generate outputs in the requested voice.

We use *task-specific prefix tokens* and additional *control prefix tokens* (cf. Figure 1), similar to Clive et al. (2021), and propose three different training objectives to train the additional control prefix tokens: implicit training, classifier guidance, and contrastive learning. We compare these three approaches and use the model without control prefix tokens as the baseline. Both implicit training and classifier guidance achieve only minor improvements in terms of control abilities but pre-

Proceedings of the Student Research Workshop associated with RANLP-2023,
pages 23–32, Varna, Bulgaria, Sep 4–6, 2023.
https://doi.org/10.26615/issn.2603-2821.2023_003

serve most of the task performance. In contrast, a contrastive learning approach achieves the highest active-passive (AP) voice generation accuracy of $\approx 95\%$. However, the semantic accuracy of the generated sentences is decreased compared to the other approaches, which is reflected in our qualitative and quantitative analyses.

The core contributions of this paper are as follows: first, we propose two new datasets that are derived from the WebNLG dataset called WebNLG-AP and WebNLG-AP-Pairs, which can be used to train on active/passive voice generation. Second, based on GPT-2 (Radford et al., 2019), we compare four training setups and assess their ability to control the voice while accurately performing the data-to-text task. Last, we make our dataset publicly available.[1]

2 Related Work

Controllable text generation is the task of generating natural language text while controlling secondary aspects of the output (Prabhumoye et al., 2020). These aspects can for instance be the sentiment, formality, persona, or content. Leng et al. (2020) control the length of generated sentences, as well as the sentence split, i.e., how many sentences are generated and how many facts each sentence contains. Zhang et al. (2022) propose a categorization of generative tasks involving control, including *Data to Text* and *Format Control*. Therefore, the WebNLG task can be seen as a control task where the control aspect is the information content, while our active-passive control falls into the category of *Format Control*.

Li and Liang (2021) propose *prefix tuning*, a prompt-based method that trains a set of continuous prompts that are prepended to the input. To increase the expressiveness of the method, a set of prefix tokens is not only learned for the input layer, but for every layer. This was later found to be beneficial for natural-language understanding (NLU) as well (Liu et al., 2021). Based on empirical results, the authors decide to reparameterize the prefix tokens using a multi-layer perceptron in order to stabilize training and improve performance. They find that prefix tuning outperforms other lightweight fine-tuning methods like adapters (Houlsby et al., 2019) in few-shot settings and provides competitive performance to regular fine-tuning in full data settings.

Clive et al. (2021) extend the prefix tuning method by adding input-dependent control tokens. For each of the domain categories, a set of control tokens is created and dynamically chosen based on the category. Contrary to our work, their main goal is to improve task performance and the control tokens are trained with a regular language modeling loss.

Split	Sample Counts	Active	Passive	Mixed
WebNLG				
Train	18,102	8,849	5,758	3,495
Dev	2,268	1,127	719	422
Test	4,928	2,546	1,485	897
WebNLG-AP				
Train	11,516	5,758	5,758	
Dev	1,438	719	719	
Test	2,970	1,485	1,485	

Table 1: The sample counts of the WebNLG and our newly created WebNLG-AP dataset for train, dev, and test split and their classification into active, passive, and mixed voice.

3 Modeling and Dataset Preparation

3.1 Task Description

We use the WebNLG dataset, which provides triples extracted from a knowledge base, e.g., {*Jones_County,_Texas : isPartOf : Abilene,_Texas*}. The task in this dataset is to generate a natural language sentence from a set of input triples. In the example above, the corresponding target sentence (also referred to as *lexicalization*) is "Abilene, Texas is part of Jones County, Texas." Counts for the different splits as well as the initial active/passive distribution are provided in Table 1. These triples are categorized into seven distinct topics. Five of them occur within the train and dev sets, namely "Airport, Astronaut, Building, Food, WrittenWork". The last two topics "Artist" and "Politician" only occur within the test set.

