Hyperparameter Optimization for Search Relevance in E-Commerce

Manuel Dalcastagné and Giuseppe Di Fabbrizio
VUI, Inc., Boston, U.S.A.

Keywords: Hyperparameter Optimization, Differential Evolution, e-Commerce Search Relevance Optimization.

Abstract: The tuning of retrieval and ranking strategies in search engines is traditionally done manually by search experts in a time-consuming and often irreproducible process. A typical use case is field boosting in keyword-based search, where the ranking weights of different document fields are changed in a trial-and-error process to obtain what seems to be the best possible results on a set of manually picked user queries. Hyperparameter optimization (HPO) can automatically tune search engines' hyperparameters like field boosts and solve these problems. To the best of our knowledge, there has been little work in the research community regarding the application of HPO to search relevance in e-commerce. This work demonstrates the effectiveness of HPO techniques for optimizing the relevance of e-commerce search engines using a real-world dataset and evaluation setup, providing guidelines on key aspects to consider for the application of HPO to search relevance. Differential evolution (DE) optimization achieves up to 13% improvement in terms of NDCG@10 over baseline search configurations on a publicly available dataset.

1 INTRODUCTION

Modern e-commerce platforms rely on search engines to help customers find relevant products from catalogs containing millions of items. Configuring these platforms is challenging and requires carefully modeling the query intent, product attributes, customer behavior, and other factors influencing relevance. Most search engines have numerous hyperparameters that can significantly impact both retrieval and ranking of results. Traditionally, these options are tuned manually in a time-consuming and often irreproducible process as the queries, products, and customer preferences evolve continuously over time.

In recent years, hyperparameter optimization (HPO) techniques have been successfully used to configure automatically many types of algorithms as well as complex machine learning models (Feurer and Hutter, 2019; Eggensperger et al., 2019). HPO employs a class of models usually called black-box or derivative-free, as no mathematical closed-form formulation of an objective function is necessary and the only requirement is a metric for numerical estimation. These techniques search through a multi-dimensional space of possible hyperparameter configurations to find the settings that optimize a performance metric such as NDCG (Wang et al., 2013).

To the best of our knowledge, there has been little work in the research community on e-commerce applications of HPO for search relevance. One notable exception is the work by Cavalcante et al. (Cavalcante et al., 2020), who used Bayesian Optimization to tune the ranking function of a customer support search application on a private dataset. However, their work did not explore different query structures, field boosting, query intent or query classification (Di Fabbrizio et al., 2024). Our work is one of the first to systematically apply HPO techniques to optimize relevance in e-commerce and to provide guidelines regarding the application of HPO to this context.

The main contributions of this work are: 1) application of differential evolution (DE) on a publicly available e-commerce dataset for search relevance optimization; 2) analysis of the dataset's label distribution impact on search relevance; 3) tuning of precision and recall-oriented Elasticsearch queries, and variants thereof, observing improvements up to 13% in terms of NDCG@10; 4) insights into the impact of field boosting, query structure, and query understanding on relevance; 5) guidelines on key aspects to consider when applying HPO to search relevance, such as the characteristics of the search space, multifidelity, or the use of multiple metrics for multi-objective optimization.

The remainder of this paper is structured as follows. Section 2 provides the problem definition. Section 3 introduces HPO and DE. Section 4 describes the WANDS evaluation dataset. Section 5 presents

Dalcastagné, M. and Di Fabbrizio, G.
Hyperparameter Optimization for Search Relevance in E-Commerce.
DOI: 10.5220/0013010500003838

In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 1: KDIR*, pages 399-407
ISBN: 978-989-758-716-0; ISSN: 2184-3228

setup, results, and analysis of the experiments. Finally, Section 6 concludes the paper and outlines potential future research.

2 PROBLEM DEFINITION

Users traditionally search by typing natural language queries that define what they are looking for (*user's intent*). As a response, a search engine retrieves and ranks a set of relevant documents from a corpus of possibly multiple document types, whose specifics are determined in dedicated document *schemas*. A *document type* is represented as a collection of named fields, also known as attributes or features, that are employed to build ranking signals quantifying the relevance of each field with respect to search queries.

