De-DSI
Decentralised Differentiable Search Index

Neague, Petru; Gregoriadis, Marcel; Pouwelse, Johan

DOI
10.1145/3642970.3655837

Publication date
2024

Document Version
Final published version

Published in
EuroMLSys 2024 - Proceedings of the 2024 4th Workshop on Machine Learning and Systems

De-DSI: Decentralised Differentiable Search Index

Petru Neague
Delft University of Technology
Delft, The Netherlands
p.m.neague@tudelft.nl

Marcel Gregoriadis
Delft University of Technology
Delft, The Netherlands
m.gregoriadis@tudelft.nl

Johan Pouwelse
Delft University of Technology
Delft, The Netherlands
j.a.pouwelse@tudelft.nl

Abstract

This study introduces De-DSI, a novel framework that fuses large language models (LLMs) with genuine decentralization for information retrieval, particularly employing the differentiable search index (DSI) concept in a decentralized setting. Focused on efficiently connecting novel user queries with document identifiers without direct document access, De-DSI operates solely on query-docid pairs. To enhance scalability, an ensemble of DSI models is introduced, where the dataset is partitioned into smaller shards for individual model training. This approach not only maintains accuracy by reducing the number of data each model needs to handle but also facilitates scalability by aggregating outcomes from multiple models. This aggregation uses a beam search to identify top docids and applies a softmax function for score normalization, selecting documents with the highest scores for retrieval. The decentralized implementation demonstrates that retrieval success is comparable to centralized methods, with the added benefit of the possibility of distributing computational complexity across the network. This setup also allows for the retrieval of multimedia items through magnet links, eliminating the need for platforms or intermediaries.

CCS Concepts: • **Information systems → Language models**; • **Computer systems organization → Peer-to-peer architectures**.

Keywords: Information Retrieval, Large Language Models (LLMs), Distributed Systems

ACM Reference Format:
Petru Neague, Marcel Gregoriadis, and Johan Pouwelse. 2024. De-DSI: Decentralised Differentiable Search Index. In *4th Workshop on Machine Learning and Systems (EuroMLSys '24), April 22, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3642970.3655837

EuroMLSys '24, April 22, 2024, Athens, Greece

ACM ISBN 979-8-4007-0541-0/24/04
https://doi.org/10.1145/3642970.3655837

1 Introduction

The proliferation of smart devices is raising concerns over privacy due to extensive data collection. This data is valuable for enhancing machine learning (ML) models but raises a significant risk of personal surveillance and loss of privacy. While the issue concerns many types of data (e.g. geolocation, personal conversations, health data, etc.), this paper attempts to take a step towards the maintenance of privacy in the field of information retrieval.

Google's federated learning, introduced in 2016, addresses privacy concerns by allowing devices to contribute to ML model improvement without sharing local data. They only exchange model updates with one or more central servers [15].

Gossip learning, a subset of algorithms of what later came to be called *decentralised federated learning*, has been proposed in 2011 to resolve the same challenge as federated learning [20, 21]. Gossip learning is fully decentralised and does not require a central server. Participants communicate directly, exchange model updates, and aggregate said updates. The great advantage of gossip learning is the lack of any infrastructure, making it both robust and easier to scale. Decentralised federated learning is now becoming a mainstream topic as studied by [14]. However, to date, Bitcoin and BitTorrent are the only examples of full decentralisation with actual broad usage. Decentralised federated learning still remains constrained to the lab [14]. Originally, decentralisation guided the development of the Internet. One of the Internet's defining principles is its lack of any single point of technical, political, or economic control [19].

Transformers and generative AI tuned for search are changing the field of information retrieval. AI-based alternatives now exist for search engine strategies such as keyword matching, BM25, IDF heuristics, and relevance ranking [33]. Traditional IR systems separate the steps of indexing, retrieval, and (re)ranking. One problem with the classical paradigm is that it is difficult to optimize various components. The various modules operate mostly separately, potentially producing suboptimal results for the architecture as a whole [44].

The leap in AI research has provoked the emergence of retrieval systems with generative models that do not rely on an explicit index anymore [17, 43, 4]. Instead, the knowledge of the documents is encoded in the parameters of a pre-trained model. The idea is that those models will be able to 'understand' queries and documents, rather than just remember and match, and as a result, be better at retrieval.

Moreover, the model can compress more information, allowing generative retrievers to occupy significantly less space than traditional retrieval methods.

The first wave of this research has explored the direct extraction of knowledge, where the model generates the answer to a question after training on a corpus of documents [25, 24, 32]. Recently a new line of work has come up that investigates the generation of document identifier strings directly from a model [36]. This novel search architecture is called *Differentiable Search Index (DSI)*. DSI uses a single Transformer model to perform both indexing and retrieval. It is co-invented by three Big Tech companies. Meta published the initial sketches for entity retrieval in 2020 [6]; Google introduced generic information retrieval in early 2022 [36]; Microsoft released *DSI-QG* (DSI with query generation) improvements in late 2022 [45].

We present De-DSI, the first successful fusion of two powerful, yet largely unexplored fields within machine learning. De-DSI combines *Decentralised Federated Learning* (DFL) with *Differentiable Search Index (DSI)*. The contributions of this work are the insights that search could be powered by LLMs trained in a decentralised manner, and that those could be used to build a decentralised public search engine. Due to our academically-pure decentralisation, this experimental search engine is owned and controlled by no one entity. We trained the Google T5 model to output document IDs in response to queries in a decentralised environment. By using sharding, we craft an ensemble of models which allows the indexing of up to 10 times more data than a single model, at the cost of accuracy.

