Genetic Generative Information Retrieval

Hrishikesh Kulkarni
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

Zachary Young
Georgetown University
Washington, DC, USA
zjy2@georgetown.edu

Nazli Goharian
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

Ophir Frieder
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

Sean MacAvaney
University of Glasgow
Glasgow, UK
first.last@glasgow.ac.uk

ABSTRACT

Documents come in all shapes and sizes and are created by many different means, including now-a-days, generative language models. We demonstrate that a simple genetic algorithm can improve generative information retrieval by using a document's text as a *genetic representation*, a relevance model as a *fitness function*, and a large language model as a *genetic operator* that introduces diversity through random changes to the text to produce new documents. By "mutating" highly-relevant documents and "crossing over" content between documents, we produce new documents of greater relevance to a user's information need — validated in terms of estimated relevance scores from various models and via a preliminary human evaluation. We also identify challenges that demand further study.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

genetic algorithm, large language models, generative information retrieval

ACM Reference Format:
Hrishikesh Kulkarni, Zachary Young, Nazli Goharian, Ophir Frieder, and Sean MacAvaney. 2023. Genetic Generative Information Retrieval. In *ACM Symposium on Document Engineering 2023 (DocEng '23), August 22–25, 2023, Limerick, Ireland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3573128.3609340

1 INTRODUCTION

Documents come in all shapes and sizes and are created by many different means, including now-a-days, generative language models. Identifying relevant documents and ranking them by relevance to the query is the aim of Information Retrieval (IR). Here, relevance attributes to timeliness and suitability of the results with reference to the query. Typically, IR uses lexical and dense approaches. Lexical approaches utilize query term occurrences [19]. On the other hand, dense approaches work in the semantic vector space where interaction between words in the queries are modelled or a single vector representation of the query is learned [21]. Although systems vary considerably in their implementations — ranging from purely lexical or dense, to hybrid or multi-stage re-ranking systems [28] — most systems these days aim to produce an "organic" result list consisting of retrieved and ranked documents directly. In contrast, Generative IR aims to provide direct answers to user queries rather than identifying sources potentially supplemented with snippets.

In Generative IR, Generative Document Retrieval, ranks relevant documents using encoder-decoder architecture while Grounded Answer Generation generates query-specific answers with a supporting document. We focus on Grounded Answer Generation where we re-frame this problem as a genetic algorithm: Genetic Generative Information Retrieval (Gen^{2}IR). Rather than making a single pass over the results, Gen^{2}IR *iteratively* makes use of a LLM as a *random genetic operator* that attempts to re-write the top results to better fit the query. The controlled mutations and cross-overs across the top candidates help in generating the new candidates. The *fitness* of each candidate is determined using a relevance scoring model — an approach that has worked well in other recent works [9]. As the population (i.e., ranked list of results) *evolves*, the most relevant mutated versions of the documents float to the top and used for future generations of documents. Finally, the process terminates when no new document is scored within the top *d* documents after a given iteration. Figure 1 gives an overview of Gen^{2}IR system.

We conducted experiments with multiple approaches for mutation and cross-over with varying parameters namely: number of mutations per iteration, sampling depth for mutations, and termination depth. A preliminary set of experiments on this technique using both automatic and human evaluators on several datasets suggest that Gen^{2}IR outperforms re-ranking based and pure generative models (generate and filter). However, we identify a few

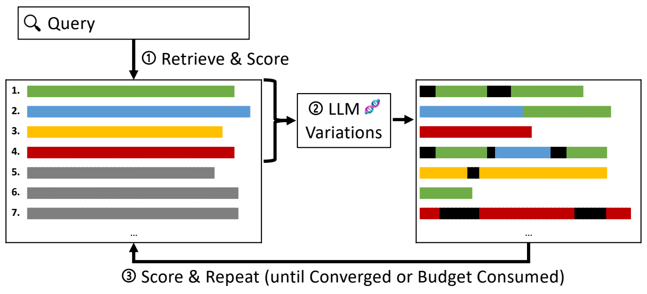

Figure 1: Overview of our Gen^{2}IR system.

challenges that remain, including the tendency for the language models to hallucinate details not present in the source documents and a misalignment of our fitness function and human preferences when dealing with similar documents (a problem that has been identified previously with respect to system evaluation [2]).

