Knowledge Graphs: Semantics, Machine Learning, and Languages
M. Acosta et al. (Eds.)

doi:10.3233/SSW230009

BiPaSs: Further Investigation of Fast Pathfinding in Wikidata

Leon MARTIN [a,1]
[a] *Media Informatics, University of Bamberg, An der Weberei 5, 96047 Bamberg.*
ORCiD ID: Leon Martin https://orcid.org/0000-0002-6747-5524

Abstract. *Purpose:* A previous paper proposed a bidirectional A* search algorithm for quickly finding meaningful paths in Wikidata that leverages semantic distances between entities as part of the search heuristics. However, the work lacks an optimization of the algorithm's hyperparameters and an evaluation on a large dataset among others. The purpose of the present paper is to address these open points.
Methodology: Approaches aimed at enhancing the accuracy of the semantic distances are discussed. Furthermore, different options for constructing a dataset of dual-entity queries for pathfinding in Wikidata are explored. 20% of the compiled dataset are utilized to fine-tune the algorithm's hyperparameters using the Simple optimizer. The optimized configuration is subsequently evaluated against alternative configurations, including a baseline, using the remaining 80% of the dataset.
Findings: The additional consideration of entity descriptions increases the accuracy of the semantic distances. A dual-entity query dataset with 1,196 entity pairs is derived from the TREC 2007 Million Query Track dataset. The optimization yields the values 0.699/0.109/0.823 for the hyperparameters. This configuration achieves a higher coverage of the test set (79.2%) with few entity visits (24.7 on average) and moderate path lengths (4.4 on average). For reproducibility, the implementation called BiPaSs, the query dataset, and the benchmark results are provided.
Value: Web search engines reliably generate knowledge panels with summarizing information only in response to queries mentioning a single entity. This paper shows that quickly finding paths between unseen entities in Wikidata is feasible. Based on these paths, knowledge panels for dual-entity queries can be generated that provide an explanation of the mentioned entities' relationship, potentially satisfying the users' information need.

Keywords. knowledge graphs, pathfinding, hyperparameter optimization, Wikidata

1. Introduction

To satisfy the users' information need more quickly, the result pages of modern web search engines such as Google[2], Bing[3], and Startpage[4] feature a variety of components in addition to the standard ranking of search results. One prominent example are knowledge panels, which are typically located in the top right corner of the result pages. These

[1] Mail: leon.martin@uni-bamberg.de.
[2] https://www.google.com (accessed 2023/05/26)
[3] https://www.bing.com (accessed 2023/05/26)
[4] https://www.startpage.com (accessed 2023/05/26)

Table 1. A subset of the information displayed in the knowledge panel variants of Google, Bing, and Startpage when the two queries *European Union* and *Alan Turing* are issued individually. Information from third-party sources like weather services has been left out. The search engines were set to English and the searches were conducted on 2023/05/26.

Query: *European Union*			Query: *Alan Turing*		
Knowledge panel of			Knowledge panel of		
Google	Bing	Startpage	Google	Bing	Startpage
Area	Description	Description	Occupation	Occupation	Description
Founding Date	Capital	Motto	Born	Born	Born
Founders	Largest metropolis	Anthem	Died	Died	Died
Awards	Official languages	Capital	Movies	Cause of death	Cause of death
Subsidiary	Official scripts	Institutional seats	Influenced by	Education	Education
	Religion	Largest metropolis	Siblings	Alma mater	Alma mater
	Demonym(s)	Official languages	Awards	Known for	Known for
	...	...		...	...

box-shaped interface elements are populated with information from purpose-built knowledge bases, namely Knowledge Graphs (KGs) [1]. In a blog entry from 2012, Google unveiled their KG as a means of improving their search engine through three primary functions: the disambiguation of entities, the generation of summaries of entities, and the provision of links to associated entities. Especially the latter two functions contribute to the composition of the knowledge panel's content. Using the example of two queries, Table 1 shows the variety of information that the knowledge panel variants of the three web search engines comprise when a query mentioning a single entity is issued. Note how the types of the queried entities, which are in this case a political union and a human being, affect the information presented in the knowledge panels. The reasons for this are twofold: First, the employed KGs use different properties to describe entities of different categories, and second, the search engines rank the entity information differently.

The examples demonstrate that knowledge panels for single-entity queries, i.e., queries mentioning exactly one entity, offer useful information. However, the quality of the knowledge panels decreases when dual-entity queries, i.e., queries mentioning exactly two entities, are issued. For instance, given the query *European Union Alan Turing*[5], Google displays no knowledge panel at all while Bing presents the same knowledge panel as for the query *Alan Turing*. Startpage shows a knowledge panel with information about a UK student exchange program named after Turing. While Startpage's result is a good attempt, the authors of [2] argue that knowledge panels for dual-entity queries could explain the relationship between the two mentioned entities, thereby potentially satisfying the users' information need without requiring them to consult the ranked search entries. Especially when the two entities are semantically distant, the way they are connected via a path in a KG, can provide valuable information for the users. The task of entity relationship explanation, which is well-known in the knowledge discovery discipline [3], is defined in [4] as follows:

> Given a pair of entities e and e', provide an explanation, i.e., a textual description, supported by a KG, of how the pair of entities is related.

[5]The searches were conducted on 2023/05/26.

The authors of [2] interpreted this task as a pathfinding problem, where a path found between e and e' in a KG serves as a means of describing the entity relationship. Web search engines like those mentioned before allow users to issue arbitrary textual queries with entities from basically any domain. Accordingly, they used Wikidata [5], a large general-domain KG, for their work since more specialized KGs do not encompass this scope. Applied to the previous example, one useful path between the entities *European Union* and *Alan Turing* in Wikidata is[6]:

Q458 (European Union) −P530 (diplomatic relation)→ Q145 (United Kingdom)
←P27 (country of citizenship)− Q7251 (Alan Turing)

Note that properties in a KG can be interpreted in both directions. Therefore, the edges of a KG can be considered as bidirectional [3]. The vast size and generality of KGs like Wikidata pose several concrete problems for pathfinding [2]:

- Uninformed search algorithms like breadth-first search might not suffice for pathfinding.
- KG interfaces struggle to deliver all edges of a queried entity.
- Users desire meaningful entity relationships.

