

Documents as Intelligent Agents: An Approach to Optimize Document Representations in Semantic Search

Oliver Strauß^a and Holger Kett^b

Fraunhofer Institute for Industrial Engineering IAO, Nobelstraße 12, 70569 Stuttgart, Germany

Keywords: Dataset Search, Agent-Based Retrieval, Semantic Search.

Abstract: Finding good representations for documents in the context of semantic search is a relevant problem with applications in domains like medicine, research or data search. In this paper we propose to represent each document in a search index by a number of different contextual embeddings. We define and evaluate eight different strategies to combine embeddings of document title, document passages and relevant user queries by means of linear combinations, averaging, and clustering. In addition we apply an agent-based approach to search whereby each data item is modeled as an agent that tries to optimize its metadata and presentation over time by incorporating information received via the users' interactions with the search system. We validate the document representation strategies and the agent-based approach in the context of a medical information retrieval dataset and find that a linear combination of the title embedding, mean passage embedding and the mean over the clustered embeddings of relevant queries offers the best trade-off between search-performance and index size. We further find, that incorporating embeddings of relevant user queries can significantly improve the performance of representation strategies based on semantic embeddings. The agent-based system performs slightly better than the other representation strategies but comes with a larger index size.


1 INTRODUCTION


Since the advent of deep learning in natural language processing a lot of progress has been made in the field of semantic search (Onal et al., 2018). Transformer models such as BERT (Devlin et al., 2018) or Sentence-BERT (Reimers and Gurevych, 2019) convert words and sentences into dense vectors while capturing their semantic context. Semantic search is then performed by embedding the query in the same way as the documents and finding the document vectors closest to the query vector in vector space. Transformer models however are limited to processing a fixed number of input tokens (Reimers and Gurevych, 2019). Although the number of tokens is constantly increasing, long documents still can not be converted to an embedding vector in one pass. One common strategy to overcome this limitation is to split long documents into passages and encode each passage separately. One can then either average over the resulting passage vectors to obtain one vector per document or index multiple vectors for each document.

In this paper we construct different document representation strategies by combining contextual semantic embeddings of a document's title and its passages into new embeddings that are subsequently added to the search index. We also combine these document-based embeddings with embedded user queries in order to adapt the documents' representations to better match the information demands of the users.

Deep semantic has been successfully applied in various domains, ranging from medicine (Lee et al., 2020) and research (Chu et al., 2023) to data search (Castro Fernandez et al., 2018; Ahmadi et al., 2022). This last aspect becomes more and more important as finding, understanding and using data is an important challenge in data ecosystems (S. Oliveira et al., 2019). A user searches data to perform a specific task (Chapman et al., 2020) and must balance the effort needed to search, evaluate and integrate external datasets with the potential benefits that using the data can potentially provide. From a data publishers perspective, the problem is to prepare, describe and publish datasets in a way that produces the most revenue with the least amount of work (Hemphill et al., 2022).

In this context we propose the idea of intelligent agents that strive to optimize the representation of

^a  <https://orcid.org/0000-0003-1421-2744>

^b  <https://orcid.org/0000-0002-2361-9733>

their document in a search index. The agents independently select different document representation strategies based solely on local information about their position in user searches and relevance feedback provided by the users. In our view a local, agent-based approach to the optimization of document representations is relevant, because approaches in which data can remain at the premises of the data provider are becoming increasingly important in data ecosystems (Nagel and Lycklama, 2021). The approach has potential applications in search infrastructures such as (distributed) data catalogues, search engines or data marketplaces to enable data users to better find, understand and evaluate relevant datasets. A (potentially distributed) agent-based system is also consistent with the concepts explored in the International Data Spaces (Nagel and Lycklama, 2021) and Gaia X¹ initiatives.

In our evaluation we investigate **(RQ1)** *what the most effective strategies to represent documents by a set of sentence embeddings are*, **(RQ2)** *whether combining document embeddings with embeddings of previously issued relevant queries can improve search performance*, and **(RQ3)** *whether an agent-based incremental optimization of the document representation based on local information yields satisfactory results*.

The contribution of this paper is **(1)** the exploration of different document representation strategies that combine contextual semantic embeddings of a document’s title, passages and user queries in various ways. Specifically, we look at linear combinations of embedding vectors, building averages over vectors and apply agglomerate clustering to query vectors before averaging them in order to capture different user intents. **(2)** We investigate the effect of combining embeddings of previously issued queries with document embeddings. **(3)** We model documents as intelligent agents, elaborating on and extending ideas presented in (Strauß et al., 2022). The evaluation is performed using the NFCorpus dataset (Boteva et al., 2016), because it provides sufficient relevance feedback for each document to drive the agent-based simulation.

2 RELATED WORK

Semantic Search. Over the last years a lot of progress has been made in the field of deep learning in natural language processing. Models like Word2Vec (Mikolov et al., 2013) or GloVe (Jeffrey Pennington et al., 2014) can encode words into vectors and per-

form similarity search by looking for nearest neighbors in the vector space. Transformer models such as BERT (Devlin et al., 2018) have been trained on huge corpora of text and use cross attention to learn to take the context of words into account. This makes them suitable for tasks ranging from semantic search to question answering or text classification. Sentence transformer models (Reimers and Gurevych, 2019) can average over the tokens of a sentence and provide a means to encode longer texts into contextual embeddings. Instead of training a customized neural model, we employ a general-purpose model to encode documents into contextual semantic embeddings.

Document Expansion. There are various approaches that try to improve query performance by expand documents with additional information. docT5query uses customized transformer models to generate synthetic queries (Nogueira et al., 2019) while (Jeong et al., 2021) use neural models to generate paraphrased text to be indexed alongside the original document. Our approach focuses on the actual title and passages and tries to combine them in an optimal way without the generation of additional text.

