Metropolis-Hasting Based Sentence Interpolation: a Mixup in Natural Language

Zilu Tang
Boston University
zilutang@bu.edu

Abstract

Interpolation between sentences is difficult due to the discrete nature of natural language (NL). In this work, we introduce Metropolis-Hasting based sentence interpolation as a sampling technique to smoothly interpolate between one sentence and another[1]. Building on top of previous work in using Metropolis-Hasting algorithm to perform constrained generation/paraphrase, we introduce additional guidance in the sampling procedure to smoothly sample sentences from the distribution of the source to the target. We intrinsically evaluate our sampling method to determine whether such strategy is as effective in interpolating sentences semantics and whether using interpolated sentences as augmentation examples could improve performance for low-resource text classification.

1 Introduction

Building deep learning models with data size constraints is difficult because of the danger of overfitting and the lack of ability to generalize outside of training data. Hence, Zhang et al. (2018) proposed the *mixup* algorithm to augment the dataset by interpolating randomly selected pairs of data and labels from two different classes. More precisely, mixup proposed to obtain the following augmentations:

$$\tilde{x} = \lambda * x_i + (1 - \lambda) * x_j \quad (1)$$

$$\tilde{y} = \lambda * y_i + (1 - \lambda) * y_j \quad (2)$$

where x_i, x_j are raw input vectors, y_i, y_j are one-hot label encodings, and λ is a mixing variable drawn from uniform distribution $U(0, 1)$.

To use mixup for the text domain is non-trivial due to the difficulty of interpolating between sentences. Words within sentences are inherently discrete, unlike pixels in images, making it difficult to optimize and search over. It is also difficult to interpolate between sentence semantics, while keeping the sentence natural, as words are interdependent in a sentence, and swapping order between adjacent words may completely change the semantics of a sentence. While one can interpolate in the embedding space (Bowman et al., 2016; Guo et al., 2019; Ghosal et al., 2022), due to anisotropic nature of sentence embeddings in modern NLP models (Ethayarajh, 2019), these methods often introduce more noise by producing data points that are not in the natural distribution.

While it may not be easy to generate a sentence that semantically interpolates two sentences, it could be easier to iteratively sample edits to the sentence until semantics slowly shifts from source to target. In our method, we consider source and target sentences, x_i and x_j, two data points sampled from the distribution of all possible natural sentences, modeled by a language model. The objective is to use an algorithm similar to Metropolis-Hasting, an instance of Markov Chain Monte Carlo (MCMC) sampling, to sample and accept proposal sentences such that the semantics of a sentence smoothly interpolates from source to target. Here are the key contributions of this work:

- We introduce sentence interpolation in the token space by sampling using Metropolis-Hasting for sentence interpolation.
- We show that by selecting spans of text to sample (rather than individual words) and target guided sampling, we can improve semantic shift and speed up mixing time during the interpolation process.
- We evaluate our method as mixup augmentations for low-resource text classification task and compare performance against baselines.

2 Background and Related Work

2.1 Metropolis-Hasting

Metropolis-Hasting is an instance of MCMC sampling method foundational to Bayesian statistics.

[1] https://github.com/PootieT/Metropolis-Hasting-Sentence-Sampling/

Given a density, it generates a sequence of correlated samples that eventually converge to the stationary distribution of the density. The algorithm can be described as:

1. Initialization: for each $x \in \mathbb{R}^d$ set up a density $q(x, \cdot)$. Choose $X_0 \in \mathbb{R}^d$ arbitrary

2. Given $X_{n-1} = x$:

   (a) Generate Y $q(x, \cdot)$, and u $U(0, 1)$.

   (b) If $u \leq min(1, \frac{\pi(Y)}{\pi(x)} \frac{q(Y,x)}{q(x,Y)})$, then set $X_n = Y$. Otherwise. Set $X_n = x$.

When dealing with sentences, we are no longer in a fixed dimension $\mathbb{R}^d$ (where d is number of words/tokens in a sentence), but we can find equivalence in the rest of the algorithm. At each timestep n, a new proposal sentence is proposed based on a proposal distribution $q(Y|x)$. This proposal can be sampled using any language model M trained with masked-language modeling (MLM) objective (more on 2.2). During acceptance calculation, $\pi(Y)$ is the prior of the sentence and can be approximated with averaged log-likelihood of each token (Eqn 5). $q(Y|x)$ can be approximated by semantic similarity ratio approximated with sentence embedding models (Reimers and Gurevych, 2019). We formally define all terms and introduce the latest relevant works in the next section.

2.2 Modeling Sequence with Transformer-based Language Models

Sentences are made up of sequences of words ($\mathbf{x} = x_1, ..., x_d$). The traditional objective of language modeling is to predict the next word given all previous words in a sequence, namely, the objective can be written as:

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{d} p(x_t|x_{<t}, \theta) \tag{3}$$

where θ are the parameters of a language model and d is the length of a sentence. During optimization, we minimize the log loss ($logp(\mathbf{x})$, Eqn 4). During evaluation, it is often easier to interpret perplexity, a related metric that is the exponential of token-wise averaged negative log-likelihood (Eqn 5). Intuitively, perplexity represents the average number of tokens the model is not sure over at each step of the word prediction. The smaller the perplexity, the better the language model is at modeling the distribution of the text.

$$logp(\mathbf{x}) = \sum_{i=1}^{d} logp(x_t|x_{<t}, \theta) \tag{4}$$

$$ppl(\mathbf{x}) = exp(-\frac{1}{d} \sum_{i=1}^{d} logp(x_t|x_{<t}, \theta)) \tag{5}$$