2 Problem Description

Developing a decentralised search engine has proven to be difficult. The main problems are the huge amounts of information to index, rampant online fraud, and high expectations of users (near-perfect, near-instantaneous results). *Differentiable Search Index (DSI)* represents a promising paradigm for information retrieval tasks using Transformers [36]. Assessing the viability of DSI within real systems with real users and enormous amounts of data is still unsolved. DSI might prove to be similar to the DHT: elegant and ineffective. It is an emerging paradigm for information retrieval, yet it still lacks decentralisation, scalability, security, privacy, practical validation, and user trails. Numerous scientists have investigated algorithms for *distributed* information retrieval [37]. However, transforming these ideas into sustainable solutions without any single point of technical, political, or economic control remains unsolved. The demise of numerous open search engine projects contains important learnings on the challenges for long-enduring sustainable solutions.

A simple query flooding approach across a peer-to-peer overlay network was first used by Gnutella in 2001 [31]. This popular system slowly collapsed and showed the need for effective search, free-riding prevention, and anti-spam measures. The YaCY search engine used a DHT to store the reverse word index in 2003. This DHT resulted in unsolved security issues such as spamming, poisoning, and sybil attacks [38]. Ultimately, it proved to be less scalable than then believed. This is caused by the mechanism to implicitly address churn and re-announce *every* document daily [29]. The leading search engine for the IPFS distributed content sharing system was shut down in 2023, after seven years of operation [9]. Basic features such as relevance ranking for random files shared on IPFS proved to be difficult to realise in a fully decentralised manner. Their central website, expensive 100-node cluster, and algorithmic improvements proved to be unsustainable without continuous grant money [7].

Table 1. Example of queries for a document ID. The query is the input into the DSI model and the docid is the response. The docid is constructed by the model token-by-token

Query	Docid
aarp spider solitaire free game	D3125778
spider solitaire free game	D3125778
spidersolitairefree	D3125778
free spider solitaire card game	D3125778
solitaire spider free	D3125778
free solitaire spider games	D3125778

3 De-DSI: Architecture and Design

Our proof-of-concept for De-DSI is simplistic yet capable of offering effective search. The cardinal design principles of our design are simplicity, scalability, feasibility, and deployability. We have access to real-world search workloads due to prior decentralised systems research which received several million user installs [26, 39, 28, 10]. Our De-DSI design and experiments are devised to realistically reflect our production environment. We aim to deploy and iteratively improve De-DSI for the coming years. Specifically, we plan to use it to enhance our open-source, decentralised, YouTube-like video search engine, which boasts 2.4 million unique installs as of February 2024 [10]. Our implementation of De-DSI, along with the experimental setups, is available online as open source.[1]

3.1 Differentiable Search Index (DSI)

Our De-DSI design is inspired by the original DSI work from 2022 [36]. In the original DSI, a single T5-based Transformer is trained to directly map queries to document identifiers (docid), in a sequence-to-sequence fashion. To this end, they trained the model on data from the *Natural Questions (NQ)* dataset [13]. Specifically, their model was trained in two phases: initial training to associate document content with

[1] https://github.com/pneague/De-DSI

a docid, and *fine-tuning* to associate each question with a matching docid. De-DSI simplifies this approach by only requiring user queries.

The *structured* approach within the original DSI generates the docid token by token. This method boasts enhanced scalability compared to the use of unstructured atomic identifiers (where each document is associated to exactly one token), as it effectively narrows down the search space with each step. Semantically structured identifiers, where each token choice in a docid is inherently meaningful, are the most advanced form of docids for the retrieval task.

The focus of our work is the retrieval of files in decentralised networks. Such systems notoriously lack good-quality metadata (often only being given the file name). This is the reason why we chose to train the model to associate queries with the docid. It is also the reason why we use the *naively structured* identifiers to represent our documents (i.e. the docids are represented by a sequence of randomly assigned characters with no inherent meaning). We show a few samples of input-output pairs belonging to one document in Table 1. The DSI method allows for retrieval of multiple ranked docids through beam search. In our investigations, we find that beam search sometimes results in hallucinations. However, most of the time, its outputs yield reasonable responses. We adopt this method when investigating the metric of top-k accuracy. That is, we count it as a success if the expected docid is within the top-k docids retrieved by the model.

3.2 Ensemble DSI

The effectiveness of DSI is inherently tied to the model's size, as there is a finite limit to the amount of information a model can fit. Since the T5-small model has fewer weights compared to models representing the state-of-the-art, scalability is bound to become a problem in systems spanning hundreds of millions of documents. Our approach of distributing many T5-small instances in a peer-to-peer (P2P) network improves the scalability of DSI.

To this end, we propose *sharding* as a means to divide the document space, and moreover divide the information load on individual models. Specifically, we propose splitting peers into groups. Each group is then responsible for only one partition (one *shard*) of the data. This means they fine-tune their T5 models on all query-docid mappings that are associated with a specific subset of the existing documents. The size of the group merely promotes robustness, since peers within the group achieve convergence by training on the same dataset. Yet, the peers only ever become aware of a subset of documents, and are oblivious to others. In order to process unseen queries, therefore, peers must consult every other group of peers, for the chance that their shard contains the relevant documents.