Agent-Based Systems. Agents have been investigated as autonomous (distributed) mechanism in software development for a few decades. They can be seen as socially intelligent, autonomous problem solvers which achieve their objectives by interaction with other similar autonomous entities (Hogg and Jennings, 2001). Agents have been used for self-organised, local optimisation in a broad spectrum of applications such as autonomous network optimisation (Fabrikant et al., 2003) or distributed energy optimisation (Hinrichs and Sonnenschein, 2017).

The term agent is ambiguously used not only for agents optimizing local objectives, but also for software components with different functionalities. For example, (Xue and Yan, 2012) use nine modules (agents) for a search system using intelligent evolution based on user queries to improve accuracy of the results. In the same sense (Mahmud et al., 2016) proposes a agent-based meta search engine for open government data. (Ciorcea et al., 2019) examine the use of multi-agent systems in the context of the World-Wide-Web. (Strauß et al., 2022) propose an agent-based system for document expansion, where each document is modeled as an agent, that tries to extend its documents with terms from relevant queries. We follow their view and consider agents as similar entities that act together in a search context to optimize their own findability but also the performance of the whole system.

¹<https://gaia-x.eu/>

3 APPROACH

3.1 Document Representation and Semantic Search

Following the approach described in (Strauß et al., 2022) we represent each document in the search index with one or more representation variants. Each variant represents the document in a different way and can be retrieved individually. Before the results are presented to the user, the variants need to be removed from the search result by only keeping the top-ranked variant of each document. In this way an agent can offer multiple representations of its document and receives feedback on the success of the respective variants.

We use semantic search via contextual sentence embeddings based the sentence transformers library (Reimers and Gurevych, 2019) as search strategy. A Transformer model such as BERT (Devlin et al., 2018) is a neural network that uses a cross attention mechanism to learn to encode tokens into vectors while taking account of their context in the text. Sentence transformers use similar tokens to encode entire sentences into a contextualized vector embedding. Using the pre-trained transformer model `all-mpnet-base-v2`² we encode the document title and passages in to embedding vectors and combine them with the strategies detailed in Section 3.2 into variants that represent the document in the search engine. Passages are built by splitting a document's content into sentences.

The search itself is performed by encoding the query in the same way and calculate the cosine similarity between the query vector and the vectors of all variants. The top_k variants with the highest similarity score are returned as result.

3.2 Document Representation Strategies

To describe the different strategies of combining the sentence embeddings of document titles, passages and queries in a compact form, the notation in Eqns. (1) - (4) are introduced. The mean over a set of embedding vectors is given by Eqn. (1). Linear combinations of two or three embedding vectors are given by l_1 in Eqn. (2) and l_2 in Eqn. (3). *Set* in Eqn. (4) is used as an abbreviation for a set of elements.

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i \quad (1)$$

$$l_1(\mathbf{a}, \mathbf{b}, \alpha) = \frac{\mathbf{a} + \alpha \cdot \mathbf{b}}{1 + \alpha} \quad (2)$$

$$l_2(\mathbf{a}, \mathbf{b}, \mathbf{c}, \alpha, \beta) = \frac{\mathbf{a} + \alpha \cdot \mathbf{b} + \beta \cdot \mathbf{c}}{1 + \alpha + \beta} \quad (3)$$

$$Set_i^N(x_i) = \{x_1, \dots, x_N\} \quad (4)$$

With the title embedding vector \mathbf{t} and the N_p embedding vectors of the document passages \mathbf{p}_i , the N_q embedding vectors of the relevant queries \mathbf{q}_i , and embeddings of relevant queries \mathbf{c}_i^k belonging to cluster C^k (with number of clusters N_c) we can construct a number of different document representations for each document. The clusters C^k are created from the embeddings of the relevant queries known at the time by agglomerate clustering using the scikit-learn library (Pedregosa et al., 2011).

The combinations defined in Eqns. (5) - (12) were investigated.

$$R_1 = \{\mathbf{t}\} \quad (5)$$

$$R_2 = \{\bar{\mathbf{p}}\} \quad (6)$$

$$R_3 = Set_i^{N_p}(\mathbf{p}_i) \quad (7)$$

$$R_4(\alpha) = \{l_1(\mathbf{t}, \bar{\mathbf{p}}, \alpha)\} \quad (8)$$

$$R_5(\alpha) = Set_i^{N_p}(l_1(\mathbf{t}, \mathbf{p}_i, \alpha)) \quad (9)$$

$$R_6(\alpha, \beta) = \begin{cases} \{l_2(\mathbf{t}, \bar{\mathbf{p}}, \bar{\mathbf{q}}, \alpha, \beta)\} & \text{if } N_q > 0 \\ R_4(\alpha) & \text{otherwise} \end{cases} \quad (10)$$

$$R_7(\alpha, \beta) = \begin{cases} Set_i^{N_q}(l_2(\mathbf{t}, \bar{\mathbf{p}}, \mathbf{q}_i, \alpha, \beta)) & \text{if } N_q > 0 \\ R_4(\alpha) & \text{otherwise} \end{cases} \quad (11)$$

$$R_8(\alpha, \beta) = \begin{cases} Set_k^{N_c}(l_2(\mathbf{t}, \bar{\mathbf{p}}, \bar{\mathbf{c}}^k, \alpha, \beta)) & \text{if } N_q \geq 2 \\ R_4(\alpha) & \text{otherwise} \end{cases} \quad (12)$$

R_1 uses the embedding of the document title to represent the document. R_2 uses the mean of the document's passage embeddings while R_4 uses this mean in a linear combination with the title embedding. In R_3 all passage embeddings represent the document while these passage embeddings are combined in R_5 with the embedding of the title in a linear combination. R_6 is the linear combination of the title embedding, mean passage embedding and the mean over the embeddings of all known relevant queries for the document. R_7 combines the embedding of each relevant query with the title embedding and the mean passage

²<https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

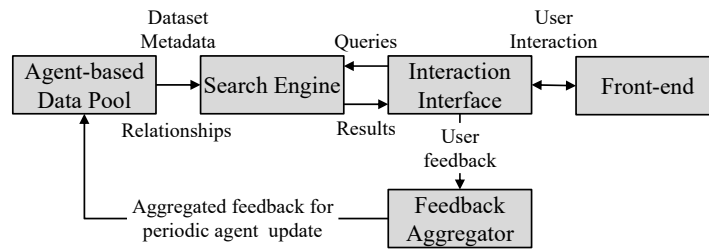


Figure 1: Architecture of the agent-based system. A feedback loop connects agents and the search engine that provides agents with information about when they were found and whether their document was considered relevant. Search engines periodically receive updated document representation from the agents.

embedding in a linear combination while R_8 uses the mean embedding of each cluster of queries instead.