However, autoregressive ways of modeling texts have many challenges with recurrent neural networks. Issues such as vanishing gradients (Pascanu et al., 2013) led to discoveries in attention mechanism (Bahdanau et al., 2015), a way to distribute features throughout the sequence during generation to combat difficulties modeling long-range dependencies. This led to discoveries of the transformer-based model (Vaswani et al., 2017), the most famous of which is BERT (Devlin et al., 2018), that uses attention modules as the main mechanism to model bi-directional inter-dependencies of words in a sentence. In addition to massively parallel attention networks, BERT introduced an alternative language modeling strategy called masked language modeling (MLM) that takes further advantage of the attention mechanisms. Namely, during training time, input sentences are randomly masked (replace with mask token "[MASK]") and the model is asked to predict the correct word under the mask. Formally, let $\Pi = \pi_1, ...\pi_k$ denote the indices of the masked tokens. Let $\mathbf{x}_\Pi$ denote the set of masked tokens in $\mathbf{x}$, and $\mathbf{x}_{-\Pi}$ denote the set of observed tokens, we can define the MLM loss formally as:

$$L_{MLM} = \frac{1}{K} \sum_{k=1}^{K} logp(x_{\pi_k}|\mathbf{x}_{-\Pi}, \theta) \tag{6}$$

During each forward pass, words are first indexed to their static word embeddings. These embeddings then pass through dense connections of the network, and arrive at their respective hidden states, or contextual word embeddings. Such embeddings are then passed through a light word prediction network to predict the logits for each word across the vocab space.

After a language model is trained on a massive amount of naturally occurring sentences, it can be finetuned with specific objectives in the downstream tasks. Sentence-BERT is one of such downstream task models finetuned to embed variable-length sentences to a fixed embedding dimension where sentences with similar semantics are closer in the embedding space (Reimers and Gurevych,

2019). In our algorithm, we use a specific variance of Sentence-BERT to estimate sentence semantics and use cosine similarity between sentence embeddings as a semantic similarity measure.

2.3 Non-autoregressive Sentence Generation

Autoregressive sequence generation has been the primary method to generate sentences using a trained language model since it aligns well with next-word prediction. However, autoregressive generations are slow, hard to control, and often suffer from exposure bias: the models do not see its (defective) generation during training, leading to hallucination, and malformed generations. Therefore, a few works have been proposed to generate non-autoregressively from MLM-trained models. Most uses masking as a primary way of generating sequences in the middle of the sentences (Wang and Cho, 2019; Ghazvininejad et al., 2019; Qian et al., 2021). Recent works have also used the diffusion process to generate whole sequences through an iterative denoising step (Gong et al., 2022; Li et al., 2022). In this work, we will also use the MLM objective to sample new sentences by masking the current sentence.

3 Method

Unlike the case of Random Walk Metropolis-Hasting algorithm (RWM), we do not sample all words at once since such sampling can change semantics drastically. Instead, we iterative sample sub-span of the sequence until the sentence converges from the source distribution to the target distribution. Algorithm 1 introduces the formal steps, and in the next section we will walk through the sampling processes step by step.

3.1 Proposal Sampling

The first section of the algorithm is to propose a new sentence given the current sentence. This part involves two 4-step: word covariance calculation, span selection, target fusion, and span generation

Word Covariance Calculation Due to the compositional tree-like structure of NL sentences, selecting single words to be edited can be deleterious to the semantics of sentence (Miao et al., 2019). Therefore, we investigate selecting a span (a proper noun, a verb phrase, etc.) at a time ($calculate\_covariance$ in Alg 1). Similar to the idea of fully-adaptive RWM algorithm, we estimate and simulate from the covariance matrix between

Algorithm 1 Metropolis-Hasting Sentence Interpolation

Input Source sentence $\mathbf{x}_{src}$, target sentence $\mathbf{x}_{tgt}$, initial temperature t_{init}, minimum temperature t_{min}, decay rate r, max steps m, generational LM M_g, perplexity LM M_p, semantic LM M_s
Output sentence samples $[\mathbf{x}_1, ..., \mathbf{x}_m]$

$t \leftarrow t_{init}$
$\mathbf{x}_0 \leftarrow \mathbf{x}_{src}$
$a \leftarrow sample\_action()$
while $i < m$ **do**
  $C \leftarrow calculate\_covariance(\mathbf{x}_i, M_g)$
  $C \leftarrow calculate\_longer\_span\_scores(C)$
  $\tilde{\mathbf{h}} \leftarrow target\_fusion(\mathbf{x}_i, \mathbf{x}_{tgt}, M_g)$
  $\mathbf{x}_{prop} \leftarrow span\_generation(\tilde{\mathbf{h}}, M_g, a, t)$
  $apt \leftarrow calculate\_acceptance(\mathbf{x}_{prop}, \mathbf{x}_i, M_s, M_p)$
  $U \leftarrow Uniform(0, 1)$
  if $U \leq apt$ **then**
    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_{prop}$
  else
    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i$
  end if
  $t \leftarrow max(t_{min}, te^{-ri})$
  $a \leftarrow sample\_action()$
end while

variables to increase the in-distribution likelihood of the proposal. We include four different ways of calculating covariance in the order of time complexity: **static**, **mixed**, **one**, **pm**(Wu et al., 2020). See details in appendix A.1.

Span (Mask) Selection Span selection samples a continuous span of tokens from a sentence to be edited. Once covariance matrix C (where each index C_{ij} indicates correlation/importance word x has to y) is obtained, we can calculate a score for spans of all sizes by averaging all their sub-span scores with dynamic programming ($calculate\_longer\_span\_scores$ in Alg 1). See pseudocode in Appendix 2.

