PROTECTED INDEXING AND QUERYING OF LARGE SETS OF TEXTUAL DATA

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority under 35 U.S.C. § 119 to NonProvisional Application No. 14/642,886, filed March 10, 2015, Provisional Application No. 62/058,759, filed October 2, 2014 and Provisional Application No. 62/110,644, filed February 2, 2015, the entire disclosures of which are herein expressly incorporated by reference.

BACKGROUND OF THE INVENTION

[0002] Exemplary embodiments of the present invention are directed to searching documents. Searching for the occurrence of one or more words can currently be performed within documents using productivity software (e.g., a word processor, spreadsheet editor, presentation software, etc.), across documents using an operating system or computer system-wide searching application, as well as on the internet using a search engine. These search techniques typically require at least the search term(s) comprising the search query to be in plaintext form, and may require both the search term(s) as well as the document(s) being searched to be in plaintext form.

[0003] The rise of cloud-based data storage has renewed interest in protocols allowing private searching of encrypted or sensitive data in a public or untrusted environment. These protocols are known as Private Set Intersection (PSI) protocols, which are also referred to as Oblivious Keyword Search, or Private Information Retrieval. These techniques provide a blind search functionality to protect the plaintext of the original query from the database provider.

[0004] Many recent approaches to PSI involve protocols with a strict set of security assumptions. For example, many approaches require that the person making the query obtain no information about the provider's database beyond the results of the intersection between

the query and the contents of the database. This can involve using an independent third party that restricts the set of legitimate queries to achieve these strict security requirements.

SUMMARY OF THE INVENTION

[0005] Although conventional PSI protocols achieve their intended purpose of protecting both the query and the data stored in the database, it has been recognized that this high level of security, as well as the attendant protocol complexity, is not necessary in all situations. There may be situations where the data of the database to be searched is publically available but there is still a need to protect the data of the query itself. For example, a person may suspect that their Social Security Number has been compromised so they would want to search for it using an internet search engine but the person may be concerned about inputting their Social Security Number into an internet search engine.

[0006] Accordingly, exemplary embodiments of the present invention are directed to identifying matches between a query and one or more documents in database, wherein the query is protected but the database can be relatively insecure because the underlying data in the database is already publically available.

[0007] An exemplary method in accordance with the present invention involves obtaining, by a processor, a plurality of artifacts in plaintext form. For ease of explanation the term artifact will be used to refer to the document(s) or other data being searched. The processor generates fingerprints for each of the plurality of artifacts by generating shingles from text within each of the plurality of artifacts so that there at least one character overlap between adjacent shingles and cryptographically hashing the shingles to generate a plurality of artifact fingerprints. The artifact fingerprints are stored in a database. The processor also receives at least one fingerprint and determines whether the received at least one fingerprint matches any of the artifact fingerprints stored in the database. An indication of an artifact containing a matched fingerprint can be output.

[0008] In certain embodiments the at least one fingerprint is a query that is received from a querying party in an encrypted form. In this case the at least one fingerprint of the query should preferably have better security than the fingerprints of the artifacts. This can be achieved by using less character overlap between adjacent shingles created from the query than adjacent shingles created from the artifacts. It can also be achieved by removing common, easily guessed words from the query prior to generation of the query fingerprint. Further protection can be achieved by identifying artificially common fingerprints in the artifact fingerprints and removing these from the set of query fingerprints before the query fingerprints are provided by the querying party to the database provider.

