The WebEngine – A Fully Integrated, Decentralised Web Search Engine

Mario M. Kubek
Chair of Communication Networks
University of Hagen
Universitätsstr. 27, Hagen, Germany
mario.kubek@fernuni-hagen.de

Herwig Unger
Chair of Communication Networks
University of Hagen
Universitätsstr. 27, Hagen, Germany
herwig.unger@fernuni-hagen.de

ABSTRACT

This paper presents a basic, new concept for decentralised web search which addresses major shortcomings of current web search engines. Its methods are characterised by their local working principles, making it possible to employ them on diverse hardware configurations. The concept's implementation in form of an interactive, librarian-inspired peer-to-peer software client, called 'WebEngine', is elaborated on in detail. This software extends and interconnects common web servers creating and forming a decentralised web search system on top of the existing web structure while –for the first time– combining modern text analysis techniques with novel and efficient search functions as well as approaches for the semantically induced P2P-network construction and its flexible management. This way, an alternative, fully integrated and powerful web search engine under the motto 'The Web is its own search engine.' is built making the web searchable without any central authority.

CCS Concepts

•**Information systems** → **Web search engines; Peer-to-peer retrieval;**

Keywords

WebEngine, web search engine, P2P-system, co-occurrence graph, librarian of the web

1. INTRODUCTION

It is definitely a great merit of the World Wide Web (WWW, web) to make the world's largest collection of documents of any kind in digital form easily available at any time and any place without respect to the number of copies needed. It can therefore be considered to be the knowledge base or library of mankind in the age of information technology. Google (https://www.google.com/), as the world's largest and most popular web search engine with its main role to connect information and the place/address where it can be found, might be the most effective, currently available information manager.

Even so, in the authors' opinion, Google and Co. are just the mechanistic, brute force answer to the problem of effectively managing the complexity of the WWW and handling its big data volumes. As already discussed e.g. in [1], a copy of the web is established by crawling it and indexing web content in big reverse index files containing for each occurring word a list of files in which they appear. Complex algorithms try to find those documents that contain all words of a given query and closely related ones. Since (simply chosen) keywords/query terms appear in millions of (potentially) matching documents, a relevance ranking mechanism must avoid that all of these documents are touched and presented in advance to the user (see Fig. 1).

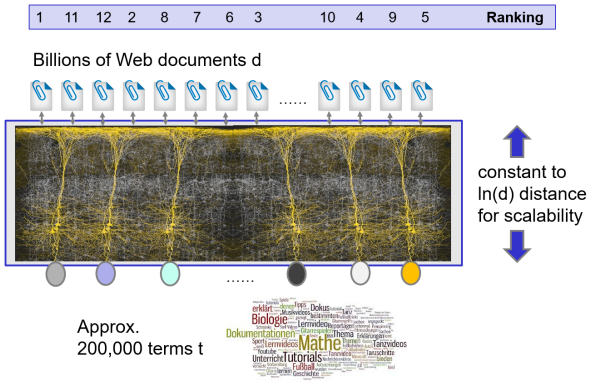

Figure 1: The dimensionality problem of the WWW

In the ranking process, the content quality and relative position of a document in the web graph as well as the graph's linking structure are taken into account as important factors. In addition to organic search results obtained from this process, advertisements are often presented next to them which are related to the current query or are derived from personalisation efforts and detected user's interests. In both cases, web search engines do not take into account probably existing (local) user knowledge. To a certain extent, this procedure follows a top-down approach as this filtering is applied on the complete index for each incoming query in order to return a ranked list of links to matching documents. The top-ranked documents in this list are generally useful.

However, due to the sheer amount of data to handle and in contrast to the bottom-up approach of using a library catalogue or asking a librarian or human expert in their role as

NLPIR 2018, September 7–9, 2018, Bangkok, Thailand

ACM ISBN 978-1-4503-6551-2/18/09…$15.00

DOI: https://doi.org/10.1145/3278293.3278294

active intermediaries between resources and users for guidance in order to find (more) actually relevant documents, the search engines' approach is less likely to return useful (links to) documents at an instant when the search subject's terminology is not fully known in advance. Furthermore, the web search results are not topically grouped, a service that is usually inherently provided by a library catalogue. Therefore, conducting research using web search engines means having to manually inspect and evaluate the returned results, even though the presented content snippets provide a first indication of their relevance.