Taking into consideration the suggestions from all peer groups means that some might be valid and relevant (as they have seen similar queries), and some will not. Hence arises the challenge of differentiating between these results. To this end, we make use of the logit scores attached with each produced output. Those scores can be thought of as the model's confidence about the respective output. More specifically, we let every peer group return the five most likely outputs given the query, using beam search. In the case that the model has seen similar queries in its training phase, it will be very confident about the document associated with said queries, and much less confident about other documents it knows. Meanwhile, models which have not seen similar queries are prone to produce a random set of docids they have learnt of, each with roughly the same probability. However, as has been pointed out by Zhou et al. [43], these models learn on different scales, and so the scores are therefore not directly comparable.

To solve this problem, we propose normalizing the scores using softmax. Let $D = \{(d_i, s_i) \mid i = 1 \leq i \leq 5\}$ be the five result candidates a model generated given some query. In this set, d_i denotes a generated docid, and s_i represents its logit score. By taking the softmax of the five scores, we can normalize the confidence scores of the models' suggestions:

$$D_{\text{softmax}} = \{(d_i, \text{softmax}(s_i)) \mid 1 \leq i \leq 5\} \tag{1}$$

Softmax allows the comparison of results of different models as it scales each of them from zero to one, so model scale variability is nullified. Thus, all scores from all models are appended to the same list, and the top-k ones with the highest scores are selected, representing the results of the model for the top-k accuracy metric. We illustrate this process, dubbed as Confidence-Ensemble, in Figure 1. Finally, by considering only the suggestions with high confidence scores, we can filter out suggestions from models that have not been trained on related queries.

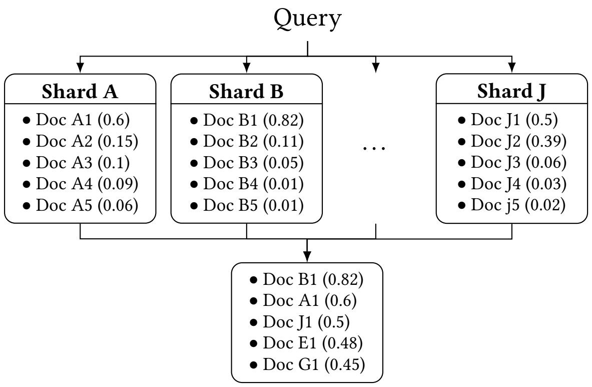

Figure 1. Confidence-Ensemble using 10 shards (i.e. 10 peer groups). The scores under each shard are post-softmax. The result of the ensemble is the top5 documents with largest post-softmax score.

4 Performance Evaluation and Experiments

As the base datasets to our experiments, we use the *Open Resource for the Curation of Answer-Snippets (ORCAS)* [5]. ORCAS is a comprehensive collection designed to support research in information retrieval, specifically for the development and evaluation of search engines. It contains 1.4 million documents and 20 million query-document pairs, obtained from Bing search histories over the course of a few months. Its data are anonymized and only pertain to English-speaking users within the United States. Despite its relatively low coverage of the entire Internet, it still represents one of the best datasets for analyzing search in the English-speaking world. A few data are shown in Table 1.

Throughout this paper the model used was the pre-trained version of the T5-small model (about 60 million parameters) for training/inference. The loss function used in all experiments was the cross-entropy function. We adopt top-k as our main metric of comparison, as explained above. This is because in this dataset we do not have relevance rankings, but only the binary correct/incorrect answer. From our experience, training a T5 model on a dataset of 1000 docids and 40 queries per docid (40 000 data points), takes about 20 hours on a Macbook with M2 Pro, and about 2 hours on an NVIDIA A4000 GPU.

In the following, we are going to evaluate our previously elaborated ideas in four successive experiments. First, we are assessing, and furthermore proving, the capability of the T5-small Transformer model to retrieve documents based on unseen queries. This evaluation occurs after the model has been trained exclusively on mappings between queries and docids, without ever being exposed to the content of the documents themselves. It is important to mention that this finding stands for web pages similar to that of the ORCAS dataset. In the second experiment, in an effort to scale up the search engine, we are employing an ensemble of DSI models. Only in our third experiment, we are adding more realistic conditions by simulating a P2P system, and we decentralise the training process. In our fourth and final experiment, we show that the T5-small can also accurately retrieve entire magnet links at low additional cost in terms of accuracy. This is meant to give support to the idea that the model can act as a real search engine.

4.1 Content-Oblivious Search

We report on a property of LLMs that has never been reported on before. Our first experiment shows emergence of an effective search engine merely by *query feeding*. The goal of this experiment is to see how many queries are needed to describe a document, such that the model can successfully generalize to new (unseen) queries. To this end, we fine-tune our model on a sample of documents and their associated queries. With larger samples of documents, there's a higher chance that document may contain similar information and thus be harder to distinguish from one another based solely on queries. This generally makes it harder for the model to correctly predict the docid given a query, and must always be taken into consideration when assessing the results. For that reason, we conduct multiple experiments with sample sizes $N = \{100, 500, 1000\}$. Furthermore, we conduct our experiments in a range of $n = 1..20$ queries per document. Naturally, we expect a better performance the more queries are fed in the model.

