Similarity Search Techniques in Exploratory Search: A Review

Mohammed Mahdi
Institute of Informatics and Computing in Energy
College of Computer Science & Information Technology, Universiti Tenaga Nasional
Selangor, Malaysia
najah.mahdi@uniten.edu.my

Abdul Rahim Ahmad
College of Computer Science & Information Technology, Universiti Tenaga Nasional
Selangor, Malaysia
abdrahim@uniten.edu.my

Roslan Ismail
Institute of Informatics and Computing in Energy
College of Computer Science & Information Technology, Universiti Tenaga Nasional
Selangor, Malaysia
roslan@uniten.edu.my

Abstract— **The past decade has seen a dramatic increase in the amount of data captured and made available to users for research. This increase amplifies the difficulties users' face in finding the data most relevant to their information needs. The document similarity search is one of the most important topics in the field of information science, especially due to the popularity of the internet applications that deal with unstructured data sources such as World Wide Web. Efficiency of similarity search has become one of the most important issues. A typical example of similarity search is in multimedia databases that manage objects without structure, i.e. images, fingerprints or audio clips. Here similarity search is involved in retrieving the most similar fingerprint to a given one. Another example is in text retrieval which is present in many systems, from simple text editors (finding words similar to a given one to correct edition errors) to big search engines (retrieving relevant documents for a given query). This study explores the use of similarity search for text data in the form of a brief review using the interface provided as a service after content-based searches has been performed. The findings will give us ideas as to how to incorporate similarity searches within others search engine architecture.**

Keywords— Information Retrieval; Search Engine; Exploratory Search; Similarity Search.

I. INTRODUCTION

Information search is increasingly important in many applications; particularly in web based distributed environments where users and applications can share various contents. With the evolution of communication technologies, the number of accessible unstructured information repositories has increased rapidly[1]. Traditional search techniques based on immediate lookup and exact matching of keywords are no longer sufficient for many emerging applications such as image retrieval and data mining and exploratory search. New models based on the unifying concepts of similarity searching or proximity searching are needed for discovering and retrieving similar or close objects to a given query[2]. Similarity search is one of the effective search techniques to find out informative data from a large amount of information, which retrieves a set of similar data to the query. Rapid advances in computer and network technologies have led to explosive growth of unstructured or semi-structured text documents such as Web pages, e-mails, news articles, and research papers. Such a vast amount of text documents has drawn much attention to text mining on large text databases[3].

This paper primarily investigates notions of Similarity search and further highlights the importance of user search intention. In this process, this study reviewed traditional search system and observed search behaviors. The contribution of this paper is in the proposal of a strategy to encapsulate the various search behaviors for modeling the user intention in the exploratory search. This paper is organized as follows. This section introduces the theme of the paper. Introduction to similarity search is given next in section II. Section III delve in more detail into content based-search. Section IV and V discusses about search in metric space and exploratory search respectively. Conclusions are finally given in section VI.

II. SIMILARITY SEARCH

Similarity search has generated a great deal of interest lately because of the abundance of applications that checked for similar text or images and documents or image copy detection. These applications characterize objects as feature vectors in very high-dimensional spaces[4,5]. User submits query object to a search engine and the search engine returns objects that are similar to the query object. The degree of similarity between two objects is measured by some distance function between their feature vectors. The search is performed by returning the objects that are nearest to the query object in high-dimensional spaces.

Similarity search is important in many applications, such as content-based retrieval, exploratory data analysis, predictive modelling and data mining. The basic problem can be stated as follows: given a set of objects, find the most similar ones to a given query object. For example, one may be interested in retrieving the most similar images to a given one from a database or in identifying those stocks whose prices evolved similarly to a specific one over the last year[6]. Retrieval of these objects is based on "similarity" rather than on "exactness". The main research in this area is focused on the development of methods that can efficiently support similarity search, since common applications involve very large amounts of data[7].

Similarity search is a function of an exploratory search system. It is to present similar sets of results and discover new ones. The search system should thus be capable of retrieving

978-1-5386-5457-6/18/$31.00 ©2018 IEEE

sets of results, which are not necessarily directly accessible with full text search. This is even more important because data is currently multimedia in nature.