Since R_6 - R_8 rely on information about queries which might not always be available, they fall back to the R_4 strategy, which is similar but does not require knowledge of relevant queries.

3.3 Documents as Intelligent Agents

In this paper we propose to represent each document by an intelligent agent, that tries to promote its document in a way that optimizes its discoverability. The intelligent agents act locally by adjusting their own presentation to the needs of the users. For this to work the agents are embedded in a system that provides a feedback loop with users, allowing to measure the success of the agents' behavior. The agents in our experiment use the representation strategies R_1 - R_8 to periodically generate new representation variants, measure the variants success based on relevance feedback and subsequently filter out unsuccessful representations.

An intelligent agent is an autonomous component that possesses internal state, interacts with other agents and its environment, makes autonomous decisions based on its state and environment and can learn and adapt to a changing environment (Jackson et al., 2017). The internal state of a dataset agent is mainly given by the document content and eventual metadata. The environment consists of a search engine that uses the metadata provided by the agents to answer user queries, and a component that captures, aggregates and transmits user feedback to agents. Agents act locally based on their state and local knowledge obtained from their environment (see Subsection 3.3.3).

Figure 1 gives an overview of the general architecture of the agent-based simulation. Documents are managed by agents that together form an agent-based data pool. The data pool is comprised of the agents and the infrastructure needed to register and update the agents. Each agent publishes its representation to a search engine that indexes the provided information

for later retrieval. Users interact with the system using a (web based) front-end that communicates with the search engine via an interaction interface component. This interaction interface manages the users' actions and the communication with the search engine. In doing so it captures relevant user actions, search queries and search results. This user feedback is an essential input for the document agents that allows them to adapt to users' needs and terminology. The user feedback is sent to the feedback aggregator component, that collects and aggregates the user feedback over a period of time and sends the aggregated information to the data pool to update the agents.

In our experiment actual user feedback is approximated by the relevance judgments that come with the used information retrieval test collection (see Section 3.3.2). Therefore the components that capture real user feedback (namely the interaction interface, front-end, and feedback aggregator in Figure 1) are not needed and are replaced by the provided relevance information described in Section 4.1.

3.3.1 Agent Environment

Each document agent operates in an environment that the agent can use to gather information for its decisions and with which the agent can interact. In the proposed approach the agents' environment consists of the following elements:

- **Search Engine** — The agent publishes its representation to the search engine that incorporates the representations into the search index. With their actions, agents influence, what information is submitted to the search engine.
- **User Feedback** — The user interacts with the search engine in order to find data that meets her information need. The interaction interface collects information from the users' actions during her search session that indicates to the agents what is being searched and which documents are considered relevant for the user's data need. As stated above, user feedback is simulated in our experi-

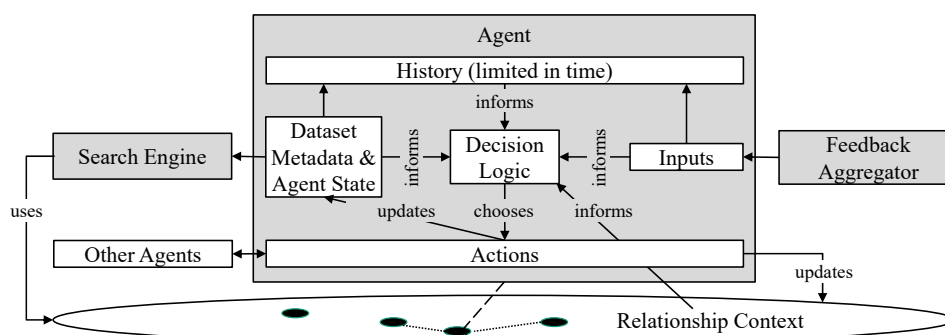


Figure 2: Internal structure of a proposed datasets agent.

ment by the relevance feedback provided by the used test collection (see Section 3.3.2).

3.3.2 User Interaction and Feedback

User feedback is an essential external input for dataset agents to learn about their relevance and success in respect to data needs and about the terminology used by different users. For example, if an agent's dataset has been returned in the result list of a query, the agent is informed about the query text q , its rank r in the list, its search engine score.

We use an information retrieval test collection to emulate user feedback. This enables reproducible results and removes the need for expensive user experiments. The topics/queries and relevance judgments provided by the test collection (see Section 4.1) replace the front-end and interaction interface components in Figure 1.

3.3.3 Agent Actions

For making sound decisions, each agent keeps track of the history of the actions it has taken and the user feedback it has received. It can analyse and exploit this information and derive its actions from this data. Since the environment and the user's needs may change over time the agents "forget" historical information after a certain amount of time (Figure 2). Although not implemented in our simulation document agents could interact and establish relationships with other agents in order to support co-operating behaviour. This is symbolized by the relationship context in Figure 2.

The agents are optimising their visibility for fitting search queries from a local perspective in a selfish way. In such a setup, rules or restrictions are required to ensure meaningful decisions. Letting agents try different strategies in parallel in the form of representation variants as in (Strauß et al., 2022) offers a pragmatic approach to evaluate an agent's performance. Instead of credit contingents or explicit re-

ward functions, the reward is measured by the success of the agents' variants and the agent logic (Section 3.3.4) needs to decide, whether a variant should be kept, deleted or whether new variants should be created. New variants are created using the representation strategies R_1 - R_8 described in Section 3.2.

