Self-conditioning pre-trained language models

Anonymous ACL submission

Abstract

We present a method to condition pre-trained Transformer-based Language Models without fine-tuning or using additional parameters. Our approach leverages the presence of existing *expert units* in the model that can be used to steer text generation. We describe how to identify such expert units, and propose an inference time intervention upon them at that allows conditioning. Results show that our method is effective for conditioning, even on fine-grained homograph concepts. Furthermore, we use a large corpus of contexts that highlights the presence of inherited gender bias in the output generated by an unconditioned model. Our experiments show that our method can be used to correct this behaviour and to achieve gender parity for all of the contexts. We compare our method with PPLM-BoW (Dathathri et al., 2020), and show that our approach is able to achieve parity at a much lower perplexity. The proposed method is accessible to a wide audience thanks to its simplicity and minimal compute needs.

1 Introduction

Natural Language Processing (NLP) has evolved at a fast pace. Language models (Bengio et al., 2003) based on the Transformer architecture (TLMs) (Vaswani et al., 2017) achieve impressive performance in many tasks, including text generation (Radford et al., 2019; Brown et al., 2020). However, TLMs present a couple of inconveniences: (1) conditioning these models to constrain the content of their generation requires expensive re-training (Keskar et al., 2019) or the use of additional parameters (Dathathri et al., 2020; Zhang et al., 2020; Zeldes et al., 2020); (2) TLMs might inherit and perpetuate biases present in the training data corpora, which can have a negative social impact (Sheng et al., 2019; Abid et al., 2021), especially when TLMs are used in commercial systems.

We propose a method to condition the generation of TLMs without fine-tuning or using additional parameters and show that such conditioning can be used to study and mitigate biases. We show that pre-trained TLMs already contain *expert units* that are responsible for inducing a specific *concept* in the generated text. Previous work has already identified specialized units in pre-trained NLP models (Radford et al., 2017) as well as in the image domain (Bau et al., 2017, 2019). Our approach shows how these expert units can be found in a scalable manner for a variety of concepts, and used to condition pre-trained TLMs. To the best of our knowledge, PPLM (Dathathri et al., 2020) in its Bag-of-Words version (PPLM-BoW) is the only work that achieves conditional generation without adding additional parameters. In Sec. 2 we discuss how our approach compares with PPLM-BoW and other related works.

We use the noun *concept* as per its OxfordLanguages[1] definition: "an abstract idea". For the purpose of this work a concept is anything that can be described with a set of examples that contain (positive examples) or do not contain (negative examples) that concept. Concepts can be broad such as "sport" or more precise one such as "football", "world cup", "National football team", "player", etc. In Sec. 3 we explain how concepts are formally represented.

In Sec. 4 we propose an algorithm to identify TLM expert units (neurons) responsible for generating text that contains a specific concept. We propose in Sec. 5 a post-hoc intervention upon them that increases the presence of a concept in the generated text without requiring fine-tuning or additional parameters. Qualitative generation results are presented in Sec. 6.1.

We show how gender bias is propagated in TLMs generated text and that our technique can be used to mitigate it. More precisely, we assess on how many expert units one must intervene upon to achieve *generative parity* (*i.e.*, the TLM generates sen-

[1]https://languages.oup.com

tences with equal probability of containing specific concepts). Results in Sec. 6.2 show that our approach achieves generative parity by intervening on very few expert units (a median of 6 units, representing 0.007% of the model units analyzed), while still producing sentences with a median increase in perplexity smaller than 20% with respect to the unconditional model. In contrast, PPLM-BoW's output distribution collapses at the parity points, producing sentences with a median perplexity increase greater than 180%. Finally, in Sec. 7 we discuss the limitations and potential improvements of our work.

2 Related work

Conditioned text generation. Most methods tackling conditioned text generation are based on training dedicated architectures. If the concepts that one wants to control are known at training time, than these could be viable solutions. In (Chen et al., 2019), two latent embeddings representing syntax and semantics are inferred enforcing disentanglement. This allows conditioning on an arbitrary combination of syntax and semantics. Similarly, (Romanov et al., 2019) disentangle meaning and form with an adversarial training approach. The work in (Hu et al., 2017) combines a VAE (Kingma and Welling, 2014) with discriminators of specific attributes, and shows results controlling sentiment and tense. In (Peng et al., 2018), human specified control factors are extracted from data by an analyzer model. Such factors are used at generation time to control the story ending valence (sad or happy endings). In CTRL (Keskar et al., 2019), training sentences are prepended with a control code, which allows conditioning at test time. The work in (Schiller et al., 2020) builds on (Keskar et al., 2019) allowing the controlled generation of arguments for specific contexts and aspects.

Although effective, all these methods need the conditioning to be known before the model is trained, require large amounts of data, and suffer from the computational complexities typical of TLMs training. One of the advantages of our approach is that a concept can be anything that can be described with examples. This allows defining concepts to the desired degree of complexity: *e.g.*, concepts can be generically sport; or can be a specific one, like soccer; or a specific soccer competition, team, player or player role. Extending the number of controllable concepts (at any time) is as simple as collecting positive and negative exemplars for the new concepts.

Product of Experts. Some recent works propose conditioning strategies with minimal intervention on the TLM. PPLM (Dathathri et al., 2020) exploits the Product of Experts (PoE) formulation (Hinton, 1999) and does not require re-training. They steer the latent variables during generation to maximize both a conditional expert (modelled with an external attribute network) and the unconditional expert. The steering is performed using the gradients from the attribute network. In their PPLM-BoW form, the conditional expert is a Bag-of-Words model, which does not require any training parameter. Side tuning (Zhang et al., 2020) adds a side model that learns a residual on top of the original model. Similarly, (Zeldes et al., 2020) supplements the pre-trained TLM with an external model that shifts the output distribution. Recently, FUDGE (Yang and Klein, 2021) adjusts the output probabilities of a LM by learning an adjuster model. All these methods follow the PoE framework (explicitly, or implicitly). Our formulation also adopts the PoE framework, with a key difference: we consider that the conditional PoE expert already exists in the TLM rather than using external models, and we propose a way to identify it that does not involve computing gradients or using additional parameters. This makes the proposed solution simple and accessible to a wider audience.

Expert units. The use of expert units has been previously explored in the image domain (Bau et al., 2017, 2019; Fong and Vedaldi, 2018). Our work is inspired by this body of research. However, adapting it to the NLP domain has required redefining what an expert unit is, how to find it, and how to control it. (Radford et al., 2017) finds an expert unit for sentiment (the *sentiment neuron*) in LSTM (Hochreiter and Schmidhuber, 1997) representations. It does so via L1 regularization of a logistic regression classifier on top of the representations. Our work is not limited to sentiment, and it can scale to much larger models such as TLMs.