This research intends to study the retrieval methods, which focus on the whole content of documents. We review the existing approximate similarity search techniques and propose a classification schema that is able to characterize them according to different aspects.

III. Content Based Search

In content-based searches, query will consist of documents, rather than keywords. The results are a set of "similar" or "related" documents. As the name implies, the search is performed over the whole content of the documents. One difficulty in matching multimedia documents is that they may have no apparent structure. Also, the number of variables to consider may be very large. In order to make sense of the sheer amount of data, the information must be condensed into meaningful pieces of information known as "features" which are engineered for all types of applications. Fig 1 presents the Content Based Search process.

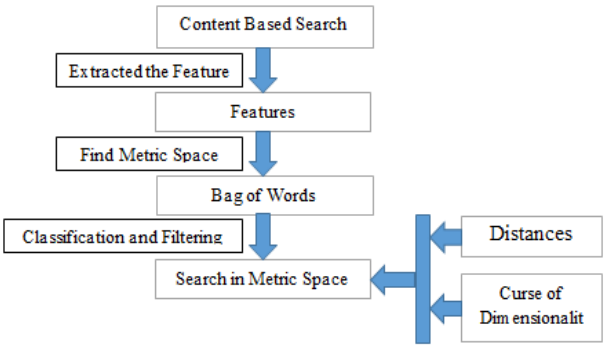


Figure 1: Content Based Search System

This brief overview outlines the main factors of a content-based search. First, the relevant features must be chosen and extracted. A metric must then be properly chosen. Finally, an algorithm must be crafted to perform the matching efficiently. Next, we detail the engineering or creation of features.

A. Features

Features are engineered to capture some aspects or characteristics of the data. For example, for text, the words and their order within each document are of interest, whereas for images, the color usage, texture composition, or shape is important. Below, we explore the simple bag-of-words model used for textual documents[8].

B. Bag-of-words

Perhaps the simplest type of feature for textual items is the bag-of-words model. The bag-of-words model regards text as an unordered collection of words. This is often an incorrect assumption for documents written in natural languages, such as English, in which the word order and grammar are important[9,10]. Nevertheless, the bag-of-words model is commonly used in document classification, such as for filtering out unwanted emails.

The method presented below is called "Bayesian filtering"[11]. We represent an email as a bag-of-words or binary vector $W(w_1, \ldots, w_n)$ where $w_i = 1$ if word i is present in the email; otherwise, $w_i=0$. Given the vector w of an email, the probability of the email is in c is denoted as $P(c|W)$, where c is either spam or ham (not spam). Using Bayes' theorem, we can write this probability as:

$$P(c|w) = \frac{p(c)\,.\,p(W|c)}{\sum_{k\in\{S,H\}} p(k).p(W|k)} \quad (1)$$

Where S and H denote spam and ham, respectively. It would be impractical to directly estimate $P(c|W)$. Instead, we make the "naïve" assumption that the words in w are conditionally independent given the class c. Under this assumption we can write:

$$P(c|W) = \frac{p(c).\prod_{i=1}^{n} P(w_i|c)}{\sum_{k\in} p(k).\prod_{i=1}^{n} P(w_i|k)} \quad (2)$$

The probabilities $P(w_i|c)$ and $p(c)$ can be estimated from a training set. The probability $P(w_i|c)$ is estimated as the frequency of the word i given the class (spam or ham) within the training set. The priors $p(S)$ and $p(H)$ can be respectively estimated as the number of emails in spam and ham in the training set. We can classify an email as spam if the following ratio is greater than a chosen threshold:

$$\frac{P(c = S|W)}{P(c = H|W)} > \lambda \quad (3)$$

Bayesian filtering above has proven successful at filtering out spam, forming the backbone of various commercial spam filtering programs, such as SpamAssassin[12] or DSPAM[13]. It has some disadvantages, however. For example, a spammer may send emails with an attached list of legitimate keywords to trick the algorithm. Another trick would include replacing some letters of highly "spammy" words or sending the email as an image. These lead us to consider features that are more sophisticated.