[0009] In other embodiments the at least one fingerprint can be selected from the artifact fingerprints and the match indicates a similarity between different artifacts within the database, which allows identification of copied or derivative texts.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Figure 1 is a flow diagram of the overall process of exemplary embodiments of the present invention;

[0011] Figure 2 is a block diagram of an exemplary system for generating and matching fingerprints in accordance with the present invention;

[0012] Figure 3 is a flow diagram of an exemplary process for fingerprinting artifacts in accordance with the present invention;

[0013] Figure 4 is a block diagram of an exemplary process for fingerprinting artifacts in accordance with the present invention;

[0014] Figures 5A and 5B are block diagrams of exemplary techniques for storing fingerprinted artifacts in a database in accordance with the present invention;

[0015] Figure 6 is a flow diagram of an exemplary process for fingerprinting queries in accordance with the present invention;

[0016] Figure 7 is a block diagram of an exemplary process for fingerprinting queries in accordance with the present invention;

[0017] Figure 8 is a flow diagram of an exemplary process for identifying whether a queried fingerprint is contained within fingerprinted artifacts in accordance with the present invention;

[0018] Figure 9A-9C are headers of three identical e-mail messages saved in different locations; and

[0019] Figure 10 is a histogram of cosine distances between stored e-mail artifacts and between a test e-mail artifact and other stored e-mail artifacts.

DETAILED DESCRIPTION

[0020] Figure 1 is a flow diagram of the overall process of exemplary embodiments of the present invention. As illustrated in Figure 1, the overall process involves collecting and fingerprinting artifacts (step 300), fingerprinting a query (step 600), and then comparing the fingerprinted query against the fingerprinted artifacts to identify any matches (step 800). The details of each of these steps will be described below.

[0021] Figure 2 is a block diagram of an exemplary system for generating and matching fingerprints in accordance with the present invention. The system 200 can comprise one or more computers that include a processor 205 coupled to memory 210, input 215, and output 220. The disclosed processes can be performed by processor 205 executing computer code stored in memory 210. The processor 205 can be any type of processor, including a microprocessor, field programmable gate array (FPGA), and/or an application specific integrated circuit (ASIC). Memory 210 can be any type of non-transitory memory. In addition to storing computer code for executing the processes described herein, memory 210 can also store the generated fingerprints. Alternatively or additionally, a separate storage medium can store the generated fingerprints. For example, the fingerprints can be stored in a

distributed key value store or non-relational, distributed database, both of which are networks of computers connected together, and often running a distributed file system, such as the Hadoop Distributed File System (HDFS), to store information in a distributed and replicated manner. Such systems allow storage of far more information than can be stored on a single computer and enable efficient retrieval and linear scaling of required computational resources as data size grows. Typical distributed key stores include Google BigTable, Apache Accumulo, and Apache HBase.

[0022] Distributed key stores store data in simple structures consisting of a row, a column family, a column qualifier, a timestamp, and a value. In Apache HBase column families are fixed throughout a table, whereas column qualifiers can be different for each entry. Moreover, distributed key stores often have limited query capabilities compared to traditional Relationship Database Management Systems (RDBMS), which can accept Structured Query Language (SQL). In contrast, distributed key value stores typically only allow querying by row ID. Although this is more limiting, the distributed key value stores more efficiently handle large volumes of data distributed across multiple machines compared to RDBMS.

[0023] Input 215 provides mechanisms for controlling the disclosed processes, including, for example, a keyboard, mouse, trackball, trackpad, touchscreen, etc. Further, input 215 can include a connection to an external storage device for providing artifacts, such as an external hard drive or flash storage memory, as well as a network connection. Output 220 can include a display, printer, and/or the like. Additionally, output 220 can include a network connection for notifying a querying party of a match between a query fingerprint and an artifact, such as by electronic mail, posting on a website or webpage, a text message, and/or the like.

[0024] As will be described in more detail below, in certain embodiments different entities will generate the fingerprints for the artifacts and the queries. For example, a first entity may operate the database containing the fingerprinted artifacts and a second entity may want to

determine whether the database contains any matches for data that the second entity does not want revealed to the first entity. In this case the first entity can have a system such as that illustrated in Figure 2 in order to collect artifacts, generate fingerprints from the artifacts and perform a fingerprint comparison, while the second entity can have a system such as that illustrated in Figure 2 in order to generate the fingerprinted query that is then provided in the protected form to the first entity.

