FUDGE: Controlled Text Generation With Future Discriminators

Kevin Yang
UC Berkeley
yangk@berkeley.edu

Dan Klein
UC Berkeley
klein@berkeley.edu

Abstract

We propose Future Discriminators for Generation (FUDGE), a flexible and modular method for controlled text generation. Given a pre-existing model $\mathcal{G}$ for generating text from a distribution of interest, FUDGE enables conditioning on a desired attribute $a$ (for example, formality) while requiring access only to $\mathcal{G}$’s output logits. FUDGE learns an attribute predictor operating on a partial sequence, and uses this predictor’s outputs to adjust $\mathcal{G}$’s original probabilities. We show that FUDGE models terms corresponding to a Bayesian decomposition of the conditional distribution of $\mathcal{G}$ given attribute $a$. Moreover, FUDGE can easily compose predictors for multiple desired attributes. We evaluate FUDGE on three tasks — couplet completion in poetry, topic control in language generation, and formality change in machine translation — and observe gains in all three tasks.

1 Introduction

Recent advances in large pretrained language models allow us to generate increasingly realistic text by modeling a distribution $P(X)$ over natural language sequences $X$. The distribution $P(X)$ may be truly unconditional, as is common in language modeling, or it may model $P(X|I)$ conditioned on some input $I$, as in machine translation or summarization.

We are frequently interested in controlled text generation, the task of generating text conditioned on an additional desirable attribute $a$ which is not already built into $P(X)$. That is, we would like to model $P(X|a)$ (or possibly $P(X|I, a)$; henceforth we will drop $I$ from the notation for simplicity). For example, $P(X)$ may be a pretrained translation model for Spanish inputs $I$ to English outputs $X$, but we may wish to additionally constrain the outputs to possess a new attribute $a$, e.g., formality, which we did not optimize for during training.

Unfortunately, once we have already obtained an unconditioned $P(X)$ defined as the output distribution of some large generative model $\mathcal{G}$, it is nontrivial to add conditioning on a new attribute $a$ without either training a new model from scratch or fine-tuning with additional data. Although in principle we can trivially sample from $P(X|a)$ via rejection sampling from $P(X)$, rejection sampling may be highly inefficient in practice. On the other hand, while generating according to attribute $a$, $P(X)$ should be left otherwise intact: in the previous translation formality example, it is pointless to generate formless English outputs if they do not preserve the original Spanish meaning.

In light of these concerns, we propose Future Discriminators for Generation (FUDGE), a flexible and modular method for modeling $P(X|a)$ which accesses only the output probabilities of the generative model $\mathcal{G}$ which defines $P(X)$. FUDGE learns a binary predictor for whether attribute $a$ will become true in the complete future, based on an incomplete sequence prefix (Sec. 3). Multiplying the output probabilities of this predictor with $\mathcal{G}$’s original probabilities and then renormalizing yields a model for the desired $P(X|a)$ via Bayes’ Rule.

We run experiments on three controlled text generation tasks — couplet completion in poetry, topic control in language generation, and formality change in machine translation — showing our method’s broad applicability. Additionally, we demonstrate the modularity of FUDGE by composing multiple attribute constraints in both the couplet and topic control tasks. In our experiments, we find that FUDGE is highly effective at attribute control, outperforming both a baseline which directly fine-tunes $\mathcal{G}$ and also a strong gradient-based method (PPLM (Dathathri et al., 2019)). Our code is available at https://github.com/yangkevin2/naacl-2021-fudge-controlled-generation.

2 Related Work

Ideally, a controlled text generation method should efficiently control for $a$ while preserving $P(X)$
as much as possible. Recent work on controlled text generation has greatly advanced our ability to control for a required attribute \( a \) flexibly and cheaply, with varying degrees of modification to the original model \( \mathcal{G} \) which defines \( P(X) \).

One line of work fine-tunes a pretrained model for a desired attribute (Ficler and Goldberg, 2017; Yu et al., 2017; Ziegler et al., 2019). The result is a class-conditional language model (CCLM). However, it is difficult to isolate the desired attribute from the distribution shift between \( \mathcal{G} \) and the fine-tuning dataset (Hu et al., 2017; John et al., 2018; Lazaridou et al., 2020), i.e., it is nontrivial to preserve the desirable qualities of the \( P(X) \) modeled by \( \mathcal{G} \). One may also need to fine-tune separately for each attribute of interest. CTRL (Keskar et al., 2019) partially addresses these issues by providing 55 attribute control codes for a large language model trained from scratch, although this is expensive. Very recently, GEDE (Krause et al., 2020) achieves strong performance by using CCLM generators as discriminators, though it relies on several heuristics. More broadly, text generation models for style transfer (Hu et al., 2017; Lampl et al., 2018b; Dai et al., 2019a), summarization (See et al., 2017; Gehrmann et al., 2018; Zaheer et al., 2020), and machine translation (Lampl et al., 2018a; Ng et al., 2019; Lewis et al., 2019) can also be viewed as CCLM’s for different “attributes.”

A second type of approach instead conditions on a desired attribute by backpropagating gradients, either to directly modify model activations (Dathathri et al., 2019; Liu et al., 2020) or to find a trigger string (Wallace et al., 2019, 2020). Such methods often exhibit a high degree of attribute control, and can be used in adversarial attacks (Wallace et al., 2020). In fact, Subramani et al. (2019) show that by carefully modifying the latent state, one can cause the base \( \mathcal{G} \) to produce arbitrary outputs.

A third class of methods, referred to as weighted decoding (WD), assumes access only to \( P(X) \) (i.e., \( \mathcal{G} \)’s output logits), and operates directly on these logits (Ghazvininejad et al., 2017; Holtzman et al., 2018; Cohn-Gordon et al., 2018; Shen et al., 2019). Compared to other approaches, WD methods are relatively interpretable in how they obtain \( P(X|a) \) from \( P(X) \), but prior WD implementations have been observed to perform poorly in controlled text generation (See et al., 2019; Dathathri et al., 2019). While FUDGE shares a Bayesian motivation with other WD methods, FUDGE follows the Bayesian factorization more closely in implementation (Sec. 3). The key distinguishing feature of FUDGE is that it models whether attribute \( a \) will be true in the future, rather than in the present. We find that FUDGE substantially outperforms previous WD approaches in our experiments (Sec. 4.2).

3 Future Discriminators for Generation

We now explain the details of our proposed method, Future Discriminators for Generation (FUDGE), and show that it corresponds to modeling the desired conditional distribution \( P(X|a) \).

For a given language generation task, assume we have an autoregressive model \( \mathcal{G} \) (e.g., a large pretrained language model) which models \( P(x_i|x_{1:i−1}) \) for tokens \( x_1, \ldots, x_i \). Letting \( X = x_{1:n} \) denote a completed sequence, \( \mathcal{G} \) can sample from \( P(X) = P(x_{1:n}) \) one token at a time by factoring \( P(X) \):

\[
P(X) = \prod_{i=1}^{n} P(x_i|x_{1:i−1})
\]

To condition on attribute \( a \), we instead model \( P(X|a) \). This requires a model for \( P(x_i|x_{1:i−1}, a) \), modifying the previous factorization:

\[
P(X|a) = \prod_{i=1}^{n} P(x_i|x_{1:i−1}, a)
\]

If we model \( P(x_i|x_{1:i−1}, a) \) directly, we obtain a class-conditional language model (CCLM). We can learn the CCLM by e.g., fine-tuning \( \mathcal{G} \) depending on the available data, possibly with some structural modification to \( \mathcal{G} \) to accommodate conditioning.

However, FUDGE instead relies on the following Bayesian factorization, exchanging \( x_i \) and \( a \) conditioned on \( x_{1:i−1} \):

\[
P(x_i|x_{1:i−1}, a) \propto P(a|x_{1:i})P(x_i|x_{1:i−1})
\]

The second term is exactly the quantity modeled by the base \( \mathcal{G} \). It then suffices to model the first term, \( P(a|x_{1:i}) \), with a binary classifier \( B \) for the attribute \( a \) given a prefix \( x_{1:i} \). Intuitively, one can view \( B \) as rescoring or reranking \( \mathcal{G} \)’s original hypotheses.

We emphasize that although \( B \) takes a prefix \( x_{1:i} \) as input, it predicts whether attribute \( a \) will in the future be satisfied for the completed generation \( x_{1:n} \). For instance, suppose we are given a dataset of examples \( \{(x_{1:n}, a')\} \) with \( a' \) being the values of binary indicators for the desired \( a \) (i.e., if \( a \) is formality, then \( a' \) is 0 or 1 when \( x_{1:n} \) is informal
Figure 1: Illustration of one decoding step in FUDGE, for an example where the desired attribute \( a \) is formality. A large pretrained model \( G \) (dark blue) outputs unconditioned probabilities. Our binary predictor (red) predicts whether the eventual completed sequence will be formal for each possible continuation (computed for each candidate \( x_3 \), e.g., “want”; holding \( a \) fixed). The probabilities for each \( x_3 \) are multiplied (purple) and then renormalized to obtain \( P(x_3|x_{1:2}, a) \), from which we sample the next token \( x_3 = \)“prefer.”

or formal respectively). For each training example \((x_{1:n}, a')\), we train our classifier \( B \) using all pairs \((x_{1:i}, a')\); that is, we construct a separate example from each prefix \( x_{1:i} \) of \( x_{1:n} \). Our approach contrasts with previous methods such as Dathathri et al. (2019), which greedily optimize for \( a \) on the immediate extension \( x_{1:i+1} \). One particular benefit is that FUDGE naturally plans for the future: in the example for generating text on the “space” topic in Table 6, FUDGE writes about a “mysterious ship” despite “ship” itself not being in the given “space”-topic bag of words, because “mysterious ship” easily leads into a mention of one of the targeted “space” words (“Earth”). Similarly, in the first couplet completion example in Table 3, FUDGE needs to rhyme with “fear” after exactly ten syllables. After seven syllables, it could reasonably generate the word “clear,” but it first generates the adverb “pretty” in order to set up the generation of “clear” as the tenth syllable.