Target Fusion Early experimentation indicates that sampling from the distribution of all sentences (disregarding target semantics) is too large of a search space. To narrow down sampling space further and speed up interpolation, we experimented with three levels of embedding fusion: static embedding fusion (**emb**), contextual embedding fusion (**hidden**), and vocab logits fusion (**logit**) ($target\_fusion$ in Alg 1). For the interpolation scheme, we tried **linear**, **exp**onential, and **polar**

(Ghosal et al., 2022). Since we interpolate between sequences with different lengths, we use several strategies to match corresponding tokens between source and target. **closest** infuses source embedding with the closest target word embedding in the same length-normalized position of the sentence. **local-n** fuses corresponding target embeddings with its neighborhood of size n before fusing with the source embedding. **global** fuses corresponding target embedding with all target word embeddings. When fusing multiple target word embeddings, we calculate a concentration α vector as distance of current word to true corresponding target word position. We use normalized α as weights for interpolation in case of **linear** and **exp**, and use α as the concentration to sample a weights from Dirichlet distribution, following Ghosal et al. (2022).

Span Generation Once a span is selected, we can choose to mask the words and insert/delete masks given the target sentence length. Therefore, at each sampling step, we compute the ratio r between the target sentence length and current sentence length and generate a mask length sampled from a Poisson distribution with $\lambda = r$. With sentence masked, we forward pass the sentence through a pretrained MLM model to generate the logits for each word, and sample word independently at each masked index from the vocabulary by softmaxing logits with temperature t ($span\_generation$ in Alg 1).

Alternative Sample Position Selection Previous methods (Miao et al., 2019; Fu et al., 2022) both sample one new word at a time and uniformly randomly choose one of three actions: delete, insert, and substitute. In Alg 1, $sample\_action$ corresponds to either randomly sampling action as described above, or calculating r in the section above.

3.2 Proposal Acceptance

Once a proposal sentence is sampled, we calculate the logprob ratio (acceptance) ($calculate\_acceptance$ in Alg 1) between proposal sentence $\mathbf{x}_{prop}$ and $\mathbf{x}_i$ with the following equations:

$$r_{sem} = (logp_{sem}(\mathbf{x}_{prop}) - logp_{sem}(\mathbf{x}_i)) * \lambda_{sem} \tag{7}$$

$$r_{ppl} = (logp_{ppl}(\mathbf{x}_{prop}) - logp_{ppl}(\mathbf{x}_i)) * \lambda_{ppl} \tag{8}$$

$$accpt = e^{r_{sem}+r_{ppl}} \tag{9}$$

where $logp_{sem}$ is calculated with log cosine similarity of sentence embedding to target sentence embedding. We use a variation of sentence BERT [2] to approximate sentence semantics. $logp_{ppl}$ is $logp$ calculated using distilled GPT-2[3]. λ_{ppl} and λ_{sem} are parameters that controls the preference for low perplexity or high semantic similarity samples. We perform ablations in 7 to show the value of such parameter.

4 Evaluation

We perform two types of evaluation methods comparing our proposed solutions to existing solutions.

Intrinsic evaluation measures how well the distribution shifts from starting sequence to the target sequence. Specifically, we are interested in the **speed** and **quality** of mixing. To evaluate the **speed** of mixing, we calculate semantic progress (**SP**), a normalized semantic similarity between source and target sentence $\in [0, 1]$, and count the steps it takes to reach 0.5 [4]. To evaluate the **quality** of samples along the path of mixing, we calculate the average and standard deviation of sentence perplexity (**ppl**) over the sampling path. A good solution should ideally produce moderate speed for mixing, yielding samples with gradually shifting semantics scores while maintaining low perplexity throughout sampling procedure. We also count rejceted samples (**RJ**), convergence steps (**CS**), convergence rate(**CR**), and report acceptance (**accpt**) statistics. We run our sampling on 10 pairs of sentences (5 pairs per class), and 5 times for each pair. Sentences are randomly sampled from IMDB train set (Maas et al., 2011).

Extrinsic evaluation should measure how useful the sampled/interpolated sentences are as mixup augmentation in improving sequence classification in low-resource settings. Given generated samples, we use **SP** as mixing scalar λ, search within a small range of the target λ, and choose the sentence with lowest perplexity as the augmentation example. We follow procedures from Kim et al. (2021) and evaluate on IMDB sentiment classification (Maas et al., 2011), finetuning a BERT-base-uncased model with 5 examples from both positive and negative classes. For augmentation, we augment 1 example from

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3] https://huggingface.co/distilgpt2

[4] Since interpolation is symmetric, an interpolation method only needs to reach 50% to be fully effective

group	method	CR	CS	max SP	ppl	RJ	accpt	acc
fusion strategies (CGMH)	no fusion	0/50	178 ± 60	0.13 ± 0.08	504 ± 300	15 ± 6	1.03 ± 0.11	59.58 ± 8.29
	emb	0/50	178 ± 60	0.16 ± 0.08	623 ± 223	13 ± 5	1.09 ± 0.48	-
	hidden	10/50	167 ± 59	0.39 ± 0.11	614 ± 254	10 ± 5	1.27 ± 0.96	56.82 ± 3.70
	logit	14/50	161 ± 60	0.42 ± 0.09	590 ± 274	9 ± 4	1.71 ± 3.16	59.31 ± 5.65
mask selection strategies	word-pm ♣	18/50	166 ± 64	0.41 ± 0.12	607 ± 276	9 ± 6	1.62 ± 3.02	59.02 ± 4.34
	span-static	33/50	96 ± 73	**0.51** ± 0.04	2266 ± 891	51 ± 44	9.49 ± 47.46	**61.28** ± 8.06
	span-one	**39/50**	91 ± 70	0.50 ± 0.05	2313 ± 788	48 ± 39	6.07 ± 16.10	55.30 ± 3.32
	span-pm	37/50	**86** ± 64	**0.51** ± 0.04	2389 ± 1111	48 ± 41	5.51 ± 10.47	60.40 ± 4.46*
	span-pm+	7/50	167 ± 64	0.32 ± 0.16	**338** ± 275	161 ± 62	130 ± 912	59.91 ± 4.05
	no aug	-	-	-	-	-	-	55.94 ± 2.84