Thus the major contributions of this paper are:

- Modeling Generative IR as a genetic algorithm
- Showing effectiveness of Gen^{2}IR through exhaustive evaluation on multiple datasets

2 RELATED WORK

IR methods are broadly classified into lexical, or word-based methods and dense, or semantic methods. The lexical and dense methods graduated to hybrid models striking the balance between flexibility and accuracy. Recently, generative IR models, which use LLMs, have spawned considerable community interest.

Lexical and Dense IR. Lexical methods focus on exact word matching delivering efficiency while dense methods go beyond the words in the semantic space to decode meaning and strive for accuracy.

Traditional approaches for retrieval use query terms and frequency of their occurrence in target documents based on exact term matching yielding efficient models like BM25 [26]. Further, query likelihood [29] and query expansion based techniques were introduced to improve retrieval accuracy, by attempting to overcome the exact term matching bottleneck.

Neural networks took retrieval beyond hand-crafted text features. In using neural models, relevance is obtained by calculating similarity between embedding space vectors of query and document, capturing latent semantic characteristics. Pre-trained transformer architectures further improved retrieval systems [10, 17].

Two primary neural retrieval models exist: interaction-based where interactions between the query terms are modelled and representation-based where a model learns a single vector representation of the query [21]. There also exist hybrid models [1] where dense models are used on top of initial efficient lexical filtering models like BM25 [16]. Additionally, proximity graph based methods are also used to execute efficient dense retrieval using nearest-neighbor search algorithms [14, 18].

Discriminative and Generative IR. IR models can also be classified as discriminative and generative models.

Discriminative models determine relevance between the query and document, learning a decision boundary separating relevant documents. Thus, Rank-SVM [30] and neural and transformer based ranking architectures like TAS-B [10] and TCT-ColBERT-HNP [17] fall under such models. While discriminative models focus on a target value, generative models overcome space boundaries and perceived bottlenecks by interacting in parametric space [13, 15].

In Generative Document Retrieval, a generate-then-read approach was shown to outperform state of the art retrieve-then-read pipeline [31]. Attempts to encode document identifiers into the model instead of using sparse/dense index based index-retrieval-rerank pipeline were made, but scaling to massive data is still a challenge [32]. Obviously, another limitation is retrieving information not available during training. To overcome these limitations different variants of generative models like contextualized IR [15] were proposed. As widely known, when LLMs generate answers to queries, hallucination is the key concern. Attempts were made to use augmentation to make LLMs generate responses grounded in external knowledge [23], but the problem persists.

Algorithm 1 Gen^{2}IR

```
Input: $q$ query, $t$ document sampling depth, $m$ mutations per iteration, $d$ termination depth
Output: $r$ query specific high relevance answer
  $R_0 \leftarrow$ Re-rankingPipline$(q, t)$                ▷ base population
  $R \leftarrow$ GeneticOperators$(R_0, m)$                 ▷ generate population
  $R \leftarrow$ FitnessFunction$(R, t)$                    ▷ worthy population
  $R_{d0} \leftarrow$ Filter$(R_0, d)$                      ▷ old top set
  $R_{d1} \leftarrow$ Filter$(R, d)$                        ▷ new top set
  while $|R_{d1} \setminus R_{d0}| \neq 0$ do               ▷ until no new doc beats top $d$
      $R \leftarrow$ GeneticOperators$(R, m)$               ▷ generate population
      $R \leftarrow$ FitnessFunction$(R, t)$                ▷ worthy population
      $R_{d0} \leftarrow R_{d1}$                            ▷ old top set
      $R_{d1} \leftarrow$ Filter$(R, d)$                    ▷ new top set
  end while
```

Genetic and Filtering Methods in IR. Genetic algorithms are evolutionary algorithms inspired from natural evolution where new generations evolve by taking selective traits from parents. There are past attempts that applied genetic algorithms to IR [7, 8, 22, 27]. However, our work differs from these prior efforts in that they focused on clustering, ranking, and query rewriting, whereas we explore genetic algorithms in the process of generating direct responses to queries from generated results.

Recent works have employed using a relevance model as a fitness function to filter the outputs of text generation models. Specifically, Jeronymo et al. [11] use a relevance function to filter out bad generated training data, and Gospodinov et al. [9] use a relevance function to filter bad generated queries for document expansion. Our work is inspired from these efforts, but differs in that we apply the fitness function as part of a larger genetic generation process.