3.2 Active-Passive Dataset

To learn the control over active or passive voice, we construct a dataset, WebNLG-AP, based on WebNLG including voice annotations. We use a heuristic to identify active or passive voice in the samples of the original dataset. To annotate the

[1] https://github.com/ValeKnappich/WebNLG-AP-RANLP

complete WebNLG dataset, we use the dependency parser provided by spaCy (Honnibal et al., 2020) configured with the "en_core_web_trf" language model. We assume that a sentence is written in passive voice if one of the following dependency tags occurs within this sentence: { NSUBJPASS, AGENT, AUXPASS }[2], where NSUBJPASS refers to a nominal subject in a passive clause, AUXPASS denotes an auxiliary verb in passive and AGENT refers to the cause or initiator of an event. For example, in "The kid was stung by a bee," *kid* is the (passive) nominal subject (NSUBJPASS), *was* the passive auxiliary verb (AUXPASS) and *bee* the agent (AGENT). This implies that passive voice has precedence over active voice, i.e., if a sentence contains multiple verbs and one of them is tagged as passive, the entire sentence is treated as passive voice. Furthermore, if a sample consists of multiple sentences, we only conclude that a sample is in active or passive voice if all sentences that are part of the same lexicalization are classified into the same voice in our WebNLG-AP dataset. Mixed samples are not treated as active or passive and are thus filtered out. Moreover, we require that the dataset includes the same number of active and passive examples and thus truncate the set of samples for the predominant voice. The distribution across all splits in WebNLG-AP is provided in Table 1.

This active/passive detection pipeline is based on a heuristic and potentially creates noise in the dataset. Since this heuristic is used to create our dataset as well as to evaluate the results in Section 4, we verify that it works robustly. To that end, we randomly sample 100 sentences that have been tagged as either active, passive, or mixed and manually inspect the results. Out of these 100 random samples, all 25 mixed samples were classified correctly and only two out of 41 active samples were misclassified; one of these two because of a typographical error in the WebNLG dataset. Edge cases arise in the 34 passive samples. As already mentioned above, a sentence with multiple verbs is considered to be in passive voice if at least one passive voice signal occurs in it. Under this assumption, there are no misclassifications. Moreover, 12 out of 34 passive voice samples do not fall under this case and have been correctly tagged as well. This leads to a total tagging accuracy of 98%. Based on these observations, we conclude that the heuristic creates very little noise while allowing us to automate both dataset creation and model evaluation.

There are also verb phrases for which there is no analogue in the opposite voice. For instance, there is no passive voice for statements about someone dying, since "is died" is not a valid grammatical form. Furthermore, some passive voice samples are instances of the so-called *stative passive*. This term refers to verbs that are formulated in passive voice and that describe a potentially static condition, e.g., *Stuttgart is located in Germany* is a fact that is formulated in stative passive and does not change over time. We treat these cases as instances of passive voice.

We additionally introduce WebNLG-AP-Pairs, which, unlike WebNLG-AP, contains only triples for which WebNLG contains both active and passive lexicalizations. This is required for the contrastive learning approach described in Section 3.3.3. There are $1,858$ samples in the train split, 225 samples in the dev split, and 459 samples in the test split.

3.3 Computational Modeling

We use the OpenPrompt framework (Ding et al., 2021) for our modeling tasks, which provides an implementation of prefix tuning. We extend this implementation to additionally use control prefix tokens, as proposed by Clive et al. (2021).

In all training setups, we first train the task-specific prefix tokens for a GPT-2 model (Radford et al., 2019) on the WebNLG dataset and verify that the results are consistent with state-of-the-art publications. In a second fine-tuning stage, we add a set of *control prefix tokens* for every control label, i.e., one set of active tokens $\{c_{a_1}, c_{a_2}\}$ and one set of passive tokens $\{c_{p_1}, c_{p_2}\}$ that are abstract numerical vectors and therefore not human-interpretable. Depending on the desired voice of the output sentence, either the active or the passive tokens are added after the prefix tokens. For a sentence that is supposed to be in active voice, the input looks as follows:

$$input = [\underbrace{p_1, p_2, p_3, p_4, p_5}_{\text{Prefix Tokens}}, \overbrace{c_{a_1}, c_{a_2}}^{\text{AP Control Tokens}}, \mathbf{X}; \mathbf{Y}]$$

where $\mathbf{X}$ is the sequence of input triples formatted as strings and $\mathbf{Y}$ is the ground truth sentence (as for example presented in Section 3.1). Since the prefix tokens are vectors and $\mathbf{X}$ and $\mathbf{Y}$ are lists of tokens, the actual concatenation happens after computing