[0025] Turning now to Figure 3, initially a set of artifacts are received via input 215 and are passed to processor 205 (step 305). The set of artifacts can be collected in any manner, including using an internet spider that crawls across the web to collect web pages and/or documents accessible via the internet. The artifact collection can be non-targeted where the spider collects web pages from random websites or it can be targeted to websites most likely to contain data that would be subject to a protected query. In this latter case the web spider can crawl across websites known to traffic in sensitive data. The term artifact is not intended to be limited to web pages and documents collected from the internet but instead can include any type of data or information that may be subjected to a search.

[0026] Processor 205 then windows the received artifacts into shingles of length n with a sequential overlap of $k_A=n-1$ (step 310). An example of this is illustrated in Figure 4 where a collection of data 405 is windowed into a set of shingles 410. In this example the shingles have a length $n=7$, and accordingly the overlap is 6. This sequential overlap can be seen by the first shingle including "BORER I" and the second shingle including the last 6 characters of the first shingle and the first character following the end of the first shingle, i.e., "ORER IN". The shingles can also be subjected to additional, optional processing, such as replacing all whitespace by a single space and removing all punctuation. It should be recognized that the particular length n of the shingles does not need to be 7; it could be a different value depending on the desired security of the hash and granularity of the possible queries.

[0027] Next processor 205 cryptographically hashes each of the shingles (step 315). Prior to cryptographic hashing, a sequence of random characters can be appended to each shingle (commonly referred to as "salting") so long as the same sequence is added to each shingle of artifacts in the database as well as the shingles of the query. In the example illustrated in Figure 4 the shingles are salted and then hashed using a Secure Hash Algorithm 512 (SHA-512). It should be recognized that this is only one exemplary type of cryptographic algorithm that can be employed with the present invention. One skilled in the art will recognize that the particular cryptographic algorithm chosen is a balance between the desired protection for the query (because the query will be subjected to the same cryptographic algorithm) and the amount of processing power, processing time, and storage space. Accordingly, the present invention can be implemented with any cryptographic algorithm to provide the desired protection to the underlying information.

[0028] Finally, processor 205 stores the cryptographically hashed shingles in a database 210 along with a unique artifact identifier (step 320). The unique artifact identifier allows the artifact to be quickly identified when there is a match between a fingerprinted shingle stored in the database and a fingerprinted query shingle. The unique artifact identifier can be, for example, a hash value computed for the entire artifact or any other unique identifier.

[0029] As discussed above in connection with Figure 2, the fingerprints can be stored in a distributed key value store. Exemplary embodiments of the present invention use unique properties of distributed key value stores to construct a unique schema to store the fingerprints. Figures 5A and 5B illustrate two examples of ways to store fingerprints in a row of a distributed key value store using unique schema. In the schema illustrated in Figure 5A the fingerprint is used as the row, column family is set to "document", the artifact identifier is the column qualifier, the time stamp represents when the data was entered into the row, and the value field is not used in the schema to save storage space. This schema allows a direct

query of the distributed key value store using the fingerprints from the query to search for similar fingerprints even though the store does not contain any of the original content or index items consisting of literal extracted pieces of the original content.

[0030] Although exemplary embodiments are being described in connection with determining whether a query appears in one or more artifacts, the present invention can also be employed to identify any derivatives of artifacts by finding latent patterns and similarities between artifacts. In this case the different artifacts will be switched in and out as the fingerprinted query that is compared to the remaining fingerprinted artifacts. To simplify this process an inverted table, such as that illustrated in Figure 5B can be employed in which the artifact identifier is stored as the row and the fingerprint is stored as the column qualifier. This secondary table allows construction of extremely efficient and highly parallel algorithms that directly measure cosine distance between two documents using an intermediate query system, such as Apache Pig. This derivative identification can be performed using a distributed graph database, such as Titan or Hama.

[0031] Now that a database of fingerprinted artifacts has been created the system is ready to receive queries. Again, the queries of the present invention are designed to be secure, which can be achieved using the method illustrated in Figure 6. Further, as also described above, in order to maintain this security a different entity can create the fingerprinted query than the entity that created the database of the fingerprinted artifacts.