From the technical point of view, the search engine's architecture carrying out the mentioned procedures has some disadvantages, too: In order to generate a refindable connection between contents and their locations and to be able to present recent results, the crawlers must frequently download any reachable web pages and thus make and store (multiple) copies of the entire web in their indexes. To achieve a high coverage and actuality (web results should cover contents that have been updated in the last 24 hours with a probability of at least 80 percent), they cause avoidable network load. Problems get bigger, once the hidden web (deep web) is considered besides regularly accessible HTML (Hypertext Markup Language) pages (surface web), too. Modern web topology models (like the evolving web graph model [2]) emanate from the fact that there are linear as well as exponential growth components, if the overall number of websites is considered. The constant crawling of these components causes especially high network load and their archiving needs a huge amount of storage capacity, too.

This brute-force method of making the web searchable is therefore characterised by a significant overhead for maintaining and updating the indexes. Furthermore, the used technical components like servers and databases are potential targets for cyber-attacks and pose a threat for the system's safety and security as well as for data protection.

As it is necessary to properly address all these problems of centralised web search engines, this paper introduces a new concept along with its technical solutions and infrastructures for future, decentralised web search relying on peer-to-peer (P2P) technology. In order to show that P2P-technology is actually useful in information retrieval tasks, the following section discusses several approaches in this regard first before deriving the respective requirements for this concept.

2. P2P INFORMATION RETRIEVAL

When it comes to using P2P-systems for the purpose of information retrieval, one has to keep in mind that –in contrast to the use case of content delivery– replica of (relevant) documents often do not exist. Thus, it is needed to find the few peers that actually can provide them. Therefore, efficient routing mechanisms must be applied to forward a query to exactly those matching peers and to keep network traffic at a low level. Thus, a suitable network structure must be set up and adapted in a self-organising manner as well. At the same time, such a network must be easily maintainable.

Some of the most important results in the field of P2P information retrieval (P2PIR) have been obtained in the SemPIR projects [3]. Their goal was to make search for information easier in unstructured P2P-networks. In order to reach this goal, a self-organising semantic overlay network using content-depending structure building processes and intelligent query routing mechanisms has been built. The basic idea of the approach applied therein is that the distribution of knowledge in society has a major influence on the success of information search. A person looking for information will first selectively ask another person that might be able to fulfil her or his information need.

In 1967, Milgram [4] has shown that the paths of acquaintances connecting any two persons in a social network have an average length of six. These so-called small-world networks are characterised by a high clustering coefficient and a low average path length. Thus, the mentioned structure building processes conceived modify peer neighbourhood relations such that peers with similar contents will become (with a high probability) direct neighbours. Furthermore, a certain amount of long-distance links (intergroup connections) between peers with unrelated contents is generated. These two approaches are implemented in order to keep the number of hops needed (short paths) to route queries to matching peers and clusters thereof low. This method is further able to reduce the network load.

In order to create those neighbourhood relations, a so-called 'gossiping' method has been invented. To do so, each peer builds up its own compact semantic profile (following the vector space model) containing the k most important terms from its documents which is periodically propagated in the network in form of a special structure-building request, the gossiping-message. Receiving peers compare their own profiles with the propagated one and

1. put the requesting peer's ID and profile in the own neighbourhood list and
2. send the own profiles to the requesting peer if the profiles are similar to each other.

Also, the requesting peer can decide based on the received profiles which peers to add to its neighbourhood list. Incoming user queries (in the form of term vectors as well) are matched with the local profile (matching local documents will be instantly returned, too), the profiles of neighbouring peers and are forwarded to the best matching ones afterwards. This mechanism differs from the mentioned approach in real social networks: in the technical implementation, the partaking peers will actively route queries from remote peers. In real social networks, people will likely just give the requesting person some pointers on where to find other persons that have the required knowledge instead of forwarding the requests themselves.

In doing so, a semi-structured overlay P2P-network is built which comprises of clusters of semantically similar peers. Additionally, each peer maintains a cache of peers (egoistic links) that have returned useful answers before or have been successfully forwarded queries to matching peers. Furthermore, the network's structure is not fixed as it is subject to dynamic changes based on semantic and social aspects.

Further approaches to P2P-based search engines are available, too. *YaCy* (https://yacy.net/de/index.html) and *FAROO* (http://www.faroo.com/) are the most famous examples in this regard. However, although they aim at crawling and indexing the web in a distributed manner, their respective client-sided programs are installed and run on the users' computers. They are not integrated in web servers or web services and thus do not make inherent use of the web topology or semantic technologies for structure-building purposes. Especially, they do not take into account semantic relationships between documents.