[2]Explanation can be found here: https://github.com/explosion/spaCy/blob/master/spacy/glossary.py

the key and value hidden states of the attention block for **X** and **Y**. This representation of the input is simplified in the way that trainable parameters are not only injected at the embedding layer but at every layer, as proposed by Li and Liang (2021). Following preliminary empirical results, we keep the task-specific prefix tokens frozen during the second fine-tuning stage. The LM weights remain frozen during the entire training.

To train the control prefix tokens, we propose three different training objectives: implicit training, classifier guidance, and contrastive learning.

3.3.1 Baseline: Implicit Training

In the baseline training procedure, we do not modify the training objective, i.e., we use cross entropy as the loss function:

$$l_{CE}(y, \hat{y}) = -\sum_i y_i \cdot \log(\hat{y}_i)$$

where y is the set of ground truth tokens and $\hat{y}$ is the set of logits. The ground truth labels are the indices of tokens in the vocabulary that are expected to be generated by the model.

The control prefix tokens are selected for each sample individually, depending on whether this sample is marked as active or passive. This approach is comparable to jointly training the model backbone with both voices and separately training two language modeling heads for the respective voices. Esentially, parts of the parameters are shared between voices to learn the task (model backbone or task-specific prefix tokens), while others are specific to the voice (separate language modeling heads or control prefix tokens). Therefore, we want the model to learn that the active control tokens are only trained on active samples and vice versa.

We find that this is not sufficient to accurately control the voice of the output sentence. Intuitively, the representations of semantically equivalent active and passive sentences are very similar. Therefore, the signal from this implicit training objective is too weak to properly guide the model. Details on the experiment results will be discussed in Section 4.2. Motivated by this finding, we propose two methods to provide a stronger, more explicit signal between the active and passive sentences.

3.3.2 Classifier Guidance

To provide a stronger signal about the voice, we first propose to use a frozen classifier on top of our model which is used to backpropagate whether the model generated the correct voice. This approach is comparable to Prabhumoye et al. (2018), who use a similar setup for style transfer. Unlike Dathathri et al. (2019), we use the classifier loss to train the tokens, rather than directly optimizing the hidden representations.

Features	Accuracy
last-token	87%
mean-3	99.18%
concat-3	99.86%
mean-5	99.86%
concat-5	100%

Table 2: Experimental results for different aggregation strategies of token embeddings for the classifier guidance approach. The *concat-5* strategy performs best.

The first step is to train the classifier. The main challenge is that we cannot train a classifier on the output text directly because beam search is not differentiable and would hence break the backpropagation. Therefore, we train it on the hidden representations of the GPT-2 model including the task-specific prefix tokens. Since GPT-2 is an autoregressive model, the hidden representation of the last token is dependent on all previous tokens and can thus be used as a sentence representation. However, we find that an aggregation of the last n token embeddings significantly improves performance. We experiment with $n \in \{1, 3, 5\}$. On top of that, we test mean and concatenation as aggregation methods and find that using the concatenation of the last 5 tokens works best. Table 2 shows how accurately the different approaches can classify active and passive voice given the respective hidden states of the model.

The classifier is an MLP with 2 linear layers, GELU activation function (Hendrycks and Gimpel, 2016), and sigmoid output. We proceed with the best classifier, attach it to GPT-2 as a second classification head and backpropagate an additional loss term:

$$L(y, \hat{y}, l, \hat{l}) = \frac{1}{w+1}(\overbrace{-\sum_i (y_i \log(\hat{y}_i))}^{\text{LM CE}} + w \underbrace{\cdot (l \cdot \log(\hat{l}) + (1-l) \cdot \log(1-\hat{l}))}_{\text{Classifier BCE}})$$

where l is the flag indicating whether the sentence is active or passive, $\hat{l}$ is the classifier output, and w

is a hyperparameter to balance the tradeoff between language modeling and classification. The total loss is normalized by $\frac{1}{w+1}$ such that high values for w don't change the magnitude of the gradients too much.