2.1 Index Time and Query Time

Modern search engines index and query documents at separate times, but decisions taken at index time might impact on both the performance and quality of results retrieved at query time. At index time, the fields of each document are analyzed and indexed: each feature is divided into tokens, mapped to a type (e.g., string, numeric, date), processed and transformed in one or more fields that are indexed (i.e., according to the signals to be modeled). Also, details about the keyword and vector algorithms to be used for ranking are usually defined at this point. For example, if using BM25 (Robertson and Zaragoza, 2009) as a ranking algorithm, its b and k_1 hyperparameters could be optimized during this phase.

Although the application of HPO is possible at both stages, doing so at index time is significantly more expensive from a computational perspective - changes to the index usually require the reindexing of the whole corpus. This work focuses only on query-time applications of HPO, but the same techniques can be applied to optimize hyperparameters with impact at index time.

2.2 HPO for Search Relevance

The application of HPO involves two steps. First, define the hyperparameters to tune (i.e., type, range, relationships with other hyperparameters) and a budget to spend for the optimization process (e.g., number of function evaluations). Second, run an optimization loop where a search algorithm iteratively explores the space defined previously to find the best possible configuration of the hyperparameters by using some user-defined metric to evaluate each configuration.

In search relevance optimization (SRO), hyperparameters correspond to properties of the search engine query (e.g., values of field boosts, type of logical operators), while user-defined metrics are information retrieval (IR) metrics like precision, recall, or NDCG. Therefore, in order to evaluate a retrieval and ranking strategy over a corpus of documents, a dataset should contain a representative set of search queries and a collection of sets of relevance labels, defining the relevance of each document that could appear in the top results of each user query.

More precisely, let $\mathcal{D}$ be a dataset of triplets (q,d,y) where q is a search query, d is a document and y is a relevance label that defines the relevance of d for q, and let $\mathcal{S}$ be a search engine with a given index structure, whose output depends on a vector of hyperparameters $\theta \in \Theta^t$ that define an optimization search space of dimension t. The optimization goal is to heuristically find the best possible configuration θ^* by using a training dataset $\mathcal{D}_{train}$ to estimate the performance of $\mathcal{S}$ during the optimization and a validation dataset $\mathcal{D}_{val}$ to prevent overfitting, so that θ^* generalizes to a test dataset $\mathcal{D}_{test}$ that was not employed during the optimization. Ideally, all these datasets should be large enough to ensure statistically sound decisions. If $\mathcal{D}$ is not large enough, methods like k-fold-cross-validation can be used to split available data in folds to be combined as k training and test sets. Therefore, the performance of any θ is estimated as $p_{train,\theta} = \mathcal{S}(\theta, \mathcal{D}_{train})$ and, at the end of the optimization, the quality of θ^* is estimated as $p_{test,\theta^*} = \mathcal{S}(\theta^*, \mathcal{D}_{test})$. Finally, to evaluate the contribution of the optimization, the whole process is repeated k times and the optimized performance of $\mathcal{S}$ is estimated as

$$p_{test} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{S}(\theta_i^*, \mathcal{D}_{test,i}) \quad (1)$$

where θ_i^* is the best configuration found at optimization i by using $\mathcal{D}_{train,i}$ as training dataset and $\mathcal{D}_{test,i}$ as test dataset from the i-th split. It is important to highlight that all splits are based on folds coming from the same initial randomized sampling process. As a result, the repeated estimation and averaging over multiple splits results in an estimate of generalization error with lower variance (Kohavi, 1995).

3 OPTIMIZATION

HPO algorithms are usually classified as model-free (e.g., variants of stochastic search like *differential evolution*) or model-based (e.g., Bayesian optimization), where a model is used to estimate the re-

sponse of the objective function to be optimized. Both approaches have advantages and disadvantages, and picking the right algorithm for the problem at hand depends on multiple factors that include search space characteristics (i.e., size, type of hyperparameters) or latency requirements (Feurer and Hutter, 2019; Bischl et al., 2023).

