Ditto: Towards Decentralised Similarity Search for Web3 Services

Navin V. Keizer*, Onur Ascigil†, Michał Król‡, George Pavlou*
*University College London, †Lancaster University, ‡City, University of London
navin.keizer.15@ucl.ac.uk, o.ascigil@lancaster.ac.uk, michal.krol@city.ac.uk, george.pavlou@ucl.ac.uk

***Abstract*—The Web has become an integral part of life, and over the past decade, it has become increasingly centralised, leading to a number of challenges such as censorship and control, particularly in search engines. Recently, the paradigm of the decentralised Web (DWeb), or Web3, has emerged, which aims to provide decentralised alternatives to current systems with decentralised control, transparency, and openness.**

In this paper we introduce Ditto, a decentralised search mechanism for DWeb content, based on similarity search. Ditto uses locality sensitive hashing (LSH) to extract similarity signatures and records from content, which are stored on a decentralised index on top of a distributed hash table (DHT). Ditto uniquely supports numerous underlying content networks and types, and supports various use-cases, including *keyword-search*. Our evaluation shows that our system is feasible and that our search quality, delay, and overhead are comparable to those currently accepted by users of DWeb and search systems.

***Index Terms*—Decentralised Search Engine, Similarity Search, Locality Sensitive Hashing, Decentralised Web.**

I. Introduction

The Internet is one of the most fundamental technological inventions of the last century and an integral part of our lives. It is used as a main source of knowledge and entertainment, and to maintain business and human relations.

While originally designed as a decentralised network of equal peers, the Internet, including Web search, evolved into an increasingly centralised, profit-oriented system, where a few tech giants control most of the market. Centralisation allowed services to scale, provide payment-free services through the monetisation of targeted advertisements, and deliver excellent quality services. However, this model consequently suffers from fundamental problems that are recently becoming increasingly visible.

When a single platform controls the majority of the market, as is the case with centralisation, an imbalance of power is created between the controlling entity and its users [1], who cannot easily migrate between ecosystems. Tech giants are incentivised to keep the environment, through which they have a leading position, as static as possible, encouraging abuse of power [2] and elimination of competition [3]. This profit-oriented approach creates policies that may influence society as a whole [4] and may lead to radicalisation [5], depression or increased suicide rates [6].

Furthermore, gathering enormous amounts of user data without any control facilitates manipulation. The problem ranges from making users addicted to the platform, to attempts to control what the users see, hear and ultimately think [7]. A centralised ecosystem also gives authoritarian regimes a convenient tool for censorship [8] and propaganda [9]. Finally, even with highly secure Cloud services, centralisation creates a serious threat where a single human error, cyber-attack or natural disaster causes a global network outage [10], [11].

The decentralised Web (*i.e.* DWeb), also known as Web3, aims to shift the power balance in favour of users through openness, equality, self-governance, and transparency. Such ecosystems are more inclusive and allow populations outside the criterion of profitability to participate. The DWeb encompasses several novel inter-operating decentralised technologies such as blockchains [12] for building trust and value transfer, and decentralised storage networks (DSN) [13] for distributed content storage.

In the current Web model, search engines, such as Google or Yahoo, are the main entry point for users accessing the Internet. The two main search categories used are *keywords search* and *similarity search*. In the former, users submit their keywords to a search engine, which returns content it has previously crawled and indexed from centrally-managed Web servers. In the latter (*e.g.* Google reverse search) users submit a base item such as an image, upon which the engine returns previously crawled items that are close to the base item in terms of a distance metric. Similarity search is also used for multiple kinds of recommendation systems, where users are shown items (*e.g.* entertainment media) that they might like based on their interaction history.

Despite the significant progress of decentralisation in recent years, one of the main challenges remains unsolved: realising *decentralised search engines*. Previous attempts to create decentralised search engines do not support both keyword and similarity search [14]–[18], do not provide result integrity [14]–[16], [18], are not completely decentralised [19], [20], are bound to a specific underlying technology [14]–[18], or require a global view of the network making them difficult to deploy in practice. We analyse existing search systems in Section VI-A.

In this work, we present Ditto, a decentralised search mechanism for DWeb services, which allows for various search use-cases on a variety of DWeb content —independent of the underlying DSN or blockchain. Ditto achieves its functionality by using a similarity search mechanism based on locality-sensitive hashing (LSH), which extracts short content signatures that maintain similarity features. Consequently, Ditto can group together content with high similarity. On top of this various search functionality beyond recommendation can be implemented (see Section III-C). Furthermore, by extracting

and hashing keywords directly from content in our algorithm, we provide a truly *decentralised keyword-search*.

Ditto does not require centralised components or a global view of the stored content, and security is guaranteed by the verifiability of LSH computations. When adding DWeb content, providers are incentivised to compute similarity signatures. Mappings from signature to content identifiers (CID) are stored using a modified distributed hash table (DHT), allowing for quick and reliable lookup. A user or application can leverage search functionality by supplying a query consisting of a signature, content type, and similarity range. Through our initial evaluation, we verify the feasibility of our system. We find that the delays in terms of signature generation and lookup of LSH are acceptable for our decentralised setting (sub-second as expected by current Web users), the overhead of participation does not greatly increase beyond current systems, and that the quality of similarity search and keyword-search is acceptable and comparable to our baseline (recall up to 57%).

To summarise, in this work we have proposed Ditto, which uniquely achieves the following:

- A general decentralised search mechanism with interoperability for various DWeb content sources and types.
- *Similarity search* for a broad number of use-cases such as recommendation, malware detection, and moderation.
- A decentralised way of achieving *keyword-search*, allowing for semantic search on the DWeb without any single root of trust.

The rest of this paper is structured as follows. Section II provides background on the building blocks used in this work. Section III presents our system assumptions and goals, and Section IV describes the architecture design of Ditto. Section V presents an experimental evaluation and verification of our system feasibility, after which we review and analyse related work in Section VI. Finally, we conclude our work in Sections VII and discuss future work.

