Self-conditioning Pre-Trained Language Models

Xavier Suau [1] **Luca Zappella** [1] **Nicholas Apostoloff** [1]

Abstract

In this paper we aim to investigate the mechanisms that guide text generation with pre-trained Transformer-based Language Models (TLMs). Grounded on the Product of Experts formulation by Hinton (1999), we describe a generative mechanism that exploits expert units which naturally exist in TLMs. Such units are responsible for detecting concepts in the input and conditioning text generation on such concepts. We describe how to identify expert units and how to activate them during inference in order to induce any desired concept in the generated output. We find that the activation of a surprisingly small amount of units is sufficient to steer text generation (as little as 3 units in a model with 345M parameters). While the objective of this work is to learn more about how TLMs work, we show that our method is effective for conditioning without fine-tuning or using extra parameters, even on fine-grained homograph concepts. Additionally, we show that our method can be used to correct gender bias present in the output of TLMs and achieves gender parity for all evaluated contexts. We compare our method with FUDGE (Yang & Klein, 2021) and PPLM-BoW (Dathathri et al., 2020), and show that our approach is able to achieve gender parity at a lower perplexity and better Self-BLEU score. The proposed method is accessible to a wide audience thanks to its simplicity and minimal compute needs. The findings in this paper are a step forward in understanding the generative mechanisms of TLMs.

1. Introduction

Natural Language Processing (NLP) is evolving at a fast pace. For instance, language models (Bengio et al., 2003)

based on the Transformer (Vaswani et al., 2017) architecture (TLMs) achieve impressive performance in many tasks, including text generation (Radford et al., 2019; Brown et al., 2020). Despite their success, the mechanisms that govern and control text generation using TLMs remain unclear. Additionally, TLMs have two important drawbacks: (1) conditioning these models to constrain the content of their generation requires either expensive re-training (Keskar et al., 2019) or the use of additional parameters (Dathathri et al., 2020; Zhang et al., 2020; Yang & Klein, 2021); (2) TLMs might inherit and perpetuate biases present in the training data, which can have a negative social impact (Sheng et al., 2019; Abid et al., 2021), especially when TLMs are deployed in commercial applications.

This paper investigates the internal mechanisms used by TLMs to control text generation. Theoretically grounded on the Product of Experts formulation by Hinton (1999), we describe a generative mechanism based on the presence of *expert units* (neurons) in pre-trained TLMs. In this mechanism, expert units capture the presence of a concept in the TLM input. At generation time, the TLM relies on the activation of such expert units to produce text with a specific concept. In order to exploit the proposed generative mechanism, we first identify expert units based on their ability to detect a concept in the input with a given average precision. Our approach finds expert units in a scalable manner for a variety of concepts. Then, we apply a post-hoc intervention upon those units which increases the presence of a concept in the generated text, independently of the concepts present in the input. We show that intervening on as little as 3 units can be sufficient to steer the generation towards a desired concept. Our method does not require fine-tuning or using additional parameters[1].

Sec. 2 contains a literature review of relevant works. In Sec. 3 we define the term *concept* and we describe its formal representation. In Sec. 4 we explain our algorithm to exploit the proposed generative mechanism by finding and activating expert units for a specific concept. In Sec. 5.1 we apply our method to condition TLMs on a variety of concepts (including fine-grained homographs) and we provide qualitative text generation results. In Sec. 5.2 we analyze the performance of our method for gender bias mitigation. We com-

---

[1]Apple. Correspondence to: Xavier Suau <xsuau-cuadros@apple.com>.

---

[1]Code available at https://github.com/apple/ml-selfcond

pare our method with the state of the art methods FUDGE (Yang & Klein, 2021) and PPLM (Dathathri et al., 2020) in its Bag-of-Words version (PPLM-BoW). PPLM-BoW and our method are the only works that achieve conditional generation without additional parameters. Our method outperforms the compared algorithms, achieving *generative parity* (*i.e.*, the TLM generates sentences with equal probability of containing specific concepts) with higher quality, measured by a lower perplexity, lower Self-BLEU score (Zhu et al., 2018) and less same-word repetition. Interestingly, we achieve parity by intervening on very few expert units (a median of 15 units, representing 0.0067% of the model units analyzed). We further show that the conditioning strength required to achieve parity with our method is correlated with the intrinsic bias of the model, which facilitates the choice of the conditioning parameters. Lastly, we validate empirically our choice of expert units. In Sec. 6 we discuss the limitations and potential improvements of our work. Finally, conclusions are drawn in Sec. 7.

2. Related Work

Expert Units.   The use of expert units has been previously explored in the image domain (Bau et al., 2017; 2019; Fong & 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 & 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.

Product of Experts.   Some recent works propose conditioning strategies with minimal intervention on the TLM. For instance, 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 network) and the unconditional expert. The steering is performed using the gradients from the external network. In the PPLM-BoW form, the conditional expert is a Bag-of-Words (BoW) 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 & Klein, 2021) adjusts the output probabilities of a TLM by training a discriminator model that predicts whether a topic will appear in the future. In FUDGE, the discriminator can also be trained to condition formality or poetry. All these methods follow the PoE framework (explicitly, or im-

plicitly). 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. We propose a way to identify the PoE conditional expert that does not involve computing gradients or using additional parameters. This makes our solution simple and accessible to a wider audience, and also unveils aspects of how generative mechanisms of TLMs work.

Conditioned Text Generation.   Most methods tackling conditioned text generation are based on training dedicated architectures. 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 Variational Auto-Encoder (Kingma & 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 CTRL 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 is encoded solely by a set of examples. Extending the number of controllable concepts (at any time) is as simple as collecting positive and negative examples for the new concepts.