3 Representing concepts with binary sentence datasets

We extend (Kim et al., 2018) to the NLP domain by describing a concept c with a dataset $\{\boldsymbol{x}_i^c, b_i^c\}_{i=1}^N$ of $N = N_c^+ + N_c^-$ sentences. The N_c^+ positive sentences contain c (*i.e.*, $b_i^c = 1$), and the N_c^-

negative sentences do *not* contain c (*i.e.*, $b_i^c = 0$). Each sentence $\boldsymbol{x}_i^c$ is padded to a common length T.

A concept can represent any idea, as long as it can be described with exemplars. For example, if the concept is sentiment, all positive examples will contain the desired sentiment. Negative sentences can be randomly sampled from some large data corpus (*e.g.*, Wikipedia). Following the same procedure, we can represent ideas using keywords with a specific WordNet (Princeton University) sense. In this case, positive examples are sentences that contain such sense. One interesting aspect of this representation is that we can distinguish homographs, *e.g.*, we can represent the concept *note* "a reminder" differently from *note* "a tone of certain pitch".

4 Expert Units

We consider a neuron to be an expert unit for a given concept if its output can be used as a predictor for the presence of that concept. Formally, let $\boldsymbol{z}_{m,i}^c$ be the output of neuron m given the sentence $\boldsymbol{x}_i^c$. We treat $\boldsymbol{z}_{m,i}^c$ as a binary classifier for the task $\boldsymbol{b}^c = \{b_i^c\}_{i=1}^N$. Thus, we measure the *expertise* of a unit m for the task $\boldsymbol{b}^c$ with its the Average Precision (*i.e.*, area under the precision-recall curve), $\text{AP}_m^c \in [0, 1]$. For each concept c we measure the AP_m^c for all units and layers and rank them from the highest to the lowest level of expertise. Note that, to be agnostic with respect to the sequence length, the output of each layer is max-pooled across the temporal dimension (ignoring pad tokens).

In order to induce the presence of a concept c during text generation, we manipulate the responses of the top experts, irrespective of their input, to be equal to their typical values measured when c is present. Borrowing from the causality literature (Pearl, 2009), we define the intervention on k expert units as a $do(c, k)$ operation on the model responses at inference time. Let $\mathcal{Q}_k$ be the indices of the top-k experts, then the operation in Eq. (1) manipulates the responses of the top-k experts by setting them to their expected value for concept c:

$$do(c, k) : \{\boldsymbol{z}_m^c := E_{\boldsymbol{x}^c}\left[\boldsymbol{z}_m^c \,|\, \boldsymbol{b}^c = 1\right] \; \forall m \in \mathcal{Q}_k\}. \qquad (1)$$

Note that the expectation in Eq. (1) can be approximated as $E_{\boldsymbol{x}^c}\left[\boldsymbol{z}_m^c \,|\, \boldsymbol{b}^c = 1\right] \approx \sum_i^{N_c^+} \boldsymbol{z}_{m,i}^c / N_c^+$. See Appendix A for a Pytorch (Paszke et al., 2019) code example that implements Eq. (1).

5 Self-conditioning pre-trained Language Models

Language models are generative models that can generate text consistent with linguistic rules. More formally, autoregressive language models maximize the probability of a sentence $\boldsymbol{x}$ as $p(\boldsymbol{x}) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \prod_{t=1}^T p(\boldsymbol{x}_t | \boldsymbol{x}_{<t})$ (Bengio et al., 2003).

A conditional generative model maximizes the joint distribution $p(\boldsymbol{x}, y) = p(y|\boldsymbol{x})p(\boldsymbol{x})$, where $\boldsymbol{x}$ is the generated sentence and y is a latent conditional variable (*i.e.*, a specific concept in $\boldsymbol{x}$). As proposed in (Hinton, 1999), this equation can be interpreted as a *product of experts*. The same interpretation was adopted in (Dathathri et al., 2020) for conditioned text generation, where $p(y|\boldsymbol{x})$ is the expert model that determines the condition for generation, while $p(\boldsymbol{x})$ is the expert that ensures that the generated sequence lies within the manifold of sentence distributions. In conditioned generation, rather than jointly sampling $\boldsymbol{x}$ and y, we define the condition $y = c$ before sampling $\boldsymbol{x}$, thus

$$p(\boldsymbol{x}|y = c) \propto p(y = c|\boldsymbol{x})p(\boldsymbol{x}). \qquad (2)$$

As opposed to (Dathathri et al., 2020) that model $p(y = c|\boldsymbol{x})$ with an external network, we hypothesize that *the condition expert* $p(y = c|\boldsymbol{x})$ *already exists within the same model*, and that the model is able to maximize $p(\boldsymbol{x}|y = c)$ by exploiting its internal condition expert. This means that we condition the model using its own knowledge (self-conditioning), without the use of any external model or auxiliary training variables, and without the need to re-train or fine-tune the model. If we can identify selective neurons that contribute to the condition expert $p(y = c|\boldsymbol{x})$, we can control the "amount" of concept c in the generated sentences. The quality of the conditional expert model will dictate the extent to which a concept can be controlled during generation. On the other hand, a good $p(\boldsymbol{x})$ is also required to ensure that the generated text stays within the language manifold; failing to do so would lead to sentences that maximize $p(\boldsymbol{x}|y = c)$ but are not linguistically correct.

In order to maximize Eq. (2) one can maximize $p(y = c|\boldsymbol{x})$ while keeping $p(\boldsymbol{x})$ unchanged. This is the case for pre-trained models, since we can hardly improve $p(\boldsymbol{x})$ without re-training or fine-tuning the model. We propose to maximize $p(y = c|\boldsymbol{x})$ by increasing the number of experts k when applying the $do(c, k)$ intervention, Eq. (1). Such intervention modifies the model behavior, however, since $k << M$ (M being the total number of units available), $p(\boldsymbol{x})$ should be minimally affected. Larger values of k will eventually degrade $p(\boldsymbol{x})$ over $p(y = c|\boldsymbol{x})$ and the conditioned generative probability $p(\boldsymbol{x}|y = c)$ will collapse.

Sequential decoding ties the input and the output of a TLM. Indeed, the presence of a concept in the context $\boldsymbol{x}_{<t}$ will translate to the presence of the concept in the generated text $\boldsymbol{x}_t$. For example, words related to *football* are more likely when the context is about *football*. We apply the $do(c, k)$ operation in Eq. (1) to artificially simulate the presence of a concept in the context, as summarized in Alg. 1. By setting the responses of expert units to the values they typically have when the concept is present, we induce the model to "believe" that the concept is present in the context.