II. Preliminaries

In this section we provide a background on the key concepts used in Ditto. Specifically, we give an overview of DHTs and the DWeb, and give a primer on similarity search using LSH.

A. Peer-to-Peer (P2P) and Distributed Hash Tables

Peer-to-peer (P2P) networks are distributed systems that partition tasks or workloads between equally privileged peers. These peers share resources such as storage, bandwidth, or computation with the rest of the network, alleviating the need for centrally managed Web servers and services. Most DWeb initiatives leverage decentralised networks based on P2P, which are generally implemented using virtual overlays on top of the physical network topology, which can be either unstructured or structured.

While unstructured overlays are formed by random connections to peers and query techniques like flooding —without a global knowledge [21]— structured overlays organise peers in a specific network topology. While this adds overhead, it improves performance, especially for the recall of rare content. Most structured overlay networks use a distributed hash table (DHT) [22]–[24], where a variant of consistent hashing is used to map content or resources to peers. Peers can then easily search the network using a hash table, allowing for efficient retrieval of content (usually in $O(log(n))$).

For example, in Kademlia [22], the most popular DHT implementation, peers randomly create a public/private key pair when joining the network on a 160-bit keyspace envisioned as a binary tree. To assign responsibility for a $\{$*key:value*$\}$ pair to the closest node, the XOR distance metric is used, due to its triangular inequality, symmetry, and unidirectionality properties.

B. Search on the Decentralised Web

The decentralised Web, also known as DWeb or Web3, aims to create decentralised alternatives to current Web services, alleviating some of the issues of the current infrastructure. Throughout this paper, when mentioning the centralisation of the current Web, we refer to the concentration of power, with a few parties controlling the majority of Web services in terms of management, governance, and ownership.

In the current Web, most users access and discover Web content using a keyword-search based workflow (although alternative workflows exist). A user generally opens a search engine and submits a number of keywords, specifying the content they are looking for. The search engine then queries its index which it has crawled proactively and ranked, and returns the results pointing to the content location. Content is generally stored at distant server infrastructures (*e.g.* the Cloud).

In the DWeb, each step of content discovery and interaction is envisioned to be decentralised and performed collaboratively by peers in the network. DWeb content is mainly stored on DSNs —P2P networks for collaborative storage. The ideas behind these networks were first developed in early P2P works two decades ago, but have recently regained interest and have improved in terms of robustness, security, and stability, leading to numerous novel storage networks emerging [13]. One key differentiating factor of these networks is that they leverage content-addressing in order to allow for efficient and scalable retrieval, and distributed control and storage of data. Instead of referring to content by its location, a self-certifying hash of the content is used as its identifier (CID), allowing retrieval of content from anywhere in the network, rather than being restricted to retrieve from only one of the content providers' locations. In order to reward honest peers for participating and collaborating by sharing their resources used for performing tasks such as search and storage, blockchains and smart contracts can be used as an additional incentive layer [25].

Blockchains and smart contracts are themselves another data source on the DWeb, particularly for financial data. They use similar hash-based names derived from Merkle trees [26] pointing to transactions. CIDs and transaction hashes are not human-readable by default, meaning that small changes to the content modify its identifier entirety. This is one of the key reasons that it is vital to have a search mechanism on top of

this storage layer. However, implementing a truly decentralised search mechanism is non-trivial, as generally there remains a root of trust like the DNS, Public-Key Infrastructure (PKI), or centralised crawling/verification servers.

C. Similarity Search

Similarity search aims to provide a mechanism for identifying content with high similarity. When a query is submitted, the goal of the algorithm is to return a number of items with high similarity of content. This has been applied in a variety of use cases [27], and implemented either using space partitioning methods such as tree-based mechanisms [28], or using hash-based approaches like Locality Sensitive Hashing (LSH) [29]. As the latter has higher performance guarantees at lower overhead, we use this method in Ditto.

The goal of LSH is to extract a short *signature* from content based on hashing, where the hashing algorithm maximises hash collisions for similar items. As signatures retain information about similarity, they can be used to perform comparisons and find similar items, rather than having to use the raw file, greatly improving performance.

Popular LSH mechanisms include random hyper-planes [30], multi-probe [31], and LSH forest [32]. In this paper, we use a minhashing [33] approach, which reduces a high dimensional content vector into a short signature using N random hash functions. First, a binary feature vector is extracted from the content, after which a large number of randomly permuted hash functions are generated, where the signature length determines the number of hash functions used. The values in the signature are then given by the position of the first row with a 1 entry following each hash function.

The minhashing approach maximises the probability that contents that are similar are mapped to the same bucket in the signature. In the general minhashing LSH algorithm, signatures are further split into buckets, creating a large number of hash-tables that capture part of the signature. When looking for similar content, a requestor checks all hash-tables to generate a candidate set, on which a similarity metric is applied to get the closest items. While a number of similarity metrics can be used [34] such as cosine similarity and Hamming distance, the minhashing approach estimates the Jaccard similarity [35], which is defined as the intersection over the union between two sets A and B:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

.

III. System Properties

In this section, we discuss the assumptions made in our design of a decentralised similarity search system. We then present our desired properties and possible use-cases of Ditto beyond similarity search.

A. Assumptions

We first note that in this work, we focus on the networking aspects and feasibility of implementing decentralised similarity search for the DWeb, and while we discuss and use data pre-processing and LSH tools, these are not our main contributions. To achieve Ditto, we have based the design of our similarity search mechanism on a number of assumptions.