3. CONCEPTUAL APPROACH

This section introduces the new concept for decentralised web search mentioned in the introduction. Beforehand, important requirements for its realisation are derived from the previous considerations.

3.1 Requirements

Based on and in continuation to the foregoing considerations and identified shortcomings of current web search engines, the authors argue that a new kind of decentralised search engine for the WWW should replace the outdated, more or less centralised *crawling-copying-indexing-searching* procedure with a scalable, energy-efficient and decentralised *learn-classify-divide-link&guide* method, that

1. employs a learning document grouping process based on a successive category determination and refinement (including mechanisms to match and join several categorisations/clusters of words (terms) and documents) using a dynamically growing or changing document collection (the local knowledge base),

2. is based on a fully decentralised, document management process that largely avoids the copying of documents and therefore conserves bandwidth,

3. allows for search inquiries that are classified/interpreted and forwarded by the same decision process that carries out the grouping of the respective target documents to be found,

4. ensures that the returned results are 100 percent recent,

5. returns personalised results based on a user's locally kept search history yet does not implicitly or explicitly propagate intimate or personal user details to any centralised authority and therefore respects data privacy and contributes to information security and

6. returns results without any commercial or other third-party influences or censorship.

Differing from the approaches cited above, the authors intend to build and maintain a P2P-network whose structures are directly formed by considering content- and context-depending aspects and by exploiting the web's explicit topology (links in web documents). This way, suitable paths between queries and matching documents can be found for any search processes. In the next subsections, the respective concept is presented.

3.2 Preliminary Considerations

In the doctrine of most teachers and based on the users' experience, the today's WWW is considered a client/server system in the classical sense. Web servers offer contents to view or download using the HTTP protocol while every web browser is the respective client accessing content from any server. Clicking on a hyperlink in a web content means to be forwarded to the content, whose address is given in the URL (Uniform Resource Locator) of the link. This process is usually referred to as surfing the web.

Nevertheless, any web server may be regarded as a peer, which is connected to and therefore known by other peers (of this kind) through the addresses stored in the links of the hosted web pages. In such a manner, the WWW can be regarded as a P2P-system (with quite slow dynamics with respect to the addition or removal of peers). However, this system only allows users to surf from web document to web document by following links. Also, these restricted peers lack client functionalities (e.g. communication protocols) offered by web browsers such that there is usually no bidirectional communication between those peers possible (simple forwarded HTTP requests neglected which are mostly initiated by web browsers in the first place).

Moreover, as an integrated search functionality in the WWW is missing so far (the aforementioned restrictions might have contributed to this situation), centralised web search engines have been devised and developed with all their many shortcomings discussed before. These problems will be inherently addressed by the subsequent implementation concept and its implementation.

3.3 Implementation Concept

In order to technically realise the mentioned decentralised web search engine, common web servers shall be significantly extended with the needed components for automatic text processing (clustering and classification of web documents and queries), for the processes of indexing and searching of web documents and for the P2P-network management. A general architecture of this concept can be seen in Fig. 2.

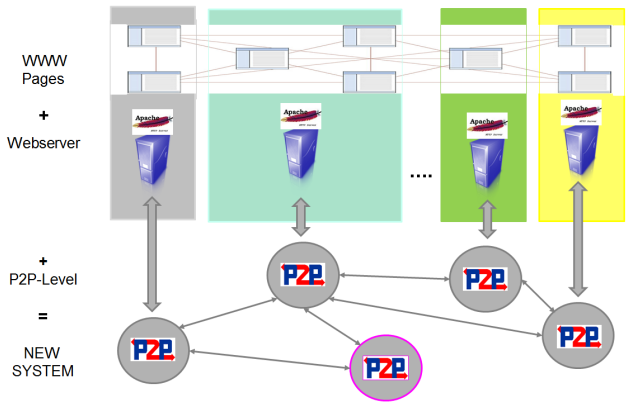

Figure 2: First concept of a decentralised, integrated web search system

The concept scheme shows that a P2P-component is attached to standard web server. Its peer neighbourhood is induced by the incoming and outgoing links of local web documents. By this means, a new, fully integrated and decentralised web search engine is created.

In the following implementation-specific elaborations, the P2P-client software 'WebEngine', which follows this concept, is described.