FUDGE’s implementation is shown schematically in Figure 1, and is quite simple in practice. FUDGE just needs to learn a \( B \) (red in Figure 1) sharing tokenization with \( G \) (dark blue). It then converts \( B \)'s output into probabilities (red table in Figure 1), and multiplies with the original output probabilities from \( G \) (dark blue table), to obtain unnormalized probabilities \( P(x_i, a|x_{1:i-1}) \) (purple table). Finally, renormalizing over the output vocabulary yields the desired distribution \( P(x_i|x_{1:i-1}, a) \). In practice, we operate in the log-probability space for numerical stability.

To improve computational efficiency, we typically choose \( B \) to be lightweight relative to \( G \). We also consider only the top 200 possibilities for \( x_i \) according to \( G \) at each step, as a cheap approximation to the full distribution, and find that this works well in practice. In each task in Sec. 4, running FUDGE on the test set takes no more than 15 minutes on a single Quadro RTX 6000 GPU.

Finally, as with other controlled generation approaches such as Dathathri et al. (2019), it is likely that augmenting FUDGE with reranking approaches such as rejection sampling could improve output quality at the cost of compute time, although we do not comprehensively evaluate such extensions in this work.

3.1 Advantages and Limitations

We highlight several additional potential advantages of FUDGE compared to directly modeling \( P(x_i|x_{1:i-1}, a) \) via e.g., a fine-tuned CCLM:

1. FUDGE requires access only to \( P(X) \) (i.e., \( G \)'s output logits) rather than \( G \) itself.

2. \( G \) can be freely swapped out for any other model that shares the same tokenization when larger models become available.

3. Given multiple conditionally independent attributes with predictors for each, FUDGE can easily condition on the combination of these attributes in a modular fashion by summing their output log-probabilities (Sec. 4.1, 4.2).

Unfortunately, like previous methods, FUDGE cannot fully guarantee that all outputs possess the desired attribute \( a \). In FUDGE’s case, this is due to the approximation inherent in modeling \( P(a|x_{1:i}) \), as well as only considering the top 200 possible \( x_i \) for computational efficiency.

\(^1\)See Appendix H for ablations on the top-200 pruning.
4 Experiments

We run experiments on a range of controlled text generation tasks to evaluate the effectiveness of our proposed method: poetry couplet completion (Sec. 4.1), topic-controlled language generation (Sec. 4.2), and machine translation formality change (Sec. 4.3). For each task we discuss the evaluation setup, the specific details of our method and baselines, and finally experimental results.

4.1 Poetry Couplet Completion

Table 1: An example couplet by William Shakespeare. Every second syllable is stressed, following iambic meter, and the last words of each line (see/thee) rhyme.

We begin with English poetry generation, a task that emphasizes well-formedness, and which has been studied in different forms by many previous works (Zhang and Lapata, 2014; Wang et al., 2016; Ghazvininejad et al., 2016, 2017). Our task here is couplet completion. Given the first line of an iambic pentameter couplet (e.g., Table 1), the model must generate a second line which (1) satisfies iambic pentameter, (2) rhymes with the first line, and (3) ends a sentence. The desired attribute \( a \) is defined as possessing all three properties, as evaluated by a rule-based checker \( F \) (Appendix A). Our test set is a collection of prefix lines of couplets, collected from the ending couplet of each of Shakespeare’s 154 sonnets.

Metrics. We consider four metrics.

1. **Success**, the fraction of couplet completions with the desired attribute \( a \), as checked by \( F \). This is the main metric.

2. **Grammaticality**, the probability of grammaticality given by a Roberta-based CoLA grammaticality model (Liu et al., 2019; Warstadt et al., 2019), averaged over all outputs.

3. **Perplexity** of the completion conditioned on the prefix. Following Dathathri et al. (2019), since our models use GPT2-Medium (Radford et al., 2019) as \( G \), we evaluate perplexity using GPT (Radford et al., 2018).\(^2\)

4. **Distinctness** of completions, measured as the number of unique unigrams, bigrams, and trigrams across all samples, divided by the total number of words (Li et al., 2015).

At test time, we decode until the model generates ten syllables followed by an end-of-sentence punctuation mark, or after the eleventh syllable (an automatic failure, since iambic pentameter requires exactly ten syllables).

Overall, because we define \( a \) using a rule-based \( F \) which is accessible during training, our formulation of couplet completion is a relatively clean task for evaluating the effectiveness of FUDGE.

4.1.1 Method and Baselines

**FUDGE Instantiation.** The obvious approach is to learn a predictor for \( F \) directly. However, the three components of \( a \) — meter, rhyme, and sentence-ending — should be roughly independent. Thus we assume conditional independence, and demonstrate the modularity of FUDGE by constructing three separate predictors to be combined at test time:

1. \( B_1(x_{1:t}, i) \) takes a text prefix \( x_{1:t} \), and predicts whether the completion \( x_{1:n} \) of prefix \( x_{1:t} \) will be in iambic meter. The model is an LSTM followed by a linear output layer.

2. \( B_2(x_{1:t}, t, r) \) takes prefix \( x_{1:t} \), the number of syllables \( t \) between \( x_i \) and \( x_n \) for \( n \geq i \), and a rhyme sound \( r \).\(^3\) It predicts whether the completion \( x_{1:n} \) has the rhyme sound \( r \) at the end of token \( x_n \). The model is an LSTM with attention dependent on \( t \) and \( r \), followed by a shallow feedforward network, and is trained via noise-contrastive estimation (Gutmann and Hyvärinen, 2010).\(^4\)

3. \( B_3(x_{1:t}, t) \) takes prefix \( x_{1:t} \) and the number of syllables \( t \) between \( x_i \) and \( x_n \) for \( n \geq i \), and predicts whether \( x_n \) ends a sentence. The model is an LSTM followed by a shallow feedforward network.

The predictors vary in architecture because \( B_2 \) and \( B_3 \) require inputs other than \( x_{1:t} \) — in truth, they are families of related predictors. We find that performance is not overly sensitive to the particulars of the predictor architectures (Appendix D).

\(^2\)See Appendix E for other perplexity measurements.

\(^3\)Two words have the same “rhyme sound” \( r \) if they rhyme according to the CMU Pronouncing Dictionary (Weide, 1998).

\(^4\)The output logits from \( B_2 \) are unnormalized, but this does not affect FUDGE after they are added to the output logits of \( G \) and softmaxed for sampling.
These truncations become inputs $x$ which enforces meter and rhyme at each decoding step using Table 2: Couplet completion results. Success (main metric), grammaticality, perplexity, and distinctness of different methods, tested on 154 prefix lines from Shakespeare sonnets. FUDGE substantially outperforms automated baselines on success and maintains high diversity, although quality unsurprisingly suffers compared to the base $G$ due to the difficult constraint $F$. Note Shakespeare’s work is often “incorrect” due to the narrowness of our metric $F$. He also scores poorly on text quality because our evaluation models are intended for more modern English.

To train the discriminators, we sample a dataset of 10 million generations of varied length from GPT2-Medium. From these generations, we sample random subsequences $x_{1:n}$ of roughly 10 to 30 syllables and truncate $t \leq 10$ ending syllables. These truncations become inputs $x_{1:t}$ to the predictors. For simplicity, we did not balance the class labels for e.g., the iambic predictor during training, although it is likely that doing so would improve performance.

At test time, we extract $r$ from the given first line of the couplet, and initialize $t = 10$, updating at each step. We then modify the output logits of $G$ by simply adding the log-probabilities from $B_1$, $B_2$, and $B_3$, demonstrating the ease of composing constraints in FUDGE.

**Baselines.** We compare to four baselines.\(^5\)

1. $G$, the original GPT2-Medium.
2. FINETUNE, a CCLM which finetunes $G$ on similar inputs to those used for $B_2$ in FUDGE. Since it is not obvious how to compose multiple CCLM’s for different attributes, we train a single CCLM for all desired properties together. We condition by prefixing the input with (1) whether the last 10 syllables of the original untruncated $x_{1:n}$ are iambic, (2) the rhyme sound at the end of $x_n$, and (3) whether a sentence ends with $x_n$. A special token is inserted 10 syllables from the end of $x_{1:n}$.
3. PPLM (Dathathri et al., 2019), which uses shallow predictors learned from $G$’s top-level hidden layer to modify $G$’s states toward increasing probability of the desired attribute via gradient ascent. We decompose the predictors into the same iambic, rhyme sound, and end-of-sentence predictors as for FUDGE, inserting an additional hidden layer in the shallow predictor when needed to incorporate additional input (the desired rhyme sound and/or number of syllables until end-of-sentence).
4. Shakespeare’s original couplet completions.