To tackle these problems, [2] proposes a bidirectional A* search algorithm [6] that considers the semantic distance between entities estimated via word embeddings of their labels to guide the search. The algorithm is parameterized with three hyperparameters α, β, and γ that weight the individual components of the employed cost function (s. Section 2). Despite the promising results, the paper leaves several open points: The algorithm's performance was evaluated based on only twelve hand-selected dual-entity queries using hand-selected configurations of the hyperparameters. Moreover, further tests (s. Section 3) with the original algorithm indicated that the entity labels alone do not yield accurate word embeddings and thus compromise the search. To tackle these open points, the present paper investigates the following research questions:

RQ1: How to estimate the semantic distances between entities more accurately?
RQ2: How to obtain realistic dual-entity queries for pathfinding in Wikidata?
RQ3: What is an optimized hyperparameter configuration for the algorithm?
RQ4: How does the optimized hyperparameter configuration perform (against other configurations) on a large dual-entity query dataset?

The remainder of the present paper is organized as follows: Section 2 investigates foundations and related work, including a thorough recapitulation of [2]. Building upon this, Section 3 focuses on answering RQ1 through RQ3. Section 4 discusses the implementation developed in the context of the present paper[7], before RQ4 is addressed in Section 5. Finally, Section 6 draws a conclusion and provides directions on future work.

[6] In this notation, the properties within the arrows (edges) connect the surrounding entities (nodes) in the respective direction. The strings with leading Q and P are Wikidata's proprietary IDs for entities and properties.

[7] The implementation and all resources required to reproduce the results are available in the GitHub repository at https://github.com/uniba-mi/bipass-wikidata-pathfinder, which is also indexed in the Software Heritage Project's archive (https://archive.softwareheritage.org; accessed 2023/05/26).

2. Foundations & Related Work

The Resource Description Framework (RDF) [7] represents a generic approach for expressing knowledge in the form of triples. In this framework, Internationalized Resource Identifiers (IRIs) are utilized as identifiers, which are a generalization of Uniform Resource Identifiers (URIs). Each triple consists of a subject, which can be an IRI or a blank node, a predicate, which is an IRI, and an object, which can be an IRI, a literal, or a blank node. The predicate denotes a property, which is a binary relation between the subject and the object expressing a statement. Although the definitions vary in the community [8], a set of RDF triples that signify real-world entities and their relationships based on a predefined ontology can be referred to as a KG. A KG can therefore be interpreted as a graph $G = (V, E)$, where V (the nodes) is the combined set of subjects and objects, and E (the edges) the instances of predicates. Regarding the pathfinding problem, only IRI nodes qualify as entities between which paths can be searched. Accordingly, a path between a pair of entities e and $e' \in V$ is a subgraph of G, where e and e' are IRIs and each node on the path is either another IRI or a blank node. KGs typically provide SPARQL Protocol and RDF Query Language (SPARQL) [9] interfaces for issuing queries.

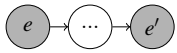
While there exist various domain-specific KGs like the Open Research Knowledge Graph (ORKG) [10] for the scientific domain, Wikidata represents a KG for the general domain. According to its statistics [11], Wikidata currently contains over 100 million entities that possess highly varied in- and outdegrees, where the indegree can exceed the outdegree by magnitudes. For example, issuing two simple SPARQL queries to the Wikidata Query Service[8] reveals that there are over 2,400 triples with the entity Q183 (Germany) as the subject and over 3.1 million with this entity as the object. Even though countries are entities with a particularly high in- and outdegree, there are numerous entities from other categories that have numbers in the thousands. The higher the indegree and outdegree of an entity, the higher the chances of encountering the entity when traversing the KG. As a result, uninformed search algorithms like breadth-first search do not suffice for fast pathfinding, in the worst case even if e and e' are directly adjacent nodes. Another problem is that the Wikidata Query Service already struggles to return mere 100,000 triples. Retrieving all adjacent entities connected to an entity via its incoming and outgoing edges is therefore not always possible.

2.1. An Algorithm for Fast Pathfinding in Wikidata

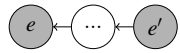
To mitigate the problems described above, the bidirectional A* search algorithm proposed in [2] only considers outgoing edges when traversing the KG and performs two simultaneous searches, one from e to e' and one from e' to e. This way, only paths of the patterns shown in Figure 1 can be discovered by their algorithm. In [2], the authors assume that this limitation is not problematic, at least in the context of Wikidata, due to its dense connectivity. Section 5 will assess this assumption.

Algorithm 1 shows a more verbose version of the strongly condensed algorithm as presented in [2]. The algorithm operates as follows: To keep track of the most promising entities to pursue during graph traversal, a priority queue is employed, which is initialized with e and e'. In each iteration, the first entity is taken from the queue. If a path

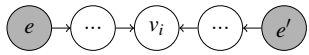
[8] https://query.wikidata.org (accessed 2023/05/26); all SPARQL query results mentioned in the present paper have been retrieved from the Wikidata Query Service on the date of visit.



(a) A direct path from e to e'



(b) A direct path from e' to e



(c) A path composed of a direct path from e to an intersecting entity v_i and a direct path from e' to an intersecting entity v_i

Figure 1. The graph patterns that can be found using the bidirectional A* search algorithm of [2]. e and e' denote the entities of a dual-entity query, between which a path was searched. Nodes with ... are placeholders for series of $n \geq 0$ entities.