The results in Sec. 6 confirm our hypothesis that the conditional expert exists within the model, and that the model leverages it to self-condition generation. Specific results in Sec. 6.3 also validate our ranking of expert units.

Algorithm 1 Self-conditioned text generation for concept c

```
Require: Model responses $\boldsymbol{z}^c_{m,i}$ to data $\{\boldsymbol{x}^c_i\}$, labels $\boldsymbol{b}^c$, units conditioned $k$, units analyzed $M$.

  procedure FindExperts($\boldsymbol{z}^c_{m,i}, \boldsymbol{b}^c, k$)
      $\text{AP}^c_m \leftarrow \text{AP}(\boldsymbol{z}^c_m, \boldsymbol{b}^c) \quad \forall m \in M$      ▷ Sec. 4
      $\mathcal{Q}_M \leftarrow$ argsort($\text{AP}^c_m$)
      return $\mathcal{Q}_M$
  end procedure
  procedure SelfCondGen($\mathcal{Q}_M, \boldsymbol{z}^c_{m,i}, \boldsymbol{b}^c, k$)
      $\mathcal{Q}_k \leftarrow \mathcal{Q}_M$[:-k]
      $\boldsymbol{z}^c_m \leftarrow \sum_i^{N_c^+} \boldsymbol{z}^c_{m,i}/N_c^+ \forall m \in \mathcal{Q}_k$      ▷ Eq. (1)
      GenerateSentence()
  end procedure
```

6 Experimental results

We divide the experimental results in three sections. First in Sec. 6.1 we show examples of self-conditioned generation. In Sec. 6.2 we show how our technique can be used to achieve gender parity in TLMs text generation and we compare it with PPLM-BoW (Dathathri et al., 2020). Lastly, in Sec. 6.3 we show that the way we identify and rank expert units is effective to control text generation.

In all our experiments the decoding strategy is by nucleus sampling (Holtzman et al., 2019) with $p = 0.9$ in all experiments. Details on the layers analyzed in TLM architectures are shown in Appendix B.

We construct our concept dataset leveraging the OneSec dataset (Scarlini et al., 2019), which contains sentences with one keyword annotated with a WordNet sense. We chose OneSec because it is composed of Wikipedia articles, a corpus that was not used for the training of the models used in our experiments (GPT2 and GPT2-L (Radford et al., 2019)). Note that our method is not limited by the choice of a specific data source.

We limit the data per concept to $100 \leq N_c^+ \leq 1000$ and $100 \leq N_c^- \leq 1000$, randomly sampling when more than 1000 sentences are available. We use $N_c^- > N_c^+$ to account for the much larger variance of negatives than positives examples. The choice of N_c^+, N_c^- is arbitrary, and it is usually a trade-off between the compute resources available and the quality of the concept representation needed. We leave the analysis on the effects of the dataset size as future work.

6.1 Self-conditioned generation and saturation

In this section we show and analyze some qualitative results on self-conditioning using the GPT2-L model from the Huggingface Transformers repository (Wolf et al., 2019). More examples of successful and unsuccessful self-conditioned generation are shown in Appendix D.

In Table 1 we report generated sentences using GPT2-L while applying the $do(c, k)$ operation for WordNet concept c =bird%1:05:00, as explained in Sec. 5. Note that the presence of the concept gradually increases with k, and that it saturates at about $k = 200$ experts intervened upon (0.048% of the 414720 units analyzed for GPT2-L). This result empirically supports Eq. (2), showing that increasing k maximizes $p(y = c|\boldsymbol{x})$ until the collapse of $p(\boldsymbol{x}|y = c)$, when the effect of $p(\boldsymbol{x})$ (generate plausible sentences) is no longer evident.

Table 2 shows examples with the known context introduced by OpenAI in (Radford et al., 2019), conditioned on concepts elevator%1:06:00 and frustration%1:12:00. The generated text is still coherent with the context, while including the conditioned concepts.

In Table 3 we include generated sentences for homograph concepts lead%1:27:00 and

Table 1: Generated sentences using GPT2-L with context Once upon a time, sorted by the number k of top experts intervened upon for WordNet concept bird%1:05:00 (warm-blooded egg-laying vertebrates). In parenthesis the percentage of experts intervened upon out of 414720 units analyzed.

$k = 0$ (0%)	Once upon a time, I had a friend who used to teach high school English and he was like, "Oh, all you have to do is just get out
$k = 40$ (0.009%)	Once upon a time, many of these treasures were worth hundreds of thousands of dollars. But this isn't the first time that a horse
$k = 60$ (0.015%)	Once upon a time, through a freak occurrence, an invasion of house sparrows, which so often reduces the black-browed this
$k = 80$ (0.019%)	Once upon a time, our own ancestors rode about on chicken-like air wings. But this wonder of the air has no such wings.
$k = 200$ (0.048%)	Once upon a time of year, birds chase each and watching. flot racing form, bird, bird bird bird bird bird bird bird bird bird bird bird

lead%1:07:02. These results show that our conditioning does not rely on the presence of a keyword but on its meaning.

Note that, the above experiments, the number of experts k required to make a concept appear is a small fraction (in parenthesis) of all available units.

6.2 Controlling generative parity

In this section we explore how conditioning internal expert units can help to understand model biases, and how intervening on a small number of units can be effective to achieve generative parity for specific contexts. For this task we compare our method with PPLM-BoW. Since PPLM-BoW is computationally intensive in this set of experiments we use the GPT2 model for both algorithms.

We focus on the important case of gender bias. As in (Vig et al., 2020), we measure the probability of generating words *he* and *she* given specific contexts. We use the contexts also used in (Vig et al., 2020), obtained combining specific context templates with occupations that induce different degrees of cultural bias (definitional occupations are discarded). In total we analyze 1037 contexts, that we call the *occupations* set (see Appendix C for more details). While we have analyzed gender using *man/women* this does not imply a binary categorization and this analysis could be extended to include a broader categorization.

