1.5em 0pt

Genetic Algorithms with Logical Mapping for Natural Language Processing Prompt Optimization in Zero-Shot Data Tagging

Anonymous Authors[1]

Abstract

To Do. Title needs work

1. Introduction

Large Language Models (LLMs) have transformed the landscape of natural language processing (NLP) with their ability to generate human-like text, perform complex reasoning, and adapt to a wide range of tasks with minimal instruction. Models such as OpenAI's GPT series, Google's PaLM, and Meta's LLaMA have pushed the boundaries of what is achievable in NLP by leveraging vast training corpora and billions of parameters. Despite these advancements, their utility in real-world applications often hinges on a critical factor: the design of effective prompts.

Prompt engineering—the art and science of crafting input queries to elicit desired responses—has emerged as a necessary practice for optimizing LLM performance. A well-designed prompt can guide an LLM to deliver high-quality, task-specific outputs, while a poorly designed prompt may result in irrelevant, incoherent, or biased responses. However, designing optimal prompts is inherently challenging due to the vast search space of possible prompts and the lack of a clear framework for systematically exploring this space. This complexity is exacerbated in social science domains where precision, consistency, and domain-specific knowledge are paramount, such as stance, ideology, and misinformation detection.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Nico: Shorten above paragraph. Prepare for a ramble. We need to be upfront that the method applies to using LLMs for data annotation. Our primary problem is that prompt engineering for data annotation requires an iterative process of drafting prompts and testing performance against validation data. However, obtaining a sufficiently sized labeled validation data for niche data domains or tasks can be difficult. Additionally, small changes in the prompt can yeild large changes in performance, especially across language models. For instance, due to internal biases, some LLMs are inherently less likely label data as "fake news", but perhaps more likley to call it "misinformation". This process can be expensive and time consuming. ML practitioners and researchers, typically have an idea of the types of prompts or modifications to prompts that could change performance, but without the lengthy iterative process of prompt engineering and validation, we do not know which prompting style will work best for the data domain, task, or language model. Using a prompt optimization method is a way to solve this, and genetic algorithms are particularly useful because they represent a class of black-box optimization algorithms that allow us to abstract away the complexities of LLMs.

To address these challenges, this paper explores the potential of genetic algorithms (GAs) as a method for optimizing prompts in the context of LLMs. Genetic algorithms are well-suited for problems characterized by large, nonlinear search spaces and an absence of explicit gradient information—conditions that align with the prompt engineering problem. By iteratively evolving populations of candidate prompts through mechanisms inspired by natural selection, such as crossover, mutation, and fitness-based selection, GAs can efficiently explore and refine prompts to maximize LLM performance on a given task.

We hypothesize that incorporating genetic algorithms into the prompt engineering process will not only enhance the quality of prompts but also provide valuable insights into the characteristics of effective LLM inputs. This study contributes to the growing body of research on automated prompt optimization, offering a framework that merges the strengths of evolutionary computation and large language models. Our results aim to demonstrate how genetic algorithms can be leveraged to improve the adaptability and efficacy of LLMs in common natural language processing

(NLP) tasks, such as stance detection, ideology identification, and misinformation detection. Additionally, we aim to show how GAs can help optimize poorly worded prompts, ultimately improving the lower bound of LLM performance by refining inputs to yield more accurate and relevant outputs.

1.1. Contributions

Our main contribution is a method to optimize zero-shot and few-shot classification prompts with GAs. Our results show that this approach can significantly improve LLM data annotation performance without additional model training. The key contributions include:

- Novel genetic representation of classification prompts which allows us to use algorithms for efficient genetic operations on text-based prompts. Our results demonstrate...
- Evaluation of three fitness functions that do not require a labeled subset of data for validation: loss-based, confidence score-based, and mixed. Our findings highlight...

In addition, we developed a comprehensive and extensible Python package that implements our methodology. The package, along with detailed experiments and documentation, is available at BLANK.

2. Related Works

Much work has been done in the automatic prompt optimization space.

Nico: moved basic GA description up. This can even be shortened

Genetic algorithms (GAs) are metaheuristic optimization algorithms inspired by the biological evolution process, mimicking Darwin's theory of the survival of the fittest (Michalewicz & Schoenauer, 1996; Katoch et al., 2021). GAs typically involve chromosome representation, fitness selection, and biologically inspired operators (Katoch et al., 2021). In a classic GA, new populations evolve through changes in genetic operators applied to individuals, where key components include chromosome representation, selection, crossover, mutation, and the fitness function.