IV. Search in Metric Space

Once features are obtained, a distance measure known as "nearest neighbor search" can be used to match the documents[14]. Nearest neighbor search however, becomes more challenging when the dimension of the feature space increases, an issue referred to as the "curse of dimensionality"[15,16]. In order to circumvent this, we propose two approaches. The first consists of increasing the efficiency of nearest neighbor search; the second consists of collating the features into textual fingerprints.

A. Distances

There are numerous distance measures for the various feature spaces. The simplest metric between real value feature vectors of a fixed dimension n are those induced by the Minkowski norm L_p:

$$d_p(v, w) = l_p(w - v) \quad (4)$$

Where v and w are two vectors in $\mathbb{R}^n$ and l_p is the Minkowski norm defined as:

$$w \mapsto L_p(w) = |w|_p = \left(\sum_{i=0}^{n} |w_i|^p\right)^{1/p} \quad (5)$$

where L_1 is the "Manhattan norm," (in reference to the distance a car in Manhattan island must travel in a rectangular street grid to reach point b from point a), L_2 is the Euclidean

distance between two points and L_∞ is the maximum norm or Chebyshev norm and corresponds to the maximum of the components.

Not all measures of similarity must be strictly induced by a norm. One well known example is cosine similarity, which measures the angle between two vectors:

$$d_{\cos}(v, w) = 1 - \frac{v \cdot w}{L_2(v) L_2(w)} \quad (6)$$

Numerous other measures of similarity exist, depending on the nature of the feature space. For example, if the feature vectors are probability vectors, that is, vectors with non-negative components that add up to one, the Kullback-Leibler divergence may be a good measure. Kullback-Leibler divergence measures the difference between two probability distributions v and w. More precisely, it measures the expected number of extra bits required to code samples from v when using a code based on w, rather than using a code based on v.

$$d_{KL}(v, w) = \sum_i v_i \log \frac{v_i}{w_i} \quad (7)$$

However, $d_K L$ is not a metric, as it is not symmetric. Also, $d_K L$ is not finite, as it tends to infinity as one of the components of w tend to zero. This would be problematic if the feature vectors had arbitrarily small components. Thus, the Jensen-Shannon is usually preferred, as follows:

$$d_{JS}(v, w) = (d_{KL}(v, m) + d_{KL}(w, m))/2 \quad (8)$$

where m = (v+w)/2. The Jensen-Shannon divergence may be considered the metric and finite version of the Kullback-Leibler divergence. After devising a feature space with a distance measure, retrieval can be reduced to matching nearby objects.

B. Curse of Dimensionality

The curse of dimensionality is a notorious problem in machine learning and data mining when feature dimension is high, thus it is useful to reduce the dimension using dimension reduction strategies. Among the dimension reduction techniques are Singular Value Decomposition (SVD), Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Perceptually Important Points (PIP) and Piecewise Aggregate Approximation (PAA), etc.[17], These dimension reduction strategies are specifically developed for time series analysis in general. However they are non-actionable in recent-biased analysis for streaming time series. This is because Traditional Time series analysis algorithms take recent data and old data as equally important. In recent-biased analysis, recent data are much more important than old ones[18]. Most existing dimension reduction techniques process time series in a batch way (i.e.) the whole time series needs to be examined again on the arrival of new data. So they are very inefficient for processing online data streams.

Curse of dimensionality causes the simple nearest neighbor search approach to be ill-defined. Nevertheless, researchers have been studying methods to make nearest neighbor search more practical and efficient in high dimensions, as will be discussed below.

C. Efficient Nearest Neighbor Search

Nearest-neighbor search is inherently expensive, especially when there are a large number of dimensions.

- The search space can grow exponentially with the number of dimensions.
- There is simply no way to build an index on disk such that all nearest neighbors to any query point are physically adjacent on disk.

Fortunately, in many cases it is sufficient to perform an approximate search that returns many but not all nearest neighbors[19]. (a feature vector is often an approximate characterization of an object, so we are already dealing with approximations anyway)[20]. For instance, in content-based image retrieval[21] and document copy detection[22] it is usually acceptable to miss a small fraction of the target objects. Thus, it is not necessary to pay the high price of an exact search.

Real feature space, or that used to discriminate between objects, may have lower dimensionality than apparent data space. In this case, the dimension of the data space can be reduced with Principal Component Analysis (PCA)[23]. However, PCA becomes impractical as the size of the dataset increases. Furthermore, adding new documents to the index may require re-calculation of PCA each time on the whole dataset.