Table 1: Intrinsic and extrinsic results evaluation across different experimental trials. All trials in fusion strategies group use random word masking method used in CGMH, but fusion occurs at different levels. ♣ corresponds to masking strategies used in PMCTG. All trials in mask selection strategies group uses logit fusion. **span-pm+** compared to **span-pm** has larger λ_{ppl}. See hyperparameters in Appendix. * indicate a t-score < 0.1.

each original data point, doubling the size of training data. We report classification accuracy (**acc**) on IMDB official test set (25K data points)

5 Baseline Methods

Due to reproducibility concerns, we compare with external baselines in Appendix 3. We experiment with variations of mask selection methods corresponding to **CGMH**(Miao et al., 2019) (sample random words and action) and **PMCTG**(Fu et al., 2022) (sample words given contribution to left-right neighbors). We also included baselines for no augmentation, and no fusion sampling.

6 Main Results

When we compare methods intrinsically, we can see fusion at **logit** level works the best by having most semantic progress and relatively low perplexity. However, semantic progress is much faster in span based methods, with **span-one** achieving most convergence rate. However, since single word masking changes semantics slowly, it increase perplexity less dramatically over sampling process. With more masks sampled at a time, there could be issues with repetitive generation due to our independent sampling procedure for each word index in a span. However, this issue can be somewhat deterred by increasing λ_{ppl} to weigh in on perplexity more during acceptance calculation (**span-pm+** vs **span-pm**). Nonetheless, such constraint comes at the cost of semantic progress.

When we look at extrinsic evaluation (**acc**), we observe **span-pm** performs (slightly) significantly better than baseline. Best average performing trial is **span-static**. Surprisingly, we do not observe a significant correlation between perplexity of the sample vs. the accuracy of the downstream task.

In addition to the main results, we performed extensive ablation studies to understand the cause and effects of different components. See Appendix A.6. For qualitative results see A.5, and for visualizations of sampling trajectories, see A.4 [5]

7 Conclusion and future directions

In this work, we investigated whether Metropolis-Hasting Sentence Interpolation is an effective way of interpolating one sentence to another. We show that by carefully selecting constraints, masking and fusion strategies, we can shift the semantics of sampled sentence from one distribution to another, and improve low-resource text classification as an augmentation method. However, our proposed fusion strategies are still naive. Though effective at shifting semantics of the samples, the perplexities remain high. Though preliminary experiments show no correlation between perplexity and downstream tasks, it is troubling to think degenerate sentences to be helpful in improving models performing in real world. Thus, it is crucial to investigate in the future what are more effective fusion and search strategies for interpolation. By restricting fusion at corresponding relative position of target sentence, we ignore the context of either sentences, introducing unnatural sampling noise (which ironically is what is needed to shift semantics). It might be better to perform mixing at higher level (phrases, clauses), or improve span-based MLM generations.

[5] Appendix experiments and explanations only go in more depth of the code, explanations of phenomenons, and personal curiosities, thus tangential the the main conclusion of the project and should be counted outside the page limit.

References

Bahdanau, D., Cho, K. H., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pages 10–21. Association for Computational Linguistics (ACL).

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ethayarajh, K. (2019). How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65.

Fu, Y., Ou, W., Yu, Z., and Lin, Y. (2022). Effective unsupervised constrained text generation based on perturbed masking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1417–1427, Dublin, Ireland. Association for Computational Linguistics.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. (2019). Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121.

Ghosal, D., Aditya, S., Dandapat, S., and Choudhury, M. (2022). Vector space interpolation for query expansion. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 405–410.

Gong, S., Li, M., Feng, J., Wu, Z., and Kong, L. (2022). Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Guo, H., Mao, Y., and Zhang, R. (2019). Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Kim, Y., Jeong, S., and Cho, K. (2021). Linda: Unsupervised learning to interpolate in natural language processing. *arXiv preprint arXiv:2112.13969*.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. B. (2022). Diffusion-lm improves controllable text generation. *arXiv preprint arXiv:2205.14217*.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Miao, N., Zhou, H., Mou, L., Yan, R., and Li, L. (2019). Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Ng, N., Cho, K., and Ghassemi, M. (2020). Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

Patel, R. and Pavlick, E. (2021). Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Qian, L., Zhou, H., Bao, Y., Wang, M., Qiu, L., Zhang, W., Yu, Y., and Li, L. (2021). Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, A. and Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36.

Wei, J. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.

Wu, Z., Chen, Y., Kao, B., and Liu, Q. (2020). Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

A Appendix

A.1 Covariance Calculation

To calculate the covariance matrix between each token in the sentence, we come up with 4 different ways using a pretrained BERT-based model. Each method requires a different amount of computational steps, yielding a trade-off between speed and quality of the covariance matrix. Formally, given a sequence of tokens $\{x_i\}_{i=1}^n$ and a model M pretrained with masked-language modeling objective (Devlin et al., 2018), we can obtain static word embedding $E_{n\times h}$, and contextual word embedding $H_{n\times h}$ through $M(E)$, where h is the hidden embedding dimension of the model M. Below are four ways of calculating covariance matrices C between words, in the order of increasing computational costs.

1. **static**: $C = E'E$. Using static embedding only, we avoid passing sentences through M

2. **mixed**: $C = \lambda_1 E'E + \lambda_2 H'H$, where $\lambda_1 + \lambda_2 = 1$ controls the weighting between static and contextual embedding. This method requires 1 pass through M for every sentence.