- A *search* is defined in our system as nodes submitting a large number of queries to the network, consisting of a number of parameters specifying the search, and in return they receive all relevant content in the network.
- The *network* in Ditto is defined as a collective of nodes in a P2P network who collaborate to provide search functionality. A node can simultaneously be a search node as well as a network node, but is not required to be both.
- When discussing *content*, we refer to data of various mime types, and stored on a number of different underlying P2P storage networks (*i.e.* multiple decentralised data sources). These collectively store a set of files and data, where each item is identified by a content identifier (CID). We remain agnostic to the storage network implementations and CID conventions. However, in the rest of the paper, for simplicity we focus on a network where the CID convention uses self-certifying names —this mirrors IPFS [36], the largest active DSN [13].
- We assume a random distribution of files to nodes that store them (*i.e.*, each participant decides independently which files to store without following any specific rules), although a DHT is used to store pointers to content providers (nodes who can provide the content).
- We assume that nodes in storage networks are incentivised to make their content available on a search system, and would participate in a collaborative search system. We assume the presence of arbitrarily malicious nodes in the P2P network that may not follow the protocol for individual gain. No single node is trusted entirely by its peers. However, at least one neighbouring node is assumed to be honest (*i.e.* nodes are not entirely eclipsed).

B. Desired Properties & System Goals

In designing a decentralised search mechanism for the DWeb (specifically one based on similarity search) we have identified the following desired goals:

1) Search Flexibility: Since the DWeb is comprised of a large number of complementary protocols and content sources (*e.g.* many different DSNs and blockchains), a search mechanism should be flexible in supporting a wide range of content types (*e.g.* text and videos), as well as various storage networks. The naming convention of identifiers (*i.e.* signatures) should also be compatible with current DSNs.

Furthermore, keyword-search should be supported as this is currently the most popular workflow, but a number of other search use-cases should be implemented, which we describe in Section III-C. It is further desired that users have control over system parameters to allow for personalised search and recommendation services. For example, the query similarity threshold and ranking policies should be user-defined.

2) Decentralisation: There should be no central or trusted entity involved in our mechanism, and global knowledge about all the files stored in the network should not be required to

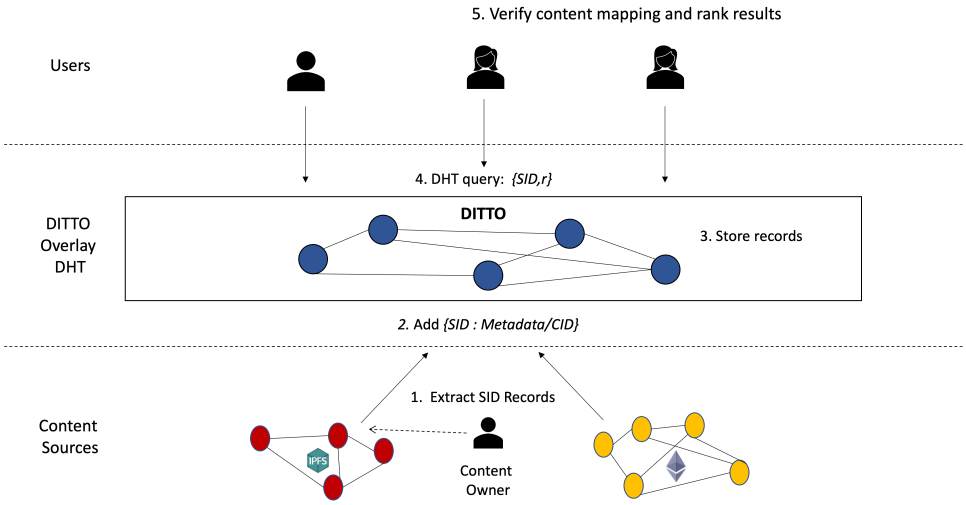

Fig. 1: Overview of Ditto, where users submit queries and are able to query from various underlying content sources.

query or participate in system upkeep. Each participant should be able to calculate signatures of files and securely verify each step of retrieval.

In order to achieve decentralisation and realise an open system, participation or resource-constraint nodes should be enabled, and therefore the system should have low-resource usage, and should not require costly global synchronization.

3) Security: Our mechanism should be secure against malicious peers and attacks. Specifically, each step of the search process should be *verifiable*. Without verifiability, any malicious nodes could return fraudulent results forcing the requesting node to accept irrelevant files.

The calculation of signatures and the distance between them is deterministic and verifiable by requesting nodes. This property is required to ensure the correctness of the returned results, and each peer should be able to independently produce signatures. Finally, the integrity of the content itself should ideally also be verifiable, for example using self-certifying naming conventions, but this is dependent on the underlying content network.

4) High Performance Guarantees: For a decentralised search mechanism to be useful, it needs to have comparable performance compared to centralised alternatives. Here, performance refers to a number of properties. First of all, user queries need to be returned quickly without a high *delay* (users expect sub-second delays in current systems). Second, the results returned need to be of high quality (*i.e.* a high *recall* of the theoretical most relevant results globally), comparable to centralised search engines.

Finally, the *overhead* of calculating the signatures and storing them should be low, in order to limit the bandwidth, storage, and computational power that peers have to dedicate to the network. If the overhead becomes too large, peers may require extra incentivisation for participation in the network, adding a layer of complexity.

Specifically, any network tasks should have sub-second delays on average user machines, and storage overhead should at most be comparable to other P2P network storage (*e.g.* storage networks).

C. Use-Cases

Our proposed similarity search system can be applied in a number of DWeb use-cases, which we will now describe.

1) Decentralised Search Engine: One of the main use-cases for Ditto is to facilitate decentralised search and replace current search engine based workflows. Users can submit content queries and retrieve all network content that is similar to it. This is extended to traditional keyword-search by hashing keywords directly and producing signatures from extracted keywords alone. We further discuss this in Section IV.

2) Recommendation Engine: Ditto can be used to create a decentralised recommendation engine for services and applications (DApps), which gives content providers and users recommendations for similar content on the DWeb. For example, music or video services can use similarity search to propose relevant items for users based on content they enjoy, without running centralised machine learning or data collection. This may also be applied to achieve decentralised and privacy-preserving personalised advertisements.