A second class of methods consists of partitioning the feature space into a tree structure. R-tree is an example of these methods[24], and many variants have been developed since. However, partitioning the feature space may increase in difficulty as the dimensionality increases, as the data is much more likely to be sparsely populated. Weber, et al. [25] Showed that after a certain number of dimensions, the R-tree like methods are not more effective than a simple linear scan of the data.

A third approach, which can be used in combination with the first two, consists of storing each dimension of the feature space separately. This technique is often referred to as "vertical decomposition". Each dimension is treated separately, depending on its significance. One example of the use of this technique is Igrid[26], which computes a similarity score based on the dimensions of the points close to the query point.

A fourth approach to increasing the practicality of nearest neighbor consists of relaxing the constraint of finding exact matches. This approach, referred as approximate nearest neighbor search, consists, in its simplest form, of only matching the best neighbors which are some $\in$ away to the query point[27]. More complicated procedures have since been devised (FLANN)[28].

D. Fingerprints

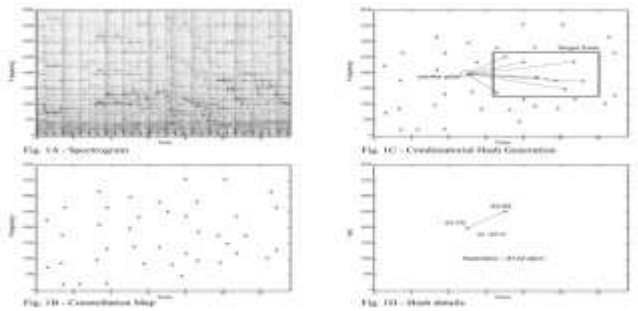
A more efficient method of comparing documents consists of comparing their respective "fingerprints", That is, to build a representation of the document which allows for fast and reliable retrieval within a database record[20]. A fingerprint should be small but robust. Changes in the object which are not perceptible to humans do not necessarily change the fingerprint. Fingerprints attempt to identify documents based on perception and are therefore very different from hash functions such as Message Digest Algorithm 5 (MD5) or

Cyclic Redundancy Check (CRC), in which a change in a single bit leads to completely different documents.

There are numerous fingerprint types for various applications. A popular example is the use of audio fingerprints by the program Shazam to identify songs playing in a real world environment[29]. Fig 2, from the Shazam article, summarizes the method employed.

- Spectrogram of the song is generated (A).
- The peaks of intensity of the spectrogram are extracted (B). The peaks of intensity are called the "constellation map" of the song. It reduces a complicated spectrogram into a sparse set of coordinates.
- Target zone for each point (anchored point) in the constellation map is defined (C).
- Each anchored point and the corresponding target zones are hashed and indexed (D).

Figure 2: Creating Fingerprint of an Audio File with Shazam (courtesy of Shazam)



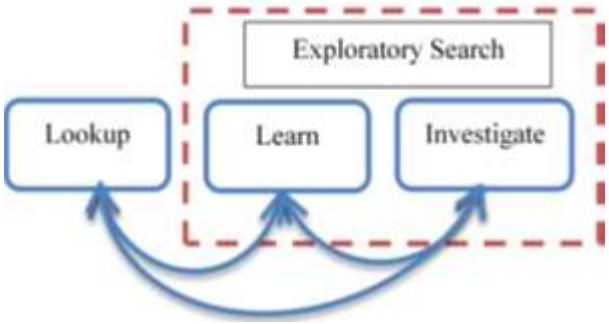
Wang [29] reported that this type of combinatorial hashing yields a speed improvement of 10,000 times for only 10 times more storage with a minimal loss of probability signal detection. The procedure exposed above is applied to create a large database index of audio files. The search consists of matching the hashes of the queried audio file with the hashes found in the index.

Fingerprinting is used both for searching exact matches and to remove "close" duplicates in a large data corpus. For example, Sinitsyn [30] described how to use audio fingerprints to perform background self-cleaning from duplicates in data management middleware. However, fingerprinting extracts certain features and collates them to permit efficient retrieval. However, for exploratory searching, retrieving sets of similar items would be more desirable than merely returning near exact matches.