All non-Shakespeare methods use top-$k$ sampling with $k = 10$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correctness</th>
<th>Text Quality</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success ↑</td>
<td>Grammar ↑</td>
<td>Perplexity ↓</td>
</tr>
<tr>
<td>$G$</td>
<td>0</td>
<td>0.52</td>
<td>44.3 ± 42.2</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>0.21</td>
<td>0.44</td>
<td>55.8 ± 98.3</td>
</tr>
<tr>
<td>PPLM</td>
<td>0</td>
<td>0.54</td>
<td>60.8 ± 66.1</td>
</tr>
<tr>
<td>FUDGE</td>
<td><strong>0.44</strong></td>
<td><strong>0.44</strong></td>
<td><strong>70.9 ± 89.4</strong></td>
</tr>
<tr>
<td>Shakespeare</td>
<td><strong>0.45</strong></td>
<td>0.29</td>
<td>333.8 ± 418.9</td>
</tr>
</tbody>
</table>

Table 2: Couplet completion results. Success (main metric), grammaticality, perplexity, and distinctness of different methods, tested on 154 prefix lines from Shakespeare sonnets. FUDGE substantially outperforms automated baselines on success and maintains high diversity, although quality unsurprisingly suffers compared to the base $G$ due to the difficult constraint $F$. Note Shakespeare’s work is often “incorrect” due to the narrowness of our metric $F$. He also scores poorly on text quality because our evaluation models are intended for more modern English.

\(^5\)A system like Hafez (Ghazvininejad et al., 2016, 2017), which enforces meter and rhyme at each decoding step using a hard constraint, could achieve perfect success rate. However, this approach relies on the meter and rhyme attributes being “prefix-checkable” at the word level: one can guarantee success by simply never selecting a word which immediately violates the constraint. This is often the case for simple rule-based constraints, but not for many other interesting attributes, such as the topic and formality attributes in our subsequent experiments. To preserve generality, FUDGE does not rely on this “prefix-checkable” property, and neither do our baselines.

\[^{1}\] To be continued...
riously. Similarly, the grammaticality and perplexity metrics are designed for our automated baselines, and thus assign poor scores to Shakespeare’s antiquated and flowery style.

FUDGE also maintains relatively fluent generation despite lower grammaticality and perplexity compared to $G$. See Table 3 for two successful examples. Interestingly, FUDGE also increases diversity compared to $G$, perhaps due to the difficult constraint $F$ forcing FUDGE to use lower-probability regions of the base distribution $P(X)$.

| And even thence thou wilt be stol’n, I fear, for this shall be the end. That’s pretty clear. |
| Or, if they sleep, thy picture in my sight I will be glad to look upon the night. |

Table 3: Two examples of successful couplet completions (in purple) generated by FUDGE.

Finally, it is possible (and trivial) to adjust the conditioning strength in FUDGE by multiplying the binary predictors’ output logits by a constant. However, this deviates from our Bayesian factorization of $P(X|\alpha)$, and we do not do so.

4.2 Topic-Controlled Language Generation

Next, we explore topic control in English language generation. The desired attribute $a$ is to be on-topic for a given topic, such as science or politics. To facilitate comparison with prior work, we largely follow the setup of PPLM (Dathathri et al., 2019): the model is provided an approximation to the topic at test time, in the form of a bag of on-topic words $W$. The goal is to sample text according to the topic approximated by $W$, starting from a generic prefix. There are 7 topics (space, politics, military, legal, science, religion, and computers) and 20 prefixes, and the model generates 3 80-token samples from each topic-prefix pair, for a total of 420 generations.

Metrics. Unfortunately, we cannot easily construct a rule-based $F$ for being “on-topic.” Additionally, following the evaluation procedure of prior work such as (Dathathri et al., 2019), we run human evaluations via Amazon Mechanical Turk for FUDGE against each baseline, comparing topic control and fluency. For each pairwise comparison, we ask 3 workers to evaluate each of 420 paired outputs. Workers were asked to mark which generation is more on topic (first, second, both, or neither), and to rate each generation’s fluency on a Likert scale from 1 to 5. We report the average fraction of outputs marked as on-topic as well as the average fluency rating for each method.

1. Success, the average number of distinct words in a heldout bag $W'$ which appear in the model output. Specifically, for each word in $W$, we add to $W'$ the closest GloVe (Pennington et al., 2014) word by cosine similarity, such that the new word does not contain (and is not contained by) any word in $W$. (This excludes e.g., most plurals.) Usage of distinct words in $W'$ measures the model’s ability to generalize $W$ to other on-topic words, of which $W'$ is a non-exhaustive set. This is our main metric.

2. Grammaticality, identical to the couplet task.

3. Perplexity, identical to the couplet task.

4. Distinctness, defined as in the couplet task. However, it is calculated separately within the 60 generations for each topic, and then averaged over the 7 topics.

Additionally, following the evaluation procedure of prior work such as (Dathathri et al., 2019), we run human evaluations via Amazon Mechanical Turk for FUDGE against each baseline, comparing topic control and fluency. For each pairwise comparison, we ask 3 workers to evaluate each of 420 paired outputs. Workers were asked to mark which generation is more on topic (first, second, both, or neither), and to rate each generation’s fluency on a Likert scale from 1 to 5. We report the average fraction of outputs marked as on-topic as well as the average fluency rating for each method.

4.2.1 Method and Baselines

FUDGE Instantiation. Since we model topics as bags of words, FUDGE uses a binary predictor $B(x_{1:i}, w)$ which takes a prefix $x_{1:i}$ and word $w$, and classifies whether $w$ appears in the future $x_{i:n}$ for $n \geq i$. (Since it is desirable to stay on topic even after successfully getting on topic, we use $x_{i:n}$ rather than $x_{1:n}$.) Training examples $(x_{1:i}, w)$ are sampled from the same dataset of 10 million GPT2-Medium generations used for the couplet task, and $B$ is trained using noise-contrastive estimation. $B$
Table 4: Topic control results. Success (main metric), grammaticality, perplexity, and distinctness for different methods. FINETUNE and WDEC often degenerate into repeating the given bag of words \( W \); this is ill-captured by perplexity, but results in poor grammaticality and distinctness. FUDGE substantially outperforms all baselines on success, including the strong gradient-based PPLM baseline, while preserving high quality and diversity.

<table>
<thead>
<tr>
<th>Method</th>
<th>On-Topic Success ↑</th>
<th>Text Quality Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Diversity Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>0.22</td>
<td>0.81</td>
<td>37.1 ± 26.9</td>
<td>0.35</td>
<td>0.78</td>
<td>0.92</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>0.28</td>
<td>0.74</td>
<td>24.9 ± 13.7</td>
<td>0.29</td>
<td>0.70</td>
<td>0.88</td>
</tr>
<tr>
<td>WDEC</td>
<td>0.14</td>
<td>0.59</td>
<td>33.8 ± 33.7</td>
<td>0.16</td>
<td>0.42</td>
<td>0.55</td>
</tr>
<tr>
<td>PPLM</td>
<td>0.48</td>
<td>0.78</td>
<td>43.1 ± 23.7</td>
<td>0.35</td>
<td>0.78</td>
<td>0.92</td>
</tr>
<tr>
<td>FUDGE</td>
<td>0.59</td>
<td>0.79</td>
<td>40.7 ± 26.3</td>
<td>0.34</td>
<td>0.75</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 5: Topic control human evaluations, pairwise comparisons. FUDGE achieves a substantially higher fraction of on-topic outputs compared to each baseline, in addition to higher average fluency (rated 1 to 5).

<table>
<thead>
<tr>
<th>Method</th>
<th>Topic</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>0.16</td>
<td>4.11</td>
</tr>
<tr>
<td>FUDGE</td>
<td>0.78</td>
<td>4.30</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>0.24</td>
<td>3.95</td>
</tr>
<tr>
<td>FUDGE</td>
<td>0.76</td>
<td>4.22</td>
</tr>
<tr>
<td>WDEC</td>
<td>0.49</td>
<td>2.50</td>
</tr>
<tr>
<td>FUDGE</td>
<td>0.75</td>
<td>4.21</td>
</tr>
<tr>
<td>PPLM</td>
<td>0.45</td>
<td>4.05</td>
</tr>
<tr>
<td>FUDGE</td>
<td>0.74</td>
<td>4.16</td>
</tr>
</tbody>
</table>

is a lightweight LSTM-based classifier similar to \( B_2 \) from the couplet task.

At test time, we can compose individual-word constraints if we assume conditional independence between words (although this may be imperfect). Given a bag of \( N \) words \( \{w_1 \ldots w_N\} \) and prefix \( x_{1:i} \), we could condition on all words in the bag appearing in the future by adding all log-probabilities \( \log P(w_1|x_{1:i}) \ldots \log P(w_N|x_{1:i}) \) to \( G \)'s logits. However, topic control does not require every word to appear; perhaps some number \( \lambda \) of on-topic words is enough to be "on-topic." Therefore, we model the topic constraint as selecting a random subset of \( \lambda \) words from the original bag, and requiring that only those \( \lambda \) words all appear. Since each of the \( N \) words is selected with probability \( \lambda N \), the quantity we add to the base \( G \) logits is \( \frac{\lambda}{N} \sum_{j=1}^{N} \log P(w_j|x_{1:i}) \) in expectation. In our experiments we use \( \lambda = 4 \), based on a fantasy-topic bag of words used for validation (Appendix C).