3.1 HPO Search Space and Latency

Due to the curse of dimensionality (Bellman, 1966), the size of the search space has a large influence on the optimization. The larger the size, the harder it is for the algorithm to find well-performing configurations of the hyperparameters. Furthermore, not all algorithms are able to scale with the number of dimensions. For example, standard Bayesian optimization (BO) based on Gaussian processes is not usually efficient on problems with more than 20 dimensions, but it excels in continuous spaces (Eggensperger et al., 2013; Frazier, 2018). In contrast, BO based on random forests and evolutionary algorithms like DE are not as efficient. Still, they are able to handle larger search spaces based on mixed hyperparameters as well.

When performance evaluations are computationally expensive, which can happen when the objective function requires training on large datasets, it might be helpful to consider multi-fidelity algorithms like Successive Halving (Jamieson and Talwalkar, 2016) or Hyperband (Li et al., 2017) to schedule monotonically the use of low-fidelity (less expensive) and high-fidelity (more expensive) evaluations during the optimization, to spend the budget efficiently. For example, DEHB (Awad et al., 2021) uses Differential evolution (DE) as an optimization algorithm to search θ in combination with a variant of *hyperband*, performing better than the more famous BOHB (Falkner et al., 2018) on a wide range of problems, including the tuning of deep learning networks.

Optimization algorithms differ as well in their parallelizability capabilities. In fact, model-free algorithms are usually more scalable since model-based methods are less parallelizable due to the presence of a common model that must be iteratively updated. For more details, refer to (Feurer and Hutter, 2019; Bischl et al., 2023).

3.2 Differential Evolution

The optimization algorithm used in the experiments is Differential evolution (Storn and Price, 1997), which is an evolutionary algorithm inspired by the concepts of biological evolution and natural selection, specifically by how the offspring inheriting the best traits of a population evolve over generations.

At the beginning of the process, a population $p_0 = (\theta_1, \ldots, \theta_n)$ is randomly sampled from Θ^t. Until some user-defined optimization budget b is consumed, DE works iteratively in three steps: mutation, crossover, and selection. During the mutation phase, each member θ of the population p_i at the current iteration i is evaluated by computing $\mathcal{S}(\theta, \mathcal{D}_{train})$. Then, a new set of n offsprings is generated by applying a scaled perturbation to each dimension of a new offspring θ_{new} resulting from the combination of randomly picked parents from p_i. A crossover operator combines each member of p_i with one of the new offsprings θ_{new}, by picking for each dimension with some probability which value from the two vectors should be used for the mutant configuration θ_{mutant}. Finally, θ_{mutant} is compared with θ, and θ_{mutant} possibly takes place of θ if its quality is better.

4 EVALUATION DATASET

The Wayfair Annotation DataSet (WANDS) is an open-source e-commerce product dataset designed to evaluate the relevancy of e-commerce product search engines (Chen et al., 2022). As described in Table 1, the WANDS dataset contains:

- 480 search queries sampled from real search logs of Wayfair, a major e-commerce retailer, with two features: query text and class. For example, a query like *smart coffee table* belongs to the *Coffee & Cocktail Tables* class. The queries were stratified sampled to cover various dimensions such as popularity, seasonality, and whether they led to customer purchases. This ensures the query set is representative of real customer search behavior.
- 42,994 products sampled from Wayfair's catalog, with nine features of which only the following five textual features were used for field boosting in the experiments: product name, class, description, category hierarchy and list of features. For example, a product named *solid wood platform bed* belongs to the *Bed* class within a category hierarchy like *Furniture / Bedroom Furniture / Beds* and has a list of features that contains information like color, material, size or weight.

  For each query, Wayfair selected a set of potentially relevant products using a combination of customer click logs, lexical search systems, and neural retrieval models. Specifically, the dataset authors employed two strategies to construct the product pool:

1. They leveraged user engagement data (clicks and add-to-cart events), hypothesizing that products users clicked on are a good approximation of potentially relevant products, while products users clicked on but didn't add to the cart could be hard negatives or almost-relevant products.
2. They further mined the product catalog using an open-source lexical search engine (Apache Solr) and a neural product retrieval system inspired by (Nigam et al., 2019). The two systems provide complementary ways to retrieve relevant products, removing the bias related to the use of a single lexical retrieval source. Moreover, this hybrid approach ensures the product set contains both obviously relevant products as well as more challenging cases that can help discriminate between different retrieval systems.