The agents are triggered periodically after the system has run for a certain time. The user feedback collected during that time is aggregated and transmitted back to the agents. Then the agents' decision cycle is triggered during which the agents determine their actions and change their representations. Finally the agents send their representation variants to the search engine. The updates of the search engine index can be collected and performed as batch job. After the update has been performed the systems collects information for the next period.

3.3.4 Agent Logic

The structure of the implemented agent is shown in Figure 3. The feedback collector takes the search results from the search engine and produces feedback signals that are sent to the agents. For each variant in a search result list that is considered relevant by the relevance judgments a positive signal is created. All other variants receive a negative signal. Each signal contains the following information: the text of the query q , the rank in the result list r , and the search engine score.

Whenever an agent receives input signals, it performs the following steps:

1. It increases its update counter t by one. t plays the role of local time for the agent.
2. It separately stores the positive and negative signals.
3. It updates the variants belonging to the signals with the information in the signal. Each variant maintains the time t_c of its creation as well as counters N_p and N_n for the number of positive or

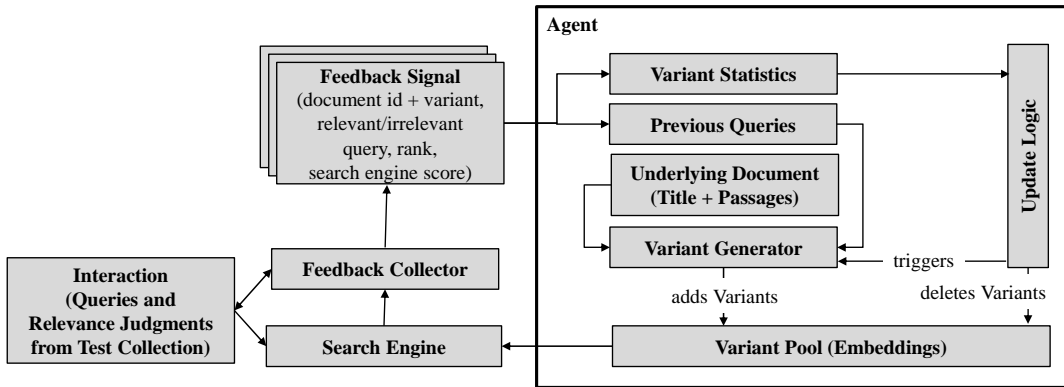


Figure 3: Internal structure of an agent in the implemented agent-based system.

negative signals it has received. The variant also tracks the ranks r_i^p and r_i^n .

4. If a positive signal was received, an update of the variants is triggered.

For variant update, the agent computes a score $s(t)$ for each variant. Following (Strauß et al., 2022) the score is calculated as the product of the mean reciprocal rank $1/N \sum_{i=0}^N 1/r_i$ and the success rate $N/(t - t_c)$ of the variant. In deviation from (Strauß et al., 2022) we also take negative feedback into account by taking the difference of the positive and negative mean reciprocal rank. This results in Eqn. (13):

$$s(t) = \frac{1}{t - t_c} \left(\sum_{i=1}^{N_p} 1/r_i^p - \sum_{i=1}^{N_n} 1/r_i^n \right) \quad (13)$$

After the scores have been calculated the variant update is performed in the following steps:

1. Variants with an age $t - t_c$ less than a grace period t_g are kept in the variants pool and not considered in the further process
2. Variants are sorted by score using the scoring function given by Eqn. (13). The N_{min} best variants are retained, and all other variants are deleted.
3. At least N_{new} variants are created using the representation strategies described in Section 3.2.

Each agent maintains a set of representation strategies in a circular list that is randomly shuffled on agent creation. Whenever new variants need to be created, the next strategy is chosen from the list and the list pointer is advanced by one. These rules allow the pool of representation variants to dynamically grow and shrink. Expansion of the pool amounts to exploring the space of possible representations while retaining at least the best N_{min} variants after the shrinking based on the variants' fitness exploits performance of the best variants.

3.3.5 Agent Update Cycle

The concrete realization of the feedback loop depicted in Figure 1 is realized in our simulations by the following procedure:

1. Initialize the agents with an initial set of at least N_{min} variants that are created using the configured representation strategies as described in Section 3.3.4.
2. Initialize the search index with the embeddings contained in all variants of all agents. Each agent is potentially represented by multiple embeddings in the index.
3. Shuffle all queries in the training set and create batches of queries.
4. Execute all queries in the current batch. Care must taken that at least $top_k = 100$ unique documents are retrieved, since potentially many variants representing the same document are in the results list.
5. Collect feedback signals.
6. Aggregate feedback signals for each agent and send it to the agent.
7. The agents update their state and update their presentation by creating new variants or deleting unsuccessful old ones according to the procedure described in Section 3.3.4.
8. A new search index is created from the embeddings of all variants.
9. If there are more batches, go back to step 4.) else continue.