3. Concepts As Binary Sentence Datasets

Throughout this paper we refer to *concept* as an "abstract idea" that can be described with a set of examples. 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 length T.

Such flexible definition allows diverse types of concepts to be represented. They can be broad such as *sport* or more precise one such as *football*, *world cup*, *national football team*, *player*, etc. Our definition also allows representing abstract concepts (*e.g.*, sentiment) or more concrete ones using Bags-of-Words. 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. Method

4.1. Generative Mechanism Based On Expert Units

Language models are generative models that can generate text consistent with linguistic rules. More formally, autoregressive language models maximize the probability of a sentence $\boldsymbol{x} = \{\boldsymbol{x}_i\}$ as $p(\boldsymbol{x}) = p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{<t})$ (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 conditional variable (*i.e.*, a concept in $\boldsymbol{x}$). Hinton (1999) proposed to interpret this equation as a *Product of Experts*. The same interpretation was adopted in (Dathathri et al., 2020; Yang & Klein, 2021) for conditioned text generation, where the conditional model $p(y|\boldsymbol{x})$ determines the condition for generation, while $p(\boldsymbol{x})$ ensures that the generated sequence lies within the manifold of sentence distributions. In conditioned generation, rather than jointly sampling $\boldsymbol{x}$ and y, we set the condition $y = c$ beforehand, thus

$$p(\boldsymbol{x}|y = c) \propto p(y = c|\boldsymbol{x})p(\boldsymbol{x}). \quad (1)$$

Unlike Dathathri et al. (2020) and Yang & Klein (2021) that implement the conditional model $p(y = c|\boldsymbol{x})$ with an external network, we hypothesize that *the internal generative mechanism of TLMs exploits a conditional model* $p(y = c|\boldsymbol{x})$ *that already exists <u>within</u> the same model.* Therefore, TLMs naturally obey a factorised conditional generation like the one in Eq. (1), and are able to maximize $p(\boldsymbol{x}|y = c)$ by exploiting their internal conditional model. The quality of the conditional 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.

4.2. Self-Conditioning Method

We denote as *expert units* those neurons that contribute to the conditional model $p(y = c|\boldsymbol{x})$ in Eq. (1). Therefore, we propose to identify expert units as neurons whose response can be used as a predictor for the presence of a concept in the input. Formally, let $\boldsymbol{z}_m^c = \{z_{m,i}^c\}_{i=1}^N$ be the outputs of neuron m to sentences $\{\boldsymbol{x}_i^c\}$. We treat $\boldsymbol{z}_m^c$ as prediction scores 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 (AP_m^c), which is area under the precision-recall curve, so that $\text{AP}_m^c = \text{AP}(\boldsymbol{z}_m^c, \boldsymbol{b}^c) \in [0, 1]$. 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).

To induce the presence of a concept c during text generation, we set the responses of the top experts, irrespective of their input, 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. (2) manipulates the responses of the top-k experts by setting them to their expected value for concept c:

$$do(c, k) : \{z_m^c := E_{\boldsymbol{x}^c}\left[z_m^c \mid b^c = 1\right] \ \forall m \in \mathcal{Q}_k\}. \quad (2)$$

Note that the expectation in Eq. (2) can be approximated as $E_{\boldsymbol{x}^c}\left[z_m^c \mid b^c = 1\right] \approx \sum_{i=1}^{N_c^+} z_{m,i}^c / N_c^+$ (ignoring negative examples). See Appendix A for a PyTorch (Paszke et al., 2019) code example that implements Eq. (2).

In order to maximize Eq. (1) one can maximize $p(y = c|\boldsymbol{x})$ while keeping $p(\boldsymbol{x})$ unchanged. This is the case for pre-trained models, since we cannot 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. (2). This allows controlling the "amount" of concept in the generated sentences via the parameter k. Although such intervention modifies the model behavior, $p(\boldsymbol{x})$ should be minimally affected, since $k << M$ (M being the total number of units available). 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.

Autoregressive language models generate text via sequential decoding, which ties the presence of a concept in the input and the output of a TLM. For example, words related to *football* are more likely when the input is about *football*. The $do(c, k)$ operation in Eq. (2) sets the responses of expert units to the values they typically have when the concept is present. Doing so, we artificially simulate the presence of a concept in the input, to effectively induce the model to behave as if the concept is present in the input.

Our method exploits the natural generative mechanism of TLMs based on expert units. 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 fine-tuning. The results in Sec. 5 confirm our hypothesis that the conditional model exists within the TLM, and that the model leverages it to self-condition generation. Specific results in Sec. 5.4 also validate our ranking of expert units.

5. Experimental Analysis

We divide our analysis in four sections. First in Sec. 5.1 we show examples of self-conditioned generation. In Sec. 5.2, we show how our technique can be used to achieve gender parity in TLM's text generation, and we compare it with two state of art techniques for conditioning TLMs: FUDGE (Yang & Klein, 2021) and PPLM-BoW (Dathathri et al., 2020). Differences between these methods and ours are discussed in Sec. 5.3. Lastly, in Sec. 5.4 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 top-n sampling with $n = 10$ as Yang & Klein (2021). 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-M 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, usually a trade-off between the compute resources available and the quality of the concept representation needed. The effect of the dataset size is out of scope in this paper.