3.3.3 Contrastive Learning

We furthermore implement *Contrastive Learning* as a means to provide a direct gradient signal between active and passive voice. To that end, we use *Contrastive Cross Entropy (CCE)* as loss function:

$$CCE(y^+, y^-, \hat{y}) = \overbrace{\sum_{t \in y^+} -\log(\hat{y_t})}^{\text{Cross Entropy}} + \underbrace{\sum_{\substack{t' \in y^- \\ t' \notin y^+}} -w_c \cdot \log(1 - \hat{y_{t'}})}_{\text{Contrastive Term}}$$

where y^+ denotes the target sentence corresponding to the currently prompted voice and y^- to the respective contrastive sample. The contrastive term iterates over all tokens only occurring in the contrastive counterpart but not in the actual target sentence. The aim is to reduce the probability of these exclusive counterpart tokens for the currently requested voice by taking the respective logit values into account. For instance, consider an example where y^+ = [Apollo, 8, is, operated, by, NASA] and y^- = [The, Apollo, 8, operator, is, NASA]. In this case, the loss function would reduce the likelihood of the tokens "The" and "operator".

The weight w_c is a hyperparameter that influences the importance of the contrastive term. In our experiments, $w_c = 3$ has yielded the best results. Since this approach requires both alternatives in terms of active and passive voice, we use our WebNLG-AP-Pairs dataset that contains at least one active and one passive sample for each input triple.

3.4 Evaluation

To evaluate the performance of our models on the WebNLG task, we use the evaluation scripts provided by Nan et al. (2021) in their Github repository.[3] We use them to calculate BLEU scores (Papineni et al., 2002) on the three categories "seen, unseen," and "all", which refer to the different categories in the dataset and indicate whether the topic occurred during training or not (hence, "seen" or "unseen" by the model during training). We also evaluate with respect to the BLEURT score (Sellam et al., 2020) since we believe that NLG tasks are much more complex than scores such as BLEU are able to reflect.

We evaluate the voice control ability using the accuracy of the AP generation process:

$$acc(\mathbf{L}, \hat{\mathbf{L}}) = \frac{\sum_i \mathbb{1}_{l_i, \hat{l_i}}}{|\mathbf{L}|}, l_i \in \hat{\mathbf{L}}$$

where $\mathbf{L}$ denotes the set of ground truth AP labels, $\hat{\mathbf{L}}$ the AP labels of generated sentences (again determined by using our heuristics as described in Section 3.2) and $\mathbb{1}_{l_i, \hat{l_i}}$ is an indicator function evaluating to 1 if both labels are equal to each other for sample i.

Hyperparameter	Value
Training	
`epochs`	5
`lr`	5e−5
`eps` (AdamW)	1e−8
`n_prefix_tokens`	5
`n_control_tokens`	2×2
Decoding	
`num_beams`	5
`top_p`	0.9
`top_k`	0
`temperature`	1

Table 3: The hyperparameters used for training the GPT-2 model.

4 Experiments

4.1 Setup

As described in Section 3.3, we train our models in two stages. We first train 5 task-specific prefix tokens on the WebNLG dataset and afterward train 2×2 control tokens using one of the approaches introduced above. In each stage, the model is trained for 5 epochs with the hyperparameters listed in Table 3. Training a model in this setup takes approximately two hours on an Nvidia V100 GPU.