(Lehman et al., 2022) show that LLMs can improve the effectiveness of mutation operators applied to genetic programming since they benefit from training data that includes sequential changes and modifications similar to what humans would make. Similarly, (Guo et al., 2024) has shown that evolutionary algorithms are an effective method for prompt optimization; however, the authors use a different algorithm, candidate fitness score, and application. (Tanaka et al., 2023) show that genetic algorithm can be used for more effective prompts, but their methodology differs significantly as they use the accuracy of the true labels as the fitness score turning the LLM into a supervised learning algorithm and potentially overfitting the prompting to the task.

This is also loosely related to the concepts of prompt fine tuning: we need to be sure to make a differentiating statement.

3. Optimizing Classification Prompts with GAs

3.1. Prompt Components for Genetic Representation

Because the space of all feasible prompts (represented by natural language) is intractable, we introduce a simple logical mapping to separate the prompts into several components for chromosome representation. These components include:

- The Classification Target- The text to be classified.
- Context - Background information about the classification task.
- Class Labels - Possible classification categories.
- Instructions - Directions for the classification task.
- Response Restrictions - Constraints on how the model should respond.

For example, consider the following prompt:

```
"Taxation is Theft!"
Stance classification is the task of
    determining the attitude or position of
     a piece of text towards a topic. What
    is the stance of the previous statement
     toward taxes?
a) for
b) against
c) neutral
Respond with only one word.
```

In the example, the classification target is "Taxation is Theft!" The context is the second sentence: "Stance classification is the task of determining..." The class labels are "a) for) against) neutral." The instructions are "What is the stance of the previous statement toward taxes?" and the response restrictions are "Respond with only one word." Few-shot prompts also include two additional prompt components: a few-shot context, which introduces and provides context for the few-shot examples, and the few-shot examples themselves.

To run a GA to optimize a prompt, our method requires the user to specify candidates for each prompt component, effectively defining a small subset of the search space. However, this task is not a significant burden, as users familiar with the data and classification task typically have a good intuition for effective prompting strategies. Moreover, this process can be assisted with LLMs, reducing the manual effort required.

Beyond selecting prompt components, our framework also

allows users to optimize the exact format and ordering of these components. By defining multiple prompt formats using format strings, users can explore different structures to identify the most effective configuration. This flexibility enables fine-grained prompt tuning while still leveraging domain expertise to guide the search.

3.2. Genetic Operations

Due to our genetic representation of prompts with logical components, we can adopt traditional methods of crossover and selection. However, introducing prompt mutations requires some creativity when working with text. We can break down mutations for prompts into three categories: swapping mutations, text-based mutations, and language-based mutations. These mutations help the algorithm explore more of the search space, potentially escaping local minima in the user-defined subspace.

3.2.1. Swapping Mutations

Swap mutations have a long precedence in GAs and can still be applied to prompts under our genetic representation. Some examples include swapping the order of few-shot examples, swapping the order of class labels, and swapping the order of certain prompt components. Adjusting the order of examples or labels may help when a specific arrangement limits the language model's ability to generalize.

3.2.2. Text-Based Mutations

Text-based mutations apply small edits to the prompt that may impact model performance. These include modifications such as adjusting punctuation, capitalization, or spacing. While minor, such changes can subtly influence how the prompt is tokenized and affect the model's ability to interepet the prompt.

3.2.3. Language Based Mutations

Introduce mutation to the language themselve using masked-language models or LLM. With MLMs we can take advantage of the masked pretraining by randomly selected a subset of words in the promp tot choose. Nico: THIS SECTION NEEDS SOME MORE WORK
Like previous work, we can also use the LLM to tune specific components. For example reword this prompt components:

3.3. Prompt Fitness

We evaluate three different fitness scores in our genetic algorithm implementation: an uncertainty quantification metric, model loss, and a multi-fitness score that is the combination of the two.