V. Exploratory Search and Search Engines

Exploratory searches encompass a combination of queries and collection browsing while collating information[31,32]. It adopts a sequence of multiple query and search sessions to allow the user searching for information to expand the knowledge of the task currently undertaken while in the process of identifying the useful information.

During the searching process, a series of complex cognitive tasks is taking place which would encourage the information seeker to learn, explore and acquire intellectual skills[33]. Searchers who are conducting exploratory searches are generally not familiar with the domain in which their goal lies, and as such in most cases would need to learn about a topic to have an understanding of how their goals can be achieved. This can, in turn, result in them being either unsure about the goals they seek, unsure about the method in which to achieve those goals, either through the technology or process available, or even a combination of both. Fig 3 presents the exploratory research process.



Fact retrieval	Knowledge Acquisition	Accretion Analysis
Know-item Search	Comprehension/ Interpretation	Exclusion/Negation
Navigation	Comparison	Synthesis
Transaction	Aggregation/ Integration	Evaluation
Verification	Socialize	Discovery
Question Answering		Planning/Forecasting
		Transformation

Figure 3: Exploratory Search [34]

VI. Exploratory Search Systems

Exploratory Search Systems (ESSs) gain advantages from new capabilities induced by the latest developments in technology as well as updated and more natural interface paradigms to promote interaction with search systems. Examples of ESSs include information visualization systems, document clustering and browsing systems, as well as intelligent content summarization systems.

According to White and Roth [31], the main goals of ESSs are to facilitate learning and investigation during the searching process and to guide the users in exploring uncharted territories. In exploratory search, users are exposed to various collections of information; therefore, it is important for an exploratory search system to summarize the information in appropriate categories in order to facilitate user exploration.

Via ESS, users are able to enhance the rate at which they gain information, determine appropriate navigational paths, and understand the encountered information. In addition, through interface features such as dynamic queries in ESS[35], users are able to identify the immediate impacts of their decisions made. Similarity search in exploratory search should be improved in order to allow for queries made of items not necessarily present in the document collection.

VII. CONCLUSION

This paper reviews and summarizes a number of selected literatures on Similarity Search Techniques. This study went beyond full text search to present content-based search, in which retrieval is performed on the whole content of the objects themselves by proper extraction of the characteristics

or features of these objects. However, the issues of high dimensionality in feature spaces can hinder the search. This paper reviewed two of the simplest features reduction employed in text applications. To perform retrieval, the distances in reduced feature space are measured and similarities are obtained using the nearest neighbor approach. In general, this paper presented a unified view of the different approaches proposed in literature. Finally, this study discussed the important problem of scheduling, presenting original results on optimality of schedules.

ACKNOWLEDGMENT

This research was sponsored and supported under the Universiti Tenaga Nasional (UNITEN) internal grant no J510050783 (2018). Many thanks to the Innovation & Research Management Center (iRMC), UNITEN who provided their assistance and expertise during the research.

REFERENCES

[1] A. Figueroa and G. Neumann, "Context-aware semantic classification of search queries for browsing community questionâ€“answering archives," Knowledge-Based Systems, vol. 96, pp. 1-13, 2016.

[2] K. Collins-Thompson, S. Y. Rieh, C. C. Haynes, and R. Syed, "Assessing Learning Outcomes in Web Search: A Comparison of Tasks and Query Strategies," in Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval, 2016, pp. 163-172.

[3] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and semantic issues of text mining," ACM Sigmod Record, vol. 36, pp. 23-34, 2007.

[4] M. Chen, Y. Li, Z. Zhang, C.-H. Hsu, and S. Wang, "Real-time, large-scale duplicate image detection method based on multi-feature fusion," Journal of Real-Time Image Processing, vol. 13, pp. 557-570, 2017.

[5] Y. Ke, R. Sukthankar, L. Huston, Y. Ke, and R. Sukthankar, "Efficient near-duplicate detection and sub-image retrieval," in ACM Multimedia, 2004, p. 5.