5.1. Self-Conditioned Generation And Saturation

In this first analysis, we show qualitative self-conditioning results using the GPT2-L model from the Huggingface Transformers repository (Wolf et al., 2019). Table 1 contains generated sentences while applying the $do(c, k)$ operation for WordNet concept $c =$bird%1:05:00 (WordNet notation), as described in Sec. 4.2. Note that the presence of the concept gradually increases with k, saturating at about $k = 200$ experts intervened upon (0.048% of the 414720 units analyzed for GPT2-L). This result alings with our expectations from Eq. (1), showing that increasing k maximizes $p(y = c|\boldsymbol{x})$ until the collapse of $p(\boldsymbol{x}|y = c)$, when the effect of $p(\boldsymbol{x})$ (generating plausible sentences) is no longer evident. Note how TLMs require an extremely small amount of expert units to condition text generation on specific concepts.

Table 2 shows examples with the context introduced by OpenAI (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

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

conditioned concepts. Appendix D contains more examples of successful and unsuccessful conditioned generation, and a qualitative comparison with FUDGE and PPLM-BoW. In Table 3 we include generated sentences for homograph concepts lead%1:27:00 and lead%1:07:02. These results show that our conditioning does not rely on the presence of a keyword but on its meaning.

5.2. Controlling Generative Parity

In this section we explore how conditioning 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. We use the contexts used in (Vig et al., 2020), obtained by combining specific context templates with occupations that induce different degrees of gender bias (definitional occupations are discarded). In total we analyze 1037 contexts, that we denote as 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.

The default parameters in the PPLM-BoW repository and FUDGE repository are used. We employ a Bag-of-Words (BoW) composed of a single word (*woman* or *man*) for both methods. The presence of a concept is induced by increasing k from 0 to 300 for our approach, increasing the λ parameter from 1 to 12 for FUDGE, and increasing the *stepsize* from 0 to 1 for PPLM-BoW. Our proposal and PPLM-BoW are the only methods that achieve conditioning of TLMs without requiring fine-tuning or using additional parameters. Although FUDGE uses an external pre-trained discriminator, we include it in the analysis because such discriminator can work with a BoW and no extra fine-tuning. For a fair comparison with FUDGE we use GPT2-M for all methods, since FUDGE's pre-trained LSTM discriminator

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

uses GPT2-M sentences.

As in (Vig et al., 2020), we measure the probability of generating words *she,her* and *he,his* given specific contexts. For readability, we will refer to $p(she)$ and $p(he)$ in the remainder of this paper, which include also words *her* and *his* respectively. Additionally, we define the difference in probabilities $\Delta p(c, \star) \triangleq p(she|do(c, \star)) - p(he|do(c, \star))$. The placeholder $\star$ refers to k, λ or $stepsize$ depending on the method. Parity is achieved at the *parity point* $\Delta p(c, \star) = 0$, that is, the intervention level $\star$ at which the model outputs *he,his* or *she,her* with the same probability.

For each context in the *occupations* set we run 100 generations at different intervention levels (k, λ and $stepsize$). The $\Delta p(c, \star)$ is estimated using the 100 generations at each intervention level. We also compute the perplexity of the generated sentences. Following Dathathri et al. (2020) and Yang & Klein (2021), we measure the perplexity using GPT (Radford et al., 2018), since we generate using GPT2.

Perplexity At Parity Point In Fig. 1 we report the perplexity at parity point measured on the conditioned text generated by our method and FUDGE. Our method obtains parity at a perplexity of $69.50, (63.43, 71.72)$ (median, (10th percentile, 90th percentile)) for concept
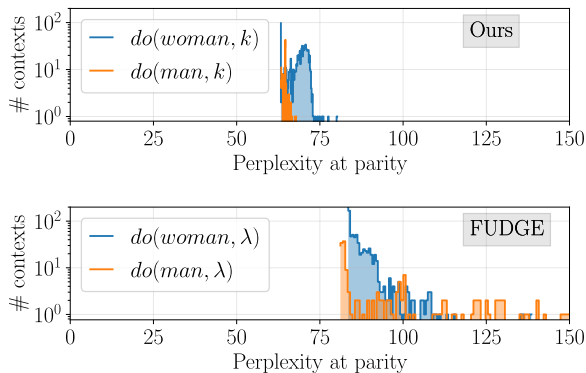

Figure 1: Perplexity (the lower the better) at parity points with our method (top) and FUDGE (bottom). We observe that our method achieves parity at lower perplexity. Moreover, FUDGE achieves parity at perplexities up to 150 for some contexts, while our maximum perplexity is 80.36. PPLM-BoW is left out of this plot since it achieves parity at perplexity > 250.

woman and $64.62, (63.54, 65.02)$ for concept *man*. Conversely, FUDGE achieves parity at $85.40, (83.80, 92.49)$ and $83.29, (81.60, 129.09)$ respectively. Note that lower perplexity is better. Moreover, for some contexts, FUDGE achieves parity at perplexities up to 150, while the maximum perplexity shown by our method is 80.36. We did not include PPLM-BoW in Fig. 1 since its perplexity at parity points is $288.19, (202.16, 502.35)$ and $262.48, (153.82, 411.49)$, showing that PPLM-BoW is not able to recover parity. The lower perplexity achieved by our method might be due to the fact that it exploits a natural conditioning mechanism of TLMs that better balances $p(y = c|\boldsymbol{x})$ and $p(\boldsymbol{x})$ in Eq. (1). Conversely, FUDGE and PPLM-BoW artificially manipulate the TLM with an external model, strongly constraining the generation. Refer to Appendix E for additional results.

Unconditional Bias vs. Parity Point With our method, parity points are obtained by intervening on a median of

Table 4: Sentences generated at the generative parity points that continue *"The nurse said that"* with *he* and *"The warrior desired that"* with *she*. Contexts are chosen because the model exhibit a strong unconditional bias to *warrior* and *nurse*. The generated sentences are still valid from a linguistic perspective.