3.3.1. Loss Based Fitness

Nico: It makes more sense to introduce loss based fitness first because people are most familiar with loss in ML

Loss based fitness is the process of evaluating the fitness of a prompt based on the loss of a language model **without the predicted label**. We use the inverse of the loss to represent fitness with the intuition being that by maximizing the inverse of the loss, we are drafting a prompt that is more likely to be produced by the language model. Our assumption is that lower-perplexity, prompts are more likely produce a correct label.

3.3.2. Confidence Score Based Fitness

Nico: Say upfront in plain english what the confidence score is. Then say what is does which is the sentence below.
Using a confidence score metric that evaluates the distance between token probabilities for the fitness function was chosen due to its simplicity and shown efficacy in evaluating language model performance in zero-shot settings with constrained choices in (Farr et al., 2024). The confidence, Let $\mathcal{T}$ represent the set of given tokens, and $P(t)$ denote the distribution of log probabilities across each token $t \in \mathcal{T}$. The log probability is then computed using the formula

$$C = \left| \max_{t \in \mathcal{T}} P(t) - \max_{t \in \mathcal{T} \setminus \{t^*\}} P(t) \right|, \tag{1}$$

where t^* is the token corresponding to the highest probability $\max_{t \in \mathcal{T}} P(t)$ (Farr et al., 2024).

3.3.3. Loss-Confidence Mixed Fitness

We also adopt a mixed fitness function that combines the loss-based fitness and the confidence score-based fitness. We use a simple weighted average of the two fitness functions. The goal of the mixed fitness function is to balance created prompts that can maximize the confidence for lower-loss prompts. Ideally, this allows us to avoid scenarios where confidence converges to the incorrect label and help draft a prompt that will help prevent the model from responding confidently to incorrect labels.

3.4. Classifying Text with Optimized Prompts

Nico: This subsection header can be improved maybe

Describe specific implementation of our algorithm.

3.4.1. Prompt-per-target Optimization

3.4.2. Optimizing the Prompts for Many Classification Targets

Nico: This includes randomly swapping (which can be viewed as a mutation I guess) in classification targets between generations and calculating fitness using the average across a sampled bactch of exmaple targets.

Nico: The notation does not help the reader and is a lengthy description of concepts that are not novel to our implementation. I'd expect the reviewer to know most of this terminology. These are still good notes to have, but we super shrink this and move most of it to section 4.3

Sam: Does a standard algorithm description work better? Yea I think we just do that...

Algorithm 1 Genetic Algorithm

Generate initial population Y_0 of m chromosomes by randomly seeding alleles from predetermined parent population.
Select $n \leq m$ parents for population Y_1 using fitness evaluation.
for $generation = 1, 2, \ldots, g$ **do**
1. Select n parent pairs from Y_i using modified Roulette Wheel Selection.
   - Fix all n available parents as a first parent
   - Select second parent using standard Roulette Wheel Selection
2. Generate n offspring O_i from parent pairs.
   - Apply uniform crossover to selected parent pairs
   - Apply random mutations, each with distinct probabilities (M_p), to each offspring in O_i:
     - Base Mutations
     - Simple Text Mutations
     - Masked Language Model (MLM) Mutations
     - Large Language Model (LLM) Mutations
3. Evaluate fitness of mutated offspring O_i'.
4. Select top n individuals from combined parents Y_i and mutated offspring O_i' for parent generation Y_{i+1}.

end for

Use algorithm in place of this

The genetic algorithm begins with an initial population Y_0 of m chromosomes. This population is generated by randomly seeding alleles from a predetermined parent population. From population Y_0, $n \leq m$ parents are selected for population Y_1 using fitness evaluation. All subsequent generations have population size n.

Let $Y_i = \{y_{i1}, y_{i2}, ..., y_{in}\}$ be defined as the current ith generation of parents. A modified version of Roulette Wheel Selection is used to choose parent chromosomes. The first parent y_{ij}, where $1 \leq j \leq n$, is chosen by fixing each chromosome in Y_i. The second parent is chosen using standard Roulette Wheel Selection from chromosomes in $Y_i \backslash \{y_{ij}\}$. This ensures that each offspring O has distinct parent chromosomes.

Let $O_i = \{o_{i1}, o_{i2}, ..., o_{in}\}$ be defined as the current ith generation of offspring. To generate offspring o_{ij}, where $1 \leq j \leq n$, uniform crossover is applied to the selected parents. Each allele from the parent chromosomes has an equal probability of being passed on to the offspring o_{ij}.