- 233,448 (query, product) pairs assigning one out of three relevance labels to the match of query and product: exact (1.0) if the product is completely relevant to the query, partial (0.5) if the product matches some but not all aspects of the query, and irrelevant (0.0) if the product is not relevant to the query.

Note that the statistics are based on the most recent version available on GitHub[1] which is slightly different from the version in (Chen et al., 2022).

A group of trained human annotators provided the labels following a rigorous set of annotation guidelines. Each *(query, product)* pair was judged by up to 3 annotators, and the ratings were aggregated using a majority vote. The WANDS dataset was constructed through multiple rounds of annotation and refinement. The inter-annotator agreement, measured by Cohen's Kappa (Cohen, 1960), improved from a moderate 0.467 in the initial months to a substantial 0.826 after a few iterations of guideline refinement and annotator training. This indicates the dataset labels are of high quality and consistency.

A key feature of WANDS is that it aims for *completeness* - i.e., for a given query, the dataset tries to include relevance labels for *all* the relevant products from the catalog subset, not just the top few results. This is achieved through an iterative "cross-referencing" process during dataset construction that identifies potentially relevant products that were not covered in the initial labeling. Completeness is important for unbiased offline evaluation as it avoids missing relevant products that could unfairly penalize certain retrieval systems. The complete, multi-graded relevance labels allow for a robust evaluation of the ranking quality of search engines using metrics like NDCG.

To evaluate the difficulty of the search relevance task in the WANDS dataset, we analyzed the distribution of relevance labels (exact match, partial match, irrelevant) across the queries. The goal was to understand how many queries have products labeled as only exact matches, only partial matches, only irrelevant, or a mixture of these labels. This analysis provides insights into the difficulty of ranking the search results for each query.

Assuming that, on average, each query contains the same proportion of exact, partial, and irrelevant labels as the overall distribution in the dataset, we found that:

- 0 queries have products with only the Exact label, 24 queries have products with only the Partial label, 1 query has products with only the Irrelevant label
- 33 queries have products with only Exact and Partial labels, 11 queries have products with only Exact and Irrelevant labels, 76 queries have products with only Irrelevant and Partial labels

This analysis reveals that 25 queries do not have an impact on NDCG, and 11 queries should have results that are relatively easy to rank. Around 100 queries are of medium difficulty, while the rest are more challenging. However, the distribution of labels across queries is not balanced, which is an important consideration for learning to rank (LtR) models (Goswami et al., 2018). If the number of labels per type is imbalanced, the model may be more prone to overfitting. For example, queries with a highly skewed distribution of exact and partial matches are easier to achieve a good NDCG score compared to queries which have a more balanced distribution of exact and partial matches.

This analysis highlights the importance of considering the distribution of relevance labels when evaluating the difficulty of the search relevance task and the potential impact on the performance of ranking models. The WANDS dataset provides a diverse set of queries with varying levels of difficulty, making it a valuable resource for evaluating and comparing different search engines and ranking algorithms in the e-commerce domain.

5 EXPERIMENTS AND RESULTS

We provide experimental results to demonstrate how hyperparameter optimization can be leveraged to

[1] https://github.com/wayfair/WANDS

Table 1: Summary of key data statistics about the WANDS dataset.

Feature	Value
Number of queries	480
Number of products	42,994
Number of (query, product) relevance labels	233,448
Relevance label scale	0-2
Relevance label distribution	Exact: 25,614 Partial: 146,633 Irrelevant: 61,201
Search queries from	Real search logs of Wayfair
Products sampled from	Wayfair's catalog
Annotators per (query, product) pair	Up to 3
Inter-annotator agreement (Cohen's Kappa)	0.826

automate solutions to many information retrieval and search problems commonly encountered in e-commerce. The focus is on optimizing hyperparameters of search queries and ranking signals used by search engines in keyword search. Elasticsearch is used as an experimental framework, but the techniques mentioned in this section are applicable to any other engine that supports the manual tuning of its components.

Experiments start from the consideration that both TF-IDF and BM25 have some ranking strategy limits, which can be partially addressed through the use of optimization for field boosting. It is worth mentioning that well-tuned boosts are critical not only to rank the expected importance of different signals but also to balance the range of the respective BM25 scores.