To start our experiments, we first sample N documents, each with 60 associated unique queries (20/20/20 for training, validation, and testing). This dataset of query-docid pairs is used as the base throughout the experiments on $n = 1..20$. In every set, the number of queries is equally distributed over all documents. In each iteration of n, we get the 'first' n queries related with each docid, from the train set. That is, the queries in $n = 1$ are also guaranteed to be part of $n = 2$, and so on. A new T5-small model is now fine-tuned on the relationship of every query-docid pair in this subset. The number of epochs is controlled by a strategy of early stopping, where the training continues until the accuracy on the validation set has not improved by at least 0.01 in the last 20 epochs. Afterwards, the state of the model with the highest accuracy is taken further to the testing phase. This is generally done to control for overfitting. Finally, we evaluate the accuracy of the fine-tuned model on the test set. For every docid that the model correctly matches to a query in the test set, we increment a score counter. The score divided by the sum of queries in the test set, reflects the accuracy, or in other words, the success rate.

The comprehensive results are depicted in Figure 2. Generally, we can observe that only very few training queries are needed to answer new queries with remarkable success. For instance, even with $N = 1000$ documents, **only two queries per document** were needed to answer unseen queries with an accuracy of above 50 %. After training on 20 queries per document, the success rate has risen to an impressive 86 %, and to 94 % for $N = 100$. All experiments experience a steep logarithmic rise after just the first few queries, and later seem to stagnate with higher numbers of queries fed. This is probably due to the queries being very similar to one another, so the success rate shrinks with higher values for N.

4.2 Ensemble-DSI for Scalable Search

As we have seen in the previous experiment, the efficacy of the search engine is subject to the number of documents in the output space. The purpose of this experiment is to increase the total number of documents in a way that scales. To this end, we are employing an ensemble of 10 models, where each is trained on distinct subsets of the data, called *shards* (i.e. the documents along with all their associated queries). For each new query, results are solicited from all models, and subsequently aggregated according to the ensemble's design.

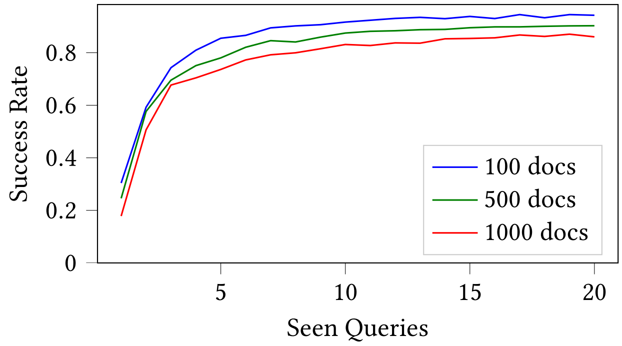

Figure 2. Success rate matching unseen queries to the correct document, based on a number of seen queries trained on.

The results of this experiment are shown in Figure 3. The error bars showcase the means and variances calculated over the accuracies of 10 shards, for top-k accuracy with $k = 1..5$. The color of the boxplots represents the method, and the y-axis shows the accuracy. That is, when asking queries from the test set of any shard, we distinguish between two methods:

- **Ensemble:** Here, the results represent the mean and standard deviation of the accuracy, achieved over all 10 shards.
- **Personal:** This refers to inference from a single model from within the respective shard (i.e. a model which has trained on queries belonging to the docid we're seeking).

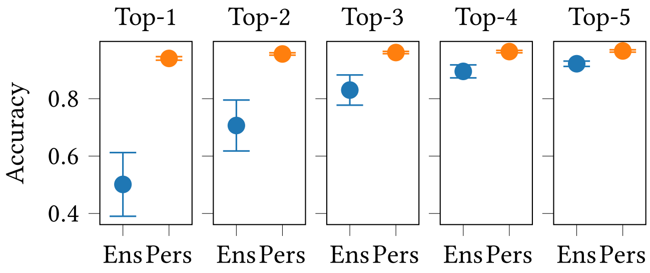

Figure 3. Results of top-k accuracies when inferring from the ensemble (Ens) vs. from only the personal model (Pers), with $k = 1..5$.

It can be seen that, as previously, peers trained well on their data (orange boxplot). In the ensemble we can observe a markedly lower top-1 mean accuracy than in the 'personal model' method. This may be due to other shards containing similar documents (which implicitly have similar queries associated with them). This would lead to the model outputting a sequence with high confidence, even though the expected sequence is not the correct one. For example, the query 'Tesla V3' may render a high confidence from a model which was trained on a document featuring the car, but also on a model which was trained on a document about renowned scientists. In this case there is a high chance that the confidences of both models would be high enough to make the end result a tossup. However, as we take into consideration metrics with a larger k (as in top-k), the chance that the right suggestion is among them approaches the chance that the personal model was asked.

Table 2. Comparison of accuracies with 1000 documents, between a single model vs. an ensemble of 10 models.

Model	Top-1	Top-5
Singular	0.860	0.923
Ensemble	0.501	0.922

We also compare the accuracy of the ensemble vs. the accuracy of a single model trained on all 1000 data in Table 2. Here too, the top-1 accuracy is badly damaged by using the ensemble method. The top-5 accuracy metric interestingly exhibits almost the same result for both the ensemble and the singular T5 method.

In this experiment it does not pay to use the ensemble (both in terms of accuracy and of higher computational cost). However, the high accuracy in the top-5 metric shows that the assembly method of the results of different T5 models works in principle. What would be needed is to find a way to reduce the confusion of models from different shards which most likely affects the metrics presented.