3 GEN^{2}IR

Traditional retrieval systems based exclusively on retrieving existing documents are limited by how content was presented by the document's original authors. This may not always be the best format for presenting results for a specific user query. Meanwhile, systems based exclusively on generation from a language model lack information provenance. Grounded Answer Generation systems aim to overcome both limitations by generating direct answers from retrieval results. We propose a Grounded Answer Generation system that uses a genetic algorithm to iteratively refine the quality of the answer for the user's query. Gen^{2}IR brings advantages of discriminative, generative and genetic approaches together. We propose cross-over and mutation operators in the context of Gen^{2}IR. In Algorithm 1, we illustrate the evolutionary process of Gen^{2}IR tailored by three parameters in producing query-specific, highly-relevant answers.

Genetic Operators are used to generate new fitter documents. They work on top of results to introduce diversity. Here, the top results are determined by using a re-ranking pipeline with BM25 as a first stage retriever and Electra model [25] for re-ranking.

Gen^{2}IR has three genetic operators:

- Randomized document mutation (summarize the document)
- Query specific document mutation (re-write the document to better answer the query)
- Multi-point document crossover (re-write the two documents to better answer the query)

We use GPT-3 [4] `text-davinci-edit-001` with default parameters to perform the above mutations and crossovers. The number of mutations performed in a single iteration is determined by parameter 'no. of mutations *m*'. While mutation brings diversity by either query specific or randomized selective change, crossover brings diversity by selective combination of multiple documents.

Population of documents is initially generated using a re-ranking pipeline. The above genetic operators generate offsprings for the next level by generating new population each iteration. The worthy population that clears the fitness criteria, participates in the evolution process. The evolution process continues until the termination criteria is reached. The termination criteria in Gen^{2}IR is determined by the parameter 'termination depth *d*'. When the top *d* documents saturate, the evolutionary process terminates.

Fitness function is used to determine the worthy population for the next iteration. We use Electra model as our fitness function. The parameter, 'document sampling depth *t*' is used to determine the number of worthy documents that can participate in the next iteration. A genetic operator for an iteration is randomly decided. In each of the *m* mutations, we randomly sample one or two documents from the worthy population (top *t* documents) based on the genetic operator.

4 EXPERIMENTAL SETUP

Via experimentation, we answer the following research questions:

RQ1 Can genetic generation improve the relevance of retrieved passages?
RQ2 Does genetic generation's iterative nature improve the relevance over a simple generate-and-filter approach of the same cost?
RQ3 Does optimizing the 'termination depth', 'document sampling depth' and 'mutations per iteration' parameters lead to improved performance?
RQ4 What is the prevalence of hallucinated content in Gen^{2}IR output?

4.1 Datasets

To evaluate Gen^{2}IR, we used the following datasets:

- **Dev (sample).** First 100 queries from the Dev (small) subset used as validation data [3].
- **TREC 2019 Deep Learning (Passage Subtask).** Dataset containing 43 queries along with manual judgements used in TREC 2019 [5].
- **TREC 2020 Deep Learning (Passage Subtask).** Dataset containing 54 queries along with manual judgements used in TREC 2020 [6].

4.2 Models and Baselines

We use BM25 for first stage retrieval in the re-ranking pipeline. We then use the Electra model [25] for re-ranking the first stage results. In every iteration, to determine worthy population, we use the Electra model to rank candidates. To evaluate, we use both model-based evaluation and a human evaluation. For models, we measure the preference between the original retrieved top document and Gen^{2}IR output using the MonoT5 model [20] and the DuoT5 model [24]. For the human evaluation, we asked an external assessor to select between the original top result or the top Gen^{2}IR result.[1] Further, the authors also annotated the results to evaluate inter-annotator agreement. We have released the code to reproduce the results of our experiments on GitHub.[2]

Table 1: Relevance as per ELECTRA, MonoT5 and Human '+' denotes cases where Gen^{2}IR output is preferred; '-' denotes cases where Re-ranking output is preferred; '=' denotes cases where both outputs are equivalent

	Dev (sample)			DL'19			DL'20		
Evaluator	+	=	−	+	=	−	+	=	−
ELECTRA*	91	9	0	39	4	0	50	4	0
MonoT5	78	9	13	34	4	5	43	4	7
tuned Gen^{2}IR									
ELECTRA*	98	2	0	42	1	0	54	0	0
MonoT5	87	2	11	37	1	5	50	0	4
Human	67	4	29	34	1	8	29	1	24