3. **one**: For each token x_i in the sentence, we replace it with $[MASK]$ token, obtaining sentence $\mathbf{x} \setminus \{x_i\}$, and obtain the contextual embedding of sentence $H(\mathbf{x}\setminus x_i) = M(\mathbf{x}\setminus\{x_i\})$. Given a distance metric $d(,)$ (we use cosine similarity), we calculate $\frac{1}{n}\sum_{j=1}^n d(H(\mathbf{x}\setminus x_i)_j, H_j)$. Intuitively, this measures the impact a word has on the rest of the tokens in the sentence. This method requires forward passing mode $n+1$ times for each sentence.

4. **pm**: Proposed by (Wu et al., 2020) and used by (Fu et al., 2022), this the perturbed masking method calculates each word x_i's impact on another word x_j through the following three step calculation:

   (a) Obtain $H(\mathbf{x} \setminus x_i)_i$ as mentioned in method 3.

   (b) Obtain $H(\mathbf{x} \setminus x_i, x_j)_i$ by masking out both x_i and x_j and forward pass through M.

   (c) Calculate $I(\mathbf{x}|x_j, x_i) = d(H(\mathbf{x} \setminus x_i, x_j)_i, H(\mathbf{x} \setminus x_i)_i)$

   Intuitively, $I(\mathbf{x}|x_j, x_i)$ measures x_j's impact on x_i. If $H(\mathbf{x} \setminus x_i, x_j)_i$ is similar to $H(\mathbf{x} \setminus x_i)_i)$, this indicates the presence or absence of x_j provides little signal in predicting x_i. This method requires forward passing model $2n$ times for each sentence.

A.2 Span Selection Python Code

Intuitively, once we obtain the covariance matrix for words, we have only obtained word pair covariance (span of size 2). In this section, we need to fill out the lower triangular part of the covariance matrix such that each lower triangular entry $C(i, j)$ corresponds to the score for the span from the i^{th} word to the j^{th} words. Once all span scores are filled, we can select any length span by sampling from the softmax of all spans' scores.

To calculate and assign scores for higher longer spans, we use heuristics and assign the score for a span as an average for all its sub-spans. We allow penalty selecting a single word by multiplying the diagonal of the covariance matrix by a penalty. We also allow penalty selecting longer spans by reducing its score by a factor to the power of a number of words in the span.

Intuitively, we use the dynamic programming trick used to calculate in box filters in computer vision to calculate our span average 2. We recognize that for a span (i, j), its score can be calculated from $C(i, j), C(i-1, j), C(i, j+1), C(i-1, j+1)$. More specifically, we can calculate it as $C(i, j) + w_1 * (C(i-1, j) + C(i, j+1)) - w_2 * C(i-1, j+1)$, where w_1, w_2 can be calculated from the size of the sub-span (or how many rows off from main diagonal in covariance matrix). Code 2 shows our dynamic algorithm

A.3 Main Results

In our main fusion strategies group, we use **closest** neighbor selection strategy with **linear** interpolation. The initial temperature is 10, except for **logit**, which is set to 1. The minimum temperature is set to 0.1, with an exponential decaying rate of 1e-4. λ_{sem} is set to 10 and λ_{ppl} is set to 0.1. In mask selection strategy, we keep initial temperature at 1, and the rest of the hyperparameters the same. For **span-pm+**, we increase $\lambda_p pl$ to 10 to filter out more lower perplexity samples.

During experimentation, the max number of sampling steps is set to the max of total number of words in either target or source sentence.

A.3.1 Reproducibility

Kim et al. (2021) reported performance compari-

```
    def calculate_higher_order_span_score(
        self, C: Tensor, single_word_logit_penalty: float = 1.0,
    ↪ higher_order_penalty: float = 0.8
    ) -> Tensor:
        n = C.shape[0]

        C = (C + C.T) / 2  # make it symmetrical if input is not strictly "
    ↪ covariance matrix"

        # set diagonal values for single word sampling
        single_word_penalty = torch.ones_like(C)
        single_word_penalty -= 1 - torch.eye(n) * single_word_logit_penalty
        C = C * single_word_penalty

        # dynamic algorithm to calculate average score of a span by averaging scores
    ↪  of all its sub-spans
        if n > 2:
            higher_order_ratio = torch.ones_like(C)
            for i in range(1, n):  # row, up to down
                for j in range(i - 1, -1, -1):   # col, right to left, fill up lower
    ↪ -triangular entries
                    # i-j is how many rows is the span off from diagonal, aka span
    ↪ length
                    off_diag = i-j
                    sub_span_coef = (off_diag) * (off_diag + 1) / 2
                    C[i, j] = C[i, j] + (C[i - 1, j] + C[i, j + 1]) * sub_span_coef
                    if (off_diag)>=2:  # if it's more than 1 diagonal offset, we
    ↪ have to subtract overlapping region
                        sub_sub_span_coef = (off_diag - 1) * (off_diag) / 2
                        C[i, j] -= C[i-1, j+1] * sub_sub_span_coef
                    C[i, j] /= (off_diag+1) * (off_diag + 2) / 2

                    # fill up higher order diagonal multiplier
                    higher_order_ratio[i, j] = higher_order_penalty ** (off_diag)

            # apply higher order span penalty
            C = C * higher_order_ratio
        return C
```

Table 2: Pseudocode to calculate span score based on the average of all its sub-spans scores.

group	method	CR	CS	max SP	ppl	RJ	accpt	acc
ours	Span-pm+	7/50	167 ± 64	0.321 ± 0.164	338 ± 275	161 ± 62	130 ± 912	59.91 ± 4.05
	no aug	-	-	-	-	-	-	55.94 ± 2.84
external comparison	no aug	-	-	-	-	-	-	61.8 ± 3.00
	EDA	-	-	-	-	-	-	62.5 ± 3.00
	Mixup	-	-	-	-	-	-	62.2 ± 1.86
	SSMBA	-	-	-	-	-	-	64.1 ± 2.10
	LINDA	-	-	-	-	-	-	67.3 ± 2.60

Table 3: Selected results comparison with (Kim et al., 2021) reported results. Our baseline differs significantly from their baseline, making the rest of the comparison a moot point.

