J. Vis. Commun. Image R. 21 (2010) 200–209

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Technologies and the development of the Automated Metadata Indexing and Analysis (AMIA) system

Pei-Ying Chiang [a,*], May-chen Kuo [a], Jessy Lee [a], C.-C. Jay Kuo [a], Todd Richmond [b], Milton Rosenberg [b], Jeff Lund [b], Kip Haynes [b], Lindsay Armstrong [b]

[a] Ming Hsieh Department of Electrical Engineering and Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2546, USA
[b] USC Institute for Creative Technologies, Marina Del Rey, CA 90292, USA

ARTICLE INFO

Article history:
Received 10 February 2009
Accepted 1 December 2009
Available online 21 December 2009

Keywords:
Text-based indexing
Multimedia search engine
Content-based retrieval
Multimedia databases
DAM
Digital asset management
Metadata extraction
Word segmentation

ABSTRACT

The Automated Metadata Indexing and Analysis (AMIA) project aims to provide an effective digital asset management (DAM) tool for large digital asset databases. We began with text-based indexing since it is still the most reliable approach as compared with other content-based media features. AMIA not only searches for the text of the file name, but also utilizes embedded information such as the metadata in Maya files. The AMIA system builds a linked map between all dependency files. We present an approach of preserving previously established metadata created by the old DAM tools, such as AlienBrain, and integrating them into the new system. Findings indicate that AMIA has significantly improved search performance comparing to previous DAM tools. Finally, the ongoing and future work in the AMIA project is described.

1. Introduction

Digital media has advanced rapidly in recent years. Due to the demand for digital assets from various industries (e.g., education, entertainment, manufacturing), digital files are being created daily and continue to grow in order of magnitude. Because of rapid file creation and editing, there are increased demands on systems to store and retrieve such files. Users benefit tremendously from effective reuse of existing resources.

The University of Southern California (USC) Institute for Creative Technologies (ICT) [1] currently holds over 400,000 files, ranging from simple photographs to complex Maya scene files with a multitude of referenced textures. Examples of typical assets used in current ICT projects are shown in Fig. 1. It is important to be able to set-up a customizable search system in order to quickly and accurately locate an asset, such as a prop that was stored in the archive the previous year. Props, characters, animation sequences, textures and other digital files are all housed in a central server, along with back-up copies. Since a large simulation can typically hold upwards of hundreds of characters, the ICT needed to discriminate between many different fields to determine the correct character, vehicle, or other file to be used or manipulated. Differences in characters' gender, ethnicity, job status, and other classifiable attributes, would all need to be searchable.

The ICT demands a Digital Asset Management (DAM) system that would utilize a structured database for effectively searching within a large repository of digital assets while meeting certain security requirements. This DAM system would also need a simple user interface to integrate into the ICT's gaming-studio style production pipeline. With increasingly complex DAM systems commercially available, the standard is constantly being revised. This, coupled with the ICT's unique needs for management software, meant that not only would the DAM systems differ greatly from one another, but that no single system would prove to be a perfect solution without custom modifications. An accurate and efficient method to index and retrieve these digital assets became a primary challenge.

Automated Metadata Indexing and Analysis (AMIA) is an ongoing project that includes a multimedia indexing/search system, such as "2D image/3D mesh retrieval", for a large art asset database. We began with text-based indexing, since it is still the most reliable approach compared to other content-based media features. We have made several contributions along this line. First, AMIA not only searches for the text of the file name, but also utilizes the embedded information, such as the metadata preserved in Maya files. The AMIA system builds a linked map among all dependency files. Second, we present a practical approach that

* Corresponding author.
E-mail address: peiyingc@usc.edu (P.-Y. Chiang).

1047-3203/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2009.12.001

Fig. 1. Digital assets developed by the ICT.

preserves established metadata from the existing DAM system (e.g., AlienBrain) and integrates them into our proposed DAM system. Third, most Digital Asset Management (DAM) systems require human input to provide metadata manually, which is not practical for a large database system. Automatic metadata extraction is essential to the management of a real-world database system. This practical issue is, however, rarely discussed in the literature. Fourth, it has been demonstrated that the AMIA system's search performance is significantly improved.