Table 2: Relevance with DuoT5

Evaluator	Dev (sample)	DL'19	DL'20
DuoT5	0.7290	0.7838	0.7174
DuoT5 (tuned)	0.7629	0.8067	0.7769

5 RESULTS AND ANALYSIS

RQ1: Relevance. In Table 1, we show that in both Electra and MonoT5 evaluations, in both the default and tuned parameters, the output of Gen^{2}IR is preferred. These results are further supported by human evaluation. Additionally, in Table 2 we show DuoT5 scores averaged over the queries. Here, values significantly greater than 0.5 show that DuoT5 finds the Gen^{2}IR output better than the re-ranking pipeline output. This is observed across parameters and datasets addressing RQ1. We also observe that the Gen^{2}IR model with tuned parameters results in more relevant answers. Equivalency is observed when the precise answer is present in the corpus and the re-ranking pipeline ranks it as the best. As a side note, the average inter-annotator agreement on a 10% randomly sampled set of Dev (sample) dataset among 5 annotators is 63%. Here, the additional assessors also used shuffled outputs similar to the external assessor to prevent bias. Even though, assessor agreement is a challenge for evaluating these types of models going forward, we note that individual annotators favor the Gen^{2}IR output over traditional re-ranking pipeline output.

RQ2: Effectiveness of Iterations. In Table 3, we list the number of queries where the output of Gen^{2}IR is prefered as compared with a popular generate and filter approach. Here we remove the termination criteria and use a common budget of 7 iterations (upper

[1]The assessor is provided shuffled outputs for unbiased assessment
[2]https://github.com/Georgetown-IR-Lab/gen2ir

Table 3: Effectiveness of Iterations

	Dev (sample) — Prefers...		
Evaluator	Gen^{2}IR	Same	Generative
ELECTRA*	99	1	0
MonoT5	70	1	29
Human	37	3	60

Table 4: Parameters

		Sampling Depth				Mutn./Iter			Term. Depth		
		2	5	10	15	4	8	12	1	2	3
Electra*	+	97	95	91	87	75	91	94	78	91	93
	=	3	5	9	13	25	9	6	22	9	7
	−	0	0	0	0	0	0	0	0	0	0
MonoT5	+	86	81	78	69	58	78	81	66	78	78
	=	3	5	9	13	25	9	6	22	9	7
	−	11	14	13	18	17	13	13	12	13	15

quartile for RQ1) with 8 mutations per iteration. We note that even though the models prefer the Gen^{2}IR output, thus highlighting the importance of iterative nature, human evaluation slightly prefers the generate and filter output. We conclude that the fitness function here is not in alignment with our human evaluator's preferences Additionally, when evaluating the same setup of Gen^{2}IR against Generative with DuoT5, we obtain a score of 0.5809 which shows that DuoT5 finds Gen^{2}IR output more relevant as compared with the popular generate and filter approach. The obtained improvement in relevance comes at the cost of increased latency due to the inherent iterative nature of genetic approach.

RQ3: Parameters. We vary the three parameters namely: document sampling depth t, mutations per iteration m, and termination depth d to draw inference about the change in performance of Gen^{2}IR as evident in Table 4. We observe decreasing preference with increasing sampling depth and better performance with increasing number of mutations but at the cost of latency. The performance saturates at termination depth of 2. In Table 1, we show that the tuned model, based on the above analysis, significantly outperforms the default parameter setup. Further, the tuned model, when tuned using MonoT5, also shows improved performance when evaluated using DuoT5 as evident in Table 2 addressing RQ3.

RQ4: Hallucination. Hallucination is a common phenomenon in deep learning based generative models [12]. Here, the generated text is unintended and fails to meet user expectations. We present a study of presence of hallucination in Gen^{2}IR output for 20 randomly sampled queries across three datasets. In 13 out of 20 cases at least some new information was found. Even though correct, presence of hallucinated contents in generated answers is one of the limitations. Further, based on the contingency tables considering hallucination and relevance, we observe that the distribution of hallucination in 'Gen^{2}IR relevant' column is approximately the same as that in the complete set. From this we infer that the relevance can be attributed to Gen^{2}IR and not to the tendency of generative models to produce unintended text addressing RQ4. Both human evaluation and MonoT5 based relevance judgements were considered for creating contingency tables for the study.

