# Learning a decoding objective from human-generated text

M.A. Thesis Proposal

13. January 2023

**Student**: Alison Y. Kim
**Supervisors**: Prof. Dr. Rico Sennrich; Prof. Dr. Ryan Cotterell; Clara Meister, M.Sc.

## 1 Overview

Natural language generation (NLG) tasks aim to produce text that is human-like, characterized not only by grammaticality, but also by fluency, comprehensibility, and relevance. One approach to such tasks is to use a probabilistic language model to generate text. Formally, a language model is a probability distribution $q$ over the set of all natural language strings, where a string is defined as a sequence of tokens from a given vocabulary. Once $q$, which is not known *a priori*, is estimated, a **decoding strategy** can be used to generate text from the estimated distribution. Specifically, a decoding strategy consists of a **scoring function**, which maps strings generated under a model's vocabulary to a real number, and a **decoding algorithm**, which autoregressively chooses the next token according to the scoring function [6]. The text generated from a language generation model (LGM), and thus the performance of an NLG tool (e.g., for machine translation), depends largely on the choice of decoding strategy [10].

When using a probabilistic model for text generation, where each string is assigned a probability, an intuitive approach to decoding may be to select the string with the highest probability under some LGM, i.e., the mode of this distribution. This approach is known formally as maximum *a posteriori* (MAP) estimation. In NLG, the *de facto heuristic*[1] MAP-based decoding algorithm is *beam search* [7, 8]. However, beam search has been found to produce texts that are repetitive or boring for some tasks [3, 2, 11], while requiring alterations to the scoring function to work well for others [5, 9].

In certain NLG tasks, stochastic decoding methods have been shown to outperform mode-seeking ones, as the former introduce diversity into outputs and seem to avoid the repetitions produced by MAP-based strategies [3, 10]. Stochastic strategies have their own drawbacks, such as producing incoherent text when low-probability items are sampled [10]. Nonetheless, the qualitatively most successful decoding strategies, at least in open-ended NLG tasks, are stochastic ones. Examples include nucleus and typical sampling, which employ ancestral sampling as their decoding algorithms and use altered scoring functions. Such alterations to the scoring function work well in practice [11, 3, 4], but are only loosely motivated by intuition.

The standard scoring function used during text generation is simply the token log-probability. Yet the need for alterations to this scoring function suggests a discrepancy between high-probability texts and human notions of high-quality texts.[2] However, these alterations are heuristic and have thus far been found through trial and error. This thesis project thus proposes an improved decoding strategy via an alteration to the scoring function — in other words, to learn the optimal transformation of some trained LGM's probability distribution output to an explicit score. In particular, the aim is to *learn* an optimal scoring function that transforms a distribution representing token attributes (discussed further in *2.1 Inputs*) to an alternative distribution $\mathbf{d} \in \mathbb{R}^{|V|}$ representing a one-hot encoded corpus $\mathcal{C}$, thereby reassigning probabilities to tokens. In essence, this approach represents a lightweight fine-tuning method to address the discrepancy between training and inference.[3]

To learn this function, attributes of the output distribution that have proven useful in existing alteration techniques shall be used. (For instance, one could take the absolute difference of a given token's information

[1] Finding the exact mode of the distribution is computationally expensive, so a heuristic is used instead.
[2] This discrepancy may arise in part due to exposure bias, or the training-inference discrepancy that occurs when an autoregressive model is trained using only ground truth inputs, i.e., without any inputs generated by the model itself.
[3] Reinforcement learning is often used to correct for this, but it is a computationally expensive and high-variance approach [1].

content and the expected information content, as done in typical sampling [4].) This learned scoring function can then be used with any decoding algorithm, such as beam search or ancestral sampling, to generate human-like text without succumbing to common issues that plague NLG frameworks.

## 2 Approach: Learning a transformation function

*Adapted from Clara's write-up: https://www.overleaf.com/project/6202110dc474ed42893f48e3*

We are given a distribution $p_{\boldsymbol{\theta}}(\cdot \mid \mathbf{y}_{<t}) \in \mathbb{R}^{|\mathcal{V}|}$. We want to learn a transformation $p_{\text{dec}} : \Delta^{|\mathcal{V}|-1} \to \Delta^{|\mathcal{V}|-1}$, i.e., a mapping from one probability distribution over the vocabulary to another probability distribution over the vocabulary. We want the resulting distribution to be one that assigns high probability to samples of human-like text. Obviously, we would hope that $p_{\boldsymbol{\theta}}$ already does this. However, how we typically build language generation models is arguably not ideal for language generation. Both the softmax used to project model outputs onto the probability simplex and the forward KL divergence (cross-entropy) loss we use to train the models ensure that the model places mass on *all* tokens in the vocabulary at each time step. Further, not all text that is used during training is "high-quality", even though the model may learn the underlying important components of language.

We want to choose some attributes of tokens $y \in \mathcal{V}$ that may be important for learning this mapping. These attributes can be context-dependent, e.g. probability of the token under the model itself $p_{\boldsymbol{\theta}}(y \mid \mathbf{y}_{<t})$, or context-independent, like the unigram probability of the token. Other context-dependent examples include an indicator variable for whether or not the token is in the top-$k$ probability tokens, or the distance between the token's information content[4] and the expected information content.[5]. Let us call these attributes $f$, and let there be an arbitrary number $K$ of them $f_{1,\ldots,K}(\cdot|\mathbf{y}_{<t}) = \mathbf{f}(\cdot \mid \mathbf{y}_{<t})$. Then, using some of the examples above, we would have $f_k(y' \mid \mathbf{y}_{<t}) = p_{\boldsymbol{\theta}}(y' \mid \mathbf{y}_{<t})$, the probability of $y'$ under the model, or $f_k(y') = \omega(y')$, the frequency of $y'$ according to some corpus.