3) Malware and Illegal Content Detection: DSNs can be used to store and distribute copyrighted and illegal content without control from centralised parties, flying under the radar of law enforcement. Distributors or this content may slightly change files from the original so they cannot be found using known image hash databases of illegal content. To combat this, similarity search can be used to do an extensive search of potentially illegal files comparing to these databases. Conversely, copyright owners can apply the same techniques to find copyright-infringing content, which they can report to law enforcement. We note that while we can detect this content, we cannot remove them from the network, and hence local moderation techniques may be implemented.

A similar technique can be used to check files for malware, which involves submitting signatures of content before open-

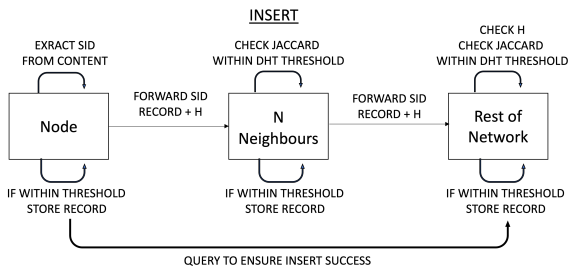

Fig. 2: Overview of inserting a similarity record.

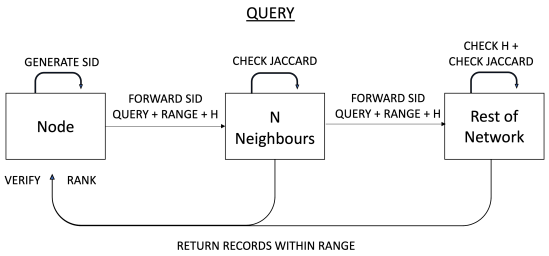

Fig. 3: Overview of a user query for a specified SID.

ing them and checking for high similarity of known databases of malware and illegal content, allowing one to asses whether the file is potentially hazardous.

4) Decentralised Moderation: As mentioned above, malware, illegal, and copyrighted content can be detected using similarity search. However, there are more types of content which may not be desired by a user based on their individual preference. For example, hate speech or extremist views may want to be avoided. As complete censorship in the network is not possible (similar to illegal/copyrighted content), users may implement local decentralised moderation strategies, which may avoid or assign a low ranking to results which are similar to content that the user has flagged to be unwanted. This method allows for transparency and control of censorship and result ranking.

IV. Architecture Design

In this section, we describe the architecture design of Ditto, which captures the desired goals outlined in Section III-B. Our system is agnostic to content data type and underlying DWeb network, but to illustrate its functionality we focus on terminology from the IPFS DSN for simplicity. We first give an overview of our system functionality, before describing individual components.

A. Overview

As described in Section III-A, Ditto takes inputs from different decentralised data sources. When a content owner uploads content to the network they simultaneously extract a *similarity identifier* (SID) based on an LSH signature. These content owners are incentivised to produce these identifiers for public content, as otherwise their content will not be searchable in the network. This assumption only applies to content for public consumption (*e.g.* Web pages), and not to private data. Signature generation is verifiable by the network, as nodes can easily recompute SIDs. The SID is structured such that it supports different data types and content networks.

To store the decentralised index, we use an overlay network based on the Kademlia ID space, but using the Jaccard similarity as the default distance metric. Nodes in the network store content records close to them in the ID space in the form of $\{$*SID:Metadata/CID*$\}$, where metadata specifies the content source and type, and CID is the identifier to query the content itself. When searching for content, a user sends a query to the overlay network consisting of a SID and a parameter r, specifying a *similarity range*. The SID can be in the form of the signature of content for which users want similar items or recommendations for example, or keywords to implement semantic search, depending on the use-case. The network then returns all results which are within the *similarity range*, and the user can filter these based on local ranking policies (*e.g.* based on content type or size) and then fetch them on the underlying data source. Figure 1 gives an overview of our system.

B. SID Generation

To generate a decentralised search index on the DHT, we first need to extract SIDs. We first extract an LSH signature and then arrange it into a SID record format.

1) Signature Generation: In order to generate an LSH signature, we need separate pipelines for different content types, as the pre-processing and hash extraction should be optimised per type. For each content type, we first need to reduce and standardise the dimensionality of our data, after which the reduced data is used to generate a feature vector, capturing the uniqueness of the content. Feature vectors are then parsed into our LSH algorithm, where parameters are tuned per mime type, which (in the text case) is based on Minhash. Our evaluation focuses on text files, as this is the base for most Web searches (HTML is text). Describing the pre-processing for each specific type is out of our scope.

In order to provide deterministic outcomes, the same LSH parameters should be available to the whole network. This could be done for example using blockchain solutions or a decentralised autonomous organisation (DAO) [37], allowing flexibility in changing the parameters. Alternatively, this can be hard-coded in the protocol, although this is not desirable.

2) Addressing: One of the main benefits of our approach is that it allows for a general DWeb search, rather than focusing on a single content source. We achieve this by formatting our SID record to include metadata. In the hash-table entry the SID is the key, and the value is given by the

path: *content_type/content_source/CID*. For example, an IPFS image would have the entry {*SID:Image/IPFS/CID*}. Using this addressing, we keep the security properties of the CID naming convention, and could add additional useful metadata such as content title, which adds human-readability.

C. Network Implementation

After generating the SID for a content item, the content owner stores the corresponding record on the DHT overlay network. Nodes close in the hash space to the content store the record, and respond to queries if their records show any content within range *r*.

ID Space: In order to structure the network peers, we use the Kademlia ID space which can be visualised as a binary tree. A node has more knowledge of close nodes in the ID space which speeds up look-ups. When nodes join the network, they randomly generate an ID in the space and start building their routing table buckets.