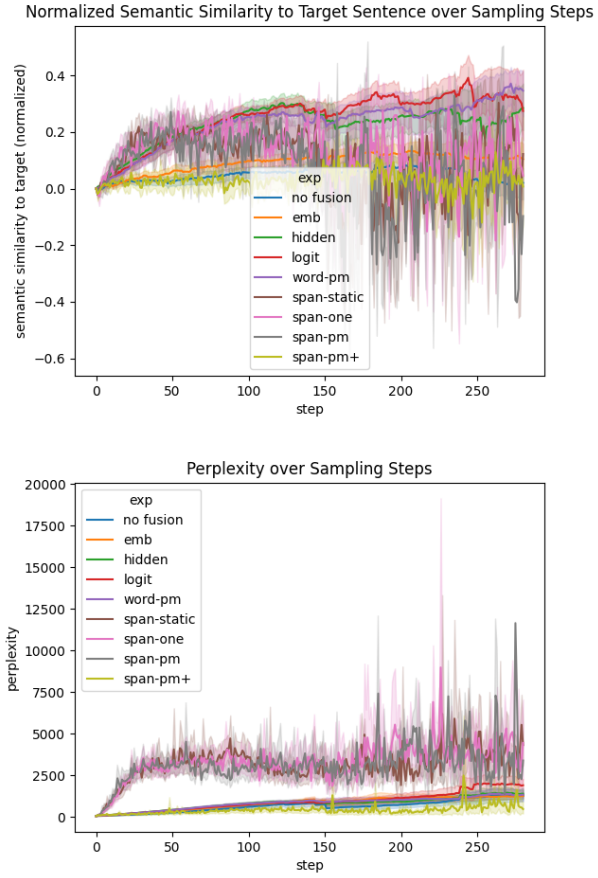

Figure 1: Semantic progress (top) and perplexity (bottom) over sampling trajectories. All names corresponds to rows in main result table 1.

son against various other methods. However, our baseline weren't matching up with theirs, after following their described experiment procedure. This is probably result of high variance of data points we selected as few-shot training example. Hence we include other methods performance in the appendix for reference. **EDA** (Wei and Zou, 2019) randomly insert/delete/swap words in a sentence. **LINDA** (Kim et al., 2021) is a deep encoder-decoder fusion model fine-tuned on 1 million sentences. **SSMBA** (Ng et al., 2020) is similar to using MLM sampling with 1 sampling step.

A.4 Sampling Trajectory Visualization

As seen in fig 1, span based sampling method archives high semantic progress in a short period and then seems to drift off with high variance at the end. One of the reasons for this is that when sampling Poisson distribution using r, the ratio between the length of target sentence and current sentence, the density for sampling zero is too high. This causes sampled sentence length to dramatically decrease and reach a certain equilibrium towards the later part of sampling, where all of the spans are equally likely to be masked out and re-sampled. In this regard, word-based sampling methods are much more reliable at incrementally increasing semantic progress, with fusion at the logit level performing the best. Looking at the perplexity trajectory, we are confirmed with results from the main table that word-based sampling methods increase perplexity less dramatically. It is good to see that **span-pm+** is able to keep perplexity low while converging at a non-trivial rate (similar to word-level sampling ones).

A.5 Qualitative Results

From Table4, we can clearly see what's happening: word-based sampling (**logit**) swaps words at each index naively, while span-based sampling is good at deleting spans but cannot generate well due to independent sampling (one way to combat this is to iteratively sample similar to (Wang and Cho, 2019))the semantic progress made from span-based methods is mostly due to deleted semantic context, leading to a more generic meaning, which sometimes matches what the target sentence means. This also indicates that semantic progress is not necessarily the best threshold to use for labeling our augmentation samples. Even after deleting most of the phrases, the semantics of the **span-pm+** example is still clearly negative. This indicates that it is perhaps more important to find and incorporate signals from class labels during augmentation with interpolation.

A.6 Ablation Studies

A.6.1 Fusion Strategies (Embeddings)

As seen in Table 5, fusion at the static embedding level (**emb**) is not an effective strategy. Understandably, by mixing static embeddings, the language model encoders may fail to understand what the token may be in the forward pass. Additionally, when interpolating contextual embeddings (**hidden**) **polar** and **linear** interpolations are both valid ways with some of the highest semantic progress. Exponential interpolation (**exp**) and mixing a neighborhood of hidden embeddings (**local**) are not fruitful in achieving any significant semantic progress. This result suggests that the hidden representations of words occupy a relatively isotropic embedding space: relatively linear operations remains valid in the space. Although the fact that adding multi-