4.2 Results

Table 4 shows the results of our four models. The first row shows the scores of the model, which only uses the five WebNLG task-specific prefix tokens on top of the GPT-2 LM. It has therefore never

[3] https://github.com/Yale-LILY/dart

Configuration	WebNLG						WebNLG-AP	
	BLEU			BLEURT			BLEU	AP-Acc
	S	U	A	S	U	A		
WebNLG	**61.13**	43.29	**54.25**	**0.40**	0.25	**0.33**	42.12	75.69
WebNLG → implicit	59.22	45.79	53.16	0.38	**0.26**	0.32	41.78	76.36
WebNLG → classifier	59.16	**46.20**	53.32	0.39	**0.26**	0.32	**42.13**	78.08
WebNLG → contrastive	52.22	34.76	44.67	0.31	0.19	0.25	35.77	**95.22**

Table 4: Experimental results in terms of BLEU, BLEURT, and AP accuracy. S, U, and A correspond to the "seen, unseen," and "all" categories, respectively.

been specialized in the AP control task. The second model uses the two additional control tokens implicitly learned by exchanging them based on the sample that is currently presented. The third model uses classifier guidance, and the last one has been trained using the contrastive dataset and loss function.

The baseline model without AP control tokens achieves the best results on the original WebNLG task. The BLEU score is 54.25 and therefore almost consistent with the results reported by Li and Liang (2021) (55.1). When it comes to active-passive generation, the baseline model already yields an accuracy of 75.69%. That is, the model already scores above random without target voice information. We conclude that there are biases in the input that hint at the target voice in some cases. Some samples only sound natural in one of the voices. For instance, in the sentence "Juan Peron belongs to the Labour Party in Argentina.", *belongs* is an intransitive verb and hence does not have a passive counterpart.

The second row in the table shows the model with additional control tokens that have been trained implicitly. It almost fully preserves the performance on the original WebNLG task, with both scores decreasing by approximately 1% in absolute terms. The AP accuracy has increased by approximately 0.8% which means that the two implicitly trained AP control tokens provide the model with some guidance in terms of active and passive generation. However, the absolute gain is rather small compared to the additional training time. As a result, we draw the conclusion that the approach of just implicitly learning these AP control tokens is not sufficient to achieve meaningful results.

Consequently, the last two rows in the table show the results of our approaches that provide an explicit signal regarding the voice. The first is the one trained with classifier guidance. As with the implicitly trained one, original task performance is almost preserved. Moreover, there is an absolute gain in AP accuracy of around 2.5%, which is around three times as high as the gain of the approach before. We experiment with the hyperparameter w that controls the tradeoff between the language modeling loss and the classifier loss and report the results in Table 5.

	BLEU	BLEURT	AP-Acc
$w = 1$	52.93	**0.32**	77.27
$w = 5$	52.95	**0.32**	76.97
$w = 10$	53.12	**0.32**	76.70
$w = 999$	**53.32**	**0.32**	**78.08**

Table 5: Results for experiments on the tradeoff between language modeling and classifier loss. Higher values for w lead to a higher weight on the classifier loss.

The model trained with contrastive learning shows by far the highest increase in AP performance with an absolute gain of almost 20% compared to the baseline. On the other hand, both BLEU and BLEURT scores significantly decrease by around 10% and 8%, respectively. This is a non-negligible performance drop since preserving performance on the WebNLG task is an important aspect when it comes to evaluating the overall quality of the model. Having such a significant decrease indicates that this model either generates unnatural-sounding sentences or it does not fully reconstruct all information provided by the input triples. We provide further insights in the qualitative analysis presented in the next section.

5 Discussion

This section provides detailed insights into the results of our models by analyzing the generated

sentences. This helps to understand why the contrastive learning approach decreases original performance scores far more significantly than the other approaches. Our analysis provides guidance for possible improvements for future research.

5.1 Contrastive Learning

As already discussed in Section 4, the model trained using a contrastive loss function loses around 10% in terms of BLEU and 8% in terms of BLEURT on the WebNLG task. Especially the much lower BLEURT score is an indicator of the reduced semantic quality of the generated sentences. Table 6 lists four different comparisons between generated sentence and the corresponding target sentence.