Let's say we have a text. We see word $y_t \mid \mathbf{y}_{<t}$. We want to learn a function that outputs a distribution $\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|}$ where $d_t = 1$ (corresponding to the index of token $y_t$) and $d' = 0$ elsewhere. We can do this by learning a linear transformation of our attributes

$$\widehat{p_{\text{dec}}}(y_t \mid \mathbf{y}_{<t}) = \mathbf{w}^\top \mathbf{f}(y' \mid y_{<t}) + b \tag{1}$$

for (learned) weights $\mathbf{w} \in \mathbb{R}^K$ and then projecting $\widehat{p_{\text{dec}}}(y_t \mid \mathbf{y}_{<t})$ onto the probability simplex (e.g. via softmax):

$$p_{\text{dec}}(y_t \mid \mathbf{y}_{<t}) = \frac{1}{Z} \exp\left[\widehat{p_{\text{dec}}}\left(y_t \mid \mathbf{y}_{<t}\right)\right] \tag{2}$$

where

$$Z = \sum_{y' \in \overline{\mathcal{V}}} \exp\left[\widehat{p_{\text{dec}}}\left(y' \mid \mathbf{y}_{<t}\right)\right] \tag{3}$$

To learn the weights $\mathbf{W}^\top$, a **loss function** must be defined. Appropriate choices for probability distributions include total variation distance (TVD), equivalent to the mean absolute error of the learned distribution $p_{\text{dec}}$, and Hellinger distance (HD). Below are their respective formulations for some sequence $\mathbf{y}$, one-hot encoded corpus $\mathcal{C}$, and vocabulary $\mathcal{V}$:

$$loss_{TVD}(\mathbf{p}_{dec}, \mathbf{d}) = \sum_{\mathbf{y} \in \mathcal{C}} \sum_{t=1}^{|\mathbf{y}|} \sum_{y \in \mathcal{V}} |\mathbf{p}_{dec}(y \mid \mathbf{y}_{<t}) - \mathbf{d}(y \mid \mathbf{y}_{<t})|$$

$$loss_{HD}(\mathbf{p}_{dec}, \mathbf{d}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{\mathbf{y} \in \mathcal{C}} \sum_{t=1}^{|\mathbf{y}|} \sum_{y \in \mathcal{V}} \left(\sqrt{\mathbf{p}_{dec}(y \mid \mathbf{y}_{<t})} - \sqrt{\mathbf{d}(y \mid \mathbf{y}_{<t})}\right)^2}$$

[4] A token $x_i$'s information content is defined as $h(x_i) = -\ln p(x_i)$, where $p(x_i)$ is the probability of $x_i$ under the model. Note: in machine learning, the natural logarithm typically replaces the binary logarithm used in canonical information theory.

[5] The expected information content of a sequence $\mathcal{X} = \langle x_1, \ldots, x_n \rangle$ is defined as $H(\mathcal{X}) = -\sum_{x \in \mathcal{X}} P(x) \ln P(x)$

# 3 Tasks

1. Literature review: organization of the existing work on the topic with an outlook on its extension.
2. Implementation of code for model training, inference, and different objective functions.
3. Experimental analysis of the performance of various objective functions using a variety of models, datasets, and generation tasks.
4. Written presentation of final thesis results.

# 4 Expected Project Timeline

**Start**: 8 August 2022
**End**: 1 June 2023

- August 2022 - September 2022: Literature review
- October 2022 - March 2023: Code implementation of existing works and their extensions
- April - May 2023: Experimental analysis of proposed approaches
- May 2023: Thesis writing

# 5 Grading

The Master's Thesis (MT) is a graded semester performance. In order to successfully complete the MT, a grade of 4.0 or higher must be obtained. The supervisor establishes the assessment criteria in a written report, which can include a presentation. In principle, the following evaluation scale is applied:

| **Grade** | **Requirements** |
|---|---|
| 6.00 | Excellent |
| 5.50 | Very good |
| 5.00 | Good |
| 4.50 | Satisfactory |
| 4.00 | Sufficient |

# References


[1] H. Dong, Z. Ding, and S. Zhang. *Challenges of Reinforcement Learning*. Springer Singapore, 2020.

[2] B. Eikema and W. Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. *CoRR*, abs/2005.10283, 2020. URL `https://arxiv.org/abs/2005.10283`.

[3] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The Curious Case of Neural Text Degeneration. Mar. 2020. URL `https://openreview.net/forum?id=rygGQyrFvH`.

[4] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell. Typical Decoding for Natural Language Generation, Mar. 2022. URL `http://arxiv.org/abs/2202.00666`. arXiv:2202.00666 [cs].

[5] K. Murray and D. Chiang. Correcting length bias in neural machine translation. *CoRR*, abs/1808.10006, 2018. URL `http://arxiv.org/abs/1808.10006`.

[6] D. Pascual, B. Egressy, C. Meister, R. Cotterell, and R. Wattenhofer. A plug-and-play method for controlled text generation. *CoRR*, abs/2109.09707, 2021. URL `https://arxiv.org/abs/2109.09707`.

[7] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.

[8] O. Vinyals and Q. V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015. URL http://arxiv.org/abs/1506.05869.

[9] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

[10] S. Zarrieß, H. Voigt, and S. Schüz. Decoding methods in neural language generation: A survey. *Information*, 12(9), 2021. ISSN 2078-2489. doi: 10.3390/info12090355. URL https://www.mdpi.com/2078-2489/12/9/355.

[11] H. Zhang, D. Duckworth, D. Ippolito, and A. Neelakantan. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online, Apr. 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.humeval-1.3.