Baselines. We compare to four baselines.

1. \( G \), the original GPT2-Medium.
2. FINETUNE, which finetunes \( G \) on the same inputs used for FUDGE. The future word is given as a prefix for conditioning. At test time, we compute logits for each prefix in the given \( W \) and use the average as the true logits, as an ad hoc way to condition on the full \( W \).
3. WDEC, a simple weighted decoding implementation which greedily considers only the immediate next token when optimizing for \( a \). Instead of using \( B \), WDEC just adds a fixed \( \lambda_{WDEC} \) to the logit for each word in \( W \). Note WDEC requires \( a \) to be well-defined at the token level, so it is not easily transferable to certain tasks (e.g., couplet completion).
4. PPLM (Dathathri et al., 2019), which modifies the activations of \( G \) to make the desired bag of words more likely at the immediate next position. We use their method without reranking for fair comparison.

All methods use top-\( k \) sampling with \( k = 10 \), following Dathathri et al. (2019)’s setup.

4.2.2 Results

FUDGE achieves the highest success by a substantial margin (Table 4), and outperforms all baselines on human evaluations in both topic relevance and fluency (Table 5). FUDGE simultaneously preserves high quality and diversity according to automated metrics. Table 6 shows two examples.

Unsurprisingly, \( G \) performs poorly on success. WDEC and FINETUNE also perform poorly, in success and especially in distinctness. WDEC frequently degenerates into repeating the given words in the bag \( W \), despite tuning \( \lambda_{WDEC} \) (Appendix C).
The desired attribute \( a \) is formality, but we cannot sacrifice the source sentence’s meaning. The latter requirement makes generation more constrained than in the couplet and topic tasks, so perplexity and distinctness are less relevant. Instead, we use the following:

1. **BLEU Score** (Papineni et al., 2002), using two of Salesky et al. (2019)’s fluent references per test example. This is our main metric.

2. **Formality**, the average probability that the model’s outputs are formal, according to an evaluator trained on the Family/Relationships domain of the GYAFC formality dataset (Rao and Tetreault, 2018). The evaluator is an LSTM followed by a linear layer.

### 4.3.1 Method and Baselines

**FUDGE Instantiation.** We assume that the attribute \( a \), formality, is conditionally independent from the original conditioning in \( \mathcal{G} \), i.e., the meaning of the Spanish input. FUDGE uses a binary predictor \( B(x_{1:n}) \) which classifies whether the text starting with prefix \( x_{1:n} \) is written in a formal style. \( B \) is an LSTM followed by a linear layer, trained on the Entertainment/Music domain of GYAFC.

At test time, FUDGE directly augments \( \mathcal{G} \)’s logits using log-probabilities from \( B \). If it’s business, then you are a student for a PHD. The masters is exactly that.

## 4.3 Machine Translation Formality Change

Finally, we turn to a somewhat more challenging task, changing formality in machine translation — specifically, from informal to formal. Given a source sentence written in an informal and conversational style, the goal is to output a translation which is also more formal. We test on the Fisher and CALLHOME Spanish–English Speech Translation Corpus (Post et al., 2013), a collection of transcribed Spanish conversations with English translations. Both the source Spanish and target English are highly informal and disfluent. Salesky et al. (2019) augment the Fisher dataset with additional parallel English translations, rewritten to be more fluent (and hence more formal); see Table 7 for an example. Our task is to translate the original informal Spanish to into more formal English. However, we assume that Salesky et al. (2019)’s fluent references are unavailable during training.
transformer model for Spanish-English. We evaluate both when $\mathcal{G}$ is fine-tuned on the original Fisher training dataset (i.e., using the original targets, not Salesky et al. (2019)’s more fluent targets) as well as zero-shot with no fine-tuning, which is challenging due to the highly informal and disfluent text.

**Baselines.** We compare to two baselines.

1. $\mathcal{G}$, the original machine translation model.
2. $\mathcal{G} + \text{ST}$, a pipeline consisting of $\mathcal{G}$ followed by a style transfer model. Our style transfer model is T5 (Raffel et al., 2020), fine-tuned on the same GYAFC Entertainment/Music domain that we used to train $\mathcal{B}$ in FUDGE.

Since we do not assume access to Salesky et al. (2019)’s more formal targets during training, it is difficult to apply PPLM to this task: PPLM’s predictor would operate on the pretrained translation model’s hidden states, thus requiring a Spanish-English translation dataset with both formal and informal English.\(^8\) We omit FINETUNE for the same reason. In contrast, FUDGE requires only the original English dataset with formality annotations. All methods use greedy decoding.

**4.3.2 Results**

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU ↑ Form. ↑</th>
<th>BLEU ↑ Form. ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G}$ (No fine-tune)</td>
<td>16.98 0.45</td>
<td>22.03 0.41</td>
</tr>
<tr>
<td>$\mathcal{G}$ (Fine-tune)</td>
<td>7.87 0.96</td>
<td>9.63 0.97</td>
</tr>
<tr>
<td>FUDGE</td>
<td>17.96 0.51</td>
<td>22.18 0.48</td>
</tr>
</tbody>
</table>

Table 8: Machine translation formality results. BLEU (main metric) and average formality for different methods, with and without fine-tuning $\mathcal{G}$ on the Fisher domain. FUDGE increases the formality of translations compared to the base model $\mathcal{G}$ while preserving or increasing BLEU score. Conversely, $\mathcal{G}$ with style transfer overfits to the formality data, resulting in near-perfect formality but losing the original meaning.

As shown in Table 8, FUDGE increases the formality of outputs compared to $\mathcal{G}$, even though the test-time formality predictor is trained on a different domain (Family/Relationships, rather than Entertainment/Music). Note that formality unsurprisingly decreases after fine-tuning $\mathcal{G}$, simply due to the informality of the fine-tuning dataset. As in the couplet task, one could adjust the strength of the formality control in FUDGE, although this is unprincipled from the view of modeling $P(X|\alpha)$.

Moreover, while FUDGE and $\mathcal{G}$ achieve similar BLEU after fine-tuning $\mathcal{G}$, FUDGE achieves higher BLEU compared to $\mathcal{G}$ when $\mathcal{G}$ is not fine-tuned on the Fisher training set. In the latter case, controlling for formality somewhat remedies the struggles of $\mathcal{G}$ when not fine-tuned on such disfluent text.

In contrast, the $\mathcal{G} + \text{ST}$ baseline achieves near-perfect formality but less than half the BLEU of $\mathcal{G}$, due to the style transfer model overfitting to the GYAFC Entertainment/Music dataset. This is similar to the distribution shift issue that we observed in topic control for FINETUNE, an issue which FUDGE largely avoids. Nevertheless, there remains substantial room for improvement on this difficult task.

<table>
<thead>
<tr>
<th>Reference In England, for example?</th>
<th>Spanish que era lo que tenía que hacer</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE</td>
<td>$\mathcal{G}$ that was what you had to do</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>Reference What’s there to do?</td>
</tr>
<tr>
<td>FUDGE Ah. in my, in England, for example.</td>
<td>Spanish ah en mi en inglaterra por ejemplo</td>
</tr>
<tr>
<td>Reference In England, for example?</td>
<td>$\mathcal{G}$ Ah, in England, for example.</td>
</tr>
</tbody>
</table>

Table 9: Example translations by $\mathcal{G}$ (fine-tuned on the Fisher dataset) and FUDGE using the same $\mathcal{G}$. Original Spanish and Salesky et al. (2019) references also shown. In this setting, FUDGE achieves similar BLEU to $\mathcal{G}$ while increasing formality. While FUDGE often simply corrects punctuation or capitalization (top), it also makes more complex adjustments (bottom). More examples in Appendix L.

**5 Discussion**

FUDGE is a principled approach to controlled text generation which models $P(X|\alpha)$ by closely following a Bayesian factorization, thus preserving the base $P(X)$ as much as possible. FUDGE achieves strong performance on a wide range of different tasks: poetry couplet completion, topic control, and informal-to-formal machine translation. Additionally, FUDGE can easily compose different attributes in a modular fashion: the meter, rhyme, and end-of-sentence constraints for couplet completion, and the individual words within each topic bag for topic control. In principle, FUDGE is applicable to any controlled generation task where we can train discriminators for the desired attribute or attributes.

\(^8\)We nevertheless ran PPLM in a somewhat convoluted setup, but found that it performed poorly (Appendix B).
6 Ethics of Controlled Text Generation

We recognize that strong controlled generation methods have the potential to produce harmful outputs and/or misinformation when used adversarially (Wallace et al., 2019, 2020). However, such methods can also be a powerful tool for mitigating harmful biases learned by large pretrained language models (Radford et al., 2019; Brown et al., 2020), for example by detoxifying language (Dathathri et al., 2019; Krause et al., 2020). Overall, we believe it is still beneficial to continue research into general controlled text generation methods such as FUDGE.