Algorithm 1 The bidirectional A* search algorithm from [2]; more verbose and with adapted notation.

procedure FINDPATH(e, e', α, β, γ, *entityLimit*)
  $priorityQueue \leftarrow \langle e, e' \rangle$
  $reachable_{source} \leftarrow \{e\}$
  $reachable_{target} \leftarrow \{e'\}$
  $visitedEntities \leftarrow \{\}$ /
  while $priorityQueue = \emptyset$ **and** $|visitedEntities| < entityLimit$ **do**
    $entity \leftarrow dequeue(priorityQueue)$
    $visitedEntities \leftarrow visitedEntities \cup \{entity\}$
    if $entity \in (reachable_{source} \cap reachable_{target})$ **then**
      return $reconstructPath(e, e'), |visitedEntities|$
    end if
    for $adjacentEntity \in getAdjacentEntities(entity)$ **do**
      $costs \leftarrow calculateCosts(e, e', adjacentEntity, \alpha, \beta, \gamma)$
      $enqueue(priorityQueue, adjacentEntity, costs)$
      if $entity \in reachable_{source}$ **then**
        $reachable_{source} \leftarrow reachable_{source} \cup \{adjacentEntity\}$
      else if $entity \in reachable_{target}$ **then**
        $reachable_{target} \leftarrow reachable_{target} \cup \{adjacentEntity\}$
      end if
    end for
  end while
  return $\perp, |visitedEntities|$
end procedure

between e and e' can be established through the currently visited entity, the algorithm terminates, returning the found path and the number of visited entities. Otherwise, the adjacent entities are retrieved and enqueued with respect to the costs of the paths leading to them. The costs are calculated by means of a cost function with the hyperparameters α, β, and γ. When the priority queue becomes empty or the entity limit, i.e., the maximum number of entities that are allowed to be visited before the search is aborted, is reached without a path being found, the algorithm terminates unsuccessfully.

By providing the measure for ranking the entities in the priority queue, the cost function (*calculateCosts* in Algorithm 1) guides the graph traversal and ultimately determines the algorithm's performance as well as the characteristics of the found paths. What remains to be discussed is therefore what cost function can meet the requirement that users desire meaningful entity relationships in the knowledge panel context, as pointed out in

Section 1. Particularly in Wikidata, numerous paths between two entities can be found, thus posing the question which candidate path is the most meaningful. Since meaningfulness is a highly complex and subjective concept, there are different approaches tackling this problem from different directions. For example, [3] investigated how *informative* subgraphs explaining the relationship between entities can be mined from entity relationship graphs. To this end, the proposed approach ranks candidate nodes according to their informativeness, which is computed using edge weights that are based on co-occurrence statistics for entities and relationships. While this statistical approach yields subgraphs that are structurally important with respect to the query entities, it disregards available semantic information like the entity labels. Thus, the found subgraphs might be prone to concept drift [12], which occurs when the semantic focus of the query is left.

In comparison, the authors of [2] argue that a path with minimal concept drift is a meaningful path. Further, they propose to assess whether a certain entity is out of a query's semantic focus by means of the semantic distances between entities. To calculate the semantic distance they use the cosine distance between the fastText[9] word embeddings [13] of the entities' labels. fastText is a well-known library by Facebook that produces static vector representations for words, while being robust against misspelling. With respect to the general cost function $f(p) = g(p) + h(p)$ of the A* search algorithm [6], [2] introduces a cost function that leverages semantic distances and the path length to calculate the costs of a path p comprising n entities as follows[10] [2]:

$$
\begin{aligned}
&g(p) \coloneqq \alpha \cdot \overline{d}(p_{[..n-1]}, e') + \beta \cdot n \\
&h(p) \coloneqq \gamma \cdot d(v_n, e') \\
&\text{where } p = \langle e, \ldots, v_n \rangle \\
&\text{and } p_{[..f]} \text{ is the sub-path } \langle e, \ldots, v_f \rangle \text{ of } p
\end{aligned}
$$

Representing the first part of formula $g(p)$, the formula $\overline{d}(p_{[..n-1]}, e')$ calculates the average of the semantic distances between all entities on the path except the last and e'. The second part is supposed to add the path length to the costs[11]. The formula $h(p)$ estimates the costs of the remaining path by means of formula $d(v_n, e')$ as the semantic distance between the last entity on the path and e'. As shown, the cost function is parameterized with three hyperparameters α, β, and γ that weight its components. To calculate the costs of a converse path in the bidirectional search, i.e., a path that starts at e' with the goal of finding a path to e, e and e' are simply interchanged in the cost function.

In addition to preferring semantically meaningful paths, the usage of this cost function as a search heuristics serves a second purpose. As explained in [2], the semantic distance of an entity to other entities tends to positively correlate with the minimal number of hops between the entities. For instance, entities that are at least five hops apart typically have a higher semantic distance than entities that are two hops apart. By prioritizing entities with a lower semantic distance to a target entity, the chances of reaching the target entity earlier are therefore higher compared to a breadth-first search.

[9] `https://fasttext.cc` (accessed 2023/05/26)

[10] The cost function was slightly modified to comply with the notation introduced above.

[11] In [2], n was used instead of $n-1$ in the second part of $g(p)$, which is a mistake because the length of a path is typically defined as the number of edges, i.e., one less than the number of nodes on the path [14]. However, this mistake does not compromise the order in the priority queue since it applies to all paths equally.

Table 2. The four configurations for the hyperparameters α, β, and γ used in [2].

Name	α	β	γ	Description
Uninformed	0	1	0	Only considers the path length, similar to breadth-first search.
Semantics-Only	1	0	1	Ignores the path length and only considers the semantic distances.
Greedy	0	0	1	Estimates the total path costs as the semantic distance of the last entity on the path to e/e' depending on the search direction.
Balanced	1	0.5	1	Leverages all components of the cost function in a balanced setting.

In recent years, novel transformer-based approaches like BERT [15] yielded excellent results, thus replacing previous approaches including fastText as the state-of-the-art for various natural language processing tasks. There are numerous variants of BERT tailored to specific domains and tasks. One example is Sentence-BERT or SBERT [16], which is able to efficiently compute accurate vector representations of sentences. For SBERT, various pre-trained models are available, which are also tailored to specific tasks. Section 3 introduces how entity descriptions, which are often one or multiple sentences, can be leveraged in addition to entity labels to improve the accuracy of the semantic distances. One potent model fine-tuned for sentences as well as short paragraphs is `all-mpnet-base-v2`[12], which is based on Microsoft's MPNet [17]. Due to the characteristics of the new input and its general performance, fastText is replaced by SBERT in combination with `all-mpnet-base-v2` for the implementation of the cost function.