5.1 BM25 Limits and Field Boosting

Scores based on TF-IDF have some shortcomings, which are partially solved by the BM25 formulation. TF-IDF's score for a term in a corpus is computed as the product of term frequency and inverse document frequency. A problem comes from the unconstrained impact of term frequency on the score (i.e., a term that appears n times in a document implies that a document is n times more relevant than another document without any occurrence). Also, the length of a document does not weight the relevance of its terms (e.g., if a term appears once in a document containing 10 words, it is considered to be as relevant as if the term appears once in a document containing 1000 words). BM25's b parameter restrains the degree to which term frequency can impact the score, determining a penalty for documents longer than the average, and the influence of common terms on the score is saturated by BM25's k_1 parameter.

Nonetheless, the scores of fields can be on different scales due to distribution differences of frequencies and document lengths and are, therefore, not directly comparable. Also, by definition, these scores are biased towards information, usually against users' needs (i.e., rare matches within a document score higher, while users usually look for popular items). Field boosting helps counterbalance the aforementioned problems by prioritizing and balancing signals from different fields. In fact, a search query usually contains more than one string and possibly multiple concepts. It does not come as a surprise that the information required to return relevant results is often stored in multiple fields.

Elasticsearch tries to solve some of TF-IDF's problems by changing how token frequencies are combined to compute scores during a multi-field search by considering the frequencies coming from multiple fields at the same time. In particular, field-centric search (e.g., `multi_match best_fields` and `most_fields`) focuses towards precision by promoting results which satisfy criteria based on the signals which are expected to match the user's search, while term-centric search (e.g., `cross_fields`, `combined_fields`) focuses towards recall, by selecting all possibly relevant search results (Turnbull and Berryman, 2016). The use of either conjunctive (AND) or disjunctive operator (OR) further pushes these queries towards precision or recall, respectively.

The combination of recall-oriented and precision-oriented clauses in a *stratified* query improves the ranking of the results returned to the user (Turnbull and Berryman, 2016). In Elasticsearch, this can be achieved using a boolean query, which matches documents satisfying boolean combinations of other queries (e.g., `multi_match` queries), where some clauses provide a recall-oriented base score that is improved by other precision-oriented clauses. For example, the base score may come from a `multi_match cross_fields` query searching in all text fields, while other scores may come from `multi_match best_fields` or `most_fields` queries based on high-quality signals.

5.2 User's Intent

Understanding the user's intent is another critical signal that significantly improves search relevance in e-commerce. This involves classifying whether the user is searching for a specific product category or asking a broader, more informational question. By using a machine learning model to predict the user's intent based on query structure, search patterns, and historical behavior, the system can adjust its ranking strategy to deliver more relevant results. For example, when seeking a specific product, search results can prioritize relevant items from the desired category. Conversely, if the user's query is informational, the system can prioritize results such as FAQs, reviews, or other informative content. Incorporating intent prediction into the search optimization process allows for more accurate recommendations and a highly personalized shopping experience.

5.3 Multi-Objective Optimization

In the experiments, we use NDCG@10 over the labeled dataset to evaluate the relevance performance of a given search engine configuration θ. The normalized discounted cumulative gain (NDCG) (Wang et al., 2013) measures the relevance of the top-ranked results, putting more emphasis on the relevance of results at higher ranks (Järvelin and Kekäläinen, 2000). This aligns well with users' behavior and preferences on e-commerce search result pages, who tend to focus mainly on the first page of results. Still, while a single metric is a good starting point for assessing the quality of search relevance performance, it might only tell part of the story.

NDCG assumes that labeled documents are uniformly distributed in the ranked list, which is usually untrue. In Section 4, we showed that even a well-built dataset like WANDS falls in more extreme situations where not all relevance labels are found for more than 100 use queries, and in some cases, only one class of relevance labels might be retrieved. A metric like NDCG cannot detect such scenarios and would return a perfect value even if some queries were evaluated, for example, only on irrelevant documents. To obtain robust evaluations, one should combine at least an order-aware metric like NDCG or Mean Reciprocal Rank with an order-unaware metric like Precision or Recall. For further details about these indicators or variants thereof, please refer to (Valcarce et al., 2018).