Acknowledgements

We thank Daniel Fried, David Gaddy, Eric Wallace, Kevin Lin, Nicholas Tomlin, Ruiqi Zhong, and the three anonymous reviewers for their helpful comments and feedback, which aided us in greatly improving the paper. We also thank the authors of Dathathri et al. (2019) for clarifying our questions about their topic control setup. This work was supported by Berkeley AI Research, DARPA under agreement HR00112020054, and the NSF through a fellowship to the first author. The content does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

References


A Details of $\mathcal{F}$ for Couplet Completion

We provide the full details of the function $\mathcal{F}$ we use to check iambic pentameter, rhyme, and sentence-ending in our couplet completion task. Note that iambic pentameter consists of two components: iambic meter as well as containing exactly ten syllables.

1. *Iambic meter:* Given a phrase, we obtain the sequence of stresses (0 for unstressed, 1 for stressed, 2 for secondary stress) for each word, according to the CMU Pronouncing Dictionary (Weide, 1998). If any word does not exist in the dictionary (almost never for non-Shakespeare methods) we return False. We treat 2 as ambiguous stress, and additionally change 1 to 2 for any monosyllabic words, i.e. we allow monosyllabic stressed words to be unstressed but not vice versa. Finally, we check that all syllables at even indices (0-indexed) are unstressed or ambiguous, and all syllables at odd indices are stressed or ambiguous.

2. *Number of syllables:* We count the number of syllables in each word based on the number of stresses according to the CMU Pronouncing Dictionary. If a word does not exist in the dictionary, we estimate the number of syllables by rounding the number of letters divided by 3 to the nearest integer.

3. *Rhyme:* Two words rhyme if and only if they both exist in the CMU Pronouncing Dictionary and are a perfect rhyme according to the dictionary.

4. *Sentence-ending:* We check if the output ends with a period, question mark, or exclamation mark.

Of course, both FUDGE and FINETUNE will fit to whatever output is given by $\mathcal{F}$. The purpose of the couplet task is to check FUDGE’s ability to fit a difficult well-formedness constraint. We simply design an $\mathcal{F}$ that corresponds to true iambic pentameter rhymes in most cases.

### A.1 Shakespeare Evaluation

Shakespeare himself performs somewhat poorly according to $\mathcal{F}$, which is designed with the automated baselines in mind, not for Shakespeare. (The same is true for our grammaticality and perplexity metrics.)

One source of error is words which are out-of-vocabulary for the CMU Pronouncing Dictionary. Such words are almost never generated by either FUDGE or our automated baselines, but appear in a fifth of Shakespeare’s lines, resulting in failures on the iambic meter and syllable checks.

Nevertheless, most of Shakespeare’s “errors” are the result of real — though slight — deviations from our very strict definitions of meter and rhyme. In particular, he frequently (1) elides syllables to fit meter, and (2) uses loose rhymes; both “error” types are likely exacerbated by differences between archaic and modern pronunciations. The example in Table 10 illustrates both types of “errors.” Although such deviations are often acceptable to a human, they are difficult to capture in an automatic metric, and we do not allow such deviations in $\mathcal{F}$. Again, Shakespeare is only included as a whimsical point of reference, and not as a serious baseline to be compared to.

| But here’s the joy; my friend and I are one;  
| Sweet flattery! then she loves but me alone. |

Table 10: An example couplet by William Shakespeare, illustrating two common deviations from the narrow definition of correctness we use in $\mathcal{F}$. For this example to follow iambic meter, one must read “flattery” in only two syllables. Moreover, “one/alone” is a loose (non-perfect) rhyme, at least in modern English.

### B PPLM Baseline in Machine Translation

As discussed in the main text, it is difficult to apply PPLM in our machine translation setup, in which $P(a|X)$ is learned from an English formality dataset without parallel Spanish. Since $P(X)$ is a Spanish-English translation model, we must obtain hidden states for training PPLM’s $P(a|X)$ by first “backtranslating” English into Spanish, accessing a second pretrained translator. For this purpose we use a second pretrained Marian transformer from HuggingFace (https://huggingface.co/Helsinki-NLP/opus-mt-en-es). Additionally, we needed to tune their suggested hyperparameters.

During evaluation, we observe that PPLM makes some reasonable modifications for formality compared to the base $P(X)$, like changing “hard” to “difficult,” but such improvements are also accompanied by occasional disfluencies and/or repetitions (although such problems plague all methods to
some degree). Overall, while PPLM achieves similar BLEU to FUDGE, it is substantially less formal (Table 11).

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU ↑</th>
<th>Form. ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPLM</td>
<td>21.94</td>
<td>0.40</td>
</tr>
<tr>
<td>FUDGE</td>
<td>22.18</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 11: PPLM baseline in machine translation formality on the fine-tuned G

C Hyperparameter Choices

FUDGE has essentially one hyperparameter in our topic control task, \( \lambda \), which controls the strength of conditioning and corresponds to the number of words in the bag which should appear in the future.

To choose \( \lambda \) in topic control, we used a separate validation bag of words (on the topic of fantasy; Appendix K.4) to select a reasonable \( \lambda \) for our main paper experiments (\( \lambda = 4 \)). Unlike in the main paper where we use heldout bags \( W' \) to measure success, during validation we simply use the original bag. We use a set of 60 generations, considering values ranging from 1 to 6 (Table 12), although the result may be somewhat noisy. Of course, different choices of \( \lambda \) result in different tradeoffs (Appendix G).

We also optimized the conditioning strength \( \lambda_{WDEC} \) for the WDEC baseline on the same fantasy bag of words, considering values ranging from 1 to 32. We selected the only value (4) which achieved reasonable success without a total collapse in diversity (Table 13), but diversity still collapsed when tested on our seven main test bags of words.

We do not optimize any model hyperparameters in the couplet completion and informal-to-formal translation tasks. LSTM’s and feedforward networks are 3 layers (including the output layer of dimension 1) and 300-dimensional unless otherwise specified. They are bidirectional (150-dimensional in each direction) for the couplet rhyme predictor and the topic control future words predictor, and otherwise unidirectional. Attention mechanisms use key-query-value attention. For the rhyme and future words predictors the output hidden state is multiplied element-wise by the embedding of the rhyme sound or future word, then concatenated to the embeddings, before the final feedforward network. Since a selling point of our method is the lightweight process of constructing and training predictors, noise-contrastive estimation is a natural choice for the rhyme and future word predictors: we avoid softmaxing over the output dimension during training. (This is primarily relevant for the future word predictor, as the number of distinct rhyme sounds is not too large, but we use noise-contrastive estimation for both for consistency’s sake.)

For the PPLM baseline, we used step size 0.01 for both couplet completion and MT after tuning, and kept their other hyperparameters fixed. For topic control we simply evaluated their provided generations instead of rerunning their model.

D Ablations on Predictor Architectures

Some variation in predictor architectures is necessary due to the diversity of our tasks (as evidenced by the difficulties in adapting PPLM). Specifically, while our core predictor architecture is word embeddings followed by LSTM and output layer, task-specific architectures vary because some “predictors” are actually families of related predictors. We model such families as a single predictor taking additional input (e.g., rhyme sound in poetry); this is needed in our poetry and topic tasks.

On these two tasks, we provide ablations with more homogenized predictors: additional inputs are simply embedded and concatenated to each input word embedding. The difference is relatively small in both cases (Tables 14 and 15). FUDGE-MOD indicates the ablated version of FUDGE.

E Alternative Perplexity Measurements

On the couplet completion task, we additionally measure perplexity using Transformer-XL (Dai et al., 2019b) and using a GPT model fine-tuned on Shakespearean language as generated by (Lau et al., 2018). We measure using Transformer-XL on the topic control task as well. Relative perplexities between most models remain largely similar when switching between GPT and Transformer-XL, with a few exceptions. Compared to the base GPT, Shakespeare’s perplexity naturally decreases while other models’ perplexities increase when measured with Shakespeare-finetuned GPT. The highly repetitive and disfluent WDEC baseline is rightly punished for this behavior when measured by Transformer-XL. PPLM also obtains slightly lower perplexity than FUDGE on topic control when
### Table 12: Results from 60 samples for FUDGE with different \( \lambda \) on topic control for a validation fantasy-topic bag of words. Note that during validation only, success directly measures use rate of words in the given bag \( W \), not a heldout bag \( W' \) as in the main paper.

<table>
<thead>
<tr>
<th>Method</th>
<th>Success ↑</th>
<th>Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE, ( \lambda = 1 )</td>
<td>0.05</td>
<td>0.80</td>
<td>38.3 ± 32.6</td>
<td>0.36</td>
<td>0.78</td>
<td>0.92</td>
</tr>
<tr>
<td>FUDGE, ( \lambda = 2 )</td>
<td>0.10</td>
<td>0.76</td>
<td>31.1 ± 17.1</td>
<td>0.35</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE, ( \lambda = 4 )</td>
<td>0.28</td>
<td>0.76</td>
<td>40.1 ± 27.6</td>
<td>0.37</td>
<td>0.77</td>
<td>0.92</td>
</tr>
<tr>
<td>FUDGE, ( \lambda = 6 )</td>
<td>0.30</td>
<td>0.72</td>
<td>46.9 ± 29.9</td>
<td>0.38</td>
<td>0.77</td>
<td>0.91</td>
</tr>
</tbody>
</table>

### Table 13: Results from 60 samples for WDEC with different \( \lambda_{WDEC} \) on topic control for a validation fantasy-topic bag of words. Note that during validation only, success directly measures use rate of words in the given bag \( W \), not a heldout bag \( W' \) as in the main paper.