The field of digital content management has developed rapidly in the last 15 years and has reached a certain level of maturity. Many traditional academic papers focus on a small test data set; therefore, they do not address the practical issues as described in our work. However, there is growing interest in the management of large multimedia databases, since the properties of large databases may not be well reflected by those of small databases. Our current work is a part of this new research trend. Here, we attempt to share our experience and provide a detailed approach to managing a large database by developing its indexing and search system. This experience will be valuable to engineers and researchers who would like to design a practical management system for a large multimedia database.

This paper consists of three parts. The first part, including Sections 1 and 2, provides an overview on the AMIA project. The second part discusses text-based indexing and search in Section 3. The third part, including Sections 4 and 5, outlines ongoing and future research and development for the AMIA project.

2. Overview of the AMIA project

The AMIA project is being developing under an ICT contract managed by the United States Army Research, Development, and Engineering Command (RDECOM) Simulation and Training Technology Center (STTC).

In order to maximize the ICT's research and development efforts in game-based training and learning systems, various underlying technologies have been employed. While this approach has proven beneficial for individual projects, it has resulted in a myriad of file formats and versions for the corpus of required digital assets. The sheer number of files and diversity of applications involved creates a massive asset management challenge. In response to this need, the AMIA effort was tasked with designing a set of creative asset development standards and tools, along with a repository of visual and audio assets.

2.1. Project goal

Having conducted a decade of immersion research development, the ICT has now developed 11 projects, creating a wealth of raw art assets (e.g., images, audio, video, 3D models, etc.) with inconsistent data management standards. Without an effective digital asset management system, these valuable resources are difficult to find and retrieve. The ICT requires a great number of art assets to create an immersive environment, and the amount of digital assets for any particular project will grow as the project develops. Therefore, it is important to provide guidelines, processes, tools and support to ensure that future art assets will be reusable. The AMIA project aims to develop a system capable of automatically indexing existing multimedia databases for faster, efficient retrieval. This system could also benefit users searching for relevant assets for use in other applications. The research vectors will concentrate on a specific set of deliverables. The first set of deliverables concern outlining a method of approach and possible software solutions (which could include computer scripts, database query strings, applications, databases, etc.) focused on one or more of the following capabilities:

I. Automatically extracting metadata.
II. Adding metadata to existing assets and repositories.
III. Automatically adding metadata as new assets are created.
IV. Organizing assets based on metadata.

The goals are to make current assets more accessible and to ensure that current and future assets have a higher degree of reusability and organization. While any software solution will initially target the existing database repositories, it will also be designed with regard to portability and applicability in other systems. The second set of deliverables are reports that focus on the current landscape of image and file analysis, along with various pertinent research topics in the areas of metadata and digital asset management and organization. In addition, there will be an effort to take a multidisciplinary approach towards extending and expanding current metadata standards relating to visual and animation assets.

2.2. Project resources

The ICT creates necessary 2D, 3D, and audio software application files from military data. The ICT currently holds over 400,000 assets, including 3D scenes, 2D images, audio, video and game engine exports. These types of assets include props (e.g., buildings, vehicles, weapons), characters (e.g., human models, rigs, animations), environments (e.g., level layouts, skies, backgrounds) and respective reference materials and textures. These raw 2D, 3D, and audio files are the most useful and versatile files within the Digital Backlot (DB).

To organize the large amount of digital assets, the ICT adopted the commercial software AlienBrain [2] and Adobe Bridge [3] as DAM tools. Parts of the assets have been manually linked to associated metadata with AlienBrain and Adobe Bridge. These metadata, which describe the attributes of these digital assets, are valuable to preserve for indexing. In addition to the metadata edited with DAM tools, there are other useful text-based attributes of assets embedded in various ways.

The following list shows the resources that contribute to AMIA's text-based indexing process:

I. AlienBrain metadata
The metadata that had been manually edited with AlienBrain was stored in the metadata files, separate from the

assets. All metadata associated with the assets was stored under the same root directory and was written into a single ".amd" file. The metadata may indicate the file type, such as 2D, 3D, the file attributes such as texture, and the security label, such as "inherited_legal_academic_use."

II. Maya embedded metadata
In a Maya file, some useful descriptions, such as the object name (e.g., "student," "wheel," "ankle"), or object attributes, were written as notes by the artist or automatically generated by the software. Maya objects may also be animated. For these objects, metadata can be extracted from the embedded metadata with our MEL scripts running on Maya.