The optimization of multiple equally important but conflicting objectives is named multi-objective optimization, where solutions that optimize all objectives simultaneously usually do not exist (Helfrich et al., 2023). In this scenario, heuristic algorithms try to find efficient, non-dominated solutions concerning the defined objectives. An alternative solution is to employ scalarization techniques to systematically approximate a multi-objective optimization problem into a regular single-objective optimization problem with the help of additional parameters such as weights and use regular optimization problems to solve the resulting scalarization. For further details about multi-objective HPO algorithms, please refer to (Feurer and Hutter, 2019; Bischl et al., 2023).

5.4 Experimental Setup

Text fields from WANDS were indexed using Elasticsearch's English analyzer, without any additional preprocessing steps. In particular, all experiments were run on Elasticsearch 8.8.2 and Python 3.10. To ensure replicability and improved comparison of results, all splits and optimization runs were carried out multiple times with a common set of random seeds. This ensured that the evaluations utilized to build estimators were paired. In addition, we computed random ranking values based on 5 repetitions, similar to how k-fold cross-validation was employed with $k = 5$. According to the experiment, the search space size varies from 8 to 27 dimensions, and each optimization run is executed up to a budget b of 400 function evaluations. Results on the test set are considered only for evaluation purposes at $b = 50$, 100, 200, and 400. Unless explicitly defined, the experiments' optimized hyperparameters were defined as in Table 2. For further details about the role of these hyperparameters in multi-field queries, please refer to Elasticsearch documentation. Finally, the DE implementation used in the experiments is the default version available on GitHub[2] from the Python package created by the authors of DEHB.

5.5 Random and Standard Baselines

In this work, we consider both random and standard ranking as baselines against which to evaluate the contribution of HPO. The random ranking provides a ranking baseline for the problem, by assigning to each document from the set of results of a retrieval strategy a pseudorandom number in the range [0,1]. As a result, it is possible to compute any performance metric on the resulting ranked list. For example, if using NDCG, higher values imply easier ranking problems. Similar considerations can be achieved analyzing the distribution of relevance labels across the dataset.

[2]https://github.com/automl/DEHB

Table 2: Hyperparameter used in the experiments.

Name	Type	Range	Default value
operator	categorical	{and, or}	or
type	categorical	{best_fields, most_fields, cross_fields}	none
minimum_should_match	ordinal	{0%, 20%, 40%, 60%, 80%, 100%}	none
tie_breaker	float	[0, 1]	0
boost	float	[0, 100]	1

Standard ranking quantifies how the standard configuration of a search engine's ranking strategy performs with respect to a completely random ranking strategy. Unlikely the general HPO scenario, where good initial configurations of hyperparameters are usually unknown, search engines come with default values that work well on average. As a consequence, the corresponding ranking performance should be considered as well as a baseline.

5.6 Optimization Improvements

The contribution of the optimization to ranking strategies is empirically estimated by showing the improvement that DE is able to achieve with respect to the standard ranking of multiple retrieval queries with a fixed structure and increasing difficulty. Results show that the optimization contributed, on average, to an improvement of approximately 0.05 in terms of NDCG@10 on 12 cases. Our optimization strategy was not only employed to fully optimize both retrieval and ranking parts of each type of Elasticsearch query used in the experiments, but it also proved its effectiveness. It was able to reach comparable results with respect to its optimized counterparts with a fixed retrieval structure, providing reassurance about its success.

Results were built on three main types of Elasticsearch queries that were increasingly difficult. In the first set of experiments (Table 3, top), basic types of multi-field query are used distinctively in combination with both conjunctive and disjunctive operators. On average, the optimization achieves an improvement of 0.07. Once optimized, precision-oriented queries achieve the same results, and therefore, only one of the two is going to be considered in the following experiments. A Boolean query is employed to build a stratified query in the second set of experiments (Table 3, middle). On average, the optimization achieves an improvement of 0.05. The best results are interestingly achieved by combining a recall-oriented query based on the conjunctive operator and a precision-oriented query based on the disjunctive operator. In the third set of experiments (Table 3, bottom), the best stratified query from the previous experiments is extended with an additional multi-field query that considers user intent. On average, the optimization achieves an improvement of 0.04, and the introduction of user intent contributes approximately 0.03 - 0.04 with respect to the best results from the previous sets of experiments.