<table>
<thead>
<tr>
<th>Method</th>
<th>Success ↑</th>
<th>Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 1 )</td>
<td>0.02</td>
<td>0.83</td>
<td>34.5 ± 23.5</td>
<td>0.36</td>
<td>0.78</td>
<td>0.91</td>
</tr>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 2 )</td>
<td>0.02</td>
<td>0.83</td>
<td>34.9 ± 23.8</td>
<td>0.36</td>
<td>0.78</td>
<td>0.91</td>
</tr>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 4 )</td>
<td>0.57</td>
<td>0.79</td>
<td>34.7 ± 23.6</td>
<td>0.33</td>
<td>0.74</td>
<td>0.86</td>
</tr>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 8 )</td>
<td>1.90</td>
<td>0.47</td>
<td>14.5 ± 19.2</td>
<td>0.04</td>
<td>0.09</td>
<td>0.12</td>
</tr>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 16 )</td>
<td>2.32</td>
<td>0.40</td>
<td>8.4 ± 9.6</td>
<td>0.01</td>
<td>0.04</td>
<td>0.06</td>
</tr>
<tr>
<td>WDEC, ( \lambda_{WDEC} = 32 )</td>
<td>2.35</td>
<td>0.41</td>
<td>7.5 ± 9.1</td>
<td>0.01</td>
<td>0.04</td>
<td>0.06</td>
</tr>
</tbody>
</table>

### Table 14: Ablation of FUDGE with a modified predictor architecture on couplet completion.

<table>
<thead>
<tr>
<th>Method</th>
<th>Success ↑</th>
<th>Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE</td>
<td>0.44</td>
<td>0.44</td>
<td>70.9 ± 89.4</td>
<td>0.40</td>
<td>0.79</td>
<td>0.78</td>
</tr>
<tr>
<td>FUDGE MOD</td>
<td>0.39</td>
<td>0.43</td>
<td>72.1 ± 66.3</td>
<td>0.41</td>
<td>0.79</td>
<td>0.77</td>
</tr>
</tbody>
</table>

### Table 15: Ablation of FUDGE with a modified predictor architecture on topic control.

<table>
<thead>
<tr>
<th>Method</th>
<th>Success ↑</th>
<th>Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE</td>
<td>0.59</td>
<td>0.79</td>
<td>40.7 ± 26.3</td>
<td>0.34</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE MOD</td>
<td>0.62</td>
<td>0.77</td>
<td>47.8 ± 51.3</td>
<td>0.33</td>
<td>0.73</td>
<td>0.88</td>
</tr>
</tbody>
</table>

### Table 16: Different perplexity measurements on couplet completion, using GPT, Transformer-XL (TFXL), and GPT fine-tuned with Shakespearean language (GPT-Shakespeare). Main paper results use GPT.

<table>
<thead>
<tr>
<th>Method</th>
<th>GPT</th>
<th>TFXL</th>
<th>GPT-Shakespeare</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{G} )</td>
<td>44.3 ± 42.2</td>
<td>84.8 ± 111.0</td>
<td>72.9 ± 62.0</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>55.8 ± 98.3</td>
<td>76.1 ± 64.6</td>
<td>69.0 ± 92.5</td>
</tr>
<tr>
<td>PPLM</td>
<td>60.8 ± 66.1</td>
<td>111.5 ± 150.4</td>
<td>120.0 ± 243.8</td>
</tr>
<tr>
<td>FUDGE</td>
<td>70.9 ± 89.4</td>
<td>137.5 ± 170.9</td>
<td>96.2 ± 117.3</td>
</tr>
<tr>
<td>Shakespeare</td>
<td>333.8 ± 418.9</td>
<td>1879.5 ± 6088.1</td>
<td>195.1 ± 228.9</td>
</tr>
</tbody>
</table>
Table 17: Different perplexity measurements on topic control, using GPT and Transformer-XL (TFXL). Main paper results use GPT.

<table>
<thead>
<tr>
<th>Method</th>
<th>GPT</th>
<th>TFXL</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>37.1 ± 26.9</td>
<td>34.1 ± 25.2</td>
</tr>
<tr>
<td>FINETUNE</td>
<td>24.9 ± 13.7</td>
<td>27.7 ± 15.6</td>
</tr>
<tr>
<td>WDEC</td>
<td>33.8 ± 33.7</td>
<td>7802.4 ± 29942.6</td>
</tr>
<tr>
<td>PPLM</td>
<td>43.1 ± 23.7</td>
<td>38.7 ± 21.0</td>
</tr>
<tr>
<td>FUDGE</td>
<td>40.7 ± 26.3</td>
<td>42.8 ± 46.9</td>
</tr>
</tbody>
</table>

Table 18: The first generation by FUDGE using $\lambda = 2$ on the space, politics, and military topics given the prefix “The issue focused on.” Words in the given bag are highlighted in red, and other related words in cyan.

G Effect of Varying Topic Control Strength

Although we use $\lambda = 4$ for FUDGE in our main paper experiments for topic control, we experiment here with varying the conditioning strength. Specifically, we experiment with $\lambda = 2$ and $\lambda = 8$. The conditioning is unsurprisingly stronger as $\lambda$ increases, as shown quantitatively in Table 19, although the perplexity increases as well.

We also provide some example generations for $\lambda = 2$ and $\lambda = 8$ in Tables 18 and 21, for the same prompts and topics as in Table 6 for $\lambda = 4$ in the main text. The $\lambda = 8$ generations remain mostly fluent and interesting, despite their worse grammaticality and perplexity.

H Effect of Varying Candidate Pruning

For computational efficiency, we only feed the top 200 candidates returned by G into FUDGE’s predictor when predicting each next token. Here, we ablate on this number in our topic control setting, testing 100 and 400 (Table 20).

I Additional Couplet Completion Examples

We provide some additional examples of FUDGE and baselines on our couplet completion task in Table 22.

We also show some unsuccessful examples for FUDGE in 23. Overall, we find that most errors are due to the rhyme and ten-syllable end of sentence constraints, or due to Shakespeare’s prefix ending in a word not in the CMU Pronouncing Dictionary (e.g., “prognosticate” in the table). FUDGE also sometimes overgenerates punctuation at the end of a sentence.

J Additional Topic Control Examples

In Tables 24, 25, and 26 we show additional example generations by our method using the same...
Table 19: FUDGE results for different values of $\lambda$ on the main 7 topics and 20 prefixes. Success and perplexity both increase as the conditioning strength $\lambda$ increases. Our main paper experiments use $\lambda = 4$.

<table>
<thead>
<tr>
<th>Method</th>
<th>On-Topic Success ↑</th>
<th>Text Quality Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Diversity Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE, $\lambda = 2$</td>
<td>0.49</td>
<td>0.79</td>
<td>38.6 ± 24.1</td>
<td>0.36</td>
<td>0.76</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE, $\lambda = 4$</td>
<td>0.59</td>
<td>0.79</td>
<td>40.7 ± 26.3</td>
<td>0.34</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE, $\lambda = 8$</td>
<td>0.76</td>
<td>0.74</td>
<td>56.5 ± 39.0</td>
<td>0.35</td>
<td>0.75</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Table 20: FUDGE results for different numbers of candidates fed through FUDGE’s predictor. Main paper results use 200.

<table>
<thead>
<tr>
<th>Method</th>
<th>On-Topic Success ↑</th>
<th>Text Quality Grammar ↑</th>
<th>Perplexity ↓</th>
<th>Diversity Dist-1 ↑</th>
<th>Dist-2 ↑</th>
<th>Dist-3 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUDGE, 100</td>
<td>0.54</td>
<td>0.77</td>
<td>48.9 ± 37.9</td>
<td>0.36</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE, 200</td>
<td>0.59</td>
<td>0.79</td>
<td>40.7 ± 26.3</td>
<td>0.34</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>FUDGE, 400</td>
<td>0.61</td>
<td>0.77</td>
<td>49.4 ± 56.9</td>
<td>0.35</td>
<td>0.76</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 21: The first generation by FUDGE using $\lambda = 8$ on the space, politics, and military topics given the prefix “The issue focused on.” Words in the given bag are highlighted in red, and other related words in cyan.

Table 22: Additional example couplet completions, comparing $G$, FINETUNE, PPLM, FUDGE, and Shakespeare. “Prefix” indicates the first line of the couplet which generations are conditioned on.
Prefix: Make thee another self, for love of me,
FUDGE: and for the joy of thee, and for love’s... If...
Shakespeare: That beauty still may live in thine or thee.

Prefix: Or else of thee this I prognosticate:
FUDGE: I O Lord Jesus Christ, my Beloved; I am
Shakespeare: Thy end is truth’s and beauty’s doom and date.

Prefix: But were some child of yours alive that time,
FUDGE: would you still have loved that child for that? I’m...
Shakespeare: You should live twice; in it and in my rhyme.

Table 23: Additional example couplet completions where FUDGE is unsuccessful. In the first, FUDGE fails to rhyme; in the second, “prognosticate” is not in the CMU Pronouncing Dictionary; in the third, FUDGE gets the syllable count wrong.

Specifically, we provide the first generation by FUDGE for 3 separate prefixes for each of the 7 topics. Virtually all examples are clearly on topic, while avoiding repetitiveness.

Additionally, we provide example generations from G, FINETUNE and WDEC in Tables 27, 28, and 29 respectively. For PPLM we refer the reader to the examples in the main paper and appendices of Dathathri et al. (2019)’s original work.