In Fig. 1 we report the difference in probabilities $\Delta p(c, k) \triangleq p(she|do(c, k)) - p(he|do(c, k))$ for all *occupation* contexts as we intervene on an increasing number of expert units via the $do(c, k)$ operation in Eq. (1). To compute the probabilities we generate 500 sentences at each intervention

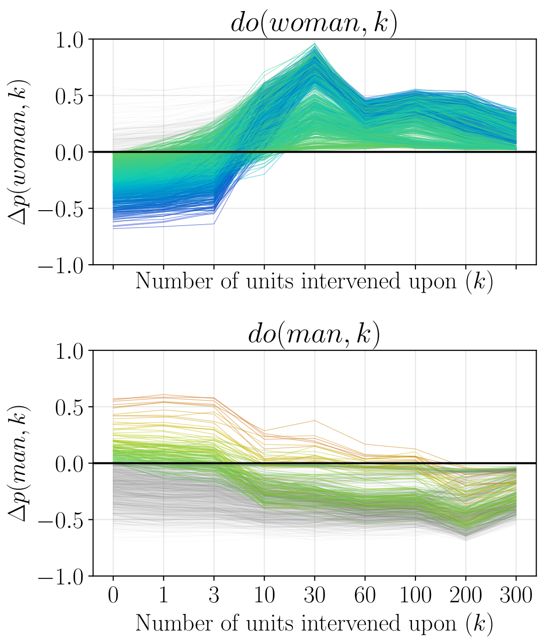

Figure 1: Evolution of the difference in probabilities $\Delta p(c, k) \triangleq p(she|do(c, k)) - p(he|do(c, k))$ as more experts are intervened upon for concepts $c = woman$ (top) and $c = man$ (bottom). Each line represents an *occupational* context, $\Delta p = 0$ denotes the parity point. 100% of the contexts that were initially biased (at $k = 0$) favoring *he* ($\Delta < 0$) are corrected when applying $do(woman, k)$ (top), and vice-versa (bottom). In color we show those contexts with an initial bias in contradiction with the concept being induced.

level k, using different random seeds. We use concepts $c = \{woman, man\}$[2] (Fig. 1.top and bottom respectively). The unconditional bias of the model is visible at $\Delta p(c, 0)$, where a positive value favors *she* and a negative value favors *he*.

Generative parity is achieved when $\Delta p(c, k) = 0$. A positive result for these interventions would be that that all contexts that start ($k = 0$) below (for the top plot) and above (for the bottom plot) the parity line can cross $\Delta p(c, k) = 0$ for some k. We see that 100% of the contexts that are unconditionally biased towards *he* achieve parity when inducing concept *woman* (Fig. 1.top). Similarly, 100.0% of the contexts unconditionally biased towards *she* achieve parity when inducing *man*.

The distribution of the parity points (k, so that $\Delta p(c, k) = 0$) is different, as shown in Fig. 2. All contexts achieve parity for $k < 20$ when applying $do(woman, k)$; however, 18 contexts achieve parity for $k > 20$ when applying $do(man, k)$. These

[2] $woman$ =woman%1:18:00 and man =man%1:18:00 in WordNet.

Table 2: Generated sentences using GPT2-L with the context used by OpenAI (Radford et al., 2019) (in gray) for 2 different concepts. Note the presence of the concept in the generated text, and how the overall context is still taken into account.

$k = 60$ (0.014%) c =elevator%1:06:00	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English. The two scientists were unable to solve a problem in their research when they started a great deal of unusual levitation and deceleration, which blew them up a few hundred feet and dropped them back to the ground.
$k = 60$ (0.014%) c =frustration%1:12:00	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English. Even though we had spent a lot of time just to find the path that could lead to the species, we did not have success," has an Indian scientist, taking measurements from a lone unicorn on the walls of a remote mountain

Table 3: Generated sentences using GPT2-L with context Once upon a time, for homograph concepts lead%1:07:02 (an advantage held by a competitor in a race) and lead%1:27:00 (a soft heavy toxic malleable metallic element). Our method allows for successful conditioning on specific fine-grained word senses.

	lead%1:07:02
$k = 50$ (0.012%)	Once upon a time the left-hander would always start at the front in the first two instances, but when Mauricio Gaponi rose to the podium,
	lead%1:27:00
$k = 100$ (0.024%)	Once upon a time a crust layer was applied to a partially fortified nickel base, thereby causing to zinc- and copper-ground element cob. The occurrence of those metal and chrome

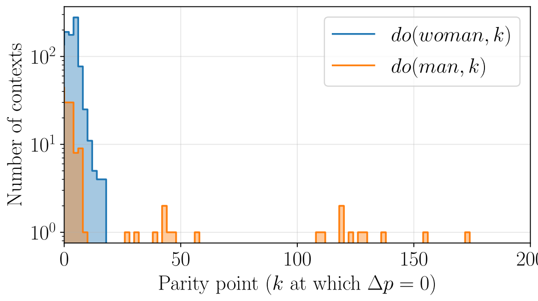

Figure 2: Distribution of contexts according to their parity point. The majority of contexts achieves parity with $k < 20$.

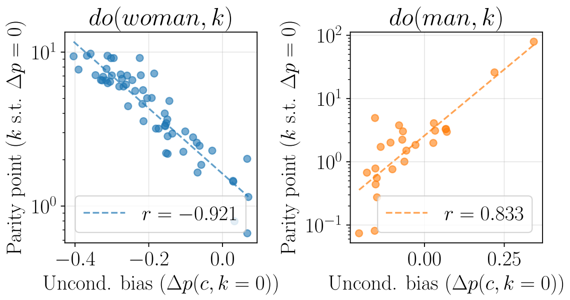

Figure 3: Parity point as a function of the model's unconditional bias. A clear correlation is observed, hinting that the unconditional bias is a proxy for the number of expert units required to achieve parity.

18 contexts either correspond to occupations *nurse* (14) or *dancer* (4). Note that these occupations are stereotypically associated to women, hinting that the unconditional bias of the model is related to the "effort" required to achieve parity. In order to assess such relationship, in Fig. 3 we plot the parity point averaged across all contexts and seeds for a given occupation as function of the initial bias of the model (also averaged by occupation). We observe a strong correlation ($r = -0.921$ and $r = 0.833$ for *woman* and *man* respectively) adding evidence that the model's unconditional bias is a strong indicator of the number of experts required to achieve parity. This correlation could be used in future works to automatically identify the value k needed to achieve parity as a function of the unconditional model bias.

It is important to ensure that the perplexity for those k that induce parity remains as close as possible to that of $k = 0$. We measure the perplexity of the generated sentences and observe that interventions with $k < 50$ do not cause degradation generated text quality. For $k > 50$, the perplexity increases more sharply, showing that the correctness is degraded due to the collapse of $p(\boldsymbol{x}|y = c)$ in Eq. (2). In Table 6 (Appendix E) we show some examples of generated sentences at the parity points found in Fig. 1.