[0032] Initially, the querying party's system 200 receives, via input 215, a plaintext query, which is provided to processor 205 (step 605). Processor 205 then windows the plaintext query into shingles of length n with a sequential overlap of $k_Q<n-1$ (step 610). An example of this is illustrated in Figure 7 where a collection of data 705 of the query is windowed into a set of shingles 710. The shingles are then compared to a stop word list and matching shingles are removed 715 (step 615).

[0033] It should be noted that the length of the shingles n constructed from the artifacts and the plaintext query are the same but the overlap k is different. There is less overlap for the plaintext query because this improves the security of the query, whereas the security of the artifacts is less of a concern because it is constructed from publicly-available data. Removing easily-guessable stop words from the query further improves the security of the query. For example, if one of the shingles of the artifacts contains an easily-guessable plaintext or "crib", (e.g., "and" or "the") the shingle containing the word will contain significantly less information than the naïve A^n possibilities and will be more easily susceptible to a brute force attack of the hash value. If there is a large overlap between adjacent shingles, once one of these cribs has been confirmed as existing within the artifact fingerprint the remaining artifact shingles are subject to an unzipping attack. Specifically, each shingle on either side of the confirmed crib is only one character shifted from the previous one so the key space required to explore for a brute force reversal of the next fingerprint is merely the size of the alphabet, making the brute force reversal for each subsequent shingle trivial for most alphabets.

[0034] In contrast, the query fingerprint of the present invention is significantly more secure by using less overlap between adjacent shingles and performing the stop word filtering. If there is no overlap between the shingles then each shingle is completely independent of the other shingles. Thus, if one shingle is correctly guessed the security of the remaining shingles would remain intact. Accordingly, when the highest security is desired the shingle overlap of the query will be $k_Q = 0$.

[0035] Returning again to Figure 6, the shingles remaining after the stop word filtering are then formed into fingerprints using a cryptographic hashing algorithm (step 620). If the artifact fingerprints were salted, the query fingerprints would also be salted, using the same salt, prior to cryptographic hashing. The security of the query can be further improved by removing fingerprints having hash values that commonly appear in the artifacts (step 625).

Because languages tend to follow Zipf's Law in which the frequency of any token is directly proportional to its rank in the language's token frequency table, the collection of fingerprints should follow a power law distribution in fingerprint frequency. Thus, it is possible to flag and remove fingerprints that significantly deviate from the power law distribution, which removes fingerprints that may represent easily guessable plaintext.

[0036] The comparison and removal of step 625 is performed at the querying entity. Specifically, the database provider provides a list of artificially common fingerprint values to the querying entity. The artificially common fingerprint values can be identified by performing the cosine distance calculation (described in more detail below) between each fingerprint in the artifact fingerprints and any cosine distances that are above a threshold value are identified as the artificially common fingerprints that are part of the set provided to the querying entity.

[0037] Now that the artifact and query fingerprints have been created these can be compared using the method of Figure 8. Exemplary embodiments of the present invention treat the fingerprints as uniformly-weighted vector spaces and the similarity between fingerprints is calculated as the cosine distance between two fingerprints. Thus, the cosine distance is used in the present invention as a similarity score indicating the likelihood of a match between fingerprints. In contrast to other weighting schemes the present invention does not presume any knowledge of the fingerprints. The uniform weighting provides a relatively simple way to compute the distance between two fingerprints. If S is defined as the set of query fingerprints, T as the set of fingerprints for a particular artifact, and k_Q and k_A as the overlap parameters between the query and artifact fingerprints, the cosine distance in this uniform weighting scheme is defined as:

$$c = \frac{|S \cap T|}{\sqrt{|S|}\sqrt{|T|}}$$

[0038] The distance score c can be used for both fingerprint similarity as well as to rank database search results. Specifically, the value c between a set of query fingerprints and the fingerprints of a particular artifact may be large enough to be significant on its own, whereas in some situations the relative values of c between the set of query fingerprints and a plurality of different sets of artifact fingerprints ranked according to lowest distance may provide a more useful indication of significance of any particular artifact relative to the query. Since the amount of overlap between adjacent tiles k is different for query fingerprint and the artifact fingerprints the distance score c will never reach a unity value.