Table 24: Generations starting with “More importantly,” by FUDGE. The first generation is selected for each prefix. The space example is somewhat tangential, while the other six are on topic. Words in the given bag are highlighted in red, and other related words in cyan.
Space: It has been shown in a pilot study in the United States and in an earlier pilot study in Europe that a combination of an advanced technology, including a laser and high-frequency pulsed light, was able to induce spontaneous cell death, which could be detected using an electroencephalogram (EEG). Our findings indicate the potential use of a small-scale laser to generate a

Politics: It has been shown that the "no" movement in France is growing, as evidenced by the increase in the vote in the national assembly on May 7th, 2012. As of now, it is a minority, and its support is shrinking with each passing day. The "no" movement has the potential to take over the government of the French Republic. France's

Military: It has been shown in several other laboratories that, while anaerobic digester systems, such as those deployed in the United States by Cummins, use a different and potentially safer process to extract and recycle the waste, their operation is also far more dangerous. We had a blast at Cummins and they are a very good company. They were very, very quick to come up with

Legal: It has been shown to be the case that a person with a criminal record is more likely to be a victim of domestic abuse and to experience more violence than the general population. Domestic abuse, whether a family member, a current or former partner or a stranger, can have devastating effects, not only on the person, but on their partner and others in their home.

Science: It has been shown to increase the efficiency of the central nerve fibers by as much as 50% in a single operation. The results of the present experiments show that it is possible to activate the central nervous system by using nanomaterials in a novel fashion and to produce a therapeutic effect on various neurological diseases by the action of a single compound in this study, the novel chemical

Religion: It has been shown that, once you become a devout Muslim, there will be an increase in your own religiosity. It can be seen from the following quote: "Islam was the religion that brought the first Muslims to Europe, and it has been the religion that will bring the first Muslims to the Americas." People have heard a number of people tell me that their religion is based on a

Computers: It has been shown using a simple and reliable approach that when the right and left sides of the network are connected by a simple method, the network will become stronger. The network, a network of nodes is connected with each receiving the information from a node that is a neighbor of the node. The neighbor node of the node receiving the information from the neighbor node is an

Table 25: Generations starting with "It has been shown" by FUDGE. The first generation is selected for each prefix. The space example seems unrelated and the military example is somewhat tangential, while the other five are on topic. Words in the given bag are highlighted in red, and other related words in cyan.

Table 26: Generations starting with "To review," by FUDGE. The first generation is selected for each prefix. The politics example is somewhat tangential but the other six are on topic. Words in the given bag are highlighted in red, and other related words in cyan.
In summary, the following are the major findings of the study: Women who use condoms at least one time per year are more likely to use contraception than those who do not. Women who use condoms at least one time per year are more likely to use contraception than those who do not. Condoms may decrease a woman’s risk of getting sexually transmitted infection (STI). This is a factor that is involved. We hope this review will contribute to this debate and to future research to shed more light on the way the world of eSports is growing. The eSports scene is growing.The oldest person, or is the youngest person who is, then you would like to see, then you will hear the second person you would like to hear from! The first thing we would like to consider is whether the “good” answer is really that good before you even get a “good” answer. The “bad” answer is the one you don’t like. So for now the good answer is: If you have a question, you need to do some basic math and get a reasonable answer.
K Topic Control Bags of Words and Prefixes

We use the exact same bags of words and prefixes as Dathathri et al. (2019) for their topic control task, with non-proper nouns lower-cased (in practice, this only changes the religion wordlist). Note our success metric in the paper matches without casing.

We additionally provide the heldout bags of words computed from the original bags (before lower-casing), which we use for the success metric. Although a few words deviate somewhat (“actress” as a synonym for “star” in the space category), overall the heldout bags do represent the desired topic.

Finally, we provide the fantasy bag of words used for selecting the $\lambda$ and $\lambda_{d_{\text{DEC}}}$ conditioning strengths for FUDGE and WDEC respectively. It is also taken directly from Dathathri et al. (2019).

K.1 Original Bags of Words

1. **Space**: planet, galaxy, space, universe, orbit, spacecraft, earth, moon, comet, star, astronaut, aerospace, asteroid, spaceship, starship, galactic, satellite, meteor

2. **Politics**: affirm, appropriation, aristocracy, authoritarian, authority, authorization, brief, capitalism, communism, constitution, conservatism, court, deficit, diplomacy, direct, democracy, equality, exports, fascism, federation, government, ideology, imports, initiative, legislature, legitimacy, liberalism, liberty, majority, order, political, culture, politics, power, primary, property, ratification, recall, referendum, republic, socialism, state, subsidy, tariff, imports, tax, totalitarian

3. **Military**: academy, advance, aircraft, ally, ammo, ammunition, armor, arms, army, arrow, arsenal, artillery, attack, attention, ballistic, barracks, base, battalion, battery, battle, battlefield, bomb, bombard, bombardment, brig, brigade, bullet, camouflage, camp, cannon, captain, capture, carrier, casualty, catapult, cavalry, colonel, combat, command, commander, commission, company, conflict, conquest, convoy, corps, covert, crew, decode, defeat, defend, defense, destroyer, division, draft, encode, enemy, engage, enlist, evacuate, explosive, fight, fire, fleet, force, formation, fort, front, garrison, general, grenade, grunt, guerrilla, gun, headquarters, helmet, honor, hospit-
tal, infantry, injury, intelligence, invade, invasion, jet, kill, leave, lieutenant, major, maneuver, marines, MIA, mid, military, mine, missile, mortar, navy, neutral, offense, officer, ordination, parachute, peace, plane, platoon, private, radar, rank, recruit, regiment, rescue, reserves, retreat, ribbon, sabotage, sailor, salute, section, sergeant, service, shell, shoot, shot, siege, sniper, soldier, spear, specialist, squad, squadron, staff, submarine, surrender, tactical, tactics, tank, torpedo, troops, truce, uniform, unit, veteran, volley, war, warfare, warrior, weapon, win, wound

4. Legal: affidavit, allegation, appeal, appearance, argument, arrest, assault, attorney, bail, bankrupt, bankruptcy, bar, bench, warrant, bond, booking, capital, crime, case, chambers, claim, complainant, complaint, confession, constitution, constitutional, contract, counsel, court, custody, damages, decree, defendant, defense, deposition, discovery, equity, estate, ethics, evidence, examination, family, law, felony, file, fraud, grievance, guardian, guilty, hearing, immunity, incarceration, incompetent, indictment, injunction, innocent, instructions, jail, judge, judiciary, jurisdiction, jury, justice, law, lawsuit, lawyer, legal, legislation, liable, litigation, manslaughter, mediation, minor, misdemeanor, moot, murder, negligence, oath, objection, opinion, order, ordinance, pardon, parole, party, perjury, petition, plaintiff, plea, precedent, prison, probation, prosecute, prosecutor, proxy, record, redress, resolution, reverse, revoke, robbery, rules, sentence, settlement, sheriff, sidebar, standing, state, statute, stay, subpoena, suit, suppress, sustain, testimony, theft, title, tort, transcript, trial, trust, trustee, venue, verdict, waiver, warrant, will, witness, writ, zoning

5. Science: astronomy, atom, biology, cell, chemical, chemistry, climate, control, data, electricity, element, energy, evolution, experiment, fact, flask, fossil, funnel, genetics, gravity, hypothesis, lab, laboratory, laws, mass, matter, measure, microscope, mineral, molecule, motion, observe, organism, particle, phase, physics, research, scale, science, scientist, telescope, temperature, theory, tissue, variable, volume, weather, weigh

6. Religion: absolute, affect, aid, angel, anthem, apostle, archangel, Archbishop, balance, ban, belief, benefit, Bible, bishop, bless, blessing, bliss, bond, bow, Buddhism, canon, Cantor, cathedral, celestial, chapel, charity, choice, Christianity, church, comfort, community, conflict, connection, conquest, conservative, control, conversion, convert, core, counsel, courage, Covenant, creative, Creator, creed, cross, Crusade, Darkness, decision, deity, destiny, Devil, disciple, discipline, discussion, divine, divinity, doctrine, duty, effect, elder, energy, essence, eternal, ethics, event, evidence, exile, Exodus, faith, family, fate, Father, favor, fundamental, gift, glory, God, gospel, grace, growth, guru, habit, hallow, halo, happiness, harmony, healing, Heaven, Hebrew, holy, honor, hope, host, humane, immortal, influence, insight, instruction, issue, Jesuit, Jesus, joy, Judaism, judgment, justice, karma, keen, Keystone, Kingdom, Latin, life, light, love, loving, marriage, meaning, mercy, Messiah, minister, miracle, mission, mortal, mosque, movement, music, mystery, nature, nun, official, oracle, order, organ, Orthodox, outlook, pacific, pagan, parish, participation, pastor, patriarch, peace, perception, personal, perspective, petition, pilgrim, politics, power, practice, prayer, prelude, presence, priest, principle, privacy, prophet, protection, purpose, query, quest, question, quiet, radiant, radical, rally, redemption, refuge, relationship, relative, religion, religious, Revelation, ritual, role, Sacrament, sacred, sacrifice, sage, saint, salvation, sanctuary, savior, scripture, scriptures, sect, security, sense, serious, serve, service, Sharia, shepherd, shrine, silence, sin, society, soul, source, spirit, spiritual, split, statue, Sunday, support, Supreme, teaching, temple, tests, text, Torah, tradition, traditional, trust, unique, unity, unknown, value, vanity, virtue, vision, voice, voices, watch, weight, whole, wisdom, wonder, yang, yin, zeal