III. Adobe Bridge metadata
The metadata previously edited at the ICT with Adobe Bridge was embedded in the file itself. Some useful file attributes are preserved along with the digital assets in Adobe Bridge format.

IV. Filename follows naming convention
The ICT has set a standard naming convention to allow better organization and sharing of assets. This naming convention defines the file name format to describe a file and a list of abbreviation to shorten the length. For example, the file name "ChrUsaCivTeenM Skater001 mesh.ma" can be parsed to the tags: "Character," "USA," "Civilian," "Teenager," "Male," "skater," and "mesh." Thus, all file names that follow the naming convention can be interpreted for further indexing.

2.3. System framework

The current system, we implement, is composed of two main programs: an indexer that re-organizes the multimedia assets in the designated folder, and a search interface that allows users to retrieve the assets they need. The overall system diagram is presented in Fig. 2. The indexer contains three parsers to process the multimedia assets, along with other related management files, to populate relevant tags to associate with each asset.

The file distributor first went through the repository and assigned the .amd file and the Maya file to the AlienBrain parser and Maya parser accordingly. Concurrently, each file name was assigned and interpreted by the filename parser as well. For example, for an asset depicting a US Army scene, related reference terms, such as "male," "solider," "USA," "tank," "Army," and so on, will be populated as tags associated with this asset. More implementation detail is described in Section 3.

The search interface allows users to type in keywords and retrieves the assets containing tags matching those keyword(s). Apache Lucene [4] is used as the backbone of the search engine. A Java program reads a list of tags and their associated files from the Tag Auto-Populator module. Then, the data are entered into the Apache Lucene engine for indexing. The more keywords an asset fits, the higher rank it will be assigned in the search result. The web interface in Fig. 3 is implemented with PHP. Other programming languages can be used to implement the search interface in the future.

2.4. System requirement

Except for the MEL script, all the functionalities of the indexer are written in Java. Therefore, running the indexer requires JDK 1.6.0 and Maya 7.0 (or higher) installed on the operating system. On the other hand, the AMIA interface works like Google Search. The system only requires a web-browser to operate the interface. However, the server designated to provide this search service requires the installation of ApacheWeb Server v2.2.4, JDK 1.6.0, and PHP v5.2.3 or higher.

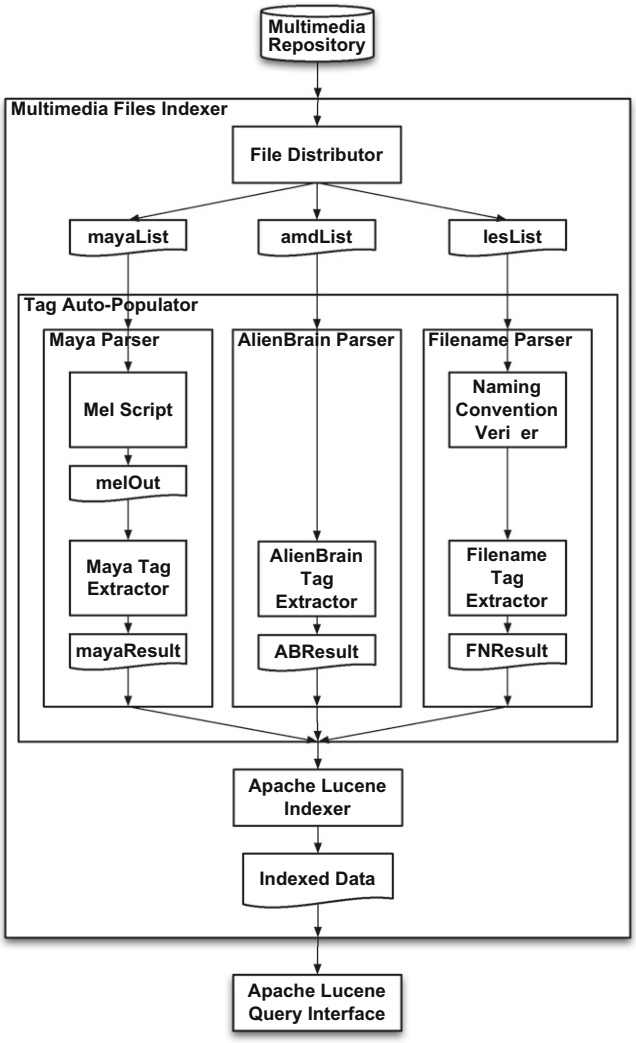

Fig. 2. Flowchart: Overall system design.