4. IMPLEMENTATION

As a prototype for this concept, the Java-based P2P-plugin 'WebEngine' for the popular Apache Tomcat (http://tomcat.apache.org/) servlet container and web server with a graphical user interface (GUI) for any standard web browser has been developed. Due to its integration with the web server, it uses the same runtime environment and may access the offered web pages and databases of the server with

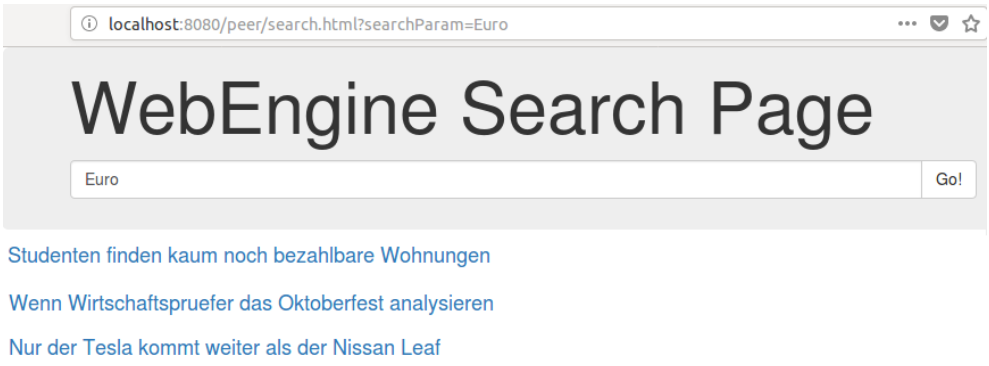

Figure 3: The graphical user interface of the WebEngine

all related meta-information. The following key points are addressed:

1. A connected, unstructured P2P-system is set up. Initially, the links contained in the locally hosted web pages of the Apache Tomcat server are used for this purpose. Other bootstrap mechanisms as known from [5] and the *PING/PONG*-protocol from *Gnutella* and other P2P-systems may be applied at a later time, too. Note, that
   - HTTP (HTTPS if possible) is used as frame protocol for any communication between the peers.
   - A fixed number of connections between the peers will be kept open (although more contactable neighbours are locally stored).
   - Furthermore, a time-to-live (TTL) counter is used to limit the number of forwarded messages.
2. All hosted web documents will be indexed in separate index files after applied stopword removal and stemming[1]. The index is updated after every change in one of the hosted web documents. It acts as a cache to answer incoming queries in a fast manner. However, it would be sufficient –as shown in [6]– to only store and use the centroid terms (single descriptive terms found in preferably large co-occurrence graphs to represent queries and whole texts alike) of local documents and the neighbouring peers' document centroids (their topical environment) in order to be able to route and answer queries properly.
3. The plug-in is able to provide a graphical user interface. In particular, a suitable search page for the requesting user (see Fig. 3) is generated.
4. Search results will be generated through a search in the local index files. Queries will also be sent via flooding to all opened connections to neighbouring peers. As they contain a unique message ID, incoming duplicates are discarded. As mentioned before, a TTL counter is applied to limit the number of hops of a query in the network. Responding peers will return their results directly to the originating peer. Multi-keyword search is possible as well.
5. Proliferation mechanisms in the plug-in are integrated to support the distribution of the WebEngine-software over the entire WWW. The P2P-client is able to recognise the peer software on other web servers addressed and offer the download of its own program, in case the peer is not running at the destination yet.

[1]In the first version of the P2P-plug-in, indexing is limited to nouns and names as the carriers of meaning.

The authors hope that the specified system rapidly changes the way of how documents are accessed, searched for and used in the WWW. The P2P-network may slowly grow besides the current WWW structures and make even use of centralised search engines when needed but may make them more and more obsolete. In this manner, the manipulation of search results through commercial influences will be greatly reduced.

4.1 The Software Components

In the previous section, the general architectural concept of the WebEngine has been outlined and depicted in Fig. 2. In a more detail manner, Fig. 4 shows the software components of the WebEngine-client. The blocks in the upper half of the scheme depict the functionality of currently running WebEngine prototype (basic implementation) presented so far with a particular storage facility to maintain the addresses of neighbouring peers.

The *Search Unit* is responsible to index local documents as well as to locally answer, forward and handle search requests issued by users. As mentioned above, in the WebEngine prototype, queries will be sent via flooding to all opened connections to neighbouring peers. However, a replacement of this basic procedure by a single-message, non-broadcasting, universal search protocol (USP), which forwards the search requests based on the centroid distance measure [7] to the target node(s), is currently being integrated.