7. Computers: algorithm, analog, app, application, array, backup, bandwidth, binary, bit, bite, blog, blogger, bookmark, boot, broadband, browser, buffer, bug, bus, byte, cache, caps, captcha, CD, client, command, compile, compress, computer, configure, cookie, copy, CPU, dashboard, data, database, debug,
delete, desktop, development, digital, disk, document, domain, dot, download, drag, dynamic, email, encrypt, encryption, enter, FAQ, file, firewall, firmware, flaming, flash, folder, font, format, frame, graphics, hack, hacker, hardware, home, host, html, icon, inbox, integer, interface, Internet, IP, iteration, Java, joystick, kernel, key, keyboard, keyword, laptop, link, Linux, logic, login, lurking, Macintosh, macro, malware, media, memory, mirror, modem, monitor, motherboard, mouse, multimedia, net, network, node, offline, online, OS, option, output, page, password, paste, path, piracy, pirate, platform, podcast, portal, print, printer, privacy, process, program, programmer, protocol, RAM, reboot, resolution, restore, ROM, root, router, runtime, save, scan, scanner, screen, screenshot, script, scroll, security, server, shell, shift, snapshot, software, spam, spreadsheet, storage, surf, syntax, table, tag, template, thread, toolbar, trash, undo, Unix, upload, URL, user, UI, username, utility, version, virtual, virus, web, website, width, wiki, window, Windows, wireless, worm, XML, Zip

K.2 Prefixes

"An illustration of", "Emphasised are", "Foundational to this is", "Furthermore.", "In brief.", "In summary", "In this essay", "It has been shown", "More importantly.", "Prior to this", "The central theme", "The connection", "The issue focused on", "The key aspect", "The relationship", "This essay discusses", "To conclude.", "To review.", "To summarise", "Views on"

K.3 Heldout Bags of Words

Note that our heldout bag construction process yielded two stopwords, which we removed; they are omitted below.

1. **Space**: actress, aeronautics, broadband, cosmonaut, cosmos, fireball, flyby, galaxies, heavens, interstellar, lander, lunar, mothership, Romulan, room, worlds

2. **Politics**: appropriated, aristocrats, authorisation, autocratic, capitalist, communist, credibility, cultural, democratic, diplomatic, efforts, energy, excise, exporting, fascist, federal, federated, freedom, gender, ideologies, immediate, imported, income, judge, jurisdiction, legislative, lengthy, minority, Nazism, progressivism, properties, purchase, ratify, referenda, remember, secondary, shortfall, socialist, subsidies, uphold

3. **Military**: aboard, academics, adjutant, advancing, airmen, allies, argue, armies, armistice, armour, army, assets, ATL, aviation, barrage, batteries, bleeding, bottom, bricks, cadre, camera, capturing, cargo, casing, casualties, citadel, civilian, civilians, clandestine, committee, companies, concern, conquered, cursor, customer, dead, decoding, defensive, deputy, detonated, dormitories, encoding, enemies, engaging, escorting, evacuating, execute, expert, explosion, fatigues, flames, flying, forcing, forming, fought, freedom, frigate, gating, glider, gun, guerilla, hand-to-hand, highest, hires, honour, howitzer, ICBM, injuries, inundate, invading, Iraq, khaki, knowledge, lace, late, launchers, leaving, lobb, longtime, Maj., manoeuvre, medical, militia, naval, offensively, offices, operation, paragraph, personnel, persuade, pirate, pistol, policeman, propel, proposal, public, pump, rear, relinquish, rescuing, rifle, riflemen, rifles, rocket, sabotaging, samurai, scouts, secluded, siege, Sgt., ship, shoulders, significant, skipper, skirmish, sloop, sonar, stationed, strategic, strategy, subsidiary, sunk, sword, taken, team, tensions, terms, threat, tribute, victory, visor, wear, won, zone, zoning

4. **Legal**: accusation, acquittal, admit, aggrieved, agreement, alleging, amendment, appearing, appellant, asserted, assertion, assault, authority, burglary, championship, convicted, conviction, criminal, custodial, debatable, decision, defensive, democratic, deposited, deputes, disagree, discoveries, dispute, disputes, edict, embezzlement, enforcement, ethical, event, exams, families, federal, felonies, findings, folder, forgive, heard, homicide, immune, incarcerated, inept, injunctive, inmates, innocence, insolvent, insolvent, investment, judgment, judicial, jurors, knowing, land-use, leave, legislative, liability, item, maintain, major, malpractice, mandamus, mediator, mutual, negligent, objecting, offender, pants, parties, passageways, pixels, police, property, prosecuting, prosecution, proxies,
purchase, quash, regulations, repress, requesting, rescind, reservation, respondent, restaurant, revelation, reversing, rulings, second-degree, sentencing, sitting, solicitor, statutory, step-by-step, sued, sworn, testify, track, transcribed, treasurer, waived, whether, widget, wrongful, wrongs

5. **Science**: action, astronomical, bacterium, bone, clinical, component, compounds, electron, electrons, evolved, flow, fuels, genomics, gravitational, humidity, hypotheses, idea, increasing, jug, ligand, magnesium, mathematics, measuring, microscope, molecular, nothing, observatory, observing, parameter, phone, physicist, physiology, pounds, rain, reason, renewable, scaling, scientific, siphon, statutes, stored, studies, system, tests, theories, transition, warming


7. **Computers**: 512MB, allows, Android, article, attribute, autocomplete, automatically, back-up, barcode, beach, binaries, button, C++, caching, cake, camera, capabilities, casing, chairs, change, cheat, chew, choice, click, coder, compiling, components, computation, computing, confidentiality, configuring, connections, console, copies, counterfeiting, creating, crucial, customer, CyanogenMod, cyber, debian, decimal, decrypt, decryption, deflate, deleting, demo, detect, developing, dialog, dialup, direction, disc, display, DNS, DSL, DVD, e-mail, edit, educational, elements, encyclopedia, Excel, execute, extract, Firefox, fixes, flames, Frequently, functionality, gamepad, garbage, glass, gmail, GPU, guest, hats, house, identifier, infected, initialize, inkjet, input, interactive, interview, ISP, iterative, Jacket, journalists, jQuery, latency, layout, little, logon, lurks, Macs, mails, mainboard, memories, mice, must, notebook, off-line, on-line, operand, original, overflow, packet, pane, paper, parasite, parsing, pasting, PDF, phishing, php, pixels, point-in-time, popup, prev, profit, pull, query, reasoning, rectangular, redirect, rename, restart, restoring, run-time, sailor, saving, search, secure, shoe, sidebar, signal, sites, Solaris, spyware, step, stored, storing, subdirectory, taxi, telephoto, text, tools, topic, torrent, touchpad, typeface, Ubuntu, update, usb, utilities, visuals, VPN, wifi, workstation, writer, XP, XSLT

K.4 Validation Fantasy Bag of Words

beast, Cerberus, demon, dragon, fairy, Frankenstein, ghost, giant, Godzilla, horror, hydra, imp, monster, mummy, ogre, ore, savage, spirit, sprite, titan, troll, undead, unicorn, vampire, witch, zombie
L. Additional Machine Translation

Formality Examples

We provide some additional examples of FUDGE against baselines on our machine translation formality task in Table 30.

| Spanish: | ah pero muy ventajoso que qué a qué qué haces en el en nada yo temen trabajo en el en el ópera pero pero yo soy un sobr es |
| G: | Ah, but very advantageous, what, what do you do in the, in the opera? I work in the, in the opera, but, but I’m an envelope |
| G+ST: | I work in the, in the opera, but I’m an envelope. I am very advantageous. What, what, what do you do in the? |
| FUDGE: | Oh, but it’s very advantageous, what, what do you do in the opera? I’m afraid of working in the opera, but I’m an envelope |
| Reference: | I also work in the library but i am a shelver |

| Spanish: | la información que que tenemos todo es propaganda entonces es portante ver otros versiones de lo que está pasando en el mundo no solamente de la versión de las disuasión de bush |
| G: | the information that, that we have is all propaganda, then is important to see other versions of what’s happening in the world, not only the version of Bush’s deterrence |
| G+ST: | The information that we have is all propaganda. It is important to see other versions of what is happening in the world, not only the version of Bush’s deterrence |
| FUDGE: | The information that we have is all propaganda, so, it’s important to see other versions of what’s happening in the world, not only the version of Bush’s deterrence |
| Reference: | The information we get is all propaganda, it’s important to see other versions of what happens in the world |

| Spanish: | y está un poco difícil verdad |
| G: | And it’s a little hard, right? |
| G+ST: | It’s a little hard, right? Is that a bit of a hard thing to do with it? I’m not sure. |
| FUDGE: | And it’s a little difficult, right? |
| Reference: | It’s a bit hard, to tell you the truth |

Table 30: Additional example translations, comparing G, G+ST, and FUDGE.

M. Software

All models are implemented in PyTorch (Paszke et al., 2019), and pretrained models G are obtained from HuggingFace (Wolf et al., 2019). Specifically, the Marian translation model is https://huggingface.co/Helsinki-NLP/opus-mt-es-en.