Future research on this topic could attempt to shard data in a semantically meaningful way, possibly by only using queries. This would mean that each shard could deal with a certain aspect of the documents in the dataset, so confusion arising from multiple shards holding similar documents would diminish, thus increasing the accuracy of the ensemble.

Another research area could be the application of a mixture of experts on the topic of De-DSI, where a 'master model' could be utilized to pick which shard to ask the query to. This would increase the scaling capabilities further by not requiring models from all shards to suggest documents. In this case, only models from a few select shards could be made to retrieve answers, reducing the computational complexity required.

4.3 Decentralised DSI

For this experiment we aim to prove the efficacy of our ensemble algorithm in the P2P setting. To this end, we simulated a network of $N = 30$ peers, and divided the data into three shards (i.e. 10 peers per group). To each shard, we assign 5000 documents. We let each peer of that shard randomly sample between 200 and 300 documents from that pool. The retrieved set of documents reflects the personal dataset S (comprising query-docid pairs) of a peer. In P2P applications, this would be equivalent to users' recent personal history of search queries and the result they selected. This way we allow peers to have differently sized datasets, but also some documents to get sampled by multiple peers, while others might not got picked at all. In case of the latter, those documents were discarded from our experiment. By these means, we attempt to relax some of the conditions posed in the previous experiment, and add the noise that is encountered in real P2P systems.

Each peer maintains one local T5-small model, and a training batch $B = \{(q_i, d_j) \mid i, j \in \mathbb{N}\}$. The training batch has a fixed size of $|B| = 32$, and (q_i, d_j) mark one data point (one query-docid pair). We train in batches to speed up the training, but also to reduce the noise that occurs if models train on individual data points. The training data is collected through gossip within the peer group. To this end, we perform a simulation in message exchange rounds (in intervals of 0.1 s). In each round, every peer sends one $(q, d) \in S$ to another random peer from its group. Thereby, each peer receives, on average, one data point per round. Incoming data points are appended to the local batch. Furthermore, every new batch is initialized with a sample $S' \subset S$ of size $|S'| = \lfloor 32/N \rceil$. The idea behind this strategy is to uniformly distribute the personal dataset throughout the entire training, avoiding overfitting.

This method of peers sending training data to each other doesn't preserve privacy. Our goal in this paper is to showcase the Ensemble-DSI and implement it in a decentralised network. Future work on this topic could look into how to implement the structure of the decentralised algorithm to train the network in a privacy non-invasive manner. One direction worth investigating could be implementing the message exchange along with an onion routing protocol [22] so that no message travelling through the network could be traced to any one peer. Other measures that could conceivably be used to preserve privacy in decentralized networks are presented in [12].

As the models converged (see Figure 3), we stopped the simulation after 8000 batches that have been processed per peer. The testing phase proceeded to sample 3 models from each of the 3 shards (i.e. 9 models in total for each inference), and use the confidence-ensemble (presented in Section 3.2) to pick a top-1 and top-5 list for each shard. Since in this case, 3 models were most likely to output the same docid (because they were trained on the same shard), a simple summing procedure over the post-softmax score of all models was used to calculate the total confidence score for each sequence. Only after the sum was calculated, the top-k suggestions of the ensemble were picked. In Figure 4, we present the accuracies on the test set for all models belonging to each of the three shards (denoted A, B, and C). The accuracies found in this experiment are in line with those found in Section 4.1, proving that the decentralised training method was successfully applied, with top-1 averaging 88 %, and top-5 at 92 %.

In Table 3, we show the performance of the ensembles. Additionally, we conduct an experiment to investigate whether adding more models from the same shard improves accuracy. Specifically, we aim to determine if merging the outcomes of various models trained on identical data leads to enhanced performance. The label in column "model pool" describes whether models exclusively from the same shard were available for sampling, or whether we could sample models from each of the three shards.

Table 3. Accuracies for our experiment on decentralised DSI training.

Shard	Top-1 Acc.	Top-5 Acc.	Model Pool
A	0.849	0.933	All shards
B	0.850	0.932	All shards
C	0.872	0.941	All shards
A	0.913	0.947	Own shard only
B	0.912	0.943	Own shard only
C	0.922	0.950	Own shard only

It can be seen that the ensemble increases accuracy when used exclusively within the same shard. Because in this ensemble we have three models from each shard, as opposed to a single one, we can see improvements in accuracy of top5 even when compared to the 92% average shown in Figure 4. This is important because it means that the ensemble can be used to improve performance as well, not just to increase the number of available data. Additionally, the method of pooling models from different shards increases accuracy to levels comparable to using one individual correct model to predict the label. In the case of Top5 it even increases it to levels beyond the performance of the average correct model (found in Figure 3) for all 3 shards.

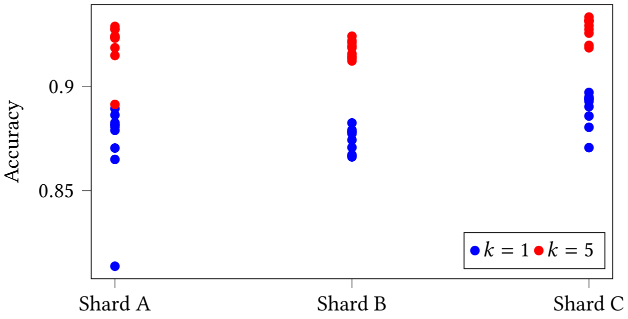

Figure 4. Accuracies on the test set, by shard and beam. Blue dots represent the top-1 accuracy, while red dots show the top-5 accuracy of one peer in the associated shard.