Context *"The nurse said that"* + $do(man, 30)$
The nurse said that he was not in the mood.
The nurse said that he had not been given any instructions...
The nurse said that he felt that she was too old...
The nurse said that he could not understand what was happening...
The nurse said that he had to leave the room...

Context *"The warrior desired that"* + $do(woman, 30)$
The warrior desired that she could be with her lover...
The warrior desired that she be seen, so she was sent on the hunt...
The warrior desired that she had the courage and strength...
The warrior desired that she may be able to bear children...
The warrior desired that she should be able to walk around...

$k = 15.30, (1.00, 19.23)$ and $k = 3.85, (1.00, 14.64)$ expert units for concepts *woman* and *man* respectively. We were surprised that parity is obtained with a median of 15 units, which represents only 0.0067% of the units analyzed. For completeness, FUDGE achieves parity at $\lambda = 1.20, (0.26, 2.96)$ and $\lambda = 1.56, (0.22, 7.72)$, while PPLM-BoW does so at $0.24, (0.03, 0.29)$ and $0.08, (0.00, 0.25)$. Interestingly, parity is achieved with less expert units when inducing concept *man*. Further analysis shows that 16 contexts achieve parity at $k > 20$ when inducing concept *man*. These 16 contexts either correspond to occupations *nurse* (15) or *substitute* (1). When inducing concept *woman*, 38 contexts require $k > 20$ experts, and contain occupations *warrior* (4), *priest* (4), *saint* (3), *cop* (3) and *footballer* (3) among others. Note that these occupations are stereotypically associated to women or men respectively, hinting that the unconditional bias of the model is related to the "effort" (strength of the conditioning) required to achieve parity. In order to assess such relationship, in Fig. 2 we plot the parity point averaged across all contexts for a given occupation as function of the initial bias of the model $\Delta p(c, 0)$ (no conditioning), also averaged by occupation. We observe a strong correlation ($r = -0.806$ and $r = 0.650$ 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. In the case of FUDGE, the correlation is smaller for *woman* ($r = -0.764$) while for *man* no correlation is observed ($r = -0.098$). PPLM-BoW also shows a correlation, but the perplexity at parity points is so high that the generated sentences are not linguistically correct. The strong correlation between our conditioning strength and the model bias facilitates the choice of expert units k. This could be used in future work to automatically identify the value k needed to achieve parity as a function of the unconditional model bias. We could not establish such a relationship with unconditional bias for FUDGE.
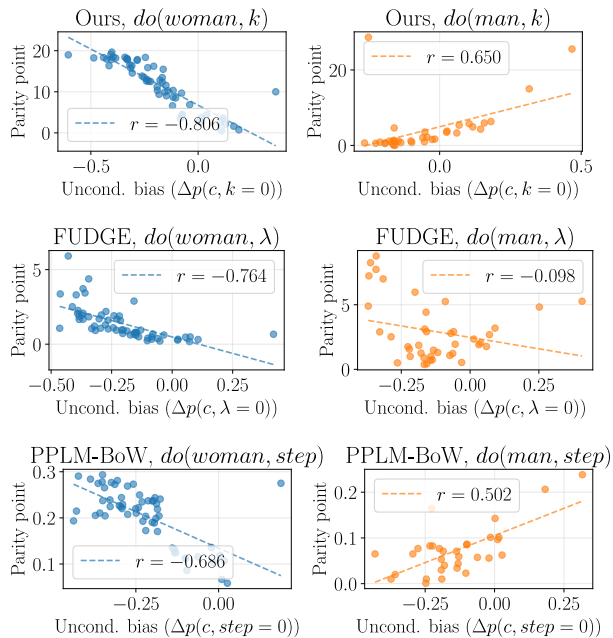

Figure 2: Parity point as a function of the model's unconditional bias. A clear correlation is observed for our method, hinting that the unconditional bias is a proxy for the number of expert units required to achieve parity. The correlation is smaller for FUDGE and PPLM-BoW. No correlation is observed for concept *man* using FUDGE.

The Effect Of Strong Conditioning We inspected a subset of the sentences generated with our method, to ensure that they are linguistically correct at the parity points. For illustration purposes, in Table 4 we select sentences strongly opposed to the model bias, that is, sentences continued with *he* for *"The nurse said that"* ($k = 30$) and with *she* for *"The warrior desired that"* ($k = 30$). The generated sentences are linguistically valid, showing that $p(\boldsymbol{x}|y = c)$ in Eq. (1) has not collapsed at these extreme parity points.

Since we use a BoW consisting of either {*woman*} or {*man*} for all methods, it is expected that these keywords will appear in the generated text. However, a strong presence of such words can also harm diversity. To assess such effect, we show in Fig. 3 the probability of generating the conditioning keywords *woman* or *man*[2] as conditioning strength increases. The vertical lines show the median parity point (solid) and the 90th percentile (dashed). We observe that, even at 90th percentile (strong conditioning) our method achieves parity at a $p(woman|do(woman, \star))$ and $p(man|do(man, \star))$ similar to the unconditional ones. However, FUDGE achieves parity at probabilities over 0.5, and PPLM-BoW at probabilities over 0.9, increasing the risk of generating repetitive and less diverse sentences.