Insert: Inserting and querying the DHT follow slightly different algorithms, because insert is not as time sensitive compared to serving user queries. In our protocol, another key parameter is the *DHT threshold*, which specifies the closeness of a node to a record for it to store it. This is because the Jaccard metric has the symmetry and triangular properties like the XOR metric, but is not unidirectional (*i.e.* the distance between a node and any two other nodes may be the same), which means there may not be a closest node, but a number of closest nodes. We can adjust the *DHT threshold* to tweak the amount of caching in the network.

To insert a record into the distributed index a node checks the Jaccard similarity to itself and their direct peers. They then place the record with any node within the threshold, and send it to the *N* closest nodes (who also forward it to their closest nodes) in terms of Jaccard similarity. Along with the record, the node sends a parameter *H* to indicate after how many hops a receiving node can discard a request, so it does not stay live in the network. Nodes receiving records verify the correctness of the {*SID:CID*} mapping before adding it.

After a time period, the node makes sure the record has been added to the network, and if it cannot find it, the above steps are repeated. Parameters *H* and *N* are again set on a protocol level, similar to the *DHT threshold* and LSH parameters.

Query: In order to query the network a node first needs a SID in which it is interested, for example based on similar content it has (as a recommendation engine). It sends the SID and a *similarity range* to its peers.

Each node that receives the query checks its local index for any entries which fall in the range, which they return to the sender. They also send it to their top N closest nodes until the termination condition H is met. In order to ensure security the node may recompute the distance of results to ensure it is correct. Alternatively, this system could easily be extended to return the top K closest results to a SID.

Ditto also implements *decentralised ranking*, as the querying node is responsible for compiling the results and ranking them based on a local policy. This can be based on similarity to the query, size, file type, or content network for example. However, our system can be extended to include metadata in the index files such as keywords, and content producer names, which allow for more fine-grained ranking mechanisms.

TABLE I: Delay of similarity calculation with different distance metrics (in 10^{-5} s).

	Mean	Standard Deviation
XOR	0.518	0.42
Jaccard	0.942	5.94
Euclidian	2.011	1.21
Hamming	2.029	2.38
Cosine	5.450	7.11

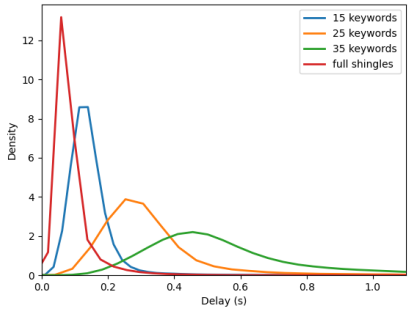

Fig. 4: Delay (s) of signature generation for full signatures and different top keyword sizes.

Keyword-search: Ditto, as described above, can be modified in order to allow for keyword-search. Rather than supplying SIDs of known content, a node can compute the SID directly based on the keywords it is interested in only. We desire a system where a user provides a number of keywords, which are then hashed into a signature, and which can then be queried in the network to receive relevant SIDs. However, if we implement this on the system described above directly (with signatures in the SID based on the content shingles), we may have extremely small similarity scores to compare. This is because the keywords are not weighted in the shingles and therefore their importance in the signature is not captured.

Instead, we can extract the most important keywords from the content and solely use these to produce a signature and SID, allowing for a better comparison of Jaccard similarity. This could be achieved using natural language processing tools. A drawback of this approach is that it requires a stronger mechanism to ensure the SID record mapping to content is correctly calculated (*e.g.* ensure that the keywords used in the signature are correct). This could be done using zero-knowledge proofs, and we discuss this further in Section V-D. We assess the feasibility of keyword-search in Ditto further in our evaluation.

V. Evaluation

In this section, we evaluate the feasibility of Ditto and verify that our design goals outlined in Section III-B are achieved.

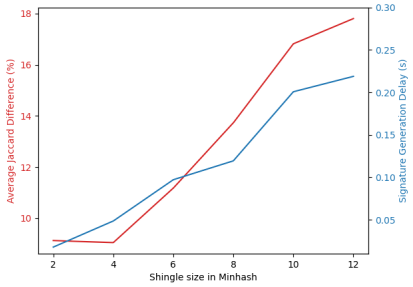

Fig. 5: Jaccard similarity difference and signature delay for varying numbers of shingle size.

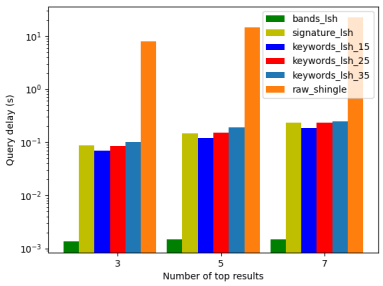

Fig. 6: Query delay (s) of search mechanisms for varying numbers of top results.

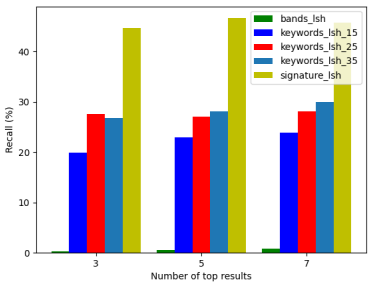

Fig. 7: Query recall (%) of search mechanisms for varying numbers of top results.

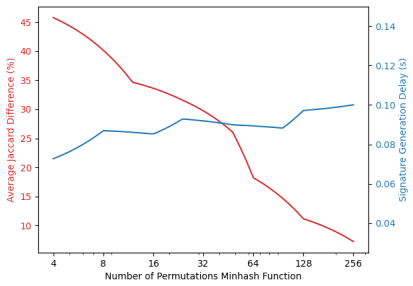

Fig. 8: Jaccard similarity difference and signature delay for varying numbers of permutation functions.

