COMPUTERIZED METHODS OF DATA COMPRESSION AND ANALYSIS

BACKGROUND

[0001] The field of the present invention relates to data compression and analysis. Specifically, the invention relates to lossless compression of text and other information that can be represented by symbols.

[0002] Previous text compression techniques are usable on only one file or document at a time. Such techniques do not scale easily to enormous data sets, *i.e.*, "Big Data," or where data is spread across many different containers. Also, previous techniques do not track the number of times a particular symbol or string of symbols appears in the uncompressed text. The number of times a particular symbol or string appears is valuable information that is useful for improving space savings, reducing processing time, and conducting contextual analysis.

SUMMARY

[0003] An example embodiment compresses symbolic information that is organized into a plurality of documents. Each of the uncompressed documents has a plurality of symbols. A document can be a string of symbols of any length (*e.g.*, a sentence, paragraph, or text file). A symbol can be any code that constitutes or carries information (*e.g.*, a letter, number, non-alphanumeric character, syllable, or word).

[0004] The example embodiment operates on an input document by identifying two adjacent symbols in the document as a symbol pair. The example embodiment stores each unique symbol pair in a compression dictionary. The compression dictionary associates each stored symbol pair with a respective replacement symbol. The example embodiment also maintains a count of the number of times each unique symbol pair appeared in the input document. The example embodiment produces a compressed output document by replacing symbol pairs with their associated replacement symbols if the count for the symbol pair exceeds a threshold

[0005] Some embodiments are programmed to repeat the compression process by using the output document as an input document for an additional pass. Certain embodiments recursively perform additional passes until no further replacements can be made (*i.e.*, in the final pass, the output document is identical to the input document). Symbol pairs identified in recursive passes can include replacement symbols from a previous pass.

[0006] Some embodiments are programmed to be applied to multiple documents. In fact, the compression ratio improves as the size of the uncompressed data increases. The larger the data set, the more likely it is that the count associated with any given symbol pair will exceed the threshold, which results in more replacements and higher compression. Compression is optimized, in some of these embodiments, when the process recurs on all the documents until no further replacements are possible.

[0007] Some embodiments increase space savings and decrease the lookup time needed to access the compression dictionary by using, as a replacement symbol, the address of the replaced symbol pair in the compression dictionary.

[0008] Some embodiments sort the entries in the compression dictionary based on the count of the number of times a symbol pair appeared. The sorting is done to place the most frequently appearing symbol pairs in more accessible memory locations than less frequently appearing symbol pair. Sorting decreases lookup times during the replacement phase.

[0009] Some embodiments use patterns of replacement symbols and the count of the number of times certain symbol pairs appeared to analyze content.

[0010] Objects and advantages pertaining to computerized methods of data compression and analysis may become apparent upon referring to the example embodiments illustrated in the drawings and disclosed in the following written description or appended claims.

[0011] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Fig. 1 is a visual representation of a text standardization process applied by an embodiment.

[0013] Fig. 2 is a visual representation of a compression dictionary used in the embodiment of Fig. 1.

[0014] Fig. 3 is a visual representation of operation of the embodiment of Fig. 2.

[0015] Fig. 4 is a flowchart that shows the operation of the embodiment of Fig. 2.

[0016] Fig. 5 is a chart showing results achieved by an experimental embodiment.

[0017] The embodiments depicted are shown schematically. The embodiments shown are examples only and should not be construed as limiting the scope of the present disclosure or appended claims.

DETAILED DESCRIPTION

[0018] Embodiments of the data compression and analysis technique can be programmed in computer code as a set of computer instructions. When running on a computer, the computer instructions control a computer processor and cause it to perform the functions described herein. The computer instructions (*e.g.*, computer software) can be stored in any computer memory accessible to the computer or encoded in computer hardware. Some embodiments are instantiated in special purpose computers or dedicated computer appliances. The documents on which the embodiments operate can be stored in the same or a different computer memory accessible to the computer system.

[0019] An example embodiment analyzes uncompressed text to identify adjacent pairs of symbols. The example embodiment replaces occurrences of a pair of symbols that appears more than a predefined number of times with an associated replacement symbol. Symbols can be alphanumeric characters,

discrete groups of alphanumeric characters such as syllables or words, or any code that constitutes and carries information.

[0020] In some embodiments, the original symbols in the uncompressed text are replaced with standard symbols during a preliminary phase. Fig. 1 is a representation of how an example embodiment standardizes uncompressed text. In certain circumstances, such as when there is a large variation in the lengths of the original symbols, standardization improves performance during subsequent operations.