The first example is an instance for which the model generated the <*pad*> token instead of a real token. It could reflect a problem that is introduced by the contrastive loss function as it strictly decreases the likelihood of some tokens being generated. As a result, the model might be confused here as there is no real token likely to be generated. The <*pad*> token is generated 86 times across 12 samples during evaluation on the full WebNLG dataset. Compared to an overall count of 1,862 test samples, we can conclude that this is indeed a smaller issue. Another issue that can be seen in this sample is that there are two distinct parts in the generated sentence that are not semantically connected at all. This is definitely reflected by both BLEU and BLEURT scores since there is not just a large difference to the target sentence, but there is also no real semantic meaning at all.

The second sample shows an instance where the same word is being generated over and over again until the maximum sequence length is reached (e.g., in this case, it is *California*). This of course also lowers the scores on the WebNLG task since there cannot be any matching n-grams for a repeating sequence of a specific word.

The third sample is an example of a generated sentence that sounds unusual and lacks some additional words. This is also an issue that occurs multiple times. The model trained with contrastive cross-entropy sometimes tends to generate unnatural-sounding sentences. A possible explanation for this behavior might be that the model focuses more strictly on generating the correct voice rather than on generating the most naturally sounding sentence. However, this is also a rare sample in which the voice does not match.

The last comparison in Table 6 shows that the contrastive model also has problems with leaving out specific information, which of course is also reflected by the BLEURT score in the end. On the one hand, the first part of the target sentence is indeed generated by the model in a semantically equivalent way. On the other hand, the second part of the target sentence is completely left out by the contrastive model. This, of course, implies a performance drop as the model tends to leave out relevant information that must be considered in evaluating the overall performance of this approach.

However, as the AP accuracy score already indicates, there are many samples on which the contrastive model performs really well. One example is the generated sentence *Antwerp International Airport is located in Antwerp, Belgium, where the German language is spoken.* The target sentence is marked as passive in the WebNLG-AP dataset and it gets clear that the model strictly followed the passive prompt when it generated this sentence. There are two semantic parts in it: One which states where the airport is located and one which describes the language spoken there. Both parts are formulated in the passive voice as requested. This makes clear that the contrastive model indeed has a strong sense of what is active and passive voice and is furthermore able to apply it during generation. Another positive example is the generated sentence *A.C. Chievo Verona is managed by Rolando Maran*. It has been correctly generated using the passive prompt. In contrast, the model with classifier guidance produced the active sentence *A.C. Chievo Verona's manager is Rolando Maran*, although it has been prompted with the passive flag. These are two of many examples in which the contrastive model strictly followed the prompt whereas the other models did not. Furthermore, these are well-formulated instances without any grammatical or other semantic issues. Hence, these are promising results to build upon.

To sum up the findings of the contrastive learning approach, there are a few issues that almost fully explain the decreased performance on the WebNLG task. First of all, the model sometimes seems to have problems with semantically unconnected parts within a generated sentence, i.e., it lacks a proper structure. Secondly, there are samples in which information is clearly missing or put into the wrong context. This is not a syntactic, but rather semantic

Ground Truth Lexicalization	Generated Lexicalization
"Juan Peron belongs to the Labour Party in Argentina."	*"Juan Perón is a member <pad> Juan Perón is the Labour Party."*
"Antioch is part of Contra Costa County in California [...]"	*"Antioch, California is part of Contra Costa County, California, California, California, California, [...]"*
"The novel "Castle" is written in English."	*"Castle is in the English language."*
"Aleksandr Chumakov was born in Moscow, Russia and he died in Russia. The leader of Moscow is Sergey Sobyanin."	*"Aleksandr Chumakov, born in Moscow, died in Russia."*

Table 6: Comparison between sentences generated using the model trained with contrastive loss and ground truth sentences. Four different error cases are shown.

issue since the model should also be able to generate correct facts. Finally, one issue is tokens being generated multiple times in a row. However, this does not happen very often. Still, this approach yields by far the best AP accuracy in our experiments. Section 6 discusses possible improvements for future research work.