2.5. System performance

Security concerns restrict the use of some data; therefore, not all of the assets are available for testing. Table 1 shows the processing time for testing the data. Most of the processes in our system can be conducted within one to 2 min, except for programs written with MEL script. In order to extract the metadata embedded in Maya files, the program must operate Maya. Thus, processing time is based on Maya's performance. Although this process is time-consuming, it is complemented offline and will not affect online search performance. The current search process takes less than 1 s.

3. Text-based indexing and retrieval

Automating multimedia file tagging is a challenging task that warrants further study. Current techniques that search for images, videos, or audio, rely primarily on any textual information associated with the media. Such an approach often relies on human interaction to provide textual annotations to facilitate media retrieval. Therefore, a more intelligent methodology for extracting information associated with multimedia files, as well as a methodology for storing these files in a well-organized textual search framework, is desirable. In this section, we first give a brief survey of text-based information retrieval. Secondly, we will to discuss the

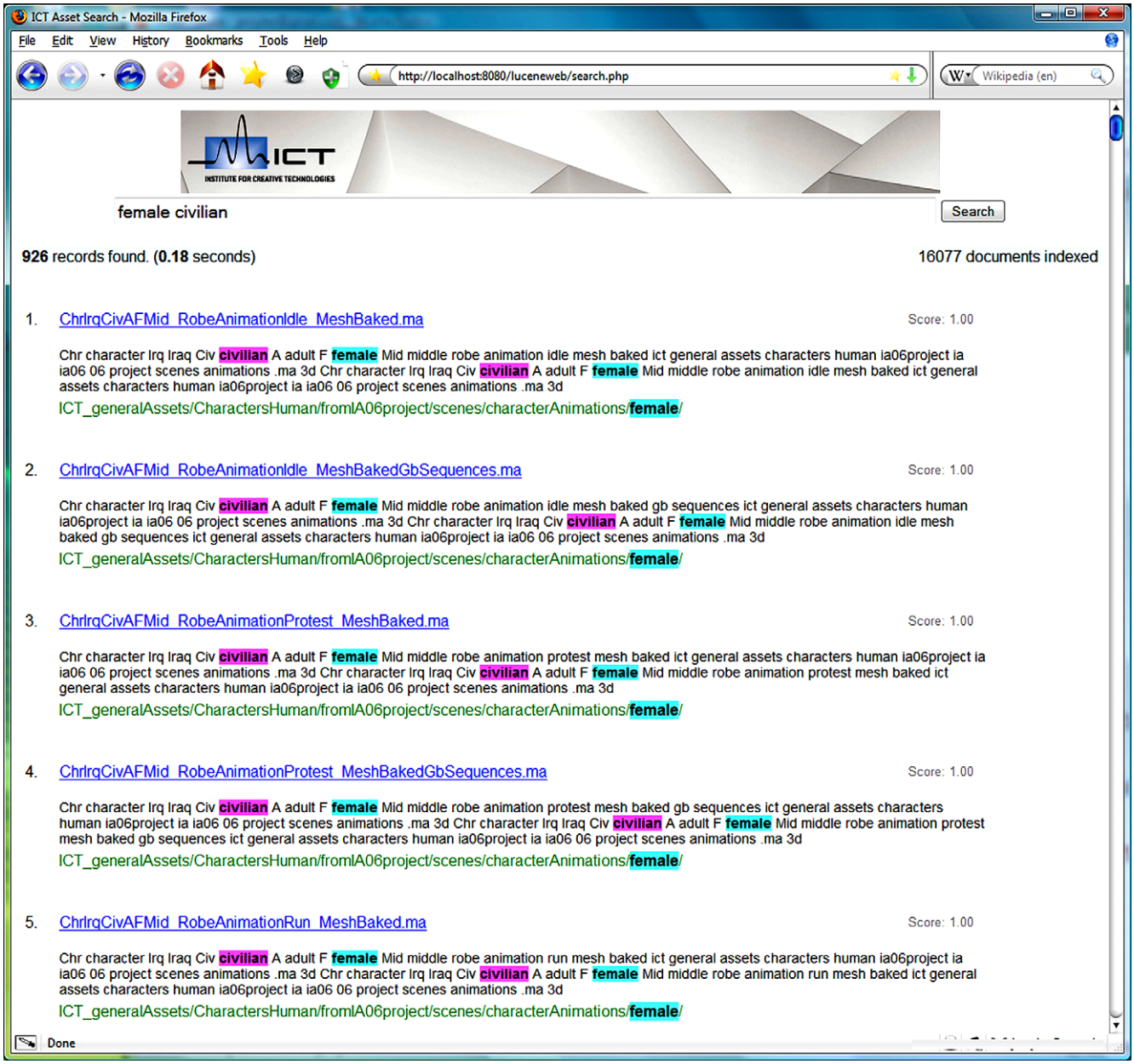

Fig. 3. Screenshot of the AMIA search interface.

Table 1
System performance.

Process	Time
Open and extract metadata in Maya files with MEL script – 1352 Maya files	30 min
Parse MEL output metadata into tags – 1352 metadata files	66 s
Parse filenames and paths into tags – 15,928 filenames	0.5 s
Parse AlienBrain metadata files into tags – seven amd files	15 s
Insert tags into Lucene – generate 14.7 MB of data	42 s
Search – within 15,928 assets	0.1 s

current progress of the AMIA project, including the implementation details and issues that we have encountered.

3.1. Survey of information retrieval

The study of text-based information retrieval inherently has a longer history than other types of media. Foundational texts [5–7] established the groundwork in information retrieval and continue to inspire future research. In addition to these works, several survey papers [8,9] also provide extensive reviews of the field. Text-based information retrieval study can be divided into four areas: indexing, clustering, storage, and search.

3.1.1. Indexing

Indexing parses through a collection of data and extracts information for searching. In the earlier days of organizing information for retrieval, human interactions were necessary in order to categorize information. Web-based search engines such as Yahoo! (http://www.yahoo.com), Excite (http://www.excite.com), and Lycos (http://www.lycos.com) provided submission forms for web administrators to submit entries for indexing. These submissions were sometimes unprocessed or delayed due to the limitations of manual processing.

Manual indexing is not a practical solution because the amount of data that needs to be indexed often grows rapidly