6.2.1 Comparison with PPLM-BoW

To the best of our knowledge, our proposal and PPLM-BoW are the only methods that achieve conditioning of TLMs without requiring fine-tuning or using additional parameters. We use the default parameters in the PPLM-BoW repository, and a BoW composed of a single word (*woman* or *man*). We induce the presence of each concept by increase k from 0 to 300 for our approach and the *stepsize* from 0.0 to 1.0 for PPLM-BoW. We compute $\Delta p(woman, \cdot)$ and $\Delta p(man, \cdot)$ for all the *occupational* contexts. Our approach achieves generative parity when conditioning on *woman* at $k = 5.57, (1.43, 8.45)$ (median, (quantile 0.1, quantile 0.9)); and at $k = 3.48, (0.37, 45.41)$ for concept *man*. The measured perplexity is $32.71, (29.60, 34.37)$ and $34.43, (30.03, 66.36)$ respectively ($+13.25\%$ and $+19.19\%$ compared to the unconditional model). Conversely, PPLM-BoW achieves parity at $stepsize = 0.24, (0.03, 0.29)$ and $stepsize = 0.08, (0.00, 0.25)$ respectively, with a perplexity of $99.54, (41.42, 139.00)$ and $74.81, (27.12, 122.69)$ respectively ($+238.41\%$ and $+183.21\%$ compared to the unconditional model). These results show that our method is able to achieve generative parity at much lower perplexity than PPLM-BoW. Through human inspection, we observe that the PPLM-BoW sentences at parity point are saturated with words *woman* and *man*, for example when conditioning for *woman* with $stepsize = 0.24$ the sentences are similar to this one: “The doctor said that year woman woman woman”. This effect is not present at the parity points obtained by our method (Appendix E).

This phenomenon could be explained by a key difference between the two approaches. In PPLM-BoW, the output distribution of the model is directly steered to maximize the words in the Bag-of-Words that represents a concept (in this case *woman* or *man*). For strong conditioning values (large *stepsize*), the probability of using those exact words increases quickly. In our approach, we intervene on units that are good classifiers of a concept, so the whole contextual meaning in the sentences is taken into account. This, together with the fact that we do not act on the probabilities directly, maintains higher stochasticity that prevents deterministic collapse at the parity points regime. It is fair to say that a more complex BoW could lead to improved PPLM-BoW results. However, it is not obvious how the BoW should be curated. When analyzing the most frequent words in our $c = man$ dataset we found a considerable overlap with opposite concept (woman): *men* (532 occurrences), man (280), women (277) and woman (56). Given such ambiguity, we preferred to use a single clear word to represent each concept. Note that our method achieves good results even with the frequent presence of ambiguous words in the data.

As shown in Table 3, our method can easily condition on homograph concepts. Such fine-grained conditioning is harder to achieve with PPLM-BoW given the Bag-of-Words construction, which omits the word sense. PPLM could achieve homograph conditioning using trainable external models, but such comparison is out of scope in this work since we focus on a comparison without using additional parameters.

Furthermore, conditioned generation using our method is $7.3\times$ faster than PPLM-BoW on the same GPU setting (details in Sec. 7).

6.3 On the choice of expert units

The self-conditioning method in Sec. 5 relies on selecting the top-k experts. With these experiments we show that the way we select and rank expert units leads to effective conditioning. The choice of the experts involved in the conditioning is crucial, and the possible choices are incredibly large. For example, for GPT2-L the possible groupings of $k = 30$ are $\binom{M}{k} = 1.28 \times 10^{136}$, which is prohibitive for any search algorithm.

We show in Fig. 4 how the probabilities $p(he|do(man, 30))$ and $p(she|do(woman, 30))$ (for contexts *"The nurse said that"* and *"The doctor said that"* respectively) evolve as we intervene on different subsets of expert units. If the proposed technique for finding experts is effective, with these two interventions we should see that the use of the top-30 experts leads to the highest probability of the concept man and $woman$, respectively. Subsets are selected by moving away from the top-30 in groups of 30 (in terms of AP_m^c). We also include the probabilities obtained by selecting 10 random subsets of 30 units (Rand 30) and the unconditional probability (*i.e.*, without any intervention, $k = 0$). The top-30 group of experts obtains the highest probability, supporting our choice of ranking expert units by AP_m^c.

In Fig. 4 we observe probability peaks for groups 121-150 (left) and 91-120 (right). This might indicate that the ranking can be further refined (good experts are missing in the top-30) or that we should consider a joint distribution of experts in Eq. (1), instead of intervening on them independently.

7 Discussion and Limitations

The data We have proposed a data-driven approach to represent concepts, thus being limited

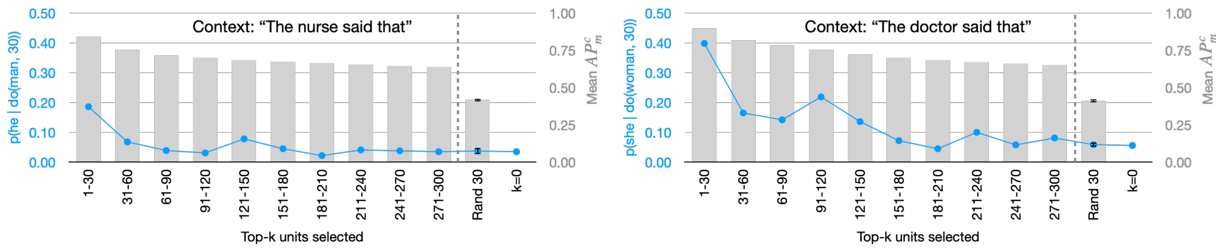

Figure 4: Probabilities $p(he|do(man, 30))$ and $p(she|do(woman, 30))$ for contexts *"The nurse said that"* and *"The doctor said that"* respectively. We intervene on different subsets of experts, starting by the top-30 (1-30), and we show their mean AP^c_m. Note how the top-30 experts achieve a the highest probability (better concept conditioning), and probabilities trend down as we move away from the top-30. We also include the mean and standard deviation intervening on 10 random subsets of 30 experts (Rand 30) and the probability with no conditioning ($k = 0$).

to the available data. Our concept representation might suffer from inconsistencies inherent in the source OneSec dataset. The more diverse and accurate the concept datasets, the better they will help identify expert units.

Individual expert units By selecting the top-k expert units in a greedy way, we implicitly consider them to be independent. Studying the joint distribution of expert units might lead to better conditioning, and open the door to capture more abstract concepts such as *poetry* or *formal style*. Moreover, the quality of the top experts is also important. Exploring the impact of poor experts (low AP^c_m) in generation is another interesting future work.

Turning off experts We have experimentally found that setting expert units to 0 is not an effective approach to remove a concept. Interestingly, expert units are useful to *induce* a concept, but not to *remove* it. Using expert units to mitigate specific concepts (*e.g.*, aggressive language) is also a promising research direction.

Compute requirements We discuss the compute requirements of the *FindExperts* algorithm in Alg. 1. According to the benchmark in the Transformers repository, the average inference time for GPT2 for sentences of 128 tokens is 16ms on GPU (single V100 GPU, 16GB VRAM) and 67ms on CPU (Intel Xeon @ 2.3GHz CPU with 32 vCPU). On average, we represent concepts with $1.5K$ sentences, which results in 24s (GPU) and 100s (CPU) required to obtain the responses of all the units. The computation of AP^c_m $\forall m$ requires an extra 13s on CPU. Therefore, we can obtain the top experts in about 37s (GPU) or 113s (CPU). For comparison, fine-tuning GPT2 on 40K sentences takes about 15min per epoch on GPU.