2.2. *Hyperparameter Optimization*

In [2], the authors evaluated the performance of the pathfinding algorithm using only the four intuitively set hyperparameter configurations shown in Table 2, which leaves room for improvement. The recent attention on machine learning fueled the investigation of hyperparameter optimization techniques, i.e., methods for automatically setting hyperparameters of objective functions to optimize performance [18]. One example is Bayesian optimization, a state-of the-art optimization framework for the global optimization of expensive blackbox functions that is applicable for a wide range of problems [19]. The Bayesian optimization framework can be broken down into two primary components [19,18]. Firstly, there is a probabilistic surrogate model that incorporates a prior distribution representing the beliefs about the unknown objective function's behavior. Secondly, an acquisition function, that measures the optimality of a series of queries, is utilized. The goal is to minimize the anticipated loss to determine the optimal sequence of queries. Based on the output of each query, the prior is revised, resulting in an informative posterior distribution over the objective function's space. Due to this incremental approach to optimization, it outperforms basic hyperparameter optimization techniques like grid search both in terms of the hyperparameter quality and efficiency.

A related alternative is called Simple(x) or just Simple [20]. While Bayesian optimization uses computationally expensive Gaussian processes to model the objective function, Simple creates a model by dividing the optimization area into simplices. The algorithm iteratively tests points within each simplex to create a more precise model. Thereby, this approach converts the optimization task into a dynamic programming problem, allowing samples to be taken without updating the entire model. Hence, Simple is employed for the optimization of α, β, and γ in Section 3.

[12] `https://huggingface.co/sentence-transformers` (accessed 2023/05/26)

3. Algorithm Improvements and a Query Dataset

Based on the insights from Section 2, this section aims to answer the research questions RQ1 to RQ3. Since RQ4 is related to the evaluation, its discussion follows in Section 5.

3.1. Accurate Semantic Distances between Entities

As described in Section 2, [2] estimates the semantic distance between two entities as the cosine distance between the vector representations of the entity labels. However, the ambiguity of entity labels alone compromises the accuracy of the vector representations and therefore the resulting semantic distances. For example, Wikidata features numerous entities that share the label *Paris*, partly from very different categories: While the entity Q90 refers to the city in France, the entity Q167646 refers to the mythological son of Priam, king of Troy[13]. During pathfinding, it is important to pin down the exact entity that is currently examined. Otherwise, paths with concept drift might be pursued when entities with alleged low semantic distances are prioritized. Therefore, RQ1 asks how the semantic distances between entities can be estimated more accurately.

We propose the two following changes to mitigate this problem. The first change is to feed entity descriptions, another resource that is available in Wikidata for most entities, in addition to entity labels to the word embedding model. For this purpose, the entity labels and entity descriptions are simply concatenated. The idea is that the additional information provided by the descriptions results in vector representations that capture the entities' meaning more accurately. Secondly, fastText is replaced by SBERT in combination with `all-mpnet-base-v2` for efficiently computing high-quality vector representations of the new input type, i.e., strings composed of entity labels and descriptions. Also note that the data within Wikidata is curated. Hence, encountering misspelling is unlikely such that fastText's robustness against them is not required.

The examples in Table 3 show the positive impact of these changes on the semantic distances. In particular, the first example shows that the additional consideration of entity descriptions affirms the semantic distance in cases where the labels alone are expressive enough to compute accurate vector representations. In contrast, the pairs of examples two/three and four/five demonstrate how the entity descriptions help to disambiguate entities with identical or similar labels, yielding more accurate semantic distances.

Further qualitative experiments with other entity pairs conform with these observations. Hence, we conclude that the additional consideration of entity descriptions improves the accuracy of the semantic distances, thereby answering RQ1. That being said, apart from entity descriptions, Wikidata provides even more entity-related information including alternative labels, labels in other languages etc. Furthermore, the (direct) neighborhood of an entity can also be seen as a description of its meaning. To retain the focus of this paper, though, these options are left open for future work as their exploration seems worthwhile to further improve the accuracy of the vector representations.

3.2. A Dual-Entity Query Dataset for Pathfinding in Wikidata

Even though the twelve dual-entity queries discussed in [2] suffice for showcasing the potential of the algorithm, their low number and artificial hand-selected nature does nei-

[13] `https://www.wikidata.org/wiki/{Q90|Q167646}` (accessed 2023/05/26)

Table 3. Comparison of semantic distances using five examples with two Wikidata entities each. d_{labels} and $d_{labels+descs}$ denote whether the presented semantic distances were calculated using SBERT vector representations of the entity labels alone or of the concatenated entity labels and entity descriptions.

Entities	Entity Labels	Entity Descriptions	d_{labels}	$d_{labels+descs}$
Q30	United States of America	country in North America	0.720	**0.793**
Q47488	International Criminal Court	intergovernmental organization and international tribunal		
Q243	Eiffel Tower	tower located on the Champ de Mars in Paris, France	**0.511**	0.499
Q90	Paris	capital and most populous city of France		
Q243	Eiffel Tower	tower located on the Champ de Mars in Paris, France	0.511	**0.758**
Q167646	Paris	mythological son of Priam, king of Troy		
Q6004986	Immigration	album by Show-Ya	0.402	**0.786**
Q841440	naturalization	process by which a non-citizen in a country may acquire citizenship or nationality of that country		
Q131288	immigration	movement of people into another country or region to which they are not native	0.402	**0.451**
Q841440	naturalization	process by which a non-citizen in a country may acquire citizenship or nationality of that country		

ther allow for a proper evaluation nor hyperparameter optimization. Accordingly, RQ2 raises the question what an appropriate dual-entity query dataset for pathfinding in Wikidata is. Considering the long-term goal of applying the algorithm for the knowledge panel generation in web search engines, such a query dataset has to be realistic in the sense that the queries have to be derived from queries issued to web search engines by actual users. However, no such query dataset (or benchmark) has been proposed so far.

In information retrieval research, the TREC [21] datasets are particularly popular and have been used for the evaluation of various information retrieval systems. The datasets are designed to enable researchers to evaluate the performance of their information retrieval systems using a common set of test collections. TREC has produced many different datasets over the years, covering a range of domains and types of text. At first glance, the datasets from the Entity Track[14] of TREC 2009, 2010, and 2011 appear to be useful for deriving a dual-entity query dataset for pathfinding in Wikidata because they include collections of entities for the evaluation of entity-oriented search systems. However, the derivation of a dual-entity query dataset from these datasets would require the artificial pair-wise combination of the single entities within the collections, which clearly contradicts the realism requirement. Furthermore, the TREC datasets only provide ClueWeb09[15] IRIs as identifiers for the entities, which would have to be expensively linked to Wikidata entities first.