within a short period of time. In addition, the method is more error-prone and inconsistent due to differences in the indexers' methods of listing a document [10]. Such limitations drive the need to research intelligent methods of automatically classifying text data [8,11–14]. Current web-based search engines utilize web crawlers to automatically "crawl" through internet web pages and index assorted content. These methods of gathering information using web-based technologies are well documented [15–18].

3.1.2. Clustering

After extracting texts from the collection of data, the information needs to be organized into clusters so that data in the same cluster will be returned together as relevant results. Organizing texts into a data structure for search is a challenging task. In addition to returning results to search queries with exact document of interest, listing similar documents is also a desirable task to further enhance the search results. Various studies investigate the methodology of clustering [19–23]. Clustering methods provide a top-town approach that groups relevant documents and returns the results together to the search query. Processing the clusters is sometimes slow and may overlook some relevant results if the documents are placed in different clusters.

Rather than clustering documents with the exact same terms together, an ontological approach takes an extra step to associate related terms together [24–27]. Although ontology is able to group more relevant documents together, the process is usually domain specific.

Another approach is to store the documents by utilizing a list of words, or tags. The list of words is extracted from the documents and stored in an alphabetical order. Upon searching for a particular text string, the keyword list is accessed directly and the associated indicators to particular documents can be retrieved quickly [28–31]. The studies of organizing the list of keywords includes B-trees, or hashing [32,33].

3.1.3. Storage

There are two ways to store text data. The first method is to provide a structured storage that adheres to a hierarchical schema. The second method is to preserve the text as one full-text document. The nature of the structure refines the search range, thus it is more efficient if the desired results exist in the refined range. However, it is difficult to define the structure of the database and to require the user to fully understand the structure in order to pose the correct query. For example, "Army" can be categorized as a "profession," a "character description," or an "institution." This method of categorization requires the user to have sufficient knowledge of structure to perform a search (i.e., what are the fields, and how are the structures applied). Although users can employ complicated query commands to search for "Army" in all possibly related categories, this method significantly diminishes search speed. In addition, most users do not like to learn complicated query language to pose search commands.