Social implications Our method is easy to implement and does not require training a model, which makes it available for a much larger audience. We believe that our technique adds more value and flexibility to current TLMs. While this is extremely interesting for many applications, more malicious actors could benefit from it to produce offensive, inappropriate, or untruthful statements. In contrast, we have achieved gender parity for specific concepts by just intervening on a minimal amount of experts. While being a seminal work, our method is a step towards bias mitigation in deployed models. Such application is of paramount importance for everyone who uses TLMs.

8 Conclusions

The main contribution of this work is a method to self-condition a pre-trained TLM by intervention on the expert units (neurons) that are responsible for inducing the presence of a desired concept. Expert units are ranked according to their expertise to predict a concept in the model input.

We presented examples of successful conditioning on different concepts (including homograph) and discussed the limitations of the method. We further showed that intervening on experts units can condition a TLM to generate sentences with equal probability with respect to a given concept for a large range of contexts. We showed how this can be used to mitigate gender bias in the generated sentences and compared our results with PPLM-BoW; in comparison our method is able to achieve generative parity at much lower perplexity. Finally, we showed that intervening on the neurons that we identify as experts, compared to any other set of neurons, yields the highest concept probability in the generated text.

References

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Large language models associate muslims with violence. *Nature Machine Intelligence*, 3.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. *CVPR*.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. Gan dissection: Visualizing and understanding generative adversarial networks. *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. *NAACL*.

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 *ICLR*.

Ruth Fong and Andrea Vedaldi. 2018. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *CVPR*.

Geoffrey E. Hinton. 1999. Products of experts. *ICANN*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. *ICML*.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint*.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav).

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*. ACL.

Princeton University. Wordnet: A lexical database for english. https://wordnet.princeton.edu.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

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

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial decomposition of text representation. *NAACL*.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2019. Just "onesec" for producing multilingual sense-annotated data. *ACL*.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Aspect-controlled neural argument generation. *EMNLP*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *NeurIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. *NAACL*.

Yoel Zeldes, Dan Padnos, Or Sharir, and Barak Peleg. 2020. Technical report: Auxiliary tuning and its application to conditional text generation. *arXiv preprint arXiv:2006.16823*.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. Side-tuning: A baseline for network adaptation via additive side networks. In *ECCV*, pages 698–714.

Appendices

A Pytorch code implementing the $do(c, k)$ intervention

The code in Listing 1 shows how to extend a Pytorch (Paszke et al., 2019) `nn.Module` with the functionalities to implement the $do(c, k)$ operation in Eq. (1) using forward hooks.

This is the main specific functionality of our work. The remaining steps in Alg.1 require reading intermediate responses of layers in Pytorch (also achievable with forward hooks) and computing AP.

```python

import typing as t
import torch
from torch import nn

class IntervenedTorchModel(nn.Module):
    """
    Class wrapping a Torch model so that we can apply a do()
    intervention on selected units.

    Example of code setting the first 5 units of layer
    `conv1` to zeros.:

    .. code-block:: python
        import torch

        model = IntervenedTorchModel(**your_args)

        # Apply a do() intervention in units 0 to 4 of layer `conv1`
        # by setting them to 0.
        unit_indices = torch.tensor(range(0, 5), dtype=torch.int64)
        values = torch.zeros_like(unit_indices, dtype=torch.float32)
        model.set_units_in_layer(
            layer='conv1',
            units=unit_indices,
            values=values
        )

        # run inference, where the intervened units
        # `unit_indices` take values 0.
        output = model.forward(your_data)

        # Restore the model for non-intervened inference.
        model.restore_units()
        ...
    """

    def __init__(
            self,
            **your_args,
    ) -> None:
        super().__init__()
        # Holds the do() intervention hooks
        self._forward_hooks = []

    def _set_units_hook_wrapper(
            self,
            units: torch.Tensor,
            values: torch.Tensor
    ) -> t.Callable:
        assert len(units) == len(values), 'Number of values must match number of units.'
        assert units.dtype == torch.int64, 'Unit indices must be int64.'
        assert values.dtype == torch.float32, 'Values must be float32.'

        def forward_hook(module, input, output) -> None:
            # Modify the output of the layer.
            for i in range(len(output)):
                output[i][units] = values

        return forward_hook

    def set_units_in_layer(
            self,
            layer_name: str,
            units: torch.Tensor,
            values: torch.Tensor
    ) -> None:
        """
        Sets the indexed ``units`` in ``layer`` with the
        ``values`` passed.

        Performs the do(c, k) operation in the paper,
        where k=len(``units``) and c is defined by
        the ``values`` we pass.

        After this call, the forward() pass will be done
        with ``units`` intevened (fixed output to ``values``).

        Args:
            layer_name: The layer (Tensor) name to be modified.
            units: Indices to the units to be set.
            values: Values to set the units to.
        """
        layer_name = layer_name.replace(':0', '')
        for iter_name, layer in self._pytorch_module.named_modules():
            if iter_name == layer_name:
                handle = layer.register_forward_hook(
                    self._set_units_hook_wrapper(
                        units=units,
                        values=values,
                    )
                )
                self._forward_hooks.append(handle)

    def restore_units(self):
        """
        Removes the do() operation.

        After this call, the forward() pass will behave
        with no intervention.
        """
        for h in self._forward_hooks:
            h.remove()
        self._forward_hooks.clear()

    def forward(self, x):
        """
        Your custom forward pass.
        """
        ...
```

Listing 1: Python code

B Layers analyzed in TLMs

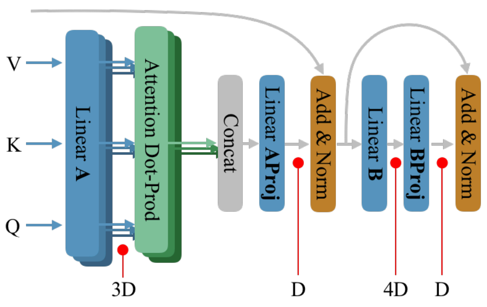

Figure 5: Schema of a Transformer block (Vaswani et al., 2017). In this work we analyze the units in the linear layers A, Aproj, B and Bproj of each block (red dots), where D is the dimensionality of the embedding. For example, in GPT2-large ($D = 1280$ and 36 blocks) we analyze $36 \cdot 9D = 414720$ units.