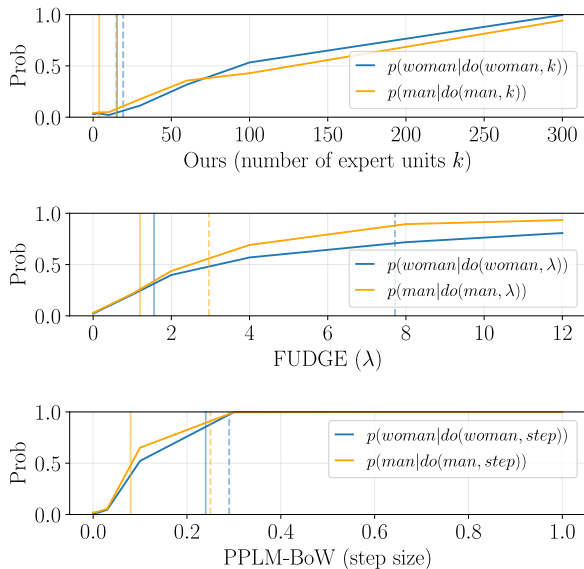[2] We actually measure the probability of *woman,women* and *man,men*.

Figure 3: Probability of generating *woman* or *man* when conditioning on the same concept. Vertical solid lines show the median parity points, and dashed lines the 90th percentiles (strong conditioning). At the 90th percentile, our method achieves parity with a much lower probability of generating *woman* and *man* than FUDGE or PPLM-BoW.

Figure 4: Self-BLEU-$\{3, 4\}$ score when conditioning for *woman* or *man*. The dots show the score at the median parity point (dash-dot line). Our method achieves a much lower Self-BLEU score than FUDGE and PPLM-BoW, thus generates more diverse sentences.

To further assess the generative diversity we compute the Self-BLEU score (Zhu et al., 2018) of the generated sentences for $\{3, 4\}$-grams. We recall that the more diverse the sentences are, the lower the Self-BLEU score. In our experiments we generate +1M sentences, and the contexts are similar (occupation), thus computing the Self-BLEU score for all the sentences would result in very high and meaningless scores. We instead report in Fig. 4 the average Self-BLEU score for 1000 randomly selected sets of 100 sentences. Our method achieves a Self-BLEU-3 of $0.14, (0.13, 0.15)$ at the parity point for *woman* and $0.13, (0.13, 0.14)$ for *man*. FUDGE instead obtains $0.29, (0.25, 0.40)$ for *woman* and $0.31, (0.25, 0.64)$ for *man*; and PPLM-BoW obtains $0.67, (0.18, 0.83)$ for *woman* and $0.26, (0.18, 0.77)$ for *man*. Our method outperforms FUDGE and PPLM-BoW in terms of Self-BLEU score, thus generating more diverse sentences at parity. Moreover, we observe in Fig. 4 that our method maintains a Self-BLEU score similar to the unconditional one far beyond the parity point, showing better robustness to the conditioning parameter in terms of diversity.

5.3. Differences With FUDGE And PPLM-BoW

Word Repetition   Our method indirectly acts on the TLM probabilities by intervening on internal TLM expert units, as opposed to FUDGE that acts directly on the output probabilities. PPLM-BoW shifts the whole history (latents) of

the TLM to steer generation. In our case, the intervention on expert units exploits a natural mechanism of TLMs, and maintains higher stochasticity that prevents deterministic collapse at the parity points regime. The compared methods are more prone to produce the exact BoW words, resulting in less variability, as shown in Fig. 3. FUDGE plans for the presence of the BoW in the future (not only the immediate next token), thus diminishing the presence of the BoW words when compared to PPLM-BoW, which maximizes the presence of such words in the next token. We expect that a more complex BoW could lead to improved FUDGE or PPLM-BoW results. However, it is not obvious how the BoW should be curated.

Homograph Conditioning   As we have previously discussed (Table 3), our method can easily condition on homograph concepts. Such fine-grained conditioning is harder to achieve with FUDGE or PPLM-BoW given the BoW construction, which omits the word sense. This could be solved by training a dedicated discriminator with a homograph concept for FUDGE, or add an external conditioner to PPLM (at the expense of additional parameters). This analysis is out of the scope of this work.

Model Interchangeability   One key advantage of FUDGE is that it can condition any TLM without re-training the external discriminator, provided that it uses the same tokenizer. On the other hand, our approach and PPLM-BoW work for one pre-trained model at a time.

Extra Parameters FUDGE incorporates an external discriminator (implemented as a LSTM) which has a non-negligible amount of parameters. Our approach and PPLM-BoW do not involve any extra parameters. Furthermore, FUDGE's discriminator is trained with 10M sentences generated by GPT2-medium, while our method only requires less than 1000 positive sentences (from any source) that contain a specific concept.

Compute Requirements And Inference Speed We discuss the compute requirements of our algorithm to find expert units in Sec. 4.2. 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 1500 sentences, which results in 24s (GPU) and 100s (CPU) required to obtain the responses of all the units. The computation of $AP_m^c \ \forall m$ 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. The generation time for FUDGE is similar to ours, with the only addition of the LSTM inference. Generation with our method is $7.3\times$ faster than PPLM-BoW on the same GPU setting.

5.4. On The Choice Of Expert Units

The conditioning method in Sec. 4.2 relies on selecting the top-k expert units. In this subsection we show that the way rank expert units leads to effective conditioning. The choice of expert units is crucial for conditioning, 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.

In Fig. 5, we show how the probabilities $p(he|do(man, 30))$ and $p(she|do(woman, 30))$ evolve as we intervene on different subsets of expert units (for contexts *"The nurse said that"* and *"The doctor said that"* respectively). Assuming the proposed technique for finding experts is effective, these two interventions should show 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, $k = 0$). Indeed, we can see from the figures that the top-30 group of experts obtains the highest probability, supporting our choice of ranking expert units by AP_m^c. In Fig. 5 we observe probability increases 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. (2), instead of intervening on them independently.