On the other hand, full-text search is more flexible. Full-text storing methods usually contain far fewer fields. The search can be programmed in a way that does not require specialized input from the user. The search results can be ranked by their distance from the query. The "distance," or disparity between the logic of the content creator and the person accessing the content, can be adjusted to the needs of the users. This reflects the nature of data management: it is difficult to constrain the manner in which users employ tagging and data management, both as creators and consumers of this information. As a result, full-text storage is more inclusive [34].

3.1.4. Search

Lexical ambiguity is one of the fundamental problems in information retrieval. Most indexers use a subset of the words contained in the content to represent the asset, but ambiguity can be an issue. The user is not as interested in retrieving files tagged with an exact term, but in retrieving those containing words with a similar meaning. For example, searching for the keyword "tank" may indicate a tank vehicle, water tank, or a tank top. Retrieval programs generally address these problems by expanding the query words by related terms from a thesaurus. By using a suggestion list of expanded queries, the user can identify what they are looking for then target and retrieve the desired files. Chirita et al. [35] propose a method to improve the inherent ambiguity of short keyword queries by expanding them with terms collected from each user's Personal Information Repository, thus implicitly personalizing the search output. They introduced five broad techniques for generating the additional query keywords by analyzing user data, and discussed factors, which help to add a flexible number of keywords to the user query.

Another approach to expand the search terms is the use of ontology [36]. This technique looks up terms associated with the search query and generates additional search terms to look for other relevant search results. The approach is limited to domain specific grouping of terms, which might only provide useful search results in constrained contexts. In addition, multilingual text search is another area that warrants further study. Although some literature has been published on the subject [37–40]; this approach cross-references documents written in different languages and organizing the data structure remain challenging.

3.1.5. Practical systems

The most widely adopted application for text-based indexing and retrieval are web-based search engines. Prominent commercial services include Google (http://www.google.com), Yahoo! (http://www.yahoo.com) and Ask.com (http://www.ask.com). These applications utilize full-text search to index and store parsed textual information from web crawler technologies. Web documents are organized as inverted keyword lists, which provide faster search performance upon information retrieval. An open source project, *Apache Lucene* (http://lucene.apache.org), provides similar implementations to the commercial search engines to software developers.

In addition to indexing general documents from the World Wide Web, specialized search engines are also available for domain specific information retrieval. Chemindustry.com (http://www.chemindustry.com) hosts a search engine for chemical related searches. GovEngine.com (http://www.govengine.com/) lists United States federal, state and local government and court sites. Lastly, Search Engine Guide (http://www.searchengineguide.com) and CustomSearchEngine.com are services that provide searching on specialized search engines.

3.2. Implementation detail: three modules of our text-based indexer

As shown in Fig. 2, the indexer is composed of three modules: File Distributor, Tag Auto-Populate and Apache Lucene Indexer. We will describe the implementation issues and details of each module.

3.2.1. File distributor

To preserve the metadata previously stored in Maya and AlienBrain files, we wrote two different parsers to process these files according to their file formats. Within the multimedia repository, Maya and AlienBrain files are handled separately from other types of files. Thus, three lists will be generated from this module and then distributed to different parsers in the module to populate

tags. The three lists are made automatically and stored in allMayaFiles.txt, allAmdFiles.txt, and allFiles.txt by default configuration.

3.2.2. Tag Auto-Populator

This module is composed of three parsers, which manage the three lists generated by the file distributor, respectively. The three parsers process every file separate and respective lists and generate tags associated with the file. The following is the detail of these three parsers, including filename parser, Maya parser and AlienBrain parser:

3.2.3. Filename parser

This parser is composed of three modules: Naming Convention Verifier, String Segmentor and File Type Identifier.

I. Naming Convention Verifier
If the filename follows the naming convention, corresponding word expansion, such as "Chr" to "character," "C" to "Child" will be performed. In the mean while, a file name such as "C_1.jpg" cannot be interpreted as "child" because the letter "C" might be named randomly without meaning. Thus, we need a verifier to check whether a file name properly follows the ICT's naming convention before interpreting keywords. Table 2. exemplifies this naming convention, as well as the tags the associated tags interpreted from the file name.