The amount of computation this ensemble method requires scales linearly with the number of shards in the retrievable dataset (since we need at least one, though as we have seen - more is better, model per shard). This limmits the application of the sharding mechanism. However, the average person uses a search engine 3 to 4 times per day [30]. Assuming one 5-beam query running on a local machine takes about 0.2 seconds (Mac M2 Pro processor), one could distribute the workload of the models in the network. A query written by a peer could be sent to other peers as well so they could 'chime in' with their suggestions. Assuming 20 shards as described above and 4 queries per day for each person, one could send a query to 5 random individuals belonging to each shard. This would lead to a processing time for the activity of online search of about $0.2 \cdot 4 \cdot 20 \cdot 5 = 80$ seconds per day per user, quite a meager amount of processing time per person.

The overhead for communicating the query and receiving suggestions from peers would be only that required by the transfer of a few bytes, representing the query or docid. Assuming regular internet connection with peers from all shards residing on the same continent, this communication overhead would be placed under 100 ms one way and another 100ms back. Adding the processing time, one search instance would involve a waiting time of around 0.4-0.5 seconds, small enough to make the search convenient.

These results confirm the plausible introduction of sharding data on P2P networks which use an LLM as a main search engine.

4.4 Decentralised video search

Finally, we demonstrate the generalizability of our method. To this end, we added support for content identifiers beyond docids, with a step towards generic URL support. This enhancement is particularly significant given the widespread popularity of video services like YouTube and TikTok, which cater to a broad audience. BitTorrent provides an open protocol for the decentralised sharing of videos [27]. This system identifies files using magnet links that contain a 40-character hexadecimal hash string [11]. We show in this experiment that De-DSI is also capable of generating document identifiers of this type, which are longer than those given by ORCAS' docids (8 characters in length).

For this experiment, we used a dataset of magnet links based on our prior crawling and dataset efforts from 2003 onwards [27, 42, 23]. We merged a magnet dataset with the ORCAS dataset by simply replacing docids with URLs. It needs to be mentioned that the URL's characters are not semantically relevant to the document they refer to, similar to the ORCAS docids in this sense. We expected that generating URLs would be error-prone as there are more tokens which have to be generated correctly in order to identify a document. If one of the 40 characters is mismatched, we count that as a failure.

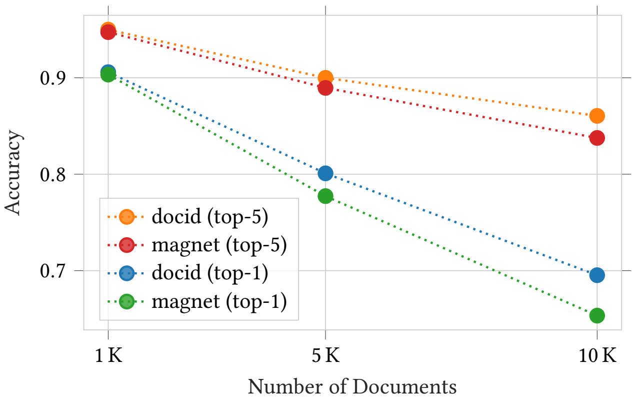

Figure 5. Comparison of top-1 and top-5 accuracy using the same dataset, with the target string represented as docid and magnet link.

The experiment follows the design in Section 4.1. We tested with 1000, 5000, and 10 000 documents, with their ID encoded either in the default ORCAS way, or with an assigned magnet link. The documents have been chosen so they have at least 40 queries associated with them, 20 in the training set, 10 in validation, and 10 in the test set. The results shown in Figure 5 are for top-1 and top-5 accuracies on the test set.

When the dataset is relatively small, the accuracies are the same for both top-1 and top-5. As more data appears in the dataset, we can see a divergence in the accuracies posted in both metrics. We hypothesize that the limited number of weights in our model efficiently captures URL patterns in scenarios with sparse data. However, as the data complexity increases, this constraint appears to hinder the model's ability to accurately recall the exact sequence of tokens in each URLs. This is merely a guess, and we intend to investigate this further in future work. However, the observed discrepancy in accuracy levels remains marginal, amounting to merely a few percentage points across a corpus of 10 K documents.

These preliminary results indicate that intermediaries such as video-sharing platforms can be decentralised. Our experimental work indicates that many entertainment platforms, e-commerce marketplaces, and financial intermediaries *could* be replaced with decentralised generative AI and various tools for decentralisation [1, 2, 8].