6 CONCLUSION

We proposed a new way to approach generative retrieval by leveraging a genetic perspective. The proposed genetic operators create new population and select top candidates to participate in evolution process based on fitness function. We present a preliminary study into the effectiveness of the approach, which suggests that the approach has the potential to improve upon existing methods. It can find application in domains demanding holistic relevance ahead of time criticality.

We acknowledge the limitations in the approach, however. First, our study is limited to a relatively small number of queries. This was, in part, due to the human cost of annotation, which does not always align with our automatic evaluation measures. Future work is needed to better align these fine-grained preferences in relevance models. Further, despite being explicitly instructed to simply edit the existing documents, the models are still prone to hallucination, which also needs to be addressed. Finally, the iterative nature of the genetic process is costly, which we need to overcome to make the approach economical to use in practice.

REFERENCES

[1] Negar Arabzadeh et al. 2021. Predicting Efficiency/Effectiveness Trade-Offs for Dense vs. Sparse Retrieval Strategy Selection. In *CIKM*.
[2] Negar Arabzadeh et al. 2022. Shallow pooling for sparse labels. *Inf. Retr. J.* (2022).
[3] Payal Bajaj et al. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *InCoCo@NIPS*.
[4] Tom Brown et al. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.
[5] Nick Craswell et al. 2020. Overview of the TREC 2019 deep learning track.
[6] Nick Craswell et al. 2021. Overview of the TREC 2020 deep learning track.
[7] Ophir Frieder et al. 1991. On the Allocation of Documents in Information Retrieval Systems. In *SIGIR*.
[8] Ophir Frieder et al. 1997. Document Allocation in Multiprocessor Information Retrieval Systems. In *IEEE Transactions on KDE*.
[9] Mitko Gospodinov et al. 2023. Doc2Query−: When Less is More. In *ECIR*.
[10] Sebastian Hofstätter et al. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *SIGIR*.
[11] Vitor Jeronymo et al. 2023. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *arXiv preprint arXiv:2301.01820* (2023).
[12] Ziwei Ji et al. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12 (2023).
[13] Volodymyr Kuleshov and Stefano Ermon. 2017. Deep Hybrid Models: Bridging Discriminative and Generative Approaches. In *Conference on Uncertainty in AI*.
[14] Hrishikesh Kulkarni et al. 2023. Lexically Accelerated Dense Retrieval. In *SIGIR*.
[15] Hyunji Lee et al. 2022. Contextualized Generative Retrieval. *ArXiv* (2022).
[16] Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2022. Efficient Neural Ranking Using Forward Indexes. In *WWW*.
[17] Sheng-Chieh Lin et al. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *ACL RepL4NLP*.
[18] S. MacAvaney et al. 2022. Adaptive Re-Ranking with a Corpus Graph. In *CIKM*.
[19] Tomas Mikolov et al. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.
[20] Rodrigo Nogueira et al. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP*.
[21] L Pang et al. 2016. A Study of MatchPyramid Models on Adhoc Retrieval. NeuIR.
[22] P. Pathak et al. 2000. Effective information retrieval using genetic algorithms based matching functions adaptation. In *Hawaii Int'l Conf. on System Sciences*.
[23] Baolin Peng et al. 2023. Improving Large Language Models with External Knowledge and Automated Feedback. arXiv:2302.12813
[24] Ronak Pradeep et al. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *ArXiv* (2021).
[25] Ronak Pradeep et al. 2022. Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. In *ECIR 2022*.
[26] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* (2009).
[27] Alexandra Russell. 2019. A Genetic Algorithm for Query Optimization. (2019).
[28] Saar Kuzi et al. 2020. Leveraging Semantic and Lexical Matching to Improve the Recall of Document Retrieval Systems: A Hybrid Approach. *CoRR* (2020).
[29] X. Xue et al. 2009. Query Substitution based on N-gram Analysis. In *SIGIR*.
[30] H. Yu et al. 2009. An Efficient Method for Learning Ranking SVM. In *KDD*.
[31] Wenhao Yu et al. 2023. Generate rather than Retrieve. In *ICLR 2023*.
[32] Yu-Jia Zhou et al. 2023. DynamicRetriever: A Pre-trained Model-based IR System Without an Explicit Index. *Machine Intelligence Research* (2023).