II. String Segmentor
Although most filenames created years ago do not follow the naming convention, those filenames were named with meaningful words. These words were concatenated with delimiters or uppercase letter as a filename (e.g., andersonheadMolly_white2.tif). The String Segmentator processes each filename and separates the concatenated words into several tags for further indexing. The delimiters including symbols (such as "_", "-", "/", and white space), uppercase letters, and numbers were pre-defined for segmenting words. For example, "andersonheadMolly_white2.tif" is divided to "anderson," "head," "Molly," "white2" and "white." Words with numbers and consecutive uppercase letters are treated differently because some meaningful words contain numbers such as "AK47" or capitalized abbreviations such as "USA," "RGB," should remain the intact.Furthermore, The directory names are also segmented and output as tags since they are meaningful. For example, a 3D female model named "Amy.ma" under the "CharacterHuman" directory will be considered as a human character. In addition, to avoid redundant tags, some common tags, such as the name of the root directory, are discarded since all of the files are under the same directory.There were various issues that we encountered when implementing the String Segmentator, which are outline in further detail in Section 3.3.1.

III. File Type Identifier
Most multimedia assets can be identified as such by their file extension. Therefore, we wrote a File Type Identifier to generate tags based on the file extension. For example, assign the tag "2D" to the file type.jpg, .bmp, .tga, and assign a tag "3D" to the file type.ma, .mb, .nif and assign a tag "animation" to the file type.kfm, .kf.

Table 2
A filename that follows the naming convention and the translated tags associated with that file.

filename	ChrUsaInfantryAdultM_Star001_Mesh.ma
Class	Character
SubClass	Usa, Infantry, Adult, Male
Descriptor	Star001
Process	Mesh
File type	ma

3.2.4. Maya parser

The Maya parser includes a MEL script and a keyword filter. The MEL script running within a Maya environment is written to transfer all of the embedded metadata of the Maya file to a separated text file (here referred to as MelOut file). The parser is interested in extracting all information concerning animation, character, skeleton, LOD, light, dynamics, and textures that may be embedded. For example, a MEL script is written to check whether the Maya file can be animated. A word "animated" outputs into MelOut as a keyword for an animation-enabled Maya file. In addition, the filenames of the reference textures have to be extracted and preserved as one of the tags so that the reference will not be lost. Furthermore, we observed that the filenames of the reference textures, such as "sky.jpg," may provide useful information about the content of the Maya file. Thus, the tags extracted from the filename of the reference texture by the filename parser are also saved.

Except for the predefined keywords extracted by the MEL script, there are other useful metadata and countless redundant data that output to the MelOut file as well. Therefore, a keyword filter is required to discard the redundant data. The keywords are categorized as keyword sets, non-keyword sets, and unknown-word sets. The keyword sets contain meaningful words such as "female," "student," "skeleton," and "animated." The non-keyword set contains useless words or words that are too common to have relevancy (e.g., "group," "translate," "scale," "rotate," "visibility," "x," "y," and "z") when parsing the metadata, the word matches within the keyword set is preserved as a tag and the word matches within the non-keyword set is discarded. All other words which do not match any of keyword and non-keyword are classified as unknown words.

One way to determine whether metadata are useful or redundant is to employ statistical analysis. To do this, we parse all words in the MelOut file and count the frequency that each word appears. Table 3 shows parts of the word counts.

This statistical result was not directly used for classifying whether a word belongs to the keyword or the non-keyword set. Since the appearing frequency is not as relevant to managing files in our project, we use this statistical result to handle words from the highest to the lowest frequency within the countless extracted words from all Maya files. For example, words "rotate," "translate," "scale," have the highest appearing frequency so that they are handled first. If these words are useless, we can get rid of numerous redundant tags in our search system by classifying them to the non-keyword set.

Defining keyword/non-keyword is subjective and application-specific. The set of application specific keywords can be trained from the assets generated from a certain application. The set of general keywords can be further divided into user-oriented domain-specific groups. In the AMIA project, the database contains a large amount of military scenes and keywords used to describe models (such as "F16", "AK47", or "M1A1"). Note that the keyword list for the AMIA project may not be suitable for other DAMs. However, the list may serve as a starting point, which can be enhanced

Table 3
Statistical analysis of examplary words.

Word	Rotate	Translate	Scale	x
Count	27,977	25,981	24,049	23,421
Word	y	z	Gamebryo	Student
Count	23,230	22,425	895	287

through user-interaction. Thus, we leave this field editable for the system administrator, who will also allow flexibility, so that the end-user may define what is important to them.