After all training queries have been processed, the index of the search engine contains the representations optimized by the agents during the training run. This index is used to perform the evaluation and the calculation of performance metrics. Since each document is represented by multiple variants, only the

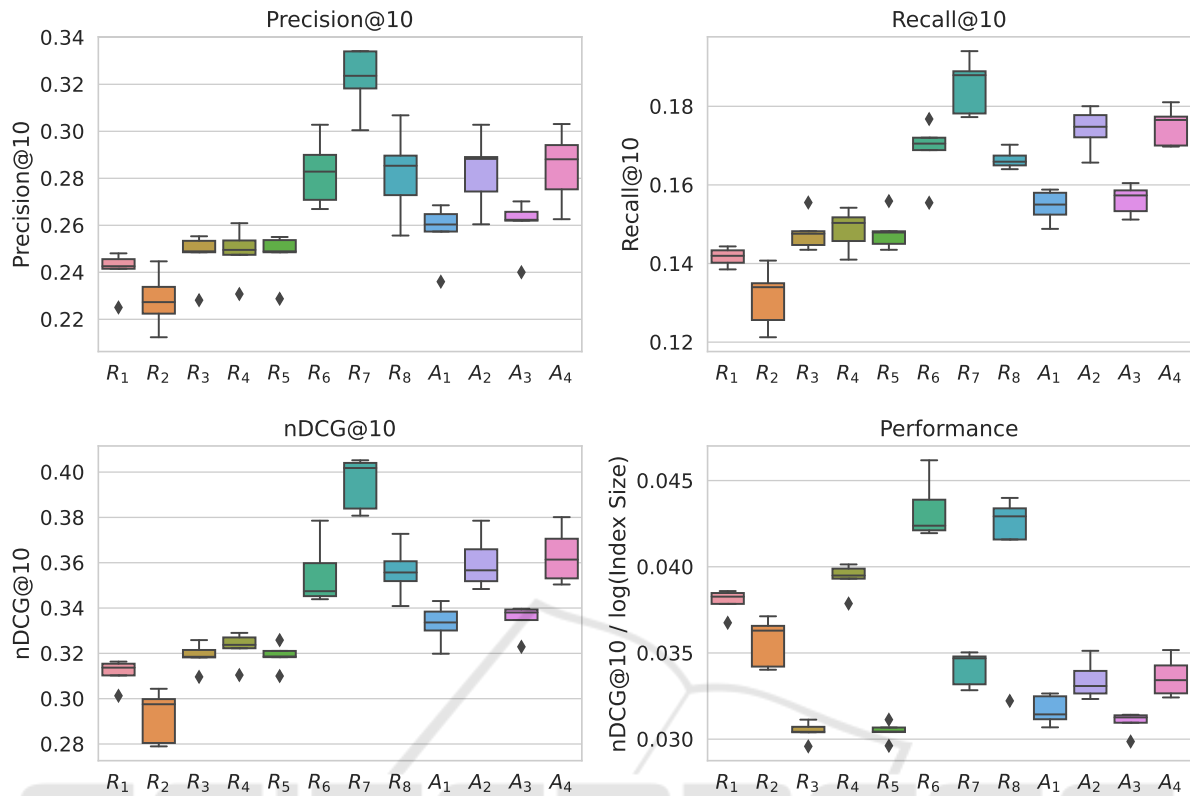


Figure 4: Box plots of the results of the cross evaluation runs for the different experiments: Precision@10 (top left), Recall@10 (top right), nDCG@10 (bottom left), and $P = nDCG@10 / \log(IndexSize)$ as an indicator of weighted performance.

top-ranked variant for a document is considered while lower-ranked variants are discarded.

4 EVALUATION

4.1 Test Collection

We chose the NFCorpus test collection (Boteva et al., 2016) as basis for the evaluation of the agent-based system, because it contains a sufficiently large amount of relevance feedback. We use the BEIR variant of this dataset (Nandan Thakur et al., 2021), since it comes with duplicate documents removed. This dataset consists of 3,633 unique documents from the medical domain that mostly come from PubMed and are written in expert terminology. Each document consists of a title and an abstract. The collection also provides 3,237 natural language queries that are written in non-technical English and have been obtained from the NutritionFacts.org website. The collection contains three types of queries that were extracted from different sources: 1,016 video queries were taken from titles of video pages, 413 non-topic

queries were extracted from the titles of pages not tagged with a topic, and 1,808 queries issued by users. An example of a document and the different query types is presented in Table 1.

The documents and queries are complemented by 134,294 automatically extracted relevance judgments. (Boteva et al., 2016) use hyperlinks between pages of the NutritionFacts website and medical articles to automatically construct relevance information. They consider a document d relevant for query document q , if q contains a direct external hyperlink to d (strong relevance) or if q links to a second similar query q' that in turn links to d (weak relevance). In our experiment we use binary relevance judgments and consider both cases as relevant. Documents not linked in the described way to queries are considered not relevant.

Because of the high number of relevance judgments, each document is considered on average relevant for over 30 queries. Since these relevant queries are driving the feedback loop of the agent-based simulation NFCorpus is a good fit for this problem. A split in training (80%), development (10%) and test sets (10%) is provided. In order to make better use of the provided data, we instead employ a cross validation strategy with stratified splits described in Section 4.2.

Table 1: Examples of a typical document and the different query types.

Document	Example
Title	Dietary Cadmium Intake and the Risk of Cancer: A Meta-Analysis
Abstract	Background Diet is a major source of cadmium intake among the non-smoking general population. Recent studies have determined that cadmium exposure may produce adverse health effects at lower exposure levels than previously predicted. We conducted a meta-analysis to combine and analyze the results of previous studies that have investigated the association of dietary cadmium intake and cancer risk. Methods We searched PubMed, EMBASE, and MEDLINE database for case-control and cohort studies that assessed the association of dietary cadmium intake and cancer risk.
Query	Example
User	liver health
Video	Eating Better to Look Better
Non-topic	What is a good source of probiotics?

4.2 Evaluation Strategy

In order to make good use of the available test data, we employ a nested cross validation strategy to measure the performance of the document representation strategies and the agent-based simulations. Since the test data can contain different types of queries, we employ stratified splits in order to retain the same mixture of the different query types in all data slices.

1. $N_{outer} = 5$ outer splits are created. Each contains a larger set of training queries and a smaller set of test queries.
2. For parameter optimization, the training queries are again split into $N_{inner} = 4$ splits.
3. All N_{inner} inner splits are evaluated and the fitness used for optimization is computed as the mean of the N_{inner} runs.
4. After a set of optimal parameters has been determined, the test queries are used to perform the final evaluation run and to calculate the performance metrics.