[0021] The example embodiment of Fig. 1 identifies and counts all occurrences of unique original symbols and stores each unique original symbol. Each original symbol is stored in a standardization memory in association with a standard symbol. In Fig. 1, standardization memory 140 associates an original symbol (in column 160) with its respective standard symbol (in column 150) and a count of how many times each original symbol appears in the uncompressed text (in column 170).

[0022] To identify unique original symbols, some embodiments are programmed to interpret certain non-alphanumeric characters (*e.g.*, a space) as denoting a break between two symbols. The example embodiment interprets a space as a break between symbols yet retains the space in the standardized document. For example, document 111 is shown with spaces separating the standard symbols. However, other embodiments drop spaces and certain other non-alphanumeric characters entirely because the location of such characters can be restored or discerned from context. For example, a space between symbols can be inferred and reproduced based on the rules of the algorithm used to identify the original symbols. Likewise, the presence of certain non-alphanumeric characters can be inferred based on the grammatical rules of the language in which the text is written.

[0023] Some embodiments interpret certain non-alphanumeric characters as part of a symbol. The example embodiment of Fig. 1 identified text 105 as a symbol even though it contains an apostrophe. Other embodiments identify non-alphanumeric characters as symbols in their own right.

[0024] Some embodiments ignore capitalization while processing symbols. Such embodiments provide space savings by storing only a uniform-case version (e.g., lowercase version) of all original symbols. Such embodiments can restore the original text, including restoring capitalization, using grammatical and contextual rules regarding sentence structure and proper nouns.

[0025] Some embodiments interpret select symbols as signaling the end of a document. The example embodiment of Fig. 1 identifies the period at 107 as the separator between document 101 and document 102. Alternative embodiments separate text into documents based on carriage returns, paragraphs, cells, fields, records, rows, columns, pages, headings, tags from a markup language, information in a table of contents, or section or chapter markings. Any human-understandable or computer-readable markers or containers can be used to divide the text into documents. Other embodiments operate on documents that are separate computer files.

standardization memory if used). Some embodiments use an existing compression dictionary. Other embodiments create the compression dictionary during operation.

[0034] Fig. 3 is a representation of document 111 as it is operated on by the embodiment of Fig. 2. The example embodiment is programmed to repeat recursively until compression is complete, *e.g.*, until no more pairs of symbols meet the threshold for replacement. The example embodiment operates on document 111 to create output document 351. Numerals 321, 331, and 341 refer to intermediate results before passes of the process carried out by the recursive embodiment; they are called "documents" here for ease of reference but can be deleted or overwritten during the next pass. Documents 112-114 are likewise processed in the same batch as document 111 to produce respective documents 352-354. Documents 322-324, 332-334, and 342-344 represent intermediate results between passes.

[0035] During each pass the example embodiment makes through the documents, the example embodiment identifies and counts the occurrence of each unique symbol pair ("the cataloging phase") and replaces symbol pairs with a count exceeding the replacement threshold ("replacement phase"). The example embodiment performs the cataloging phase on all documents before turning to the replacement phase.

[0036] Alternative embodiments identify and replace symbol pairs in one operation by processing the text of each document in sequential order. In such alternative embodiments, a pass through a document involves comparing each sequential symbol pair to entries of the compression dictionary. If the comparison results in a match, the compared symbol pair's count is incremented, and if the replacement threshold is met, the symbol pair is replaced with a replacement symbol. If the comparison does not produce a match, the symbol pair is identified as unique and added to the compression dictionary. Because such alternative embodiments do not use replacement symbols until a symbol pair's count exceeds the threshold for replacement, such embodiments schedule replacement of earlier instances of a symbol pair once the symbol pair's count exceeds the replacement threshold.

[0037] In the identification or cataloging phase of a pass, the example embodiment identifies symbol pairs in an input document by pairing each symbol with the next adjacent symbol in the document until the last symbol in the document is reached. All symbols, except the first and last symbol, in the input document are therefore included in two symbol pairs: One symbol pair formed from pairing the symbol with the preceding symbol, and one symbol pair formed from pairing the symbol with the following symbol.

[0038] For example, when processing document 111, the example embodiment recognizes symbols 311 and 312 as a first pair and symbols 312 and 313 as a second pair. The example embodiment continues pairing symbols until it pairs symbol 318 and 319. The first symbol (symbol 311) and the last symbol (symbol 319) are each only paired with one other symbol (i.e., symbol 312 and symbol 318, respectively). All other symbols in the document are included in two symbol pairs.

documents are added to the set of documents, the counts for below-threshold symbol pairs could increase to the point where they meet the replacement threshold. Some alternative embodiments store below-threshold portions of the compression dictionary in a separate archive rather than the compression dictionary. The archive can be accessed when new documents are added.