mixup ratio		examples
0.5	source (neg)	**Watching this movie is like eating a banquet of nothing but meringue**. It initially looks great but ultimately provides NO satisfaction–none. The plot is a muddled mess about a toy factory and the forces of evil. So, how is it possible that with this basic plot AND Robin Williams that the movie still turns out so badly?! It's because the picture is all appearance with no substance whatsoever–much like the terrible Popeye picture Williams did at the beginning of his film career. The film must have cost a fortune but perhaps there wasn't enough money left over to hire writers who had graduated grade school. The film is one unfunny joke that goes on and on and on and on. I really am **unsure why it was made in the first place–it certainly wasn't made to provide any sort of entertainment**.
	target (pos)	Dark Angel is a cross between Huxleyś Brave New World and Percyś Love in the Ruins–portraying the not too distant future as a disturbing mixture of chaos and order, both in the worst sense of the word. Once one swallows the premise that all modern technology can be brought to a standstill by "the Pulse," it provides an entertaining landscape for exploring the personalities of and relationships between the two primary characters–Max (the Dark Angel/bike messenger) and Logan (the rich rebel). It seems uneven, perhaps a result of a variety of authors, but is held together by the energetic, beautiful, and charming Jessica Alba, who seems both strong and calloused yet vulnerable and sensitive. I think that Fox has done it again.
	logit	watching this angel is is like eating a banquet of's brave new newingue. it initially looks great but ultimately provides no satisfaction - none the < br / > < br future the plot a muddled mess about, both factory the worst forces of evil the word., how is one is possible with this basic plot and robin williams that be movie still turns so?!'by s because,, picture is all appearance with no substance whatsoever - - like the and and and between they the two picture did at the beginning his film career. the film must have cost money but perhaps there wasn't enough money left, to hire writers who had graduated grade school. < br / a < br / > the film is one unfunny joke that goes on and and on and on. i really am unsure why it was made in the first vulnerable, - - it certainly wasn't made fox has provide any sort entertainment.
	span-pm+	**watching this movie plot is a muddled mess** am **unsure why it was made in the first place** - - **it certainly wasn't made to provide any sort of entertainment**.

Table 4: Interpolation examples

fusion method	CR	CS	Max SP	ppl	RJ	accpt
no	0/50	178 ± 60	0.129 ± 0.078	504 ± 300	15 ± 6	1.03 ± 0.11
emb, closest, linear	0/50	178 ± 60	0.163 ± 0.077	623 ± 223	13 ± 5	1.09 ± 0.48
emb, closest, exp	0/50	178 ± 60	0.120 ± 0.063	466 ± 165	13 ± 5	1.01 ± 0.06
hidden, closest, linear	10/50	**167** ± 59	**0.390** ± 0.107	614 ± 254	10 ± 5	1.27 ± 0.96
hidden, closest, exp	0/50	178 ± 60	0.091 ± 0.076	404 ± 135	16 ± 7	1.00 ± 0.08
hidden, closest, polar	**12**/50	166 ± 60	0.379 ± 0.116	644 ± 290	10 ± 5	1.20 ± 0.39
hidden, local5, exp	0/50	166 ± 60	0.097 ± 0.061	**403** ± 127	16 ± 9	1.01 ± 0.13
hidden, local5, polar	0/50	178 ± 60	0.187 ± 0.100	428 ± 133	11 ± 6	1.04 ± 0.07

Table 5: Intrinsic evaluations with static and contextual embedding fusions. For terminologies in fusion method, refer to 3.1

ple neighborhood embedding does not work suggests that hidden embeddings still represent distinct "modes" of distributions that do not mix well when combined together on a big scale. Though, **hidden, closest, linear** and **hidden, closest, polar** are two trials with the highest perplexities, which may mean that in order to achieve semantic shift, there may have to be some sacrifice in terms of perplexity.

A.6.2 Fusion Strategies (Logits)

Seen in Table 6, any target side neighborhood mixing (**local**, **local5**, or **global**) is detrimental to the sampling process. Unexpectedly, **polar** and **exp** interpolation outperform **linear**. This could be due to more "spiky" nature of vocab logits: they are supposed to represent a (unormalized) density of what words are likely to appear. Therefore, when mixing, a non-linear method is more appropriate. This could also explain why methods other than **closest** perform poorly: noises from adjacent vocab logits may cancel out the signals.

To further understand how interpolate-able contextual embeddings and logits are, we performed additional probing experiments in A.7.

A.6.3 Acceptance Parameters

Since we have multiplicative scalars λ_{sem} and λ_{ppl} to control the acceptance calculation given proposal's semantic similarity to target sentence and perplexity, we also looked at how these parameters affect the sampling process. In Table 7 top 3 rows, we observe that λ_{ppl} can be an effective way of curbing perplexity gain in the sampling process. However, this objective necessarily makes the sampling process harder as it restricts the search space for only sentences that are natural. Given the same search budget, a higher λ_{ppl} achieves less semantic progress. On the other hand, lower λ_{sem} lowers the semantic progress. When λ is small, almost all samples are accepted.

A.6.4 Temperature Annealing

Although sampling with higher temperature could increase probability of generating rarer words, in Table 8 we found that increasing temperature has an adverse affect on semantic progress. It is likely that higher temperature introduced too much noise, rather than benefiting the search process with a diversity (paraphrasing, or using synonyms). Although we did not verify whether such increase in diversity could have positive impact on extrinsic evaluation, which would be interesting to observe.

A.6.5 Word Masking Strategies

In addition to main table result 1, we included **span-mixed** here. It seems that by averaging covariance matrix obtained from static embedding and contextual embedding does not provide much more signal than just using static embedding **span-static**.

A.7 Interpretability Probing of Fusion Strategies

To understand how interpretable hidden representations of the language models are, and whether their meanings are interpolated (as opposed to simply mixing distributional modes), we designed this probing experiment. We came up with several pairs of sentences, each with a word masked out. The sentences are written in a way that the identity of the masked words should be contextually obvious to predict. We then interpolate the contextual embedding or logits of the two masked words in the pair, and compare sampled words against human expected output.