Hence, another approach was pursued. The dataset from the Million Query Track[16] of TREC 2007 consists of 10,000 realistic textual queries for web search engines. Using this as a starting point, the following procedure was applied to each query of the TREC dataset to derive a dual-entity query dataset for pathfinding in Wikidata:

[14] https://trec.nist.gov/data/entity.html (accessed 2023/05/26)
[15] https://lemurproject.org/clueweb09/index.php (accessed 2023/05/26)
[16] https://trec.nist.gov/data/million.query07.html (accessed 2023/05/26)

1. GENRE[17] [22], a state-of-the-art Wikidata entity linker, is employed to look for Wikidata entities in the query. If two or more entities are recognized, the next step is taken. Otherwise, the query is dropped.
2. Since GENRE only provides the entity labels and not the necessary entity IDs, Wikidata is queried using SPARQL to retrieve them. To minimize incorrect matches[18], only entities with the exact labels are retained. If two or more entities are identified, the next step is taken. Otherwise, the query is dropped.
3. Finally, $\frac{n(n-1)}{2}$ pairs of entities are composed, where n is the number of recognized and successfully identified entities. The two IDs of each entity pair represent one dual-entity query, which is finally stored.

As an example, consider the query *children books on the effect of music on plants* with number 4480 from the TREC dataset. In the first step, GENRE recognizes three entities within this query, namely entities with the labels *book*, *music*, and *plant*. Querying Wikidata in the second step identifies the entities as Q571, Q638, and Q756, which are reasonable matches[19]. In the final step, $\frac{3(3-1)}{2} = 3$ dual-entity queries are composed, namely ⟨Q571, Q638⟩, ⟨Q571, Q756⟩, and finally ⟨Q638, Q756⟩. Note how the queries correctly reflect the order of occurrence of the entities in the original TREC query. In total, 1,196 dual-entity queries are derived using this procedure[20].

3.3. Optimization of the Hyperparameters α, β, and γ

The dual-entity query dataset not only allows for a proper evaluation of the pathfinding algorithm but also the optimization of its hyperparameters. For the reasons explained in Section 2, the Simple optimizer is leveraged to find an optimized hyperparameter configuration for the pathfinding algorithm, which represents the answer to RQ3. For this purpose, the optimizer requires an objective function that accepts a hyperparameter configuration and returns some objective value to assess the performance of this configuration. Depending on the use case, the optimizer's task is then to either minimize or maximize the objective value by testing different configurations.

Given the web search engine context, the goal is to find paths between entities in Wikidata fast, i.e., with few visited entities, such that the result can be displayed quickly. Hence, the number of visited entities is chosen as the objective value to be minimized. To account for the low-latency requirement of web search engines, the entity limit is set to 100. Furthermore, the pathfinding algorithm must be able to reliably find paths for unseen dual-entity queries because users are allowed to enter arbitrary queries. To evaluate this ability, the optimization operates on only 20% or 239 of the dual-entity queries while the remaining 80% or 957 queries serve as the test set for the evaluation in Section 5. Since the queries within the TREC 2007 Million Query Track dataset are ordered arbitrarily, the first 239 dual-entity queries are sampled for the optimization.

[17] https://github.com/facebookresearch/GENRE (accessed 2023/05/26)

[18] GENRE cannot always link entities correctly since it relies on the context of the entity mentions [22], which is sparse in many TREC queries. As explained before, entity labels alone are also ambiguous.

[19] Instead of Q571, a better match might have been Q8275050 (children's book), which demonstrates that entity linking is still an open problem; cf. https://www.wikidata.org/wiki/{Q571|Q638|Q756|Q8275050} (accessed 2023/05/26).

[20] The derived dual-entity query dataset is available in the provided GitHub repository[7] in the CSV format.

Algorithm 2 The algorithm for optimizing the hyperparameters α, β, and γ.

Require: A list *samples* with 239 dual-entity queries, the pathfinding procedure FINDPATH, the *minimize* function of the Simple optimizer

```
procedure TESTCONF(α, β, γ)
    score ← 0 // lower is better
    entityLimit ← 100
    for ⟨e, e'⟩ ∈ samples do
        path, numberOfVisitedEntities ← FINDPATH(e, e', α, β, γ, entityLimit)
        if path == ⊥ then
            score ← score + numberOfVisitedEntities × 2
        else
            score ← score + numberOfVisitedEntities
        end if
    end for
    objectiveValue ← score / |samples|
    return objectiveValue
end procedure

procedure PERFORMOPTIMIZATION
    interval ← [0.0; 1.0]
    iterations ← 150
    minValue, α, β, γ ← minimize(TESTCONF, interval, iterations)
    return minValue, α, β, γ
end procedure
```

In the literature on Bayesian optimization, the recommendations for the number of iterations vary depending on factors like the number of hyperparameters and the employed acquisition function (cf. [23]). [20] reports that Simple approximates the global optimum of an objective function with two hyperparameters at about 25 iterations. Thus, generous 150 iterations are used for our problem with three hyperparameters. Since the hyperparameters α, β, and γ represent weights of the cost function components, specifying that each of them can assume a value in the interval [0.0;1.0] suffices. With respect to these settings, Algorithm 2 presents the PERFORMOPTIMIZATION procedure for optimizing the hyperparameters α, β, and γ. To this end, the procedure calls the Simple optimizer's *minimize* function to minimize the *objectiveValue* returned by the TESTCONF procedure. Hence representing the objective function, TESTCONF accepts a candidate hyperparameter configuration and returns the average number of entities visited during pathfinding across all queries from *samples* using this configuration as the *objectiveValue*. If no path is found before the entity limit is reached, the number of visited entities is doubled as a penalty[21] for the particular query.

Figure 2 shows the *objectiveValue* yielded in each iteration of the optimization process as well as the *minValue*, i.e., the so far smallest *objectiveValue*. As depicted, a good *minValue* is already found in the 17th iteration and no major improvements are observed until the 150th iteration. Significant improvements beyond 150 iterations are therefore not expected. The lowest *minValue*, i.e., 59.347, was encountered in the 101st iteration using the hyperparameter configuration

$$\alpha = 0.699,\ \beta = 0.109,\ \gamma = 0.823$$

[21] Tests with other penalty factors and high fixed values as a penalty did not yield better results.