[0039] Because the present invention uses different overlaps for the query and artifacts (i.e., $k_Q \neq k_A$) then the cosine distance will only vary between 0 and $\sqrt{\frac{n-k_A}{n-k_Q}}$. Using the example above where $n = 7$, $k_A = 6$, and $k_Q = 0$, the scores would only vary between 0 and 0.3778. Accordingly, using these parameters the closer a cosine distance score is to 0.3778 the more likely it is that there is a match. If desired the all scores can be adjusted by a factor proportional to n and the difference between k_Q and k_A so that the scores are distributed between 0 and 1, which is the range of values when the same overlap is used for the query and artifacts (i.e., $k_Q = k_A$). Regardless of the particular weighting scheme, so long as k_Q and k_A remain constant throughout the system all scores will be adjusted equally across database comparisons so that relative scores between the query and artifacts remain useful.

[0040] Accordingly, first a median artifact fingerprint similarity score c_A is calculated within the database using the cosine distance calculation (step 805). Next, a similarity score between the set of query fingerprints and the artifact fingerprints c_{QA} is calculated using the cosine distance calculation (step 810), which produces a set of similarity values c for each comparison between the set of query fingerprints and each fingerprint stored in the database. The median artifact similarity score c_A is compared to the similarity score between the query

fingerprint and the artifact fingerprints c_{QA} (step 815) to determine whether the difference between these scores is greater than or equal to a threshold value (step 820). The threshold used here is designed to balance occurrences of false matches against missed matches. Thus, one skilled in the art can set the threshold to the desired balance between these two by, for example, setting it to the top 1% quantile value, thus eliminating 99% of the scores as false matches.

[0041] When the difference between these similarity scores is less than the threshold ("No" path out decision step 820), then there are no matches (step 825), and an indication of this is output. If, however, the difference between the similarity scores is greater than or equal to the threshold ("Yes" path out of decision step 820), then a match has been found and an indication of the plaintext document containing the queried information is output (step 830). Specifically, referring again to Figure 5A, the artifact fingerprint having a similarity score greater than or equal to the threshold is searched in the database using the row value and the artifact fingerprint having the matching row value is examined to identify the artifact ID in the column qualifier field.

[0042] In order to appreciate the operation of the present invention, as well as how to interpret the similarity scores, two examples of implementations of the present invention will now be presented.

[0043] **Example 1**

[0044] The first implementation involved using Charles Dickens' Tale of Two Cities and Miguel de Cervantes' Don Quixote as two separate artifacts. These two texts were separately fingerprinted according using the method described above in connection with Figure 3. The following query was then fingerprinted using the method of Figure 6:

> It was the best of times
> it was the worst of times
> it was the age of wisdom

it was the age of foolishness

[0045] It will be recognized that this query is a direct quote taken from the beginning of Tale of Two Cities. The cosine distance calculation described above was performed separately for the fingerprinted query and the two fingerprinted artifacts, which resulted in similarity scores of Don Quixote = 0.0016376817020082647; Tale of Two Cities =0.008143011160437408. In the abstract the similarity score for Tale of Two Cities appears quite small. This is due to the fact that although the query is a direct quote from Tale of Two Cities, it only appears once in the entire text. However, when this similarity score is compared to the similarity score for Tale of Two Cities is five times greater than for Don Quixote. An appropriately selected threshold for step 820 would indicate that a match for the query was found in Tale of Two Cities but not in Don Quixote.

[0046] **Example 2**

[0047] The second implementation involved a subset of emails made public by the Federal Energy Regulatory Commission (FERC) during its investigation of Enron Corporation. These e-mails were stored on a Hadoop cluster of 10 nodes having a total of 80 TB of storage. The subset used as the artifacts consisted of 3,000 e-mails from the larger set made available by FERC. These e-mails were fingerprinted as the artifacts as a Map-Reduce job using the Apache Pig platform. The query was a single e-mail from this dataset that had two near duplicates, the difference being the e-mails were stored in different folders of the user's email account. Figure 9A illustrates the header of the e-mail used as the query and Figures 9B and 9C illustrate headers of the two near duplicates. The different storage locations of these e-mails are described in the "X-Folder:" field. The bodies of these e-mails were identical.