To reach an optimal prompt, small random changes (mutations) are applied to each offspring in O_i. In this methodology, mutations can be classified as:

- Base Mutations
- Simple Text Mutations
- Masked Language Model (MLM) Mutations
- Large Language Model (LLM) Mutations

These mutations are used to explore the search space and avoid local minima. Each mutation is given a probability (M_p) to be applied randomly to an allele present in $o_{ij} \in O_i$. When applied, these operations generate the population of mutated offspring O_i'. The mutated offspring O_i' are then scored using fitness evaluation. These offspring are compared to the current generation of parents Y_i and the top n prompts are passed on to the next generation.

These operations—selection, crossover, and mutation—are repeated iteratively until the maximum number of generations has been reached.

4. Experimental Set Up

4.1. Data

To provide easy comparability and use, we leveraged common benchmark datasets for the stance, ideology, and misinformation datasets.

4.1.1. Stance Detection

Stance detection is defined as "a classification problem where the stance of the author of the text is sought in the form of a category label from this set: Favor, Against, Neither" (Ng & Carley, 2022). For stance detection, we used the SemEval-16 dataset provided by (Mohammad et al., 2016). The SemEval-16 dataset consists of approximately 5000 tweets in relation to one of five targets: Hilary Clinton, Legalization of Abortion, Feminism, Climate Change, and Atheism. To allow optimal prompt convergence, we treated each target class as its own data set.

4.1.2. IBC

Ideology can be defined as "the shared framework of mental models that groups of individuals possess that provide both an interpretation of the environment and a prescription as to how that environment should be structured"(North & Denzau, 1994). We evaluated our methodology on the Ideological Books Corpus (IBC) from (Sim et al., 2013)

with sub-sentential annotations (Iyyer et al., 2014). The IBC dataset consists of 4326 sentences of which 1701 are labeled conservative, 600 neutral, and 2,025 liberal.

4.1.3. MISINFO

4.2. Models

We use Llama3B-Instuct, huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct, ...

To adapt large language models (LLMs) for GA prompt optimization, we introduce a few implementation-specific optimizations. First, we select the classification label by identifying the token with the maximum log-probability for each label after a forward pass through the network. This approach avoids the complexities associated with autoregressive text generation and decoding schemes. Additionally, we use population sizes that fit into a small number of inference batches (often one), enabling efficient calculation of prompt fitness.

4.3. Genetic Algorithm Set Up

Fill in what mutations with what probabilities do we use. What do we use for crossover, what do we use for selection. What population sizes do we use, how many generations do we test? How do we set up our specific experiments?

5. Results

6. Discussion

7. Conclusion

7.1. Availability

The authors of this work believe in open-source and reproducible research. The open-source models used are available on huggingface at `huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct`. The package, along with detailed experiments, documentation, and data is available at BLANK.

References

Farr, D., Cruickshank, I., Manzonelli, N., Clark, N., Starbird, K., and West, J. Llm confidence evaluation measures in zero-shot css classification, 2024. URL https://arxiv.org/abs/2410.13047.

Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers, 2024. URL https://arxiv.org/abs/2309.08532.

Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., and Daumé III, H. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 633–644, 2014.

Katoch, S., Chauhan, S. S., and Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

Lehman, J., Gordon, J., Jain, S., Ndousse, K., Yeh, C., and Stanley, K. O. Evolution through large models, 2022. URL https://arxiv.org/abs/2206.08896.

Michalewicz, Z. and Schoenauer, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

Mohammad, S. M., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June 2016.

Ng, L. H. X. and Carley, K. M. Is my stance the same as your stance? a cross validation study of stance detection datasets. *Information Processing & Management*, 59(6): 103070, 2022.

North, D. and Denzau, A. Shared mental models: Ideologies and institutions. *Kyklos*, 47:3–31, 02 1994. doi: 10.1111/j.1467-6435.1994.tb02246.x.

Sim, Y., Acree, B. D. L., Gross, J. H., and Smith, N. A. Measuring ideological proportions in political speeches. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S. (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 91–101, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1010.

Tanaka, H., Mori, N., and Okada, M. Genetic algorithm for prompt engineering with novel genetic operators. In *2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter)*, pp. 209–214, 2023. doi: 10.1109/IIAI-AAI-Winter61682.2023.00047.