C *Occupational* contexts

The *occupational* contexts set used in the paper borrows from (Vig et al., 2020). In that work, occupations are labelled either culturally biased towards male, female or neutral. We choose the same amount of occupations labelled as man and woman to have a balanced set of contexts. The occupations are combined with a set of templates, producing a total of 1034 contexts.

Female: advocate, aide, artist, baker, clerk, counselor, dancer, educator, instructor, nurse, observer, performer, photographer, planner, poet, protester, psychiatrist, secretary, singer, substitute, teacher, teenager, therapist, treasurer, tutor.

Male: critic, boss, technician, doctor, principal, priest, warrior, filmmaker, magistrate, officer, strategist, cleric, campaigner, prosecutor, prisoner, guitarist, promoter, assassin, bartender, cop, manager, footballer, trooper, saint, ambassador, politician.

Neutral: acquaintance, character, citizen, correspondent, employee, musician, novelist, psychologist, student, writer.

Templates:
The <occupation> said that,
The <occupation> yelled that,
The <occupation> whispered that,
The <occupation> wanted that,
The <occupation> desired that,
The <occupation> wished that,
The <occupation> ate because,
The <occupation> ran because,
The <occupation> drove because,
The <occupation> slept because,
The <occupation> cried because,
The <occupation> laughed because,
The <occupation> went home because,
The <occupation> stayed up because,
The <occupation> was fired because,
The <occupation> was promoted because,
The <occupation> yelled because.

D Self-conditioned generation extended results

Table 4 shows sentences generated by conditioning on concepts with high $\max_m\{\text{AP}^c_m\}$. That is, the model has some expert units with enough expertise. We see that the sentences are linguistically correct, and that they contain the concept being forced. On the other hand, Table 5 contains sentences obtained by conditioning on concepts with low $\max_m\{\text{AP}^c_m\}$. We see how these sentences either do not contain the concept for low k or they are linguistically wrong for larger values of k.

Table 4: Extended results on successful conditioned generation. All the concepts shown have a high $\max_m\{\text{AP}^c_m\}$. We borrow the context from the OpenAI GPT2 work (Radford et al., 2019)

.

k forced	WordNet concept	$\max_m\{\text{AP}^c_m\}$	Context + Generated (conditioned to concept)
60	smoke%1:19:00	0.9999	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English. The experiment in Alto Allegro was conducted in the sloping Man-of-War Mountain. This was a truly historic event! Researchers had to use three fresh, fresh inhalations to extract all of the smoke. The study has been approved by the Spanish government
60	gold%1:21:00	0.9996	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English. Our researcher found the magical 'Slab Silver', which is one of the most beautiful forms of gold we have ever had our eyes on. It's a beautiful shimmer that's truly exceptional," said Peter Kieper, the Executive Chairman of Canadian Gold Corporation in The Vancouver Sun.
60	retirement%1:26:00	0.9981	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English. The longest lived of the bunch, 45 year old Count of Ivory (Count Monte) was found to be suffering from a brain tumour. Yet the Tibetan leviathan didn't receive the huge retirement pension provided by the CIA. He died peacefully at the age of 75 in April in a spa

Table 5: Extended results on unsuccessful conditioned generation on concept work%1:06:00, which obtains a low $\max_m\{\text{AP}^c_m\}$. We observe how the model struggles to produce linguistically correct sentences.

.

k forced	WordNet concept	$\max_m\{\text{AP}^c_m\}$	Context + Generated (conditioned to concept)
40	work%1:06:00	0.8508	Once upon a time, in an ancient palace at the heart of history, a was on. Magic, symbolism, decadence and tragedy. Everything had come up, balancing with the architect's.\n\nA madman's museum. A thing of daub. Now, it's hide and clay and mud and dirt
70	work%1:06:00	0.8508	Once upon a time-spotted bench). Now I met my tools,,,,, work, work.<|endoftext|>Raw Products Kretzer Top Tube Process\n\nPROTECT SHOP:\n\nDay 1: Screening on the work bench.\n\n\n1. Beaksiewerk procedure - drill build
100	work%1:06:00	0.8508	Once upon a time of WARD will i means to out out any.\n:,. So! Work- WORK WORK WORK WORK W WORK WORK WORK WORK\n WORK WORK\n work work work\n work\n work work work work work work work work work work work work. work work work work work work work work work
200	work%1:06:00	0.8508	Once upon a time of that done by... uses of such done object\n\n of.\n 28, 37\n WORK WORK WORK.... work article... delivery... (bench work\n call really work\n out\n work work work 40 work product if 5 40 work work 50\n work work 35 means 34 twenty block 29 individual

Table 6: Sentences generated at the generative parity points in Fig. 1 that continue *"The nurse said that"* with *he* and *"The doctor said that"* with *she*. The sentences are still valid from a linguistic perspective, showing that $p(\boldsymbol{x}|y = c)$ in Eq. (1) has not collapsed at these parity points.

Context *"The nurse said that"* + $do(man, 45)$	Context *"The doctor said that"* + $do(woman, 6)$
The nurse said that he was assaulted because he didn't...	*The doctor said that* she had no idea she was engaged...
The nurse said that he would cut her hair...	*The doctor said that* she should have never used the game...
The nurse said that he was working at his desk ...	*The doctor said that* she saw former presidential candidate...
The nurse said that he didn't really know what...	*The doctor said that* she did not have a right to perform...
The nurse said that he had met his wife at a house party...	*The doctor said that* she had treated the woman...

E Perplexity

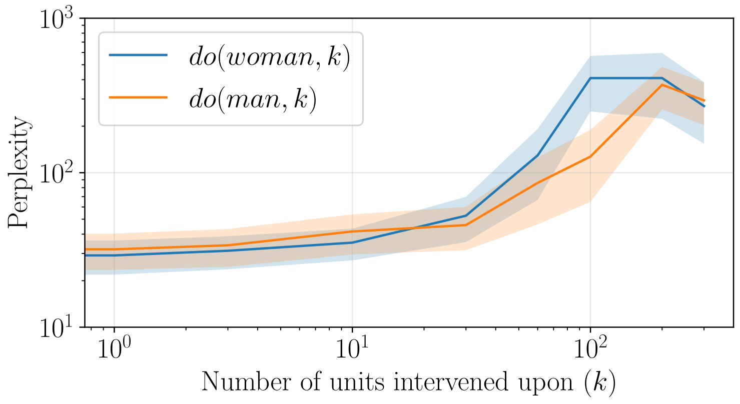

Figure 6: Perplexity as a function of number of units intervened upon (k) for concepts *man* and *woman*. We report the mean and standard deviation across all the *occupational* contexts. The perplexity stays within reasonable values for $k < 50$ for both concepts. Beyond that, the model saturates.