[0048] The median similarity score calculated in step 805 was 0.04326. The fingerprinted query was then compared to each of the 3,000 individual fingerprinted artifact e-mails and

the two near duplicates produced similarity scores of 0.9896 and 0.9874. These similarity scores are illustrated in Figure 10, where it can be seen that there is a significant difference between the median similarity score and the similarity scores of the two near duplicates. Thus, the similarity scores of the two near duplicates indicate a high likelihood that these e-mails contain the queried information, which is also indicated by the substantial difference between these similarity scores and the median similarity score. Accordingly, these two separate evaluations of the similarity scores provide a high degree of confidence that the two near duplicates contain the queried information.

[0049] As will be appreciated from the discussion above, exemplary embodiments of the present invention provide advantageous techniques for identifying matches between a query and a set of artifacts in a way that ensures the security of the plaintext query so that the operator of the database containing the artifacts typically cannot determine the original plaintext query. Instead, a fingerprinted artifact containing some or all of the query is identified and provided to the querying party in the artifact's original plaintext form. The querying party can then take any further action, such as contacting the operator of the website that contained the queried information to request that the information be removed, and/or contacting the authorities to take appropriate action with respect to the public posting of private, confidential information.

[0050] The foregoing disclosure has been set forth merely to illustrate the invention and is not intended to be limiting. Since modifications of the disclosed embodiments incorporating the spirit and substance of the invention may occur to persons skilled in the art, the invention should be construed to include everything within the scope of the appended claims and equivalents thereof.

CLAIMS

1. A method, comprising:

obtaining, by a processor, a plurality of artifacts in plaintext form;

generating, by the processor, fingerprints for each of the plurality of artifacts by

generating shingles from text within each of the plurality of artifacts so that there at least one character overlap between adjacent shingles; and

cryptographically hashing the shingles to generate a plurality of artifact fingerprints;

storing the artifact fingerprints in a database;

receiving, by the processor from a querying party's system, at least one fingerprint that has been cryptographically hashed by the querying party's system;

determining, by the processor, whether the received at least one fingerprint matches any of the artifact fingerprints stored in the database; and

outputting, in plaintext form, the artifact containing a matched fingerprint.

2. The method of claim 1, wherein the processor and database are part of a first system controlled by a first entity, and wherein the first system is a different system than the querying party's system.

3. The method of claim 2, wherein the at least one fingerprint is generated by:

receiving, by a second processor of the querying party's system, a plaintext query;

generating, by the second processor, a plurality of query shingles from the plaintext query so that there is no character overlap between adjacent shingles;

cryptographically hashing, by the second processor, the plurality of query shingles to form a plurality of query fingerprints.

4. The method of claim 3, wherein prior to cryptographically hashing the plurality of query shingles by the second processor, the method further comprises:

comparing, by the second processor, the plurality of query shingles to a stop word list; and

removing, by the second processor, any shingle of the plurality of query shingles matching a word in the stop word list.

5. The method of claim 3, wherein prior to determining whether the received at least one fingerprint matches any of the artifact fingerprints stored in the database, the method comprises:

receiving, by the second processor, a set of artificially common fingerprints in the artifact fingerprints; and

removing, by the second processor, any fingerprints from the plurality of query fingerprints matching any of the fingerprints in the set of artificially common fingerprints.

6. The method of claim 3, wherein the shingles generated from the text within each of the plurality of artifacts and the shingles generated from the plaintext query are salted prior to the cryptographic hashing.

7. The method of claim 1, wherein the received at least one fingerprint is obtained from the stored artifact fingerprints, the method further comprising:

outputting an indication of an artifact containing the received at least one fingerprint, thereby indicating that the output artifacts contain at least some common text.