Analogously, in its lower half, Fig. 4 depicts the components which are yet to be integrated and tested at the time of writing this paper (extension). As it is planned to turn the WebEngine into a powerful 'Librarian of the Web' [6], more sophisticated, centroid-based methods for the local management of document collections (their cataloguing, classification and topical clustering), the semantically induced query interpretation and targeted forwarding to neighbouring peers as well as the decentralised construction and maintenance of hierarchical library structures usually comprising a large number of connected peers have been devised and are carried out by these components. The following components are currently being integrated:

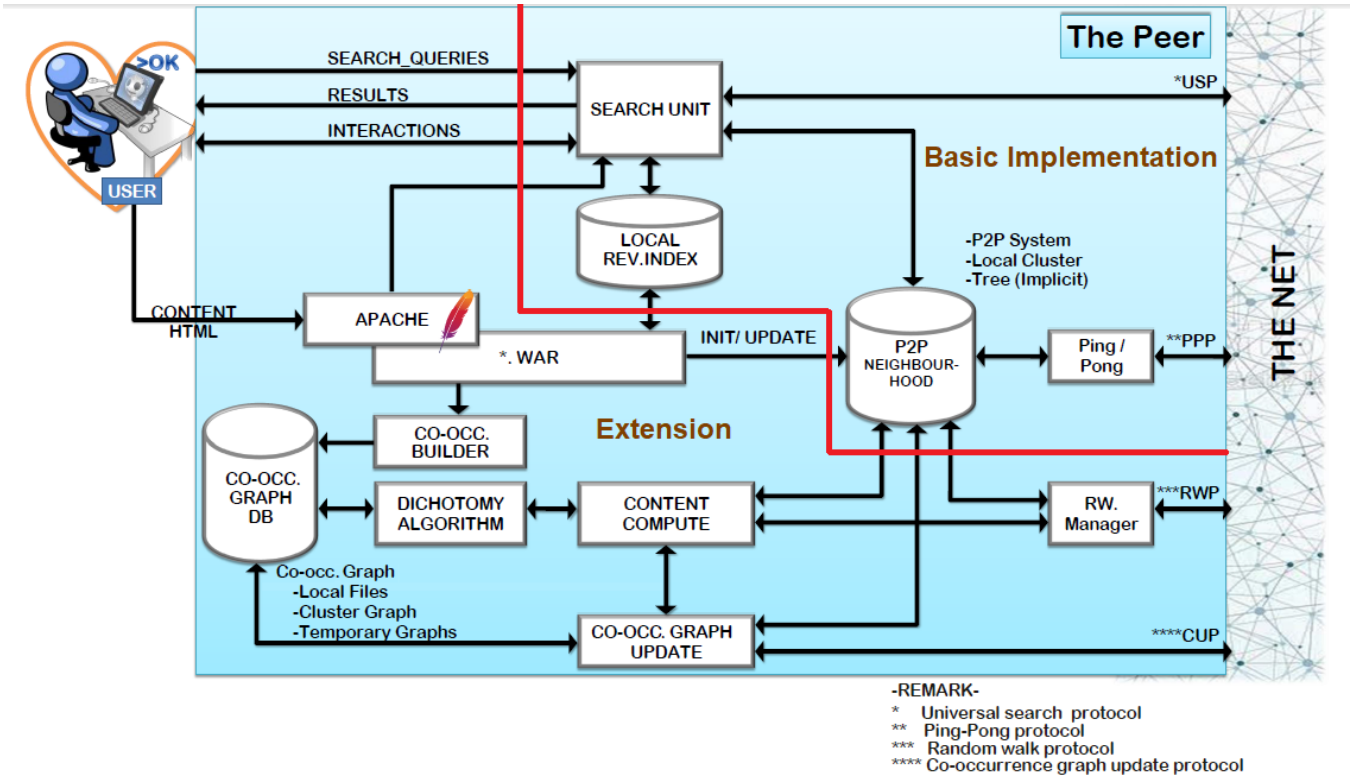

Figure 4: The WebEngine's internal structure

- As the decentralised library management is –in contrast to the top-down algorithm presented in [6]– carried out using random walkers, a particularly structured data unit circulating in the P2P-network, in the actual implementation of the WebEngine, a special *RW-Management* unit is added that carries out a special random walker protocol (RWP). Its working principles and methods are described in detail in [8]. Also, the mentioned USP will be additionally extended such that random walkers will not only be used to generate the tree-like library but to perform search operations in it as well. For this purpose, special random walkers with query data as their payload will be sent out. This payload is matched and exchanged with other random walkers in the network when appropriate until matching documents are found.