5.2 Classifier Guidance

Classifier Guidance resulted in a small but significant improvement of the AP accuracy compared to implicit training. However, the results are in parts unintuitive to us. A larger increase in AP accuracy and a larger decrease in task performance would have been expected from this explicit signal about the voice. Instead, we observe a small improvement in AP accuracy and even slightly better task performance compared to implicit training (see Table 4). Furthermore, the results from experimenting with the loss weights in Table 5 elicit a very counterintuitive trend: increasing the weight of the classifier loss improves task performance and has no apparent correlation with the AP accuracy. Further investigations are necessary to better understand these results and make better use of this technique.

6 Conclusion and Outlook

We studied controllable active-passive generation using continuous prefix prompts on top of a generative language model. Our goal was to explicitly guide a generative model into generating sentences formulated in either active or passive voice. Three different approaches that we tested have shown different strengths and weaknesses. It becomes clear that there is a trade-off between the semantic quality of generated sentences and the strict enforcement of active or passive voice. The model which is trained using contrastive learning achieves the best results in terms of controllable AP generation. However, the quality of the generated sentences decreased, as reflected by BLEU and BLEURT scores, as well as a qualitative analysis of the generated sentences. On the contrary, the model trained using classifier guidance was able to maintain the task performance quite well but only improved voice control very slightly.

Future work should investigate how to maintain more of the task performance while achieving a high AP accuracy. The most apparent technique towards that goal would be to combine the contrastive learning and classifier guidance objective functions, given that their results were complementary: one is good at maintaining task performance while the other is good at controlling the voice. It could also be beneficial to include additional loss functions in the experiments, like less strict contrastive loss functions or directly optimizing the BLEURT score (Sellam et al., 2020; Shu et al., 2021). Furthermore, a promising direction for future research is instance-dependent prompt tuning (Tang et al., 2022; Jin et al., 2022), where the prefix tokens depend on the input rather than being static per task or control class.

In the broader context of applying prompt tuning methods to controllable text generation, future research should investigate whether our insights from studying active-passive voice control generalize to other control tasks such as sentiment or formality control. In particular, it would be interesting to see if control prefix tokens are composable (flexibly controlling multiple aspects at the same time) and transferable between domains and datasets (Su et al., 2021; Gu et al., 2021; Vu et al., 2021).

Limitations and Broader Impact

This work results from a student research project conducted in the summer of 2022 as part of graduate studies. A few months after project completion, new state-of-the-art models like ChatGPT and GPT-4 have been made publicly available. As a result, our work does not consider models that have been released past the summer of 2022. Therefore, this paper does not investigate the AP performance of these new models. This is subject to future research. We believe that our work still provides valuable contributions and insights into prefix and prompt tuning when relying on the more traditional fine-tuning paradigm, which itself is still a very commonly used technique for low-resource and domain-specific settings.

Acknowledgements

We would like to thank Esra Dönmez and Thang Vu from the IMS at the University of Stuttgart for their great supervision and always helpful input during this research project.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jordan Clive, Kris Cao, and Marek Rei. 2021. Control prefixes for text generation. *arXiv preprint arXiv:2110.08329*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 179–188. Association for Computational Linguistics.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Matthew Honnibal, Ines Montani, Sofie Van Ladeghem, and Adrian Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2022. Instance-aware prompt learning for language understanding and generation. *arXiv preprint arXiv:2201.07126*.

Yuanmin Leng, François Portet, Cyril Labbé, and Raheel Qader. 2020. Controllable neural natural language generation: comparison of state-of-the-art control strategies. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 34–39, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for 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 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

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.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. 2020. Exploring controllable text generation techniques. *arXiv preprint arXiv:2005.01822*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Raphael Shu, Kang Min Yoo, and Jung-Woo Ha. 2021. Reward optimization for neural machine translation with learned metrics. *arXiv preprint arXiv:2104.07541*.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, et al. 2021. On transferability of prompt tuning for natural language understanding. *arXiv preprint arXiv:2111.06719*.

Tianyi Tang, Junyi Li, and Wayne Xin Zhao. 2022. Context-tuning: Learning contextualized prompts for natural language generation. *arXiv preprint arXiv:2201.08670*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *arXiv preprint arXiv:2201.05337*.