4.3 Metrics

We use the retrieval metrics precision, recall and the ranking metrics normalized discounted cumulative gain (nDCG) and calculate these metrics for the top 10 hits in each search (thus e. g. Precision@10 in Table 2). nDCG is well suited to measure realistic

user behaviour (Fuhr, 2018) and is therefore preferred over mean average precision (MAP) and mean reciprocal rank (MRR). We also record the index size I resulting from each run. Since the results of the N_{outer} cross evaluation runs were unevenly distributed and contained outliers, we report median scores instead of mean values.

Since representing each document in the search index with multiple variants increases the index size I which leads to higher computational cost for each search, there is a trade-off between the potentially increased search performance through additional variants and the computational cost of these additional variants. In order to capture this trade-off, we introduce a performance indicator P such that $P = nDCG@10/\log I$.

4.4 Representation Strategies

In a first step we evaluated the different document representation strategies defined in Section 3.2 outside the agent-based simulation in a setting, where all knowledge about queries and relevance judgments from the training set was used to collect the relevant queries for each document. We applied the cross validation approach described in Section 4.2. In every run each strategy produced the corresponding embedding representations for each document using input from the training queries. The created embeddings were aggregated in the search index of the semantic search engine. The test queries were then submitted to the search engine and only the top-ranked variant for each document were kept in the result list. Finally, the relevance judgments were used to create the metrics to evaluate the retrieval performance of the run.

In the optimization step the parameters α and β of the strategies R_4 - R_8 were optimized to maximize the nDCG@10 metric using the RBFOpt library (Costa and Nannicini, 2018; Nannicini, 2021). The results are shown in Figure 4 and in Table 2 in rows R_1 - R_8 . Table 5 shows the parameter values of the evaluation runs.

4.5 Agent-Based Experiments

Four agent-based configurations A_1 - A_4 were tested that use different sets of representation strategies. All four configurations use the strategies R_1 - R_4 that solely rely on the document title and passages. In addition, A_1 uses R_5 , A_2 uses R_6 , A_3 uses R_7 , and A_4 uses R_8 . Optimization of the parameters t_g , N_{min} , and N_{new} in addition to the α -, β - and γ -parameters of the representation strategies was performed using the RBFOpt library. The agent-based configuration were evaluated

Table 2: Overall effectiveness of the eight document representation strategies (R_1 - R_8) and the four agent-based simulations (A_1 - A_4 , used strategies are indicated in brackets). In all runs `all-mpnet-base-v2` was used as embedding model. Plain numbers indicate the median over all five cross evaluation splits. Numbers in parenthesis indicate the standard deviation. For each run the final size of the index is given. The best results are highlighted in boldface. The nDCG@10 baseline methods marked with (*) were taken from Table 2 in (Nandan Thakur et al., 2021).

Exp.	nDCG@10	Precision@10	Recall@10	Performance P	Index Size I
BM25*	0.325	-	-	-	-
docT5query*	0.328	-	-	-	-
BM25-CE*	0.350	-	-	-	-
R_1	0.314 (0.006)	0.243 (0.009)	0.142 (0.002)	0.038	3633 (0)
R_2	0.298 (0.012)	0.227 (0.012)	0.134 (0.008)	0.036	3633 (0)
R_3	0.318 (0.006)	0.249 (0.011)	0.148 (0.005)	0.030	35130 (0)
R_4	0.324 (0.007)	0.250 (0.011)	0.150 (0.005)	0.039	3633 (0)
R_5	0.319 (0.006)	0.249 (0.011)	0.148 (0.005)	0.030	35130 (0)
R_6	0.347 (0.015)	0.283 (0.015)	0.171 (0.008)	0.042	3633 (0)
R_7	0.402 (0.012)	0.324 (0.014)	0.188 (0.007)	0.035	107492 (2028)
R_8	0.356 (0.012)	0.285 (0.019)	0.166 (0.002)	0.043	24036 (45622)
A_1 [R_1 - R_4 , R_5]	0.334 (0.009)	0.260 (0.013)	0.155 (0.004)	0.031	36948 (4601)
A_2 [R_1 - R_4 , R_6]	0.357 (0.012)	0.288 (0.016)	0.175 (0.006)	0.033	47908 (93)
A_3 [R_1 - R_4 , R_7]	0.338 (0.007)	0.262 (0.012)	0.157 (0.004)	0.031	49659 (2)
A_4 [R_1 - R_4 , R_8]	0.361 (0.012)	0.288 (0.016)	0.177 (0.005)	0.033	49540 (102)

using cross validation (Section 4.2) and the procedure described in Section 3.3.5. The results are shown in Figure 4 and in Table 2 in rows A_1 - A_4 . Table 4 shows the parameter values of the evaluation runs.

All experiments were performed on a NVIDIA A100-SXM4-40GB GPU.

5 DISCUSSION

Concerning our research questions defined in Section 1 we can draw the following conclusions from our experiments.

(RQ1) *What are the most effective strategies to represent documents by a set of sentence embeddings?* A useful reference and benchmark for the NFCorpus test collection is provided as part of the BEIR benchmark (Nandan Thakur et al., 2021). It reports a nDCG@10 baseline performance of 0.325 for the lexical search method BM25 (Robertson and Zaragoza, 2009) and a best score of 0.350 for the BM25+CE re-ranking method (Wang et al., 2020). The results of our experiments summarized in Table 2 are not directly comparable with these numbers, since the evaluation strategy was different. They still provide a weak indication of the performance of the tested methods. When comparing strategies R_1 - R_5 that do not incorporate knowledge from user queries, R_4 which combines the title embedding with the mean if the passage embeddings performs best in terms of the median nDCG@10 (0.324) and P

(0.039) measures. Of the strategies that incorporate user queries the representation strategy R_8 , which is the linear combination of the title embedding, mean passage embedding and the mean over clustered relevant queries achieves the best P -performance (0.043) with a median nDCG@10 score of 0.356. The best median nDCG@10-performance of 0.402 is achieved by strategy R_7 which combines title embeddings and average passage embeddings with all relevant user queries. This leads to a P value of only 0.035. We conclude that the strategy R_8 that uses agglomerate clustering over user queries offers the best trade-off between index size I and nDCG@10 performance followed by R_6 . This is also visible in Figure 5.