[0050] In various embodiments, the replacement symbols can be associated with a symbol pair when the symbol pair is first identified or meets the replacement threshold. The example embodiment assigns replacement symbols after both sorting and truncation so that the most frequently used replacement symbols will have the shortest length and can be placed in the most accessible memory locations. In circumstances where a replacement symbol is the address of its associated symbol pair, each replacement symbol is associated with its symbol pair by virtue of the symbol pair's position in the sort order.

[0051] Alternative embodiments can assign replacement symbols any time before the replacement phase begins. Such alternative embodiments can sort the compression dictionary based on the counts or maintain the entries in their original order. After sorting, some embodiments delete the counts to save additional space.

[0052] During the replacement stage of a pass, the example embodiment creates a compressed output document by replacing symbol pairs in an input document with replacement symbols from compression dictionary 200. Alternative embodiments create a new output document distinct from the input document by inserting either the original symbols or replacement symbols in place of a pair of the original symbols. Such alternative embodiments later delete the input document. The example embodiment creates output document 321 (see Fig. 3) by overwriting symbol pairs in input document 111 with replacement symbols where appropriate.

[0053] The compression dictionary for the example embodiment only contains entries for symbol pairs that exceed the replacement threshold because entries with counts not exceeding the threshold were not retained. Thus, the example embodiment is programmed to replace the first symbol pair that matches an entry in the compression dictionary. When processing document 111, the example embodiment identifies the pair of symbols 311 and 312 (Fig. 3) and compares it to the entries of compression dictionary 200 (Fig. 2). The pair of symbols 311 and 312 is found in compression dictionary 200 (see entry 212). Because a match is found, the pair of symbols 311 and 312 is replaced with replacement symbol 325 in output document 321. The dictionary also contains the pair of symbols 312 and 313 (see 214), and the count exceeds the threshold for replacement (see 215). However, the pair of symbols 312 and 313 is not considered for replacement, because the example embodiment already replaced symbol 312.

[0054] Embodiments that do not truncate the compression dictionary must compare symbol pairs in a document to the compression dictionary and also check the count before replacement. Alternative embodiments analyze multiple possible replacements to determine which would maximize the number of total replacements or maximize the space savings. Other embodiments determine which pairs to replace

based on minimizing the length of the replacement pairs or the lookup time required to decompress the document. Still further alternative embodiments attempt to replace the first pair of symbols and, if no replacement is possible, skip over the pair that could be formed by combining the second symbol of the first pair with the next sequential symbol.

[0055] When the example embodiment cannot replace a symbol pair, it continues to consider the next symbol pair. For example, when the pair of symbols 315 and 316 is compared to the entries in the compression dictionary, no match is found. The example embodiment, therefore, moves to the next symbol pair (the pair of symbols 316 and 317). Embodiments that create a new output document rather than overwriting the symbols of the input document insert into the output document the first symbol of a symbol pair that cannot be replaced. If that last symbol of a symbol pair is the last symbol in a document, and the symbol pair does not match any of the entries in the compression dictionary, both symbols of the symbol pair must be inserted into the output document.

[0056] As described, the example embodiment is programmed to perform recursive passes until no further replacements are possible. Thus, the example embodiment begins a second pass by applying cataloging and replacement phases to document 321 to produce document 331. In the symbology of Fig. 3, symbol "R3P2" in document 331 refers to row 3 of the second pass. Symbol pairs identified in a second or subsequent pass can be two replacement symbols. See replacement symbol 335, which refers to the sequence of replacement symbol R4 (see 325) followed by replacement symbol R3 (see 326 in the first pass in document 321). Alternatively, symbol pairs identified in a later pass can be a combination of a replacement symbol and an uncompressed symbol. See replacement symbol 336, which refers to the sequence of uncompressed symbol S300 followed by replacement symbol R3. Compression dictionary 200 of Fig. 2, therefore, lists in row R3 of column 220 (P2) the sequential pair R4 followed by R3. Review of column 210 reveals that the sequence R4 followed by R3, in turn, denotes symbol S1 followed by S79 followed by S75 followed by S5. Thus, in this example, after the second pass (P2), the example embodiment has replaced the four-symbol "phrase" S1-S79-S75-S5 with the single symbol R3P2.

[0057] Sequential passes by the example embodiment result in additional columns 220 and 230 (Fig. 2) that store symbols pairs from each respective pass. In the example embodiment, each pass in which unique symbol pairs are discovered results in additional columns being added to compression dictionary 200 to store the newly identified symbol pairs and their respective counts (see columns 221 and 231). Fig. 2 presents symbols discovered in later passes in separate columns to illustrate the recursive nature of the embodiment.

[0058] The example embodiment only makes replacements from the column or section of compression dictionary 200 created during the instant pass. Storing replacement symbols in pass-specific sections reduces lookup times because the embodiment does not need to search through entries having replacement symbols that will not be used in the current pass.