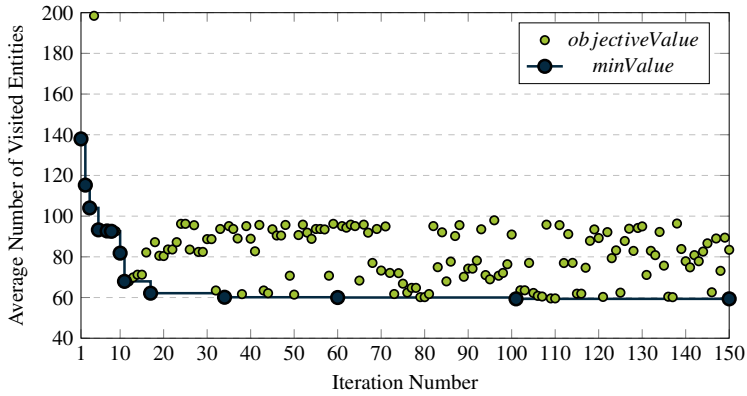


Figure 2. The optimization results.

which represents the optimized hyperparameter configuration and thus the answer to RQ3. Interestingly, β is significantly lower than the other hyperparameters. This supports the assumption that leveraging semantic distances as part of the search heuristics can reduce the number of visited entities. In the course of answering RQ4, Section 5 elaborates on this point.

4. The Implementation: BiPaSs

Another point that needs to be addressed even though it is not directly related to the answering of the research questions is the implementation. Originally, the prototype implementing the pathfinding algorithm was written in Python[22]. For the present paper, a full re-implementation[7] was produced. For future reference, we call the implementation Bidirectional Pathfinding System (BiPaSs). The new implementation uses Rust[23] for the pathfinding algorithm itself and leverages state-of-the-art data structures including a Fibonacci-heap-based priority queue. As shown in Figure 3, the implementation comprises four components. The first one is the *Query Factory* that implements the procedure from Section 3 for deriving the dual-entity query dataset from the TREC dataset. The resulting dual-entity queries are provided to the *Pathfinder* component, which contains the pathfinding algorithm as well as the code for running the hyperparameter optimization and the benchmark. For calculating the semantic distances using SBERT and `all-mpnet-base-v2`, it interacts with the *Wembed API* component via HTTP. For retrieving entity data from Wikidata, it interacts with the *Wikidata API* component, which is a wrapper for issuing SPARQL queries to the Wikidata Query Service, also via HTTP. The *Pathfinder*, the *Wikidata API*, and the *Wembed API* thus constitute the pathfinding system. For ease of use and reproducibility, all components are Docker-ized[24]. Additionally, a significant number of the HTTP interactions are cached, allowing for rapid reproduction of the results despite the restrictive query limits of the Wikidata Query Service, which are responsible for the major portion of the pathfinding duration.

[22] `https://www.python.org` (accessed 2023/05/26)
[23] `https://www.rust-lang.org` (accessed 2023/05/26)
[24] `https://www.docker.com` (accessed 2023/05/26)

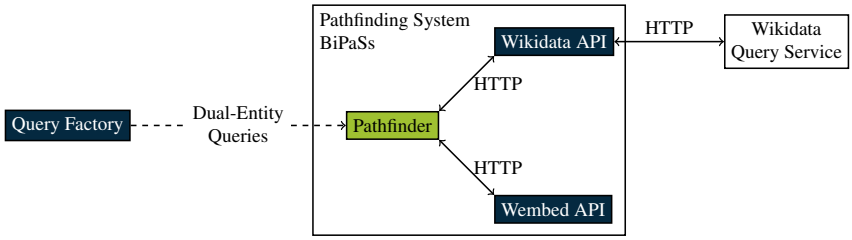


Figure 3. The components of the implementation. Components with a blue background are implemented in Python, the components with a green background in Rust. Arrows indicate communication between components. If the line is dashed, the communication takes place before and not during the actual pathfinding.

5. Evaluation & Discussion

To answer RQ4, the optimized hyperparameter configuration is benchmarked against all hyperparameter configurations introduced in [2] (s. Table 2). This includes the Uninformed configuration with α, β, and γ set to 0, 1, and 0 respectively that represents a baseline as it mimics bidirectional breadth-first search. For the pathfinding itself, the entity limit of 100 introduced in Section 3 is retained. As the test set of queries, the remaining 80% of the dual-entity query dataset, i.e., 957 unseen queries, are employed.

The first quantitative metric to be discussed is the coverage of the test set, i.e., the number of queries, for which a path was found, divided by the total number of queries. As shown in Figure 4, all configurations that consider semantic distances result in a higher coverage than the Uninformed configuration at only 55.6%. Due to the imposed entity limit, this supports the assumption that the usage of semantic distances as part of the search heuristics increases the chances of finding a path with fewer visited entities. However, the numbers of the other configurations show a considerable spread: While the Greedy configuration (59.4%) barely surpasses the Uninformed configuration, the Optimized configuration results in a coverage of 79.2%, the highest coverage achieved. Therefore, the optimization can be considered successful even though the Semantics-Only configuration from [2] actually turned out to be a strong guess as it results in a coverage of 73.6%. Raising the entity limit would increase the coverage of all configurations but also the pathfinding duration, which is problematic in the web search engine context.

Next, Figure 4 also reveals that the Optimized configuration visits 24.7 entities on average to find paths in the successful cases[25]. With 23.4, only the Semantics-Only configuration beats this. The configurations with a higher β, i.e., Balanced and Uninformed, need to visit the most entities to successfully find paths on average (30.0 and 39.5 respectively), which explains their lower coverage. The Optimized configuration also features a β value higher than 0, though. This indicates that β can affect the pathfinding positively up to a certain threshold, beyond which its impact becomes negative.