[6] V. T. Lee, A. Mazumdar, C. C. del Mundo, A. Alaghi, L. Ceze, and M. Oskin, "POSTER: Application-Driven Near-Data Processing for Similarity Search," in Parallel Architectures and Compilation Techniques (PACT), 2017 26th International Conference on, 2017, pp. 132-133.

[7] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," IEEE Transactions on Image Processing, vol. 27, pp. 949-963, 2018.

[8] T. Deselaers, D. Keysers, and H. Ney, "Features for image retrieval: an experimental comparison," Information retrieval, vol. 11, pp. 77-107, 2008.

[9] M. Vazirgiannis, "Graph of Words: Boosting Text Mining Tasks with Graphs," in Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 1181-1181.

[10] H.-J. Yoon, L. Roberts, and G. Tourassi, "Automated histologic grading from free-text pathology reports using graph-of-words features and machine learning," in Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on, 2017, pp. 369-372.

[11] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in Learning for Text Categorization: Papers from the 1998 workshop, 1998, pp. 98-105.

[12] J. Mason, "Filtering spam with spamassassin," in HEANet Annual Conference, 2002, p. 103.

[13] Zdziarski, "The DSPAM project," 2004.

[14] L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen, "Efficient Metric Indexing for Similarity Search and Similarity Joins," IEEE Transactions on Knowledge and Data Engineering, vol. 29, pp. 556-571, 2017.

[15] R. da Silva Villaca, R. Pasquini, L. B. de Paula, and M. F. Magalhaes, "Hcube: A server-centric data center structure for similarity search," in Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on, 2013, pp. 82-89.

[16] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, 2002, pp. 380-388.

[17] T. N. Phan, J. Küng, and T. K. Dang, "eHSim: an efficient hybrid similarity search with MapReduce," in Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, 2016, pp. 422-429.

[18] J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques: Elsevier, 2011.

[19] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold, "Clustering for approximate similarity search in high-dimensional spaces," IEEE Transactions on Knowledge and Data Engineering, vol. 14, pp. 792-808, 2002.

[20] J. Wenyu and Y. Rongshan, "High-performance, very low power content-based search engine," in Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on, 2013, pp. 1-6.

[21] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg, "Terabyte-scale image similarity search: experience and best practice," in Big Data, 2013 IEEE International Conference on, 2013, pp. 674-682.

[22] R. Negrel, D. Picard, and P.-H. Gosselin, "Compact tensor based image representation for similarity search," in Image Processing (ICIP), 2012 19th IEEE International Conference on, 2012, pp. 2425-2428.

[23] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," Chemometrics and intelligent laboratory systems, vol. 2, pp. 37-52, 1987.

[24] A. Guttman, R-trees: a dynamic index structure for spatial searching vol. 14: ACM, 1984.

[25] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in VLDB, 1998, pp. 194-205.

[26] C. C. Aggarwal and P. S. Yu, "The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space," in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 119-129.

[27] S. A. Nene and S. K. Nayar, "A simple algorithm for nearest neighbor search in high dimensions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pp. 989-1003, 1997.

[28] M. Muja and D. G. Lowe, "Flann, fast library for approximate nearest neighbors," in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009.

[29] A. Wang, "An Industrial Strength Audio Search Algorithm," in ISMIR, 2003, pp. 7-13.

[30] A. Sinitsyn, "Duplicate song detection using audio fingerprinting for consumer electronics devices," in 2006 IEEE International Symposium on Consumer Electronics, 2007, pp. 1-6.

[31] R. W. White and R. A. Roth, "Exploratory search: beyond the query-response paradigm (Synthesis lectures on information concepts, retrieval & services)," Morgan and Claypool Publishers, vol. 3, 2009.

[32] M. N. Mahdi, A. R. Ahmad, and R. Ismail, "Paradigm Extension of Faceted Search Techniques A Review," Journal of Telecommunication, Electronic and Computer Engineering (JTEC), vol. 9, pp. 149-153, 2017.

[33] M. Dörk, "Visualization for Search: Exploring Complex and Dynamic Information Spaces," Doctoral dissertation, University of Calgary, 2012.

[34] G. Marchionini, "Exploratory search: from finding to understanding," Communications of the ACM, vol. 49, pp. 41-46, 2006.

[35] B. Shneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, Designing the user interface: strategies for effective human-computer interaction: Pearson, 2016.