5 Related Work

The potential of scaling up model-based retrievers by employing a distributed model has also been addressed in another study. Zhou et al. [43] proposed *DynamicRetriever*, a model which showed improved accuracy over even the most advanced variant of DSI (DSI with semantically structured docids). In their study, the authors randomly partitioned a collection of 3.2 million documents into 32 distinct subsets,

Table 4. Technological progress of AI for information retrieval

Date	Title	Inventor	Generative AI	Lifelong Learning	Decentralised	Web Scale
Sep 2011	SGD [21]	Szeged Univ.	-	-	✓	-
Feb 2022	DSI [36]	Google	✓	-	-	-
Mar 2022	DynamicRetriever [43]	Renmin Univ.	✓	-	*partial*	-
Apr 2022	SEAL [3]	Meta	✓	-	-	-
Jun 2022	NCI [40]	Microsoft	✓	-	-	-
Dec 2022	DSI++ [16]	Google	✓	✓	-	-
Apr 2023	GenRet [35]	Baidu	✓	-	-	-
Apr 2024	**De-DSI**	**Our team**	**✔**	-	**✔**	-

each containing 100 000 documents. Each subset functioned as the training set for a model. That is, 32 individual models were trained on different datasets, respectively. This allowed the models to be much smaller, as fewer documents had to be memorized on an individual basis. In the distributed setting, groups of peers would be assigned different models, thus training on different subsets of the data. At retrieval, the query is sent to each group, and from each model a list of the top 100 documents, with their relevance scores, is retrieved. These items (3200 in total) are merged into a final ranking list. Their experiments yielded a sharp decline in accuracy over the non-distributed setting. The authors concluded that this is due to the inconsistent scale of scores learned by the independently trained models. Our model used softmax to aggregate the results, thus comparing them on the same scale.

The advent of DSI has motivated other researchers to develop techniques and advancements that would yield better retrieval accuracy. It has been shown in the architectures for DSI-QG [45] and NCI [40], for instance, that the generation and feeding of artificial queries on the basis of the documents' contents has the capability to significantly improve retrieval performance [45, 40]. Furthermore, Meta proposed SEAL [3], a system that uses n-grams from documents as docids, effectively improving efficacy. The performance of SEAL has been topped in GenRet [35], where an autoencoder is trained to tokenize documents into semantic docids.

While all those efforts have proved to be effective, they rely on the knowledge of document contents, which usually is not given in decentralised applications. In addition to techniques that exploit this knowledge, however, the authors of NCI [40] also proposed a novel prefix-aware weight-adaptive (PAWA) decoder, as well as an updated regularization loss function. Both have shown positive effects on the search engine's performance. As those experiments have been conducted on the basis of semantic docids, it is not clear what the effects of the PAWA encoder or the altered loss function would be on De-DSI.

Finally, DSI++ [16] addresses lifelong learning in the context of DSI. To this end, they propose two solutions. Firstly, they leverage *Sharpness-Aware Minimization (SAM)* to steer the model towards flatter loss basins, enhancing stability and reducing the propensity for catastrophic forgetting. Secondly, they employ a generative memory designed to produce pseudo-queries based on previously indexed documents. These pseudo-queries are then utilized for rehearsal, further bolstering the model's ability to retain and recall information over extended periods.

We have summarised these developments in Table 4. As can be seen, De-DSI is the first work to take a step into decentralizing generative AI for retrieval at web scale.

6 Conclusion

With De-DSI, we merged two fields and simultaneously brought them a small step closer to user trials and broad societal usage.

First, research into DSI represents a significant improvement in efficiency and efficacy compared to the classical index-retrieve-rerank architecture of information retrieval. By removing the need for document term indexing, and training merely on query-docid pairs, we eased the process further. Our key finding is that the mere provision of queries is sufficient to turn an open source Transformer into a public search engine. Additionally, we find that magnet links can directly be retrieved by the DSI method, with minimal impact on accuracy given a relatively small dataset.

Secondly, our De-DSI ensemble model shows self-scaling properties in our experiments. Although the computational complexity of the ensemble increases linearly with the number of shards in the retrievable dataset, we envision a distribution of the workload within the network to address this issue [41]. Each part of the global network could specialise in a certain flavour of content and build-up of a stable community. This stable community in turn enables strong security, for instance, with self-sovereign identities [34] and state-of-the-art Sybil-tolerant trust frameworks [18].

Acknowledgment

This work was funded by Dutch national NWO/TKI science grant BLOCK.2019.004.

References

[1] SMA Abbas. 2009. A gossip-based distributed social networking system. In *2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. IEEE, 93–98.

[2] Joost Bambacht and Johan Pouwelse. 2022. Web3: a decentralized societal infrastructure for identity, trust, money, and data. *arXiv preprint arXiv:2203.00398*.

[3] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35, 31668–31683.

[4] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. Corpusbrain: pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 191–200.

[5] Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. Orcas: 18 million clicked query-document pairs for analyzing search. *arXiv preprint arXiv:2006.05324*.

[6] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

[7] Mathijs de Bruin. 2021. Filecoin open grant proposal: scaling out ipfs-search.com along with ipfs. https://github.com/filecoin-project/devgrants/blob/master/open-grant-proposals/ipfs-search-scale-out.md. [Accessed 07-02-2024]. (2021).

[8] Martijn de Vos, Georgy Ishmaev, and Johan Pouwelse. 2022. Decentralizing components of electronic markets to prevent gatekeeping and manipulation. *Electronic Commerce Research and Applications*, 56, 101220.

[9] Frido Emans. 2023. Bump in the road. https://blog.ipfs-search.com/bump-in-the-road/. [Accessed 07-02-2024]. (2023).

[10] Alexander Gorishnyak. [n. d.] GitHub Release Stats — qwertycube.com. https://qwertycube.com/github-release-stats/?OWNER=tribler&REPO=tribler. [Accessed 22-02-2024]. ().

[11] Arvid Norberg Greg Hazel. 2017. Bep$_0$009.$rst_post---bittorrent.org$. https://www.bittorrent.org/beps/bep_0009.html. [Accessed 29-03-2024]. (2017).