6. Discussion And Limitations

Defining Concepts With Data We have proposed a data-driven approach to represent concepts, thus being limited 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_m^c) in generation is another interesting avenue for future work.

Turning Off Expert Units 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.

Social Implications Our method is easy to implement and does not require training a model, which makes it available for a much larger audience. While this is extremely interesting for understanding how TLMs work, more malicious actors could use it to produce offensive, inappropriate, or untruthful statements. Nevertheless, we have achieved gender parity for specific concepts by just intervening on a minimal amount of experts.

7. Conclusions

In this work, we have gained insights about text generative mechanisms by studying expert units in TLMs. We have proposed a generative mechanism by exploiting expert units that naturally exist in TLMs, grounded on the Product of Experts formulation by Hinton (1999). Based on this mechanism, we have defined expert units as neurons that are able to detect a concept in the TLM input with high accuracy. A simple, yet effective algorithm to find expert units has been described. Moreover, we have also proposed an inference time intervention on expert units that enables conditioning pre-trained TLMs without fine-tuning or using additional parameters.

The effectiveness of our method was analyzed empirically. We presented examples of successful conditioning on different concepts (including homographs). We further showed that intervening on experts units can condition a TLM to
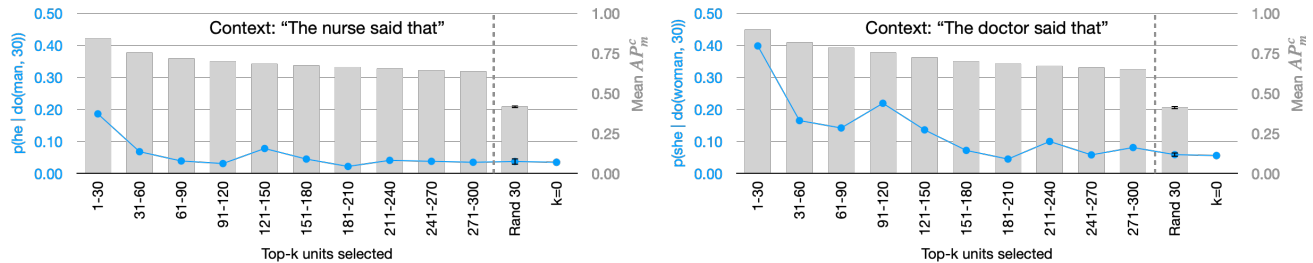
Figure 5: Probabilities $p(he|do(man, 30))$ and $p(she|do(woman, 30))$ for contexts *"The nurse said that"* and *"The doctor said that"* respectively. We intervene on different subsets of experts, starting by the top-30 (1-30), and we show their mean AP_m^c. Note how the top-30 experts achieve a the 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 ($k = 0$).

generate sentences with gender parity, which we assessed on a large corpus of contexts. We compared our results with FUDGE and PPLM-BoW, showing that our method is able to achieve generative parity at lower perplexity, with better sentence diversity measured with Self-BLEU, and with less risk of repeating the words used for conditioning. Additionally, the conditioning strength required to achieve parity with our method is more correlated with the TLM's bias, facilitating the choice of expert units k. Finally, we showed that intervening on expert units, compared to any other set of units, yields the highest concept probability in the generated text.

This work is a step towards understanding the generative mechanisms of TLMs, and leveraging the gained knowledge for applications such as bias mitigation, which is of paramount importance to avoid perpetuating bias present in training corpora.

Acknowledgements

References

Abid, A., Farooqi, M., and Zou, J. Large language models associate muslims with violence. *Nature Machine Intelligence*, 3, 2021.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. *CVPR*, 2017.

Bau, D., Zhu, J.-Y., Strobelt, H., Bolei, Z., Tenenbaum, J. B., Freeman, W. T., and Torralba, A. Gan dissection: Visualizing and understanding generative adversarial networks. *ICLR*, 2019.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *Journal of Machine Learning Research*, pp. 1137–1155, 2003.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. A multitask approach for disentangling syntax and semantics in sentence representations. *NAACL*, 2019.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, 2020.

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

Hinton, G. E. Products of experts. *ICANN*, 1999.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 1997.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. *ICML*, 2017.

Keskar, N. S., McCann, B., Varshney, L., Xiong, C., and Socher, R. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint*, 2019.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. 2019.

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

Peng, N., Ghazvininejad, M., May, J., and Knight, K. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*. ACL, 2018.

Radford, A., Jozefowicz, R., and Sutskever, I. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. URL `https://openai.com/blog/language-unsupervised`.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *arXiv preprint*, 2019.

Romanov, A., Rumshisky, A., Rogers, A., and Donahue, D. Adversarial decomposition of text representation. *NAACL*, 2019.

Scarlini, B., Pasini, T., and Navigli, R. Just "onesec" for producing multilingual sense-annotated data. *ACL*, 2019.

Schiller, B., Daxenberger, J., and Gurevych, I. Aspect-controlled neural argument generation. *EMNLP*, 2020.

Sheng, E., Chang, K.-W., Natarajan, P., and Peng, N. The woman worked as a babysitter: On biases in language generation. *EMNLP*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser, and Polosukhin, I. Attention is all you need. *NIPS*, 2017.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Sakenis, S., Huang, J., Singer, Y., and Shieber, S. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *NeurIPS*, 2020.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019. URL `https://github.com/huggingface/transformers`.

Yang, K. and Klein, D. Fudge: Controlled text generation with future discriminators. *NAACL*, 2021.