1. **Source**: Roses are [MASK], violets are

fusion method	CR	CS	Max SP	ppl	RJ	accpt
no	0/50	178 ± 60	0.129 ± 0.078	**504** ± 300	15 ± 6	1.03 ± 0.11
logits, closest, linear	2/50	178 ± 59	0.319 ± 0.121	659 ± 262	11 ± 6	1.25 ± 0.93
logits, local, linear	0/50	178 ± 60	0.225 ± 0.087	634 ± 258	13 ± 6	1.10 ± 0.24
logits, local5, linear	0/50	178 ± 60	0.165 ± 0.071	622 ± 288	14 ± 6	1.03 ± 0.09
logits, global, linear	0/50	178 ± 60	0.154 ± 0.074	646 ± 326	15 ± 7	1.02 ± 0.04
logits, closest, polar	**14**/50	**164** ± 62	0.388 ± 0.124	577 ± 251	9 ± 5	1.35 ± 1.15
logits, closest, exp	12/50	170 ± 58	**0.412** ± 0.101	613 ± 263	9 ± 5	1.20 ± 0.37

Table 6: Intrinsic evaluations with logits fusions. For terminologies in fusion method column, refer to 3.1

λ_{sem}	λ_{ppl}	CR	CS	Max SP	ppl	RJ	accpt
10	0.1	12/50	170 ± 58	**0.412** ± 0.101	613 ± 263	9 ± 5	1.20 ± 0.37
10	1	15/50	164 ± 63	0.393 ± 0.109	606 ± 278	11 ± 5	3.25 ± 14.9
10	10	9/50	167 ± 60	0.294 ±0.161	**230** ± 123	76 ± 45	0.82 ± 0.88
1	0.1	**14**/50	**163** ± 62	0.364 ± 0.130	612 ± 278	1 ± 1	1.00 ± 0.00
0.1	0.1	9/50	169 ± 70	0.388 ± 0.100	656 ± 310	0 ± 1	0.99 ± 0.00

Table 7: Intrinsic evaluations with different λ_{sem} and λ_{ppl} values.

t_{init}	CR	CS	Max SP	ppl	RJ	accpt
1	**14**/50	**161** ± 60	**0.417** ± 0.089	**590** ± 274	9 ± 4	1.71 ± 3.16
10	12/50	170 ± 58	0.412 ± 0.101	613 ± 263	9 ± 5	1.20 ± 0.37
20	12/50	164 ± 58	0.403 ± 0.101	613 ± 263	8 ± 4	1.31 ± 0.68

Table 8: Intrinsic evaluations with different initial temperature.

mask sampling method	CR	CS	Max SP	ppl	RJ	accpt
word-random	14/50	161 ± 60	0.417 ± 0.089	**590** ± 274	9 ± 4	1.71 ± 3.16
word-pm	18/50	166 ± 64	0.413 ± 0.120	607 ± 276	9 ± 6	1.62 ± 3.02
span-static	33/50	96 ± 73	0.506 ± 0.042	2266 ± 891	51 ± 44	9.49 ± 47.46
span-mixed	33/50	96 ± 73	0.506 ± 0.042	2266 ± 891	51 ± 44	9.49 ± 47.46
span-one	**39**/50	91 ± 70	0.499 ± 0.046	2313 ± 788	48 ± 39	6.07 ± 16.10
span-pm	37/50	**86** ± 64	**0.509** ± 0.040	2389 ± 1111	48 ± 41	5.51 ± 10.47

Table 9: Intrinsic evaluations with different mask sampling methods. For explanation of the method see A.1.

blue. **Target**: Bananas usually have the color [MASK]. **Expected Interpolation**: Orange

2. **Source**: In US, cars go on the [MASK] side of the road **Target**: Most people are right handed, so it's rare to someone who is [MASK] handed **Expected Interpolation**: middle / center

3. **Source**: Firefighters are [MASK] people: they save lives while risking their own. **Target**: Thieves are [MASK] people: they take advantage of others for their own gain. **Expected Interpolation**: average (no clear answer)

4. **Source**: I left the car [MASK] at my desk, so I can't start the car. **Target**: Brazil is home to one of the largest wild habitat in the world: the Amazon [MASK]. **Expected Interpolation**: (no clear answer)

In first two example, we test for color spectrum and spatial orientation. In third example we test for subjective attribute understanding, and in the last example, we are looking for anomolies or surprising results.

As observed in Table 10, mixing at hidden level generated more sensible results, and linear and polar interpolations are both valid. However, we found that interpolation at either contextual embedding or logits level are more or less mixing modes of distribution, rather than semantically interpolating between two concepts. Due to scope of the project, we aren't able to investigate deeper modes of fusion. Perhaps given more representation power, we could see such semantic interpolation. Grounding a statistical machine with real world can indeed be difficult, as observed in (Patel and Pavlick, 2021).

Masked Word	Red + Yellow = Orange	Left + right = Middle	Selfless + selfish = average	Key + forest = ?
Hidden, linear	Purple, yellow, red	Left, wrong, mean	Honorable, mythical, of	keys,belt, tiger
Hidden, exp	Violet, unused tokens	ologists, auschwitz, unused tokens	Scrape, sarcasm, unused tokens	Mento, Bombardier, unused tokens
Hidden, polar	Purple, yellow, green	Wrong, left, flip	Good,skilled, dangerous	Keys, ids, rainforest
Logit, linear	Red, pure, have	‘	anonymous, helping, unstable	one
Logit, exp	have	‘	ordinary	one
Logti, polar	Have, white, red	‘	Responsible, competent, attractive	one

Table 10: Interpretability probing results. The top row (column header) represent the two words masked in each sentence in the pair and their expected interpolated word. The left most column represent the fusion method used to interpolate during sampling. Each cell records the top 3 most commonly generated words out of a trial of 10.