8. The method of claim 1, wherein the database storing the artifact fingerprints is a distributed key store.

9. The method of claim 1, wherein the plurality of artifacts are obtained using a web spider that collects websites as it crawls across the internet.

10. A method, comprising:

obtaining, by a processor, a plurality of artifacts in plaintext form;

generating, by the processor, fingerprints for each of the plurality of artifacts by

generating shingles from text within each of the plurality of artifacts so that there at least one character overlap between adjacent shingles; and

cryptographically hashing the shingles to generate a plurality of artifact fingerprints;

storing the artifact fingerprints in a database;

receiving, by the processor, at least one fingerprint;

determining, by the processor, whether the received at least one fingerprint matches any of the artifact fingerprints stored in the database; and

outputting an indication of an artifact containing a matched fingerprint,

wherein the determination of whether the received at least one fingerprint matches any of the stored artifact fingerprints comprises

generating, by the processor, a plurality of query vectors from the at least one cryptographically hashed shingle;

generating, by the processor, a plurality of artifact vectors from the artifact fingerprints;

performing, by the processor, a cosine distance calculation using the plurality of query vectors and the plurality of artifact vectors to generate a plurality of distance scores; and

selecting, by the processor, a vector of the plurality of artifact vectors having a lowest distance score.

11. The method of claim 10, further comprising:

determining, by the processor, a median cosine distance for the plurality of artifact vectors, wherein the median cosine distance is used to set a threshold against

which the lowest distance score is compared to determine whether the vector of the plurality of artifact vectors having the lowest distance score is a match of one or more of the query vectors.

12. A method, comprising:

obtaining, by a processor, a plurality of artifacts in plaintext form;

generating, by the processor, fingerprints for each of the plurality of artifacts by

generating shingles from text within each of the plurality of artifacts so that there at least one character overlap between adjacent shingles; and

cryptographically hashing the shingles to generate a plurality of artifact fingerprints;

storing the artifact fingerprints in a database;

receiving, by the processor, a plurality of query fingerprints, each of which is generated by cryptographically hashing shingles of a plaintext query, wherein a character overlap between shingles generated from the plaintext query is less than the at least one character overlap of the shingles generated from the artifacts;

determining, by the processor, whether any of the plurality of query fingerprints matches any of the artifact fingerprints stored in the database by performing a cosine distance calculation; and

outputting an indication of an artifact containing a matched fingerprint.

13. The method of claim 12, wherein

the cosine distance calculation is performed between each of the artifact fingerprints stored in the database to produce a median artifact similarity score,

the cosine distance calculation is performed between each of the plurality of query fingerprints and each of the artifact fingerprints to produce individual query fingerprint scores, and

a match between the any of the query fingerprints and any of the artifact fingerprints is determined based on both the median artifact similarity score and the individual query fingerprint similarity scores.

14. The method of claim 12, wherein there is no character overlap between shingles generated from the plaintext query.

15. The method of claim 12, wherein the processor and database are controlled by a first entity and the at least one fingerprint is received from a second entity, and wherein the plurality of query fingerprints are generated by:

receiving, by a second processor from the second entity, a plaintext query;

generating, by the second processor, a plurality of query shingles from the plaintext query so that there is no character overlap between adjacent shingles;

cryptographically hashing, by the second processor, the plurality of query shingles to form the plurality of query fingerprints.

16. The method of claim 15, wherein prior to cryptographically hashing the plurality of query shingles by the second processor, the method further comprises:

comparing, by the second processor, the plurality of query shingles to a stop word list; and

removing, by the second processor, any shingle of the plurality of query shingles matching a word in the stop word list.

17. The method of claim 15, wherein prior to determining whether any of the received query fingerprints match any of the artifact fingerprints stored in the database, the method comprises:

receiving, by the second processor, a set of artificially common fingerprints in the artifact fingerprints; and

removing, by the second processor, any fingerprints from the plurality of query fingerprints matching any of the fingerprints in the set of artificially common fingerprints.