- The processing of random walker data is performed by the *Content Compute* unit, which needs to access the term co-occurrence graph databases (e.g. constructed and stored using the graph database system Neo4j (https://neo4j.com/)), which in turn contain the co-occurrence graph data of

  1. each web document offered by the local WWW server,
  2. the local term cluster which the node is responsible for,
  3. temporary operations of the random walkers to build or update the hierarchical library structures.

- The remaining units support various operations on co-occurrence graphs:
  - In order to construct co-occurrence graphs, a co-occurrence graph builder is implemented in a separate unit.
  - The needed document clustering (see [6]) is carried out by the *Dichotomy Algorithm* unit.
  - the exchange of co-occurrence information between peers is supported by the co-occurrence update protocol (CUP) controlled by a respective separate unit.

As the WebEngine makes heavy use of graph databases for the storage and retrieval of co-occurrences, their role is discussed in the next subsection, too.

4.2 Graph Databases

When taking a look at the WebEngine's architecture and functionalities from a technological point of view, it becomes obvious that it is necessary to be able to manage large graph structures efficiently and effectively. Graph database systems such as Neo4j (https://neo4j.com/) are specifically designed for this purpose. Also, they are well-suited to support graph-based text mining algorithms. This kind of databases is not only useful to solely store and query the herein discussed co-occurrence graphs with the help of its property graph model, nodes (terms) in co-occurrence graphs can be enriched with additional attributes such as the names of the documents they occur in as well as the number of their occurrences, too. Likewise, the co-occurrence significances can be persistently saved as edge attributes. Graph databases

are thus an immensely useful tool to realise the herein presented technical solutions. Therefore, the WebEngine makes especially use of embedded Neo4j graph databases for the storage, traversal and clustering of co-occurrence graphs and web documents.

5. CONCLUSION

This paper presented the concept of a novel, decentralised web search engine as well as its P2P-based implementation, called the 'WebEngine'. Its features and software components have been elaborated on in detail. Specifically, it utilises existing web technologies such as web servers and links in web documents to create a decentralised and fully integrated web search system. In doing so, the structure of the generated P2P-network is directly induced by exploiting the web's explicit topology. As an extension to the well-known Apache Tomcat servlet container, the WebEngine is easy to install and maintain for administrators. Internally, it makes use of graph-based text analysis techniques. Therefore, the graph database system Neo4j has been used for the persistent storage and retrieval of terms, links between documents as well as the determination of their semantic relations. As such, a decentralised web search system is created that –for the first time– combines state-of-the-art text analysis techniques with novel, effective and efficient search functions as well as methods for the semantically oriented P2P-network construction and management. The basic implementation of the WebEngine is currently being greatly enhanced by numerous additions which will turn it into a modern 'Librarian of the Web'. The WebEngine will be made publicly available in late 2018.

6. REFERENCES

[1] R. Eberhardt, M. M. Kubek, and H. Unger. Why google isn¨t the future. Really not. In *Autonomous Systems 2015*, pages 268–281. VDI Verlag, 2015.

[2] A. Broder et al. Graph Structure in the Web: Experiments and Models. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, pages 309–320, Amsterdam, The Netherlands, 2000.

[3] Website of the DFG-project 'Search for text documents in large distributed systems'. http://gepris.dfg.de/gepris/projekt/5419460, 2009.

[4] S. Milgram. The Small World Problem. In *Psychology Today*, 2:60–67, 1967.

[5] P. Kropf, J. Plaice, and H. Unger. Towards a Web Operating System. In *Proc. of the World Conference of the WWW, Internet and Intranet (WebNet'97)*, pages 994–995, Toronto (CA), 1997.

[6] M. M. Kubek and H. Unger. Towards a librarian of the web. In *Proc. of the 2nd International Conference on Communication and Information Processing*, ICCIP '16, pages 70–78, New York, NY, USA, ACM, 2016.

[7] M. M. Kubek and H. Unger. Centroid terms as text representatives. In *Proc. of the 2016 ACM Symposium on Document Engineering*, DocEng '16, pages 99–102, New York, NY, USA, ACM, 2016.

[8] M. M. Kubek and H. Unger. A Concept Supporting Resilient, Fault-tolerant and Decentralised Search. In *Autonomous Systems 2017*, Fortschritt-Berichte VDI. 10(857):20–31, VDI-Verlag Düsseldorf, 2017.