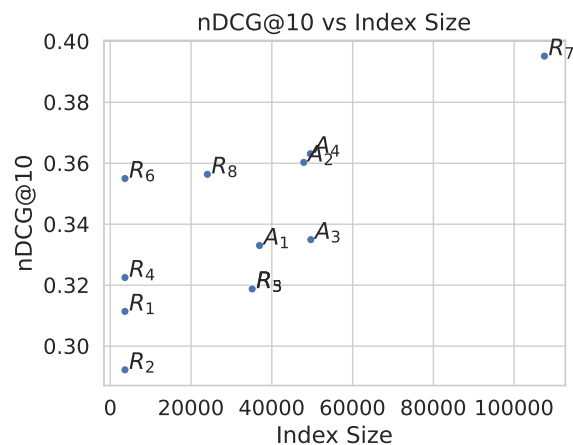


Figure 5: Trade-off between search performance (nDCG@10) and index size I .

Table 3: Comparison of the effectiveness of the conducted experiments. The comparison is based on the first split of the cross validation process, therefore results differ from Table 2, that shows the median over all five cross validation splits. The best results are highlighted in boldface. Superscripts denote significant differences in paired Student’s t-test with $p \leq 0.01$.

#	Model	NDCG@10	P@10	Recall@10
a	R_1	0.315	0.243 ^b	0.143
b	R_2	0.300	0.227	0.135
c	R_3	0.318 ^b	0.249 ^b	0.143
d	R_4	0.329 ^{ab}	0.250 ^b	0.150 ^b
e	R_5	0.319 ^b	0.249 ^b	0.143
f	R_6	0.344 ^{abce}	0.271 ^{abcde}	0.169 ^{abcde}
g	R_7	0.404 ^{abcdefhijkl}	0.324 ^{abcdefhijkl}	0.194 ^{abcdefhijkl}
h	R_8	0.352 ^{abcde}	0.273 ^{abcdei}	0.164 ^{abcde}
i	A_1	0.338 ^{abce}	0.260 ^{abcde}	0.155 ^{bce}
j	A_2	0.352 ^{abcde}	0.274 ^{abcdeik}	0.172 ^{abcdeik}
k	A_3	0.338 ^{abce}	0.262 ^{abcde}	0.151 ^b
l	A_4	0.353 ^{abcdeik}	0.275 ^{abcdeik}	0.170 ^{abcdeik}

(RQ2) *How strong is the improvement in performance if combining document embeddings with are combined with embeddings of previously issued relevant queries?* While strategies R_1 - R_5 use no information from user feedback, the other approaches do make use of additional information from user feedback in form of relevant and irrelevant queries. We observe, that unsurprisingly the strategies R_6 - R_8 which use information from relevant queries significantly outperform the other strategies. In order to test this observation, we compared the runs of the first cross validation split of each method and performed a paired Student’s t-test with $p \leq 0.01$ (Table 3). We find that R_6 - R_8 perform significantly better than R_1 - R_5 .

(RQ3) *Does an agent-based incremental optimization of the document representation based on local information yield satisfactory results?* The results of A_2 and A_4 outperform R_6 and R_8 on which they are based in terms of nDCG@10 but have a significantly lower P -value because they create a larger index. If this is not an issue, then the performance of A_2 and A_4 is acceptable. We suspect that the potential of the agent-based approach is not fully realized in the presented experiments since co-operative aspects such as information sharing between similar agents has not been investigated in our experiments.

It has to be noted that the representation strategies R_3 , R_5 , R_7 and R_8 produce multiple variants for each document and thus trade a bigger index size for potentially higher retrieval performance. This also applies to the agent-based simulations. Figure 5 visualizes this trade-off. It can be observed that the strategies R_4 , R_6 and R_8 offer the best compromise between index size I and nDCG@10-performance.

Another observation is, that there is a rather big variation in some parameters as can be seen from Tables 4 and 5.

6 CONCLUSION AND FUTURE WORK

In this paper we propose to represent each document in the search index by several different contextual embeddings. We define and evaluate eight different strategies to combine embeddings of document title, document passages and relevant user queries by means of linear combinations, averaging, and clustering. In addition, we apply an agent-based approach to search whereby each data item is modeled as an agent that tries to optimize its metadata and presentation over time by incorporating information received via the users’ interaction with the search system. The agents only act on local information available to them. This allows them to operate in a distributed environment, such as data catalogues in data spaces.

We validate the document representation strategies and the agent-based approach in the context of medical information retrieval via semantic search based on contextual sentence embeddings and similarity search. We evaluate eight different representation strategies and test them in the context of the agent-based system. We find that strategy R_8 which is the linear combination of the title embedding, mean passage embedding and the mean over the clustered embeddings of all known relevant queries offers the best trade-off between nDCG@10-performance and index size. We further find, that incorporating embeddings of relevant user queries can significantly

Table 4: Median parameters for the experiments A_1 - A_4 (standard deviation included in parenthesis) determined by parameter optimization during cross validation.

Exp.	N_{Min}	N_{New}	t_g	α (R_4)	α	β	γ
A_1	8.0 (0.4)	1.0 (3.5)	0.0 (2.2)	3687.0 (851.6)	8072.5 (929.7)	-	-
A_2	2.0 (0.0)	28.0 (0.0)	6.0 (0.0)	2611.3 (0.0)	3280.6 (0.0)	7411.1 (0.0)	-
A_3	29.0 (0.0)	23.0 (0.0)	27.0 (0.0)	6200.9 (0.0)	9531.2 (0.0)	2818.0 (0.0)	-
A_4	24.0 (2.9)	2.0 (8.6)	0.0 (8.9)	4802.4 (1963.3)	3567.4 (1058.0)	8370.3 (1406.8)	65.8 (44.8)

Table 5: Median parameters for the experiments R_4 - R_8 (standard deviation included in parenthesis) determined by parameter optimization in the five cross validation splits.