5.7 Retrieval Relaxation Improvements

All results show that, on average, queries using the conjunctive operator perform worse than queries adopting the disjunctive operator. In particular, random ranking results allow us to infer that performance values can be improved by relaxing the matching requirements and retrieving more potentially relevant documents that could be otherwise excluded from further ranking refinement. This behavior aligns with modern multi-stage IR systems that rely on multiple ranking phases, where the first phase focuses on recall and successive steps towards precision (Dang et al., 2013; Zhou and Devlin, 2021).

6 CONCLUSIONS

This work demonstrates the potential for HPO techniques to substantially improve the search relevance of e-commerce engines with minimal human effort in a reproducible and automatic process, providing insights into the impact of field boosting, retrieval query structure, and query understanding on relevance, as well as guidelines on the application of HPO to search relevance in e-commerce.

By leveraging the WANDS evaluation dataset and DE as HPO algorithm, we automatically optimized both retrieval and ranking strategies of Elasticsearch queries, improving NDCG@10 up to 13% with respect to baseline configurations. The introduction of the user's intent in the search strategy, defined as correspondence between the category of user query and document, brought an improvement of up to 4 %. Finally, results showed that the relaxation of the retrieval strategy led to significantly better results. Default search engine configurations leave significant room for relevance improvements that can be un-

Table 3: Best results from the first set (top), the second set (middle), and the third set of experiments (bottom). All performance metrics are expressed as averaged NDCG@10 with standard deviation, and results with highest average are in bold for each column.

Query type	Operator	Space Size	Random	Standard	Optimized
cross_fields	OR	8	0.53 ± 0.01	0.60 ± 0.00	0.73 ± 0.02
cross_fields	AND	8	0.49 ± 0.00	0.52 ± 0.01	0.59 ± 0.02
best_fields	OR	8	0.54 ± 0.01	0.60 ± 0.01	0.73 ± 0.01
best_fields	AND	8	0.46 ± 0.00	0.48 ± 0.01	0.52 ± 0.04
most_fields	OR	8	$\mathbf{0.60 \pm 0.00}$	$\mathbf{0.69 \pm 0.00}$	0.73 ± 0.01
most_fields	AND	8	0.49 ± 0.00	0.51 ± 0.01	0.52 ± 0.04
optimized	optimized	10	/	/	$\mathbf{0.75 \pm 0.02}$
stratified	OR, OR	17	0.62 ± 0.00	0.71 ± 0.00	0.74 ± 0.02
stratified	OR, AND	17	0.59 ± 0.00	0.64 ± 0.00	0.74 ± 0.03
stratified	AND, OR	17	$\mathbf{0.64 \pm 0.00}$	$\mathbf{0.72 \pm 0.00}$	$\mathbf{0.75 \pm 0.02}$
stratified	AND, AND	17	0.52 ± 0.00	0.55 ± 0.01	0.58 ± 0.02
optimized	optimized	21	/	/	0.74 ± 0.02
stratified, most_fields	AND, OR, AND	21	0.65 ± 0.00	0.74 ± 0.00	0.77 ± 0.02
stratified, cross_fields	AND, OR, OR	21	0.65 ± 0.00	0.74 ± 0.00	0.78 ± 0.03
optimized	optimized	27	/	/	$\mathbf{0.78 \pm 0.02}$

locked with HPO, through a reproducible process that does not keep humans in the never-ending loop of manual search relevance optimization.

Picking the best algorithm for search relevance optimization depends on various factors including the size and type of hyperparameters, as well as multi-fidelity and multi-objective requirements. Evolutionary algorithms like DE are capable of handling large mixed search spaces, but unless the size of the search space goes beyond hundreds of dimensions, random-forests-based BO is another possible option. Furthermore, when performance evaluations are expensive due to the need for large datasets, options such as multi-fidelity HPO algorithms should be considered. Finally, to obtain robust configurations, one should consider multi-objective HPO algorithms to optimize for both order-aware and order-unaware metrics, or to create a scalarization of such metrics to apply regular HPO algorithms like DE.

While the optimal configuration will vary for each search application, this work establishes a general framework, methodology, and best practices for applying HPO to improve search relevance. With the increasing availability of easy-to-use HPO libraries and their integration with popular search engines, we believe this is a highly promising direction to improve the search experience for e-commerce customers with less manual effort and greater reproducibility.