[12] Ehsan Hallaji, Roozbeh Razavi-Far, Mehrdad Saif, Boyu Wang, and Qiang Yang. 2024. Decentralized federated learning: a survey on security and privacy. *IEEE Transactions on Big Data*, 10, 2, (Apr. 2024), 194–213. doi: 10.1109/tbdata.2024.3362191.

[13] Tom Kwiatkowski et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 453–466.

[14] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. 2023. Decentralized federated learning: fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys and Tutorials*, 25, 4, 2983–3013. doi: 10.1109/comst.2023.3315746.

[15] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. Communication-efficient learning of deep networks from decentralized data. (2023). arXiv: 1602.05629 [cs.LG].

[16] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2022. Dsi++: updating transformer memory with new documents. *arXiv preprint arXiv:2212.09744*.

[17] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. In *Acm sigir forum* number 1. Vol. 55. ACM New York, NY, USA, 1–27.

[18] Bulat Nasrulin, Georgy Ishmaev, and Johan Pouwelse. 2022. Meritrank: sybil tolerant reputation for merit-based tokenomics. In *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 95–102.

[19] Mark Nottingham. 2023. RFC 9518: Centralization, Decentralization, and Internet Standards — datatracker.ietf.org. https://datatracker.ietf.org/doc/rfc9518/. [Accessed 22-02-2024]. (2023).

[20] Róbert Ormándi, István Hegedüs, and Márk Jelasity. 2011. Efficient p2p ensemble learning with linear models on fully distributed data. *CoRR abs/1109.1396*.

[21] Róbert Ormándi, István Hegedűs, and Márk Jelasity. 2013. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25, 4, 556–571. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.2858. doi: https://doi.org/10.1002/cpe.2858.

[22] Paolo Palmieri and Johan Pouwelse. 2014. Key management for onion routing in a true peer to peer setting. In *Advances in Information and Computer Security: 9th International Workshop on Security, IWSEC 2014, Hirosaki, Japan, August 27-29, 2014. Proceedings 9*. Springer, 62–71.

[23] [n. d.] Peer to peer trace archive. http://p2pta.ewi.tudelft.nl/datasets/. [Accessed 28-02-2024]. ().

[24] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*.

[25] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

[26] Johan Pouwelse. 2000. Open information pools. In *2000 USENIX Annual Technical Conference (USENIX ATC 00)*.

[27] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. 2005. The bittorrent p2p file-sharing system: measurements and analysis. In *Peer-to-Peer Systems IV*. Miguel Castro and Robbert van Renesse, (Eds.) Springer Berlin Heidelberg, Berlin, Heidelberg, 205–216.

[28] Johan A Pouwelse et al. 2008. Tribler: a social-based peer-to-peer system. *Concurrency and computation: Practice and experience*, 20, 2, 127–138.

[29] ProbeLab. 2023. Amino (the public ipfs dht) is getting a facelift. https://blog.ipfs.tech/2023-09-amino-refactoring/. [Accessed 07-02-2024]. (2023).

[30] Lily Ray. 2019. We surveyed 1,400 searchers about google - here's what we learned. https://moz.com/blog/new-google-survey-results. Accessed: 2024-02-24. (2019).

[31] Matei Ripeanu. 2001. Peer-to-peer architecture case study: gnutella network. In *Proceedings first international conference on peer-to-peer computing*. IEEE, 99–100.

[32] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

[33] Amit Singhal et al. 2001. Modern information retrieval: a brief overview. *IEEE Data Eng. Bull.*, 24, 4, 35–43.

[34] Quinten Stokkink and Johan Pouwelse. 2018. Deployment of a blockchain-based self-sovereign identity. In *2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 1336–1342.

[35] Weiwei Sun et al. 2024. Learning to tokenize for generative retrieval. *Advances in Neural Information Processing Systems*, 36.

[36] Yi Tay et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35, 21831–21843.

[37] Almer S Tigelaar, Djoerd Hiemstra, and Dolf Trieschnigg. 2012. Peer-to-peer information retrieval: an overview. *ACM Transactions on Information Systems (TOIS)*, 30, 2, 1–34.

[38] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. 2011. A survey of dht security techniques. *ACM Computing Surveys (CSUR)*, 43, 2, 1–49.

[39] Jun Wang, Marcel JT Reinders, Johan Pouwelse, and Reginald L Lagendijk. 2005. Wi-fi walkman: a wireless handheld that shares and recommends music on peer-to-peer networks. In *Embedded Processors for Multimedia and Communications II*. Vol. 5683. SPIE, 155–163.

[40] Yujing Wang et al. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35, 25600–25614.

[41] Niels Zeilemaker, Zekeriya Erkin, Paolo Palmieri, and Johan Pouwelse. 2013. Building a privacy-preserving semantic overlay for peer-to-peer networks. In *2013 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 79–84.

[42] Niels Zeilemaker and Johan Pouwelse. 2014. 100 million dht replies. In *14-th IEEE International Conference on Peer-to-Peer Computing*, 1–4. DOI: 10.1109/P2P.2014.6934318.

[43] Yu-Jia Zhou, Jing Yao, Zhi-Cheng Dou, Ledell Wu, and Ji-Rong Wen. 2023. Dynamicretriever: a pre-trained model-based ir system without an explicit index. *Machine Intelligence Research*, 20, 2, 276–288.

[44] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: a survey. *arXiv preprint arXiv:2308.07107*.

[45] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

Received 26 February 2024