As the final metric, Figure 4 shows the average length of the paths found using the five hyperparameter configurations, again only considering the successful cases[25]. There are two groups of configurations. The first group comprises the Optimized, the Balanced, and the Uninformed configuration that produce paths with lengths of less than five on

[25]Only the numbers for successfully found paths are considered here due to the imposed entity limit. Otherwise, the entity limit would skew the results because it is not known whether a path would have been found using a certain configuration after visiting, for example, 105 or 1,005 entities.

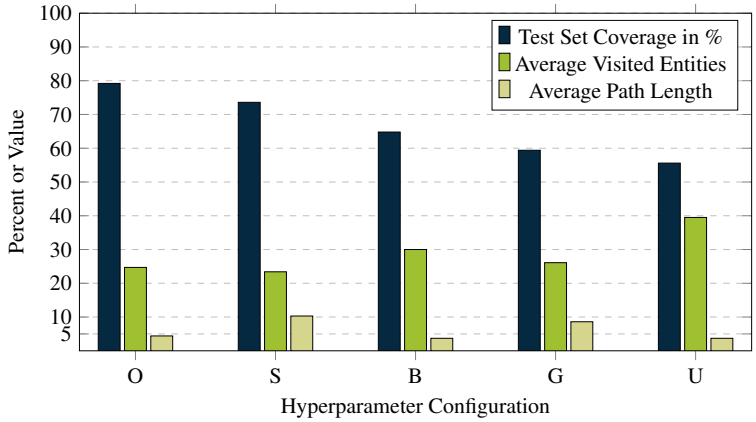


Figure 4. The benchmark results in terms of the coverage of the test set ($n = 957$), the average number of visited entities, and the average path length. The characters on the y-axis denote the examined hyperparameter configurations: **O**ptimized, **S**emantics-Only, **B**alanced, **G**reedy, and **U**ninformed.

average. The second group, i.e., the Semantics-Only and the Greedy configuration, yield average path lengths about twice as high. Note that the members of the former group use a β value of higher than 0, whereas the members of the latter use a β value of exactly 0. This indicates that β plays a key role in controlling the path lengths.

In summary, the answer to RQ4 is as follows: The Uninformed configuration reaches a lower coverage than the configurations that consider semantic distances, thereby supporting their utility as part of the search heuristics. Given the web search engine context, the Optimized configuration represents the best option as it reaches the highest coverage of the test set with a low average of visited entities and a moderate average path length. Its high coverage also supports the assumption by [2] that limiting the algorithm to only consider outgoing edges is unproblematic. At first, the Semantics-Only configuration seems to be a competitive configuration, as well. However, the high average length of the paths found with this configuration raises doubts about their usefulness for users.

To investigate the usefulness of entity relationships, a representative user study has to be conducted in the future. The goal of this study is to deepen the understanding of the meaningfulness of entity relationships in the web search engine context and to assess to what extent the cost function and hyperparameter configurations comply with the users' perceived meaningfulness of entity relationships. Nevertheless, to give an idea of the paths found by the different configurations, Table 4 presents a few examples. Generally, the examples conform with the quantitative results. One interesting observation is that the Semantics-Only path for Query A exhibits concept drift even though the employed configuration fully depends on semantic distances. Also, the Balanced and Uninformed configurations fail to find paths for Query B given the entity limit. The Optimized configuration produces adequately long paths that retain the semantic focus of the queries.

6. Conclusion

With the goal of investigating the open points of [2], the key contributions of the present paper include the improvement of the semantic distances leveraged in the pathfinding

Table 4. Examples of paths found using the five hyperparameter configurations. For conciseness, the predicates have been left out.

Configuration	Path found for Query A ⟨Q7958 (explanation), Q46857 (scientific method)⟩
Optimized	Q7958 (explanation) → Q352842 (teaching) → Q11862829 (academic discipline) ← Q336 (science) ← Q46857 (scientific method)
Semantics-Only	Q7958 (explanation) → Q352842 (teaching) → Q133500 (learning) → Q14819853 (learning or memory) → Q2996394 (biological process) → Q64732777 (biological phenomenon) → Q420 (biology) → Q7205 (paleontology) → Q1069 (geology) → Q7991 (natural science) → Q2522419 (hard science) → Q336 (science) ← Q46857 (scientific method)
Balanced	Q7958 (explanation) → Q352842 (teaching) → Q11862829 (academic discipline) ← Q336 (science) ← Q46857 (scientific method)
Greedy	Q7958 (explanation) → Q352842 (teaching) → Q11862829 (academic discipline) → Q336 (science) → Q46857 (scientific method)
Uninformed	Q7958 (explanation) → Q151885 (concept) → Q5891 (philosophy) ← Q1799072 (method) ← Q46857 (scientific method)
Configuration	**Path found for Query B ⟨Q81938 (pain), Q482853 (vertebral column)⟩**
Optimized	Q81938 (pain) → Q408801 (celecoxib) → Q52849 (ankylosing spondylitis) → Q7577457 (spinal disease) → Q1979420 (human vertebral column) → Q482853 (vertebral column)
Semantics-Only	Q81938 (pain) → Q169872 (symptom) ← Q12136 (disease) ← Q1595418 (remedy) ← Q179661 (treatment) ← Q701216 (pharmacotherapy) ← Q12140 (medication) ← Q11190 (medicine) ← Q514 (anatomy) ← Q515083 (extremities) ← Q62513663 (lower limb) ← Q6027402 (human leg) ← Q23852 (human body) ← Q5170145 (core) ← Q160695 (torso) ← Q133279 (back) ← Q482853 (vertebral column)
Balanced	No path could be found within the entity limit.
Greedy	Q81938 (pain) → Q898407 (venlafaxine) → Q410142 (solute carrier family 6 member 4) → Q14330969 (brain development) → Q1073 (brain) → Q28947902 (cranium) → Q13147 (skull) → Q1377526 (axial skeleton) → Q482853 (vertebral column)
Uninformed	No path could be found within the entity limit.

algorithm's cost function, the introduction of a dual-entity query dataset for pathfinding in Wikidata, the optimization of the algorithm's three hyperparameters, and an evaluation of the algorithm on the said dataset with respect to the examined hyperparameter configurations. The provided re-implementation completes the picture.