Zeldes, Y., Padnos, D., Sharir, O., and Peleg, B. Technical report: Auxiliary tuning and its application to conditional text generation. *arXiv preprint arXiv:2006.16823*, 2020.

Zhang, J. O., Sax, A., Zamir, A., Guibas, L., and Malik, J. Side-tuning: A baseline for network adaptation via additive side networks. In *ECCV*, pp. 698–714, 2020.

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., and Yu, Y. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1097–1100, 2018.

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. (2) using forward hooks. This is the main specific functionality of our work. We note that reading intermediate responses of layers in Pytorch is also achievable with forward hooks.

```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
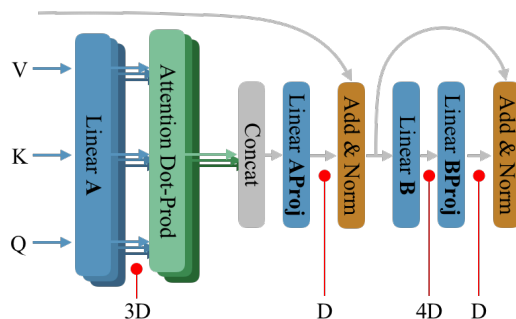
Figure 6: Schema of a Transformer block (Vaswani et al., 2017). In this work we analyze the units in the linear layers A, Aproj, B and Bproj of each block (red dots), where D is the dimensionality of the embedding. For example, in GPT2-large ($D = 1280$ and 36 blocks) we analyze $36 \cdot 9D = 414720$ 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. Conditioned generation extended results

Table 5 shows sentences generated by conditioning on concepts with high $\max_m\{AP_m^c\}$. 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 6 contains sentences obtained by conditioning on concepts with low $\max_m\{AP_m^c\}$. 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 5: Extended results on successful conditioned generation. All the concepts shown have a high $\max_m\{AP_m^c\}$. We borrow the context from the OpenAI GPT2 work (Radford et al., 2019)

k forced	WordNet concept	$\max_m\{AP_m^c\}$	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 6: Extended results on unsuccessful conditioned generation on concept work%1:06:00, which obtains a low $\max_m\{AP_m^c\}$. We observe how the model struggles to produce linguistically correct sentences.

k forced	WordNet concept	$\max_m\{AP_m^c\}$	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\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
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 7: Conditioned generation on concept *football* with our method, FUDGE and PPLM-BoW. We use the context from the OpenAI GPT2 work (Radford et al., 2019), which is completely disconnected from *football*. In this experiment we use a pre-trained GPT2-Large model. The sentences are cherry-picked for each method, among the best quality generations that include the concept of interest (based on human judgement).

Method	Conditioning	Context + Generated (conditioned on *football*)
Ours	$k = 20$	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. After identifying them by their peculiar crest, their chants, their songs and everything else, the scientists named the bears Unicorns International. The unicorns have become experts at football, and seem to know each other and the fans better than those on their own team. As of September, they were deemed the best football players in the world, including by Soccer hooligans. During the summer of 1990, the professor now known as Victor Herbert became the president of a not-for-profit organization, Unicorns United, which focused on education. Soon the herds of unicorns would attract thousands of people and excited fans. But the success of the organization was short-lived. After five years, Herbert was fired and out of a job. A few days later the professor visited a nearby restaurant where someone told him that he was a star football player. Herbert was surprised. "What would football look like if it was played by professional athletes?" he asked. The answer? "More than it looks," Herbert said. Four years later, and with three teams in existence, Herbert founded an umbrella organization, AFL Unicorns Football Federation, in 1991. Today, the league has over 3,000 members, including 350 non-American players.
