Efficient Keyword Generation using Pretrained Language Models

Cédric Goemaere[0000−0001−7308−2885], Thomas Demeester[0000−0002−9901−5768], Tim Verbelen[0000−0003−2731−7262], Bart Dhoedt[0000−0002−7271−7479], and Cedric De Boom[0000−0003−0763−8114]

Ghent University, Ghent, Belgium

Abstract. Pretrained language models can generate impressive bodies of text but often lack controllability, thereby limiting their practical use. While there is a large amount of literature on sentiment or topic control, this is not the case for keyword generation, where the goal is to generate a sentence that contains a predetermined target keyword. One study solves the problem formally, but its suggested solution is impractical, as it involves fine-tuning the entire pretrained language model for every single target. We improve on this study by optimizing the implementation of its training framework, using a mathematically more rigorous loss function and designing several parameter-efficient solutions that act as guided decoding strategies of the pretrained language model, which we view as a black box. According to our results, all our designs meet the criteria for efficient keyword generation. One approach is even capable of zero-shot generalization. Moreover, due to their simplicity, the inner workings of all our designs are interpretable, which allows for an explanation of the observed results.

Keywords: keyword generation · algorithmic efficiency · explainable AI · pretrained language models · natural language processing

1 Introduction

The capabilities of a pretrained language model (LM) are impressive, but its steerability is often limited to a simple input prompt. In practice, the user may want to impose a goal or intent on the model, e.g. a plot in story generation.

As a workaround, it is possible to modify a language model in such a way that it either is fine-tuned on the given task or can accept an additional input to specify the target. The most practical approaches are those that are as efficient as possible in terms of parameter count, training/inference time and data requirements. To that end, we view the original pretrained LM as a black box and add simple trainable layers at its output. Additionally, this creates a split between 'modeling language' and 'steering towards the target', thereby allowing the inner workings of the approach to be interpreted. To the best of our knowledge, we are the first to use this approach for controllable text generation.

While topic or sentiment are popular control targets [3], we instead focus on the task of keyword generation, where the goal is to generate a sentence that contains a predetermined target keyword[1].

[1] This paper is a thesis abstract. The full Master's thesis can be found **here**.

2 Methodology

Khalifa et al. formally solved the problem of keyword generation using pretrained LMs and designed a practical and stable training framework called KL-Adaptive DPG [2]. They chose to train a policy the size of the original LM, which limits its practical usability. Instead, we will design more efficient approaches.

Inspired by earlier work that tries to translate the cosine similarity between two word embeddings into a conditional probability [1], we suggest an approach called 'SimPolicy'. It trains a small feedforward neural network that takes 2 inputs, namely a single token's probability according to the LM and its cosine similarity to the target keyword, and produces the token's new output probability. This policy is not aware of the exact LM, token or target keyword it is working with, it only knows the given input numbers. This allows it to generalize in a zero-shot manner to previously unseen LMs and target keywords.

A different approach, named 'EmbeddingPolicy', is to multiply the LM's output probability with a target-specific constant prior over the vocabulary, similar to how a constant vector represents a specific word embedding.

While SimPolicy has the advantage of zero-shot generalization, EmbeddingPolicy has more knowledge of the specific tokens and targets it is working with. By combining these two policies, it is possible to reap the benefits from both. We call these hybrid policies 'SimbeddingPolicy' and 'EmbeSimPolicy', depending on the order of the components.

3 Results

All policies manage to include the keyword in the sentence significantly more often than the original LM (see Fig. 1). Except for SimPolicy, all policies are rather stable over the horizontal axis, which represents target keywords of increasing document frequency from left to right. While the keyword inclusion scores are quite low overall, this should not affect the end user, thanks to the low overhead of additional generations in parallel.

As a result of steering the LM towards the target, most policies suffer from a slight loss in fluency as indicated by a higher perplexity under the original LM (see Fig. 2). Only EmbeddingPolicy manages to maintain fluency.

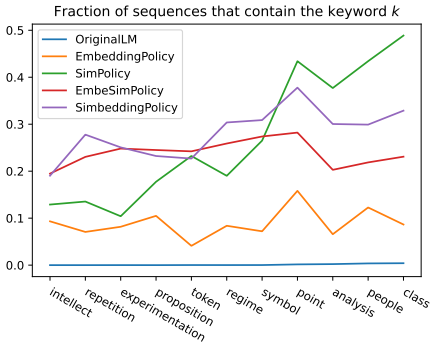

Fig. 1. Keyword inclusion score

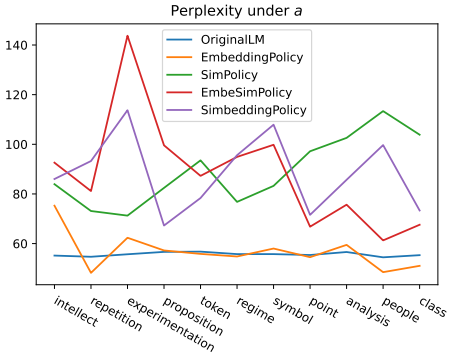

Fig. 2. Output perplexity

References

1. Blok, S.V., Medin, D.L., Osherson, D.N.: Probability from similarity (2002)
2. Khalifa, M., Elsahar, H., Dymetman, M.: A Distributional Approach to Controlled Text Generation. CoRR **abs/2012.11635** (2020), https://arxiv.org/abs/2012.11635
3. Zhang, H., Song, H., Li, S., Zhou, M., Song, D.: A survey of controllable text generation using transformer-based pre-trained language models (2022). https://doi.org/10.48550/ARXIV.2201.05337, https://arxiv.org/abs/2201.05337