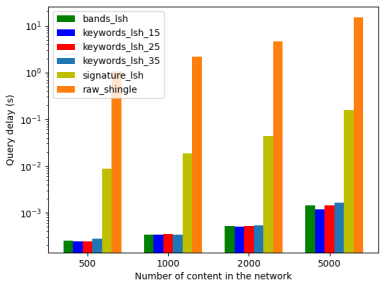

Fig. 9: Query delay (s) of search mechanisms for varying numbers of content in the network.

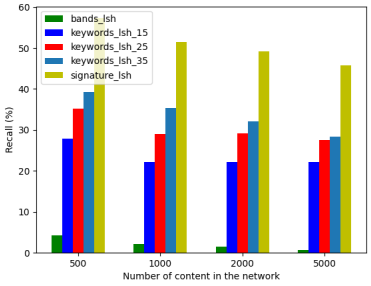

Fig. 10: Query recall (%) of search mechanisms for varying numbers of content in the network.

As our work is the first to explore LSH as a general search mechanism including keyword-search for DWeb settings, we focus mainly on its application and feasibility, leaving further evaluation of network and DHT aspects for our future work.

Setup: For our evaluation[1] we use a Wikipedia mirror which is a large dataset of content which can be found on the IPFS network. We have taken a subset of the Wikipedia articles in Dutch from the archive[2] to use as a dataset in the rest of this evaluation. Our LSH implementation can be seen as a lower bound, as we do not propose any optimisations, and therefore its main use is to establish that there is value in using LSH as a DWeb search mechanisms.

To calculate LSH signatures we use a Minhash implementation in Python[3]. We first extract the text from a page and produce shingles with length k, where k is set based on the length of the text (generally k=6 for our data). The algorithm then calculates the signature based on the specified number of permutation functions.

A. Similarity Metric

Our proposed system relies heavily on the calculations of Jaccard similarity for querying and inserting content into the decentralised index. To verify that this does not deteriorate performance significantly, we first compare the performance of this metric against other known similarity metrics.

We have taken a random subset of our dataset and computed signatures, on which we measure the delay in computing a similarity score. As shown in Table I, Jaccard similarity is relatively fast and comes close to XOR performance. We note that XOR is not applicable in our system as it uses the longest prefix match, instead of comparing sets within the signature. XOR is not suitable as this may result in a low distance in signatures, even though there is a lot of overlap in the content (but in different positions), particularly for keyword-search. For this reason, we focus our evaluation on the LSH specifics, rather than performing a comparative analysis with an XOR-based DHT. We plan on investigating our DHT properties separately in future work.

B. Minhashing Performance

Next, we assess the performance of the minhashing algorithm in terms of the delay and the accuracy in translating the information in the original document to a short signature. We take two subsets of our dataset, generate the signature, and store this along the raw shingles. We then randomly pick content pairs and compute the Jaccard similarity of the raw shingles and compare this to the signature, and we store the percentage difference between the two similarity scores. We also measure the delay of generating signatures.

[1] https://github.com/navinkeizer/ditto
[2] https://wiki.kiwix.org/wiki/Content_in_all_languages
[3] github.com/ekzhu/datasketch

Figure 5 shows how both the delay and the difference go up with higher values of k, suggesting that for our use-case a smaller k-value is appropriate. Figure 8 shows that the delay and accuracy are inversely correlated with higher numbers of permutation functions. However, in either case the delays are reasonable (sub-second) in our system, meeting our high performance goal. We note that this delay is not often incurred, as it is only calculated when adding or verifying content.

C. Search Performance

Finally, we evaluate the performance of search using LSH. We take our dataset and take subsets of different sizes in order to analyse scalability. We measure the overhead, query delay, and the recall, which we define as the percentage overlap of items returned compared to our baseline (*i.e.* on raw shingles). We implement a search system based on comparing the similarity of raw shingles as the baseline, and compare this against an implementation of traditional Minhash LSH using bands, and our signature comparisons.

After taking a randomised subset, we feed the shingles and signatures into our various search mechanisms. We then take a smaller subset and query for results on each mechanism, comparing the results. We have not optimised the sorting/searching algorithm on the stored LSH data, so we can view this as a lower bound on performance.

Furthermore, we implement **keyword-search** by extracting the top keywords from the text file using Automatic Keyword Extraction[4] and hashing these directly. We are then able to perform queries of keywords only (extracting the top keywords from the query subset), which we compare to our other implementations.

In terms of overhead, storing signatures instead of raw shingles reduces the size in memory by about 10^3 times. Figures 6 and 7 show respectively the delay and recall for varying the number of results returned by our search. We can observe that while the delay is very low for lsh_bands, the recall is extremely poor. Comparatively our signature_lsh approach performs much better, and keyword-search also works well, especially when setting the number of top keywords extracted for the signature at 35. We also observe that using raw_shingles is not a feasible option as there are large delays that grow linearly.

We observe the same when sweeping across the number of content in the system in Figures 9 and 10, where delays grow linearly, but remain manageable for signature_lsh, and are very low for keyword-search. In terms of recall, signature_lsh performs best, while keyword-search also achieves reasonable quality. In terms of generation delays, Figure 4 shows that using full signatures generally achieves delays of under 0.1s, while the keyword signature delays grow with larger numbers of keywords extracted due to our extraction algorithm overhead.

To summarise, we have shown that keyword-search is a feasible option with comparable delay, recall, and overhead compared to native similarity search, capturing our first design goal of search flexibility. We also show that our proposed system achieves the goal of high performance, as delays are sub-second, and recall is between 30-50% of baseline. Our system is also lightweight enough with low-resource usage to be deployed in a decentralised setting. In Section V-D we discuss how the security property can be met in more detail.

[4] github.com/LIAAD/yake

D. Security Analysis

As mentioned in Section III-B, one of our desired properties is security, both in terms of SID record mapping, as well as protection against malicious peers, who may not respond or return incorrect content records, or may not participate in network tasks like computing SIDs. Our system mitigates against these threats in a number of ways.