[0059] Alternative embodiments store symbol pairs identified in different passes in a common column or memory area. Such embodiments can increase space savings because they use less complex replacement symbols. For example, an embodiment that stores all identified symbol pairs in the same column of the compression dictionary could use a replacement symbols that consists entirely of a row identifier. In any embodiment, close attention to the choice of symbology for the replacement symbol is capable of producing increased compression.

[0060] As shown in Fig. 3, a third pass performed on document 331 produces document 341, and a fourth pass performed on document 341 produces document 351. If the output of fourth pass (document 351) is the same as the output of the third pass (document 341), and assuming this remains true for all other documents that are being compressed concurrently with document 111, then the example embodiment stores the last output version of each document, and the compression process terminates. The example presented in Fig. 3 only shows four passes of the process applied to the example text. However, any number of passes can be performed. In some embodiments, the number of passes can be a fixed number. In other embodiments, a maximum number of passes can be set, but the process is programmed to terminate if no further replacements are made before the cap is reached.

[0061] As more passes are performed on a set of documents, the number of symbol pairs identified and the count of each identified symbol pair tends to decrease, hence the incremental size reduction provided by each additional pass decreases. However, each additional pass will consume additional processing resources. To balance the size reduction against processing resources, certain embodiments are programmed to terminate the process if the size reduction resulting from a given pass falls below a certain limit (*e.g.*, if a given pass produces less than a half of a percent size reduction).Other embodiments terminate the process based on a different factor, such as the number of new symbol pairs found being less than a fixed amount or the count of the most frequently occurring symbol pair falling below a certain level. In another example, an embodiment can be constructed to terminate the process if the count of the most frequently occurring symbol pair in a pass falls below a predetermined percentage of the count of the most frequently occurring symbol in the first pass.

[0062] Once the criterion for terminating the process has been met, some embodiments are programmed to replace any identified symbol pairs found in the pass that triggered the termination condition. Other embodiments terminate the process immediately upon recognizing the termination condition, without replacing symbol pairs found in that last pass. Still further embodiments terminate mid-pass if an insufficient number of symbol pairs are found after processing enough text to determine that the resultant space savings would fall below a predetermined amount. Any symbol pairs that are not replaced would not need to be stored in the compression dictionary or assigned a replacement symbol.

[0084] In the case of controlling access to a secure location, an example embodiment would first use the compression process, as described herein, to compress a set of documents, each representing one of a group of known and authorized persons, and to produce a compression dictionary. Later, when requesting access to the secure entity, an unknown person initiates a process by which measurements are taken of his or her face. The unknown person's face could be measured by a non-invasive method (e.g., calculating the required values from an image captured by one or more cameras). The required values are then arranged into a new document that is compressed using the same compression dictionary. The system compares the replacement symbols in the compressed version of the new document to those contained in the previously compressed documents and outputs the results (e.g., just that a match exists and access should be granted or more specifically the believed identity of the unknown person). In some embodiments, the system can have the unknown person purport to identify himself or herself by other means, such as through voice announcement or a card swipe, and check the document containing information derived from the captured image of the unknown person's face against only the document containing pre-stored information about the person that the unknown person claimed to be. In other embodiments, the system can compare the document containing the captured image of the person's face against all documents containing pre-stored information about all authorized persons.

[0085] In the facial recognition system, the symbology of facial measurements in the documents can be exact numerical measurements (*e.g.*, pupillary separation = 64.57 mm or eye color (RGB) = 161, 202, 241). Use of exact measurements can produce a high degree of certainty in any match that is found but is also likely to result in false negatives. Slight variances in the measurement conditions or equipment calibrations over time could result in misidentification. Thus, it is advisable that the symbology use approximate measurements or values representing ranges in which the measurements fall. The symbology of the ranges can include numerical values of the range, an average value for the range, or an arbitrary symbol that represents the range.

[0086] Each facial recognition document needs to have a sufficient number of measurements to avoid misidentification. Too few measurements results in a risk of falsely identifying a match. Another use of the facial recognition process is to investigate and improve facial recognition systems based on the process discussed herein, because by analyzing a sufficiently large set of documents representing known faces, the system can be used to determine how many measurements are required and the optimal order or groupings of measurements necessary to produce sufficient points of comparison.

[0087] Other practical applications of the process use recognized commonalities in compressed documents to analyze the content of the documents. Symbol pairs that are counted and associated with replacement symbols have an underlying meaning. Thus, thoughts, preferences, or other information about the author or subject of a document can be inferred from the symbol pairs. Data collected during the cataloging phase can provide a wealth of information of such information.