This work focuses on optimizing keyword-based search, but it is worth noting the complementary role of dense vector search using learned semantic representations (Mitra and Craswell, 2018). In many search use cases where user queries primarily consist of named entities like product names or brands, exact keyword matching remains critical and even preferable. However, modern search engines offer hybrid search capabilities that combine the strenghts of sparse keyword-based retrieval with dense vector search. This hybrid approach is commonly used in retrieval augmented generation (RAG) architectures, as purely semantic search can miss obvious keyword matches needed for accurate product retrieval in e-commerce and for more strongly grounded factual knowledge retrieval (Lewis et al., 2020).

Finally, other several exciting avenues for future work in this area include:

- Exploration of the benefits that HPO can bring to hybrid search, such as improvements to the fine-tuning process of embedding models used in dense vector search or the configuration of other hyperparameters used in multi-stage IR systems;
- Application of multi-objective optimization to jointly optimize multiple metrics that measure different aspects of the results;
- Investigation of possible interactions as well as differences between HPO and LtR techniques for search relevance.

REFERENCES

Awad, N. H., Mallik, N., and Hutter, F. (2021). DEHB: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization. *Proceedings of IJCAI*.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., et al. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

Cavalcante, L., Lima, U., Barbosa, L., Gomes, A. L., Éden Santana, and Martins, T. (2020). Improving Search Quality with Automatic Ranking Evaluation and Tuning. In *Anais do XXXV Simpósio Brasileiro de Bancos de Dados*, Brasil.

Chen, Y., Khrennikov, D., Ferrer, I., and Verberne, S. (2022). WANDS: A Dataset for Web-based Product Search. In *European Conference on Information Retrieval*, pages 61–75. Springer.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*.

Dang, V., Bendersky, M., and Croft, W. B. (2013). Two-stage learning to rank for information retrieval. In *Advances in Information Retrieval*. Springer.

Di Fabbrizio, G., Stepanov, E., and Tessaro, F. (2024). Extreme Multi-label Query Classification for E-commerce. In *The SIGIR 2024 Workshop on eCommerce*, Washington, D.C., USA.

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., et al. (2013). Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, Nevada. Curran Associates, Inc.

Eggensperger, K., Lindauer, M., and Hutter, F. (2019). Pitfalls and best practices in algorithm configuration. *Journal of Artificial Intelligence Research*, 64:861–893.

Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*.

Feurer, M. and Hutter, F. (2019). *Hyperparameter Optimization*, chapter 1, pages 3–38. Springer, Cham.

Frazier, P. I. (2018). Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs.

Goswami, A., Zhai, C., and Mohapatra, P. (2018). Learning to rank and discover for e-commerce search. In *14th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2018)*, pages 331–346, Germany. Springer.

Helfrich, S., Herzel, A., Ruzika, S., and Thielen, C. (2023). Using scalarizations for the approximation of multiobjective optimization problems: towards a general theory. *Mathematical Methods of Operations Research*, pages 1–37.

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*.

Järvelin, K. and Kekäläinen, J. (2000). IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval*, New York, NY, USA. Association for Computing Machinery.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: a novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816.

Mitra, B. and Craswell, N. (2018). An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*.

Nigam, P., Song, Y., Mohan, V., Lakshman, V., Ding, W. A., Shingavi, A., Teo, C. H., Gu, H., and Yin, B. (2019). Semantic product search. In *Proceedings of KDD*, New York, NY, USA. Association for Computing Machinery.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359.

Turnbull, D. and Berryman, J. (2016). *Relevant Search: With applications for Solr and Elasticsearch*. Manning Publications Co., USA.

Valcarce, D., Bellogín, A., Parapar, J., and Castells, P. (2018). On the robustness and discriminative power of information retrieval metrics for top-n recommendation. In *Proceedings of the 12th ACM conference on recommender systems*, pages 260–268.

Wang, Y., Wang, L., Li, Y., He, D., and Liu, T. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. In *COLT 2013 - The 26th Annual Conference on Learning Theory*.

Zhou, G. and Devlin, J. (2021). Multi-vector attention models for deep re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5452–5456.