First of all, peers who upload content as public files are incentivised to calculate the SID mapping, as their content otherwise will not be discoverable. Second, using Jaccard similarity for inserting along with the *similarity threshold* parameter means that there will be redundancy in the network in the form of caching, as multiple close nodes store SID records, meaning that even if one or more nodes are malicious, there is still a high probability of finding the file from an honest peer.

In the querying process, nodes can verify that returned results are actually within the similarity range wanted with a lightweight Jaccard check. Furthermore, the mapping from CID to signature is verifiable as the algorithm and its parameters are set on a protocol level and can be computed by any peer. When using keywords directly, this mapping can also be verified if the keyword extraction is standardised. However, we may also use additional mechanisms like zero-knowledge proofs, avoiding the need to re-verify integrity at all nodes that come across the file. We aim to explore this solution further in future work.

VI. Related Work

In this section, we provide an overview of related work. We focus on decentralised search for DWeb content, as well as decentralised similarity search.

A. Decentralised Search on the DWeb

A number of works have proposed decentralised keyword-search mechanisms, similar to current search engine workflows based on crawling and indexing. Li et al. [19] proposed DeSearch, which decouples state from computation by using a centralised Cloud solution to store the index, while maintenance of the index uses decentralised workers executing verifiable tasks.

A number of works present improved decentralisation by storing the index on a P2P network. For example, SIVA [14] and Wang and Wu [15] propose to store a decentralised index for the IPFS [36] DSN directly on the IPFS DHT. Other initiatives [16], [18] have attempted to translate the centralised search engine workflow to a decentralised setting.

However, these search engines fail to capture the needs of a truly open and decentralised network, as they are not entirely decentralised in their index storage, or require a single root-of-trust for naming consistency and provenance. Furthermore, these approaches require lots of network participation for tasks like indexing and crawling, requiring additional incentives, which have not been implemented and it is unclear who will pay for it (monetary inflow source).

In contrast, Ditto hardly requires additional work for network peers who already actively store content. This is due to the fact that we do not copy the crawling/indexing model directly, but instead use similarity search. Further, for content producers, it is an extra incentive to participate in the system for their content to be found.

B. Decentralised Similarity Search

There has been lots of work on similarity search architectures in terms of implementations and optimisation techniques, and specifically in locality-sensitive hashing. A number of works [38], [39] have focused on implementing generic distributed versions of LSH, proposing a number of performance improvements. However, they either rely on centralised components, or they do not specify the networking implementation and solely focus on the algorithm details.

Some works have proposed to use a DHT to store the mapping of similarity signatures to content [40], [41]. Haghani et al. [42] leverage a cyclic DHT space based on Chord [23] to provide nearest neighbour search and queries within a range. While peers are organised in local DHTs, their system does leverage gateway peers.

Other works include Bahmani et al. [43], who improve network cost of LSH by proposing a layered LSH method on two distributed frameworks. Hamming DHT [44] implements a DHT where identifiers are generated based on LSH and the hamming distance metric is used in maintaining a Chord based ring structure. The works mentioned generally focus on increasing performance of LSH frameworks in a decentralised setting, but are not tailored towards the DWeb like Ditto, meaning that they lack details on specific security and privacy considerations.

Within the DWeb setting, Fujita [45] argues for implementing similarity search on IPFS using a DHT, but lacks details in areas like signature generation and network implementation. Yuan et al. [46] implement a LSH-based image retrieval scheme using blockchain smart contracts and distributed storage on IPFS.

Ditto differentiates itself in a number of ways from prior works. We provide interoperability and support for different content networks and types, which is required in a DWeb environment. Furthermore, the wide range of DWeb use-cases we described, and specifically keyword-search using LSH has not been explored prior.

VII. Conclusion and Future Work

In this work we have presented Ditto, a decentralised search mechanism for the DWeb based on similarity search. Ditto supports numerous decentralised content networks and types, and uniquely implements decentralised keyword-search. Our evaluation verifies the feasibility and shows that our system goals of flexibility, decentralisation, security, and performance are met.

While this paper focuses on the feasibility of decentralised search, we aim to explore the networking specifics in future work. Particularly, we aim to explore design specifics and evaluation of our proposed DHT, as well as unstructured and hybrid approaches for implementing decentralised search, in order to improve performance.

References

[1] V. Lehdonvirta, *Cloud Empires: How Digital Platforms Are Overtaking the State and How We Can Regain Control*. MIT Press, 2022.

[2] "Epic games vs apple," in *Wikipedia*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Epic_Games_v._Apple

[3] "Google and facebook ad deal." in *New York Times*, 2021. [Online]. Available: https://www.nytimes.com/2021/01/17/technology/google-facebook-ad-deal-antitrust.html

[4] M. Moore, *Tech Giants and Civic Power*. CMCP, Policy Institute, King's College London, Apr. 2016.

[5] "Youtube algorithms might radicalise people," in *The Conversation*, 2022.

[6] "Facebook calls links to depression inconclusive while researchers disagree," in *NPR*, 2022.

[7] "The all conquering quaver," in *The Economist*, 2022.

[8] "Russian internet blocked facebook meta," in *The Verge*, 2022.

[9] "China rumps up propaganta machine," in *MSN*, 2022.

[10] J. Swearingen, "When amazon web services goes down, so does a lot of the web," *New York Magazine*, 2018.

[11] A. Cuthbertson, "Facebook down: Users report issues with messenger and instagram," *The Independent*. [Online]. Available: https://www.independent.co.uk/life-style/gadgets-and-tech/facebook-down-messenger-instagram-not-working-b1950938.html

[12] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, and A. Zohar, "Survey on blockchain networking: Context, state-of-the-art, challenges," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–34, 2021.

[13] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks," 2021.