Exp.	α	β	γ
R_5	250.1 (340.9)	-	-
R_6	513.8 (62.8)	917.0 (52.1)	-
R_7	995.7 (1.3)	489.7 (55.5)	-
R_8	6.9 (444.0)	5.9 (232.8)	994.7 (446.2)

improve the performance of representation strategies based on semantic embeddings. The agent-based system performs slightly better than the other representation strategies in terms of nDCG@10 (nDCG@10 = 0.361 for A_4) but come with a larger index size.

The performed experiments indicate, that the agent-based approach to search can yield promising results. In the future other aspects of the proposed approach such as inter-agent relationships, co-operation or a more sophisticated agent logic based on reinforcement learning could be explored. The availability of test data with a large number of queries and the associated relevance judgments is a prerequisite and constraining factor for these experiments, since this information is needed to drive the feedback loop of the agent-based system. Producing these test collections is an additional challenge to be addressed by future work.

ACKNOWLEDGEMENTS

This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Incentives and Economics of Data Sharing Funding Action (IEDS0001). The author is responsible for the content of this publication.

REFERENCES

Ahmadi, N., Sand, H., and Papotti, P. (2022). Unsupervised Matching of Data and Text. *arXiv*.

Boteva, V., Gholipour, D., Sokolov, A., and Riezler, S. (2016). A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In Ferro, N., Crestani,

F., Moens, M.-F., Mothe, J., Silvestri, F., Di Nunzio, G. M., Hauff, C., and Silvello, G., editors, *Advances in Information Retrieval*, volume 9626 of *Lecture Notes in Computer Science*, pages 716–722. Springer International Publishing, Cham.

Castro Fernandez, R., Mansour, E., Qahtan, A. A., Elmagarmid, A., Ilyas, I., Madden, S., Ouzzani, M., Stonebraker, M., and Tang, N. (2018). Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In *IEEE 34th International Conference on Data Engineering*, pages 989–1000, Piscataway, NJ. IEEE.

Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.-D., Kacprzak, E., and Groth, P. (2020). Dataset search: a survey. *The VLDB Journal*, 29(1):251–272.

Chu, X., Liu, J., Wang, J., Wang, X., Wang, Y., Wang, M., and Gu, X. (2023). CSDR-BERT: a pre-trained scientific dataset match model for Chinese Scientific Dataset Retrieval.

Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., and Zimmermann, A. (2019). A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web. In IFAAMAS, editor, *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*.

Costa, A. and Nannicini, G. (2018). RBFOPt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*.

Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C. H., and Shenker, S. (2003). On a Network Creation Game. *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, pages 347–351.

Fuhr, N. (2018). Some Common Mistakes In IR Evaluation, And How They Can Be Avoided. *ACM SIGIR Forum*, 51(3):32–41.

Hemphill, L., Pienta, A., Lafia, S., Akmon, D., and Bleckley, D. A. (2022). How do properties of data, their curation, and their funding relate to reuse? *Journal of the Association for Information Science and Technology*.

Hinrichs, C. and Sonnenschein, M. (2017). A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents. *International Journal of Bio-Inspired Computation*, 10(2):69–78.

- Hogg, L. M. J. and Jennings, N. R. (2001). Socially Intelligent Reasoning for Autonomous Agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31(5):381–393.
- Jackson, J. C., Rand, D., Lewis, K., Norton, M. I., and Gray, K. (2017). Agent-Based Modeling. *Social Psychological and Personality Science*, 8(4):387–395.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014). GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Jeong, S., Baek, J., Park, C., and Park, J. C. (2021). Un-supervised document expansion for information retrieval with stochastic text generation.
- Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics (Oxford, England)*, 36(4):1234–1240.
- Mahmud, S. M. H., Rabbi, M. F., and Guy-Fernand, K. N. (2016). An Agent-based Meta-Search Engine Architecture for Open Government Datasets Search. *Communications on Applied Electronics*, 4(7):21–25.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations*.
- Nagel, L. and Lycklama, D. (2021). Design Principles for Data Spaces: Position Paper.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych (2021). BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Nannicini, G. (2021). On the implementation of a global optimization method for mixed-variable problems. *Open Journal of Mathematical Optimization*, 2:1–25.
- Nogueira, R., Lin, J., and Epistemic, A. (2019). From doc2query to docttttquery. *Online preprint*, 6:2.
- Onal, K. D., Zhang, Y., Altingovde, I. S., Rahman, M. M., Karagoz, P., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., McDonnell, T., Nguyen, A. T., Xu, D., Wallace, B. C., de Rijke, M., and Lease, M. (2018). Neural information retrieval: at the end of the early years. *Information Retrieval*, 21(2-3):111–182.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Robertson, S. and Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- S. Oliveira, M. I., Barros Lima, G. d. F., and Farias Lóscio, B. (2019). Investigations into Data Ecosystems: a systematic mapping study. *Knowledge and information systems*, pages 1–42.
- Strauß, O., Kutzias, D., and Kett, H. (2022). Agent-Based Document Expansion for Information Retrieval Based on Topic Modeling of Local Information. In *2022 9th International Conference on Soft Computing & Machine Intelligence (ISCMCI)*, pages 198–202. IEEE.
- Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *arXiv*.
- Xue, B. and Yan, G.-l. (2012). Research on multi-agents information retrieval system based on intelligent evolution. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology (ICCSNT 2012)*, pages 1042–1045, Piscataway, NJ. IEEE.