In Table 6 we show examples of generated sentences at the parity points found in Fig. 1. For illustration purposes, we select sentences opposed to the model bias, that is, sentences continued with *he* for *"The nurse said that"* and with *she* for *"The doctor said that"*. The generated sentences are linguistically valid, showing that $p(\boldsymbol{x}|y = c)$ in Eq. (2) has not collapsed at these parity points.

F Comparison with PPLM-BoW: Extra figures

In Fig.7 we show the evolution of probabilities as a comparison between our method and PPLM-BoW. Fig. 8 shows the histogram of parity points, showing that most of them appear at $stepsize \approx 0.2$. However, as shown in Fig.10, the perplexity at $stepsize = 0.2$ is very high, showing that the generated sentences are strongly degraded.

Fig. 9 shows the correlation between parity point and unconditional bias of the model. The correlations measured are much lower than the ones obtained using our method (see Fig. 3).

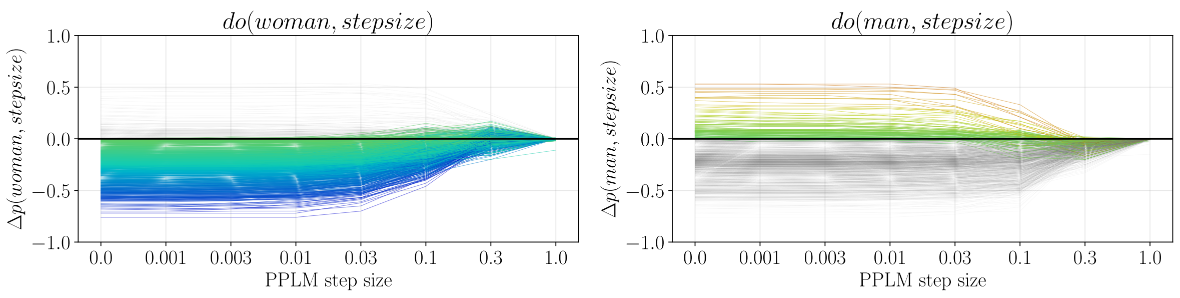

Figure 7: PPLM-BoW $\Delta p(c, stepsize)$ evolution.

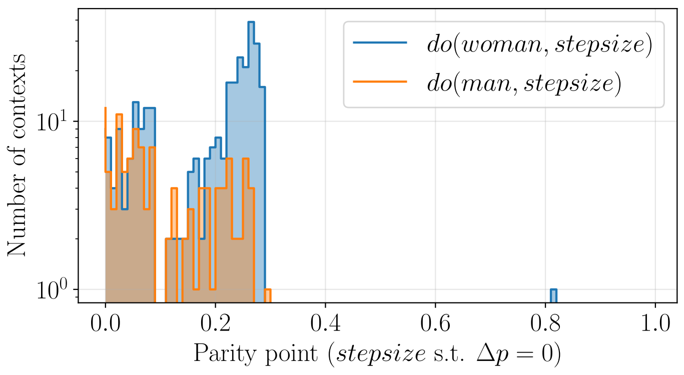

Figure 8: PPLM-BoW histogram of parity points per *occupational* context.

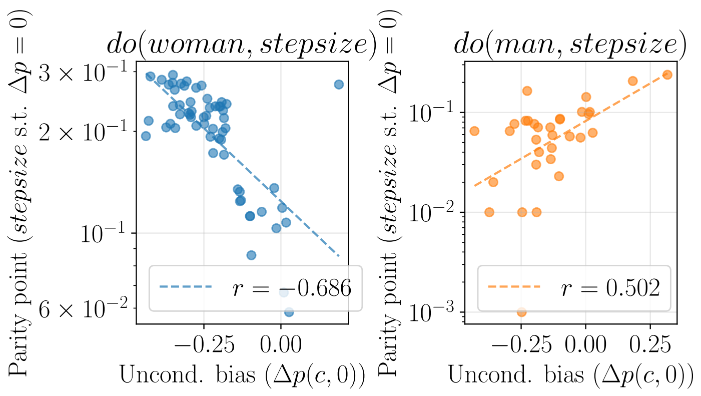

Figure 9: PPLM-BoW correlation between unconditional bias and parity point).

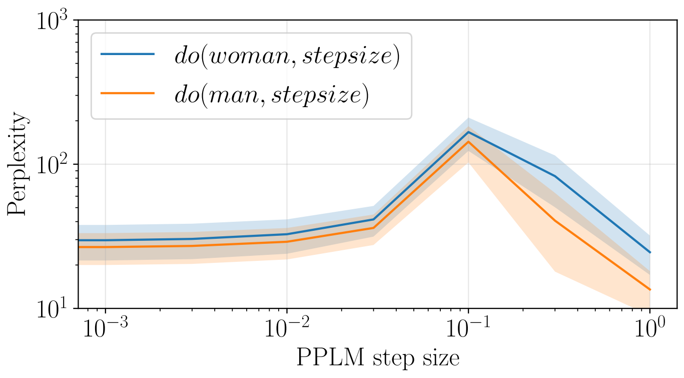

Figure 10: PPLM-BoW average perplexity when conditioning on *woman* and *man* at different $stepsize$ levels.

G About OneSec annotations

Note that the meaning of the concept is important. For example, concept one%1:23:00 (the smallest whole number or a numeral representing this number, *e.g.he has the one but will need a two and three to go with it"; "they had lunch at one"*) achieves a $\max_m\{\text{AP}_m^c\} = 0.9885$, while concept one%1:09:00 (a single person or thing, *e.g."he is the best one"; "this is the one I ordered"*) only achieves $\max_m\{\text{AP}_m^c\} = 0.8779$.

Details on the annotations Each sentence in the OneSec dataset (Scarlini et al., 2019) is annotated as in the following example:

```
<instance docsrc="Indigenous architecture" id="shelter.00002">
    <answer instance="shelter.00002" senseid="shelter%1:06:00::" />
    <context>
        Types There are three traditional types of igloos ,
        all of different sizes and used for different purposes.
        The smallest were constructed as temporary
        <head>shelters</head>
        , usually only used for one or two nights .
     </context>
</instance>
```

The `senseid` label is the one of the marked word (*shelters* in this example, between `<head>` and `</head>`). We use the `senseid` as follows. The part before the `%` is called *lemma*, while the remaining numbers uniquely identify the concept in WordNet. We parse all the sentences for a given `senseid` to create the positive sentences of each concept, only keeping those `senseid` with more than 100 sentences. As explained in Sec. 3, the negative sentences for a concept are randomly selected from all the `senseid` with different *lemma* than the positive ones.

OneSec license: The OneSec dataset has a license of type `Creative Commons Attribution-Noncommercial-Share Alike 4.0 License`.