[14] N. Khudhur and S. Fujita, "Siva-the ipfs search engine," in *2019 Seventh International Symposium on Computing and Networking (CANDAR)*. IEEE, 2019, pp. 150–156.

[15] F. Wang and Y. Wu, "Keyword search technology in content addressable storage system," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2020, pp. 728–735.

[16] L. Zhu, C. Xiao, and X. Gong, "Keyword search in decentralized storage systems," *Electronics*, vol. 9, no. 12, p. 2041, 2020.

[17] B. Ramirez, "The graph network in depth," https://thegraph.com/blog/the-graph-network-in-depth-part-1, 2020.

[18] M. Zichichi, L. Serena, S. Ferretti, and G. D'Angelo, "Governing decentralized complex queries through a dao," in *Proceedings of the Conference on Information Technology for Social Good*, 2021, pp. 121–126.

[19] M. Li, J. Zhu, T. Zhang, C. Tan, Y. Xia, S. Angel, and H. Chen, "Bringing decentralized search to decentralized services," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021, pp. 331–347.

[20] ipfs search, "ipfs-search documentation," https://ipfs-search.readthedocs.io/en/latest/index.html, 2021.

[21] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proceedings First International Conference on Peer-to-Peer Computing*, 2001, pp. 99–100.

[22] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, p. 149–160, Aug. 2001. [Online]. Available: https://doi.org/10.1145/964723.383071

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, p. 161–172, aug 2001. [Online]. Available: https://doi.org/10.1145/964723.383072

[25] J. Benet and N. Greco, "Filecoin: A decentralized storage network," *Protocol Labs*, 2018.

[26] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[27] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam, and C. Crushev, "A survey on locality sensitive hashing algorithms and their applications," *CoRR*, vol. abs/2102.08942, 2021. [Online]. Available: https://arxiv.org/abs/2102.08942

[28] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '84. New York, NY, USA: Association for Computing Machinery, 1984, p. 47–57. [Online]. Available: https://doi.org/10.1145/602259.602266

[29] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 518–529.

[30] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 380–388. [Online]. Available: https://doi.org/10.1145/509907.509965

[31] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, ser. VLDB '07. VLDB Endowment, 2007, p. 950–961.

[32] M. Bawa, T. Condie, and P. Ganesan, "Lsh forest: Self-tuning indexes for similarity search," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 651–660. [Online]. Available: https://doi.org/10.1145/1060745.1060840

[33] A. Broder, "On the resemblance and containment of documents," 06 1997.

[34] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *CoRR*, vol. abs/1408.2927, 2014. [Online]. Available: http://arxiv.org/abs/1408.2927

[35] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Finding Similar Items*, 2nd ed. Cambridge University Press, 2014, p. 68–122.

[36] J. Benet, "Ipfs-content addressed, versioned, p2p file system," 2014. [Online]. Available: https://arxiv.org/abs/1407.3561

[37] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: Concept, model, and applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, 2019.

[38] D. Li, W. Zhang, S. Shen, and Y. Zhang, "Ses-lsh: Shuffle-efficient locality sensitive hashing for distributed similarity search," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 822–827.

[39] Y. Hua, B. Xiao, D. Feng, and B. Yu, "Bounded lsh for similarity search in peer-to-peer file systems," in *2008 37th International Conference on Parallel Processing*, 2008, pp. 644–651.

[40] I. Bhattacharya, S. Kashyap, and S. Parthasarathy, "Similarity searching in peer-to-peer databases," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 329–338.

[41] A. Smirnov and A. Ponomarev, "A hybrid peer-to-peer recommendation system architecture based on locality-sensitive hashing," in *Proceedings of 15th Conference of Open Innovations Association FRUCT*, 2014, pp. 119–125.

[42] P. Haghani, S. Michel, and K. Aberer, "Distributed similarity search in high dimensions using locality sensitive hashing," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, 2009, pp. 744–755.

[43] B. Bahmani, A. Goel, and R. Shinde, "Efficient distributed locality sensitive hashing," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 2174–2178. [Online]. Available: https://doi.org/10.1145/2396761.2398596

[44] R. da Silva Villaca, L. B. de Paula, R. Pasquini, and M. F. Magalhaes, "Hamming dht: Taming the similarity search," in *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, 2013, pp. 7–12.

[45] S. Fujita, "Similarity search in interplanetary file system with the aid of locality sensitive hash," *IEICE TRANSACTIONS on Information and Systems*, vol. 104, no. 10, pp. 1616–1623, 2021.

[46] B. Yuan, J. Peng, C. Li, and W. Qiu, "Ppirb:achieving an privacy-preserving image retrieval scheme based on blockchain," in *2022 5th International Conference on Data Science and Information Technology (DSIT)*, 2022, pp. 1–6.

[47] J. Arkko, B. Trammell, M. Nottingham, C. Huitema, M. Thomson, J. Tantsura, and N. ten Oever, "Considerations on internet consolidation and the internet architecture," IETF, Tech. Rep., 2020.

[48] C. Castillo, "Fairness and transparency in ranking," *SIGIR Forum*, vol. 52, no. 2, p. 64–71, jan 2019. [Online]. Available: https://doi.org/10.1145/3308774.3308783

[49] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Facebook and online privacy: Attitudes, behaviors, and unintended consequences," *Journal of computer-mediated communication*, vol. 15, no. 1, pp. 83–108, 2009.

[50] "Fact sheet how breaking up amazon can empower small business." in *Institute for Local Self-Reliance*, 2022. [Online]. Available: https://ilsr.org/fact-sheet-how-breaking-up-amazon-can-empower-small-business/

[51] V. Buterin, "A next-generation smart contract and decentralized application platform," 2015.

[52] E. Pariser, *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group , The, 2011.

[53] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *SIGMOD Rec.*, vol. 14, no. 2, p. 47–57, jun 1984. [Online]. Available: https://doi.org/10.1145/971697.602266