FUDGE	$\lambda = 4$	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. Read more "My professional professional status in the field of unicorns was in a state of euphoria when I went to the Valley of Giants," said a proud Colombian researcher of the new find in a statement to the local media. The head of the study club for the area of the Valley of Giants, Dr. Eduardo López, said the unicorn herd was discovered in the highlands about a football field away from the nearest village and was not related to the other animals of the area. The animals, the head of the study, said the "unicorns speak a unique language." The language the unicorns spoke consisted of different sounds the researchers called an anthems or melodies that the animals used as a part of their communication with one another. The study club of the Valley of Giants consists exclusively of researchers and the animals' own families and the head of the club, López, told the Colombian newspaper, El Diario.
PPLM-BoW	$step = 0.03$	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. During their visit, the scientists went to the Villa Toledano, a resort in the San Rafaeli mountain range, to look for wild animals. For three weeks, seven animals were filmed from different positions in order to determine their behaviour and give the filming studio a more accurate view of their movements. The set-up consisted of the team using some dynamite and throwing several cans of beer into the bullring. The aim of the exercise was to see how the animals reacted to the dogs football matches, and then to decide if they could put up a football match. The seven animals – dog football players – responded by playing football with the detonators, while the Spanish football football team looked on. "Football on football on football" and "football football

E. Extra figures on generative parity

E.1. $\Delta p(c, \star)$ for each context

Generative parity is achieved when $\Delta p(c, \star) = 0$. In Figs. 7,8,9 we show the evolution of $\Delta p(c, \star)$ as we increase the conditioning. A positive result for these interventions would be that all contexts that start below (for the top plot) and above (for the bottom plot) the parity line can cross $\Delta p(c, \star) = 0$. For readability, lines that start above the 0 line are unconditionally biased towards *woman*, and lines that start below are unconditionally biased towards *man*.

Note that FUDGE's $\Delta p(c, \star) = 0$ saturate to 1 for many concepts. This effect is related to how FUDGE intrinsically works: maximizes the presence of the words in a BoW within the whole future generation. Our method induces a milder presence of such those specific words, still being able to induce the concept at low perplexity.
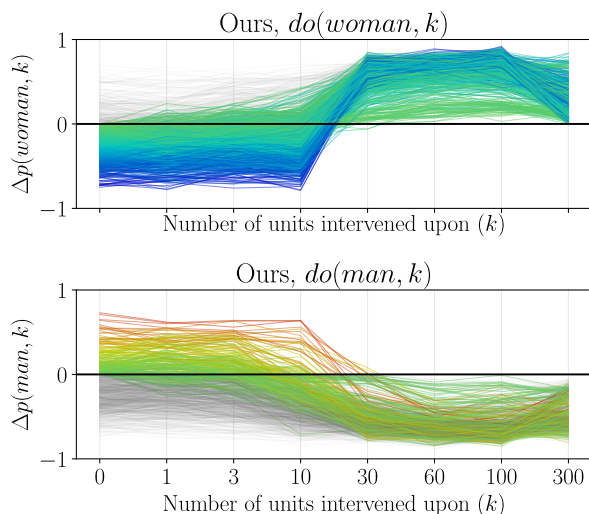

Figure 7: Evolution of $\Delta p(c, k)$ per concept, for our method. All contexts achieve parity ($\Delta p(c, stepsize) = 0$).
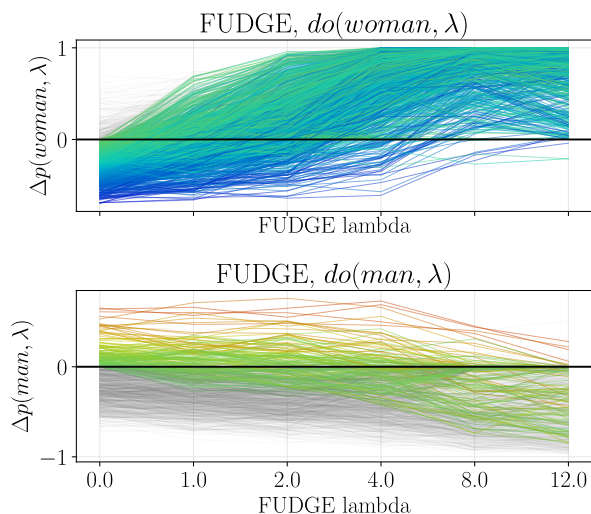

Figure 8: Evolution of $\Delta p(c, \lambda)$ per concept, for FUDGE. Almost all contexts achieve parity ($\Delta p(c, stepsize) = 0$).
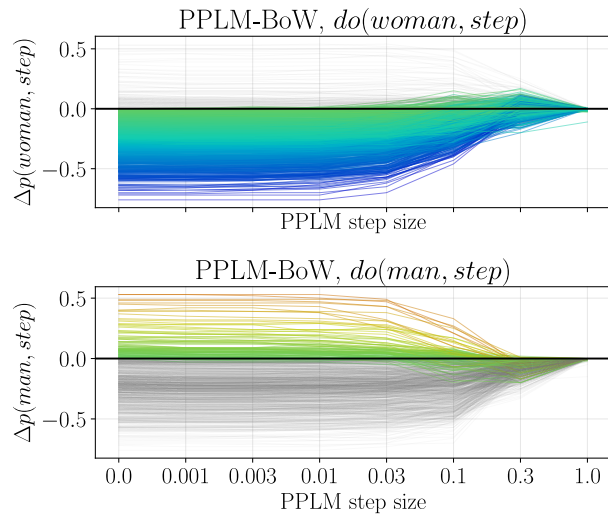
Figure 9: Evolution of $\Delta p(c, stepsize)$ per concept, for PPLM-BoW. In this case, $\Delta p(c, stepsize)$ collapses to 0 because both $p(man)$ and $p(woman)$ also collapse. The perplexity at the points where $\Delta p(c, stepsize) = 0$ is extremely high (over 250).
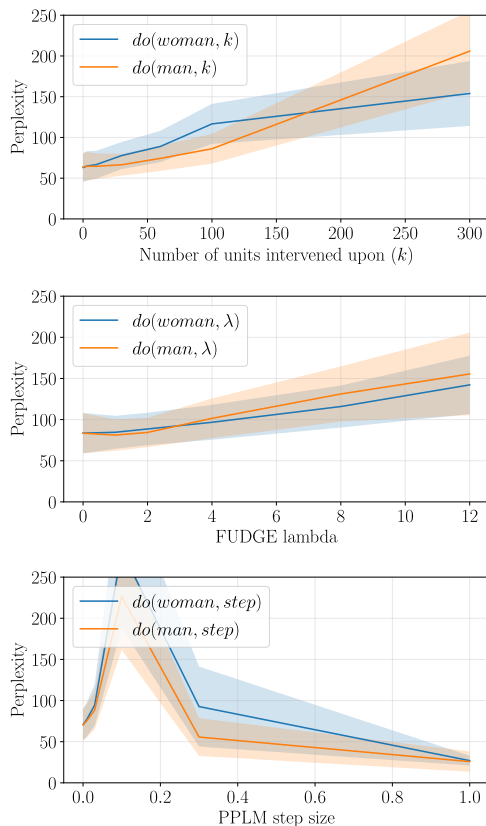
E.2. Perplexities

Figure 10: Perplexity as a function of the conditioning strength for concepts *man* and *woman*. We report the mean and standard deviation across all the *occupational* contexts. The perplexity of our method and FUDGE follow a similar trend. However, as we saw in Sec. 5.2, the perplexity at parity points is lower with our method. The perplexity of PPLM-BoW increases dramatically.

F. 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.`