Apart from the leads on future work mentioned above, one important point is the integration of the pathfinding algorithm in an end-to-end application where dual-entity queries can be entered, upon which the pathfinder is issued, such that a knowledge panel explaining the relationship between the query entities can be generated and finally be displayed. In this regard, it has to be investigated which kind of presentation yields the best user experience. For instance, an actual text describing the relationship could be generated using natural language generation techniques based on the found paths. At the same time, graph-based visualizations are conceivable, as well.

Instead of adopting the cost function introduced in [2], one could investigate alternative cost functions that also take the meaning of the predicates into account. Intuitively, predicates that express taxonomic relations might be more accessible for non-expert users than more specialized predicates.

Finally, a point raised in [2] should also be repeated here, namely how multi-entity queries that mention more than two entities could be served. While the obvious option is to simply issue a pathfinder between all pairs of entities and concatenate the resulting paths, an approach that tries to identify an entity that connects the query entities with minimal global concept drift is also conceivable, for example.

References

[1] Singhal A. Introducing the Knowledge Graph: Things, not Strings. Official Search Blog. 2012. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html (accessed 2023/05/26).

[2] Martin L, Boockmann JH, Henrich A. Fast Pathfinding in Knowledge Graphs Using Word Embeddings. In: Schmid U, Klügl F, Wolter D, editors. KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI, Bamberg, Germany, September 21-25, 2020, Proceedings. vol. 12325 of Lecture Notes in Computer Science. Springer; 2020. p. 305-12. Available from: https://doi.org/10.1007/978-3-030-58285-2_27.

[3] Kasneci G, Elbassuoni S, Weikum G. MING: mining informative entity relationship subgraphs. In: Cheung DW, Song I, Chu WW, Hu X, Lin J, editors. Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009. ACM; 2009. p. 1653-6. Available from: https://doi.org/10.1145/1645953.1646196.

[4] Reinanda R, Meij E, de Rijke M. Knowledge Graphs: An Information Retrieval Perspective. Found Trends Inf Retr. 2020;14(4):289-444. Available from: https://doi.org/10.1561/1500000063.

[5] Vrandecic D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Commun ACM. 2014;57(10):78-85. Available from: https://doi.org/10.1145/2629489.

[6] Hart PE, Nilsson NJ, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Trans Syst Sci Cybern. 1968;4(2):100-7. Available from: https://doi.org/10.1109/TSSC.1968.300136.

[7] Cyganiak R, Hyland-Wood D, Lanthaler M; W3C. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation. 2014. https://www.w3.org/TR/rdf11-concepts (accessed 2023/05/26).

[8] Ehrlinger L, Wöß W. Towards a Definition of Knowledge Graphs. In: Martin M, Cuquet M, Folmer E, editors. Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), September 12-15, 2016. vol. 1695 of CEUR Workshop Proceedings. Leipzig, Germany: CEUR-WS.org; 2016. Available from: http://ceur-ws.org/Vol-1695/paper4.pdf.

[9] Harris S, Seaborne A; W3C. SPARQL 1.1 Query Language. W3C Recommendation. 2013. https://www.w3.org/TR/sparql11-query (accessed 2023/05/26).

[10] Jaradeh MY, Oelen A, Farfar KE, Prinz M, D'Souza J, Kismihók G, et al. Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In: Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, November 19-21, 2019. Marina Del Rey, CA, USA: ACM; 2019. p. 243-6. Available from: https://doi.org/10.1145/3360901.3364435.

[11] Wikidata; Wikidata. Statistics. Wikidata. 2023. https://www.wikidata.org/wiki/Special:Statistics (accessed 2023/05/26).

[12] Dietz L, Kotov A, Meij E. Utilizing Knowledge Graphs for Text-Centric Information Retrieval. In: Collins-Thompson K, Mei Q, Davison BD, Liu Y, Yilmaz E, editors. 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR 2018). ACM; 2018. p. 1387-90. Available from: https://doi.org/10.1145/3209978.3210187.

[13] Mikolov T, Grave E, Bojanowski P, Puhrsch C, Joulin A. Advances in Pre-Training Distributed Word Representations. In: Calzolari N, Choukri K, Cieri C, Declerck T, Goggi S, Hasida K, et al., editors. Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA); 2018. p. 52-5. Available from: http://www.lrec-conf.org/proceedings/lrec2018/summaries/721.html.

[14] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, 3rd Edition. MIT Press; 2009. Available from: http://mitpress.mit.edu/books/introduction-algorithms.

[15] Devlin J, Chang M, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 4171-86. Available from: https://doi.org/10.18653/v1/n19-1423.

[16] Reimers N, Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Inui K, Jiang J, Ng V, Wan X, editors. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. Association for Computational Linguistics; 2019. p. 3980-90. Available from: https://doi.org/10.18653/v1/D19-1410.

[17] Song K, Tan X, Qin T, Lu J, Liu T. MPNet: Masked and Permuted Pre-training for Language Understanding. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H, editors. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual; 2020. p. 16857-67. Available from: https://proceedings.neurips.cc/paper/2020/hash/c3a690be93aa602ee2dc0ccab5b7b67e-Abstract.html.

[18] Feurer M, Hutter F. Hyperparameter Optimization. In: Hutter F, Kotthoff L, Vanschoren J, editors. Automated Machine Learning - Methods, Systems, Challenges. The Springer Series on Challenges in Machine Learning. Springer; 2019. p. 3-33. Available from: https://doi.org/10.1007/978-3-030-05318-5_1.

[19] Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N. Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proc IEEE. 2016;104(1):148-75. Available from: https://doi.org/10.1109/JPROC.2015.2494218.

[20] Stroeml C; Microsoft. Simple: Simple(x) Global Optimization. GitHub. 2018. https://github.com/chrisstroemel/Simple (accessed 2023/05/26).

[21] TREC; National Institute of Standards & Technology. Text REtrieval Conference (TREC) Home Page. TREC. 2018. https://trec.nist.gov/ (accessed 2023/05/26).

[22] Cao ND, Izacard G, Riedel S, Petroni F. Autoregressive Entity Retrieval. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net; 2021. Available from: https://openreview.net/forum?id=5k8F6UU39V.

[23] Snoek J, Larochelle H, Adams RP. Practical Bayesian Optimization of Machine Learning Algorithms. In: Bartlett PL, Pereira FCN, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States; 2012. p. 2960-8. Available from: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.