The unknown keyword list serves as a candidate list for keywords. Currently, we preserve these unknown words as we would preserve keywords. However, a tag scoring method can be used to differentiate priorities between tags. The system can define the importance of each term in the unknown keyword list to fit users' interests. This tuning can be done manually or automatically. For example, we can tune the importance based on the query frequency of a term.

3.2.5. AlienBrain parser

The AlienBrain Parser is written to extract the keywords and values that are stored in the AlienBrain metadata file (*.amd). These .amd files can be exported from AlienBrain, and stored in the format (Filename—Keyword—Value). Each line shows a pair of keyword and its value referring to a certain file. A sample data of the .amd file is shown in Fig. 4.

3.2.6. Apache Lucene indexer

The tags previously generated by the Tag Auto-Populator module are sorted in XML format files. Then, a list of tags and their associated files are extracted from these XML files and entered into the Apache Lucene engine for indexing. We worked on the Apache Lucene's scoring algorithms to adjust the order of returned results. In Apache's default, to evaluate the score for document "D," given query "Q," is in the following equation:

$$coord(Q,D)\sum_{q_i}(tf(q_i,D)idf(q_i)boost(q_i)norm(q_i,D)), \quad (1)$$

where $Q = q_1, q_2, \ldots$, each q_i is a tag used in the query. Thus the five factors are:

(A) *coord(Q, D)*: The number of terms in the query that are found in the document *D*.
(B) *tf(q_i, D)*: The square root of the occurrence of the tag q_i in document *D*.
(C) *idf(q_i)*: The inverse document frequency of "q_i". The greater the amount of documents that contain the tag q_i, the less important the tag is deemed.
(D) *boost(q_i)*: A user option to emphasize a tag during the search. The user can type in "female^2 teacher" as the query to emphasize the tag "female."
(E) *norm(q_i, D)*: The product of three boosting functions computed upon adding the document into the database.

The three functions are:

(i) The importance of the document:
The importance of a document is user-dependent. For example, Maya documents are of higher weight than Photoshop documents if a user is targeting 3D assets. On the other hand, if the user is interested in the 2D design, Photoshop documents may be deemed more relevant. The importance of a document is embedded information, which is defined when it is added to the Apache Lucene index. The importance value can be tuned upon request. In our implementation, we gave the same importance value to every document.
(ii) The importance of each field in the document:
This is also user-dependent as the importance of the document.
(iii) The inverse of the number of tokens in each field:
This will be discussed below in Item II.

```
\sourceimages\brick.jpg | structure_type| 2D
\sourceimages\brick.jpg | structure_subtype| Texture
\sourceimages\brick.jpg | structure_tiled_texture| not tiled
\sourceimages\brick.jpg | structure_texture_layer| Color
\sourceimages\brick.jpg | inherited_legal_usc_only|False
\sourceimages\brick.jpg | inherited_legal_gov_use|True
```

Fig. 4. Sample data in AlienBrain.amd files.

The order takes five factors into consideration. Factors A and B are applicable to our search. Factors D, E(i) and E(ii) are user options, and therefore are not applicable. However, the default setting of C and E(iii) is not very suitable for searching for files associated with the queried tags. We analyzed the effect of each factor, and here we modify them to better suit our needs.

I Omitting the factor of the inverse document frequency of q_i.
In the default setting of C, the default value of idf(q_i) is:

$$idf(q_i) = 1 + \log\frac{\#\_of\_documents}{\#\_of\_documents\_containing\_q_i + 1}$$

The more documents that contain the tag q_i, the less important q_i will be. However, in our application, this falsely penalized the upper class words such as "Characters," "Props," and the tags that characterize file extensions (since they appears in more documents). Therefore, this factor was removed from the scoring evaluation process.

II Omitting the factor of the inverse of the number of tokens in each field
In the default setting of E(iii), the inverse of the number of tokens in each field falsely penalized the files that were associated with more tags. A Maya file containing a larger set of scenes will have more tags than another Maya file containing a simple object; however, the same tag in both files should be of equal importance. For example, file_1, which contains 100 buildings, will have more tags than file_2, which contains only 1 building. The tag "building" will appear in both file_1 and file_2, and should be equally important in both files. As a result, we removed this factor in the scoring evaluation process.

In addition, we made two more modifications to the Apache default indexer to meet the project's requirements. One modification was merging the tags of duplicated files in the system. Because the tags generated from three different parsers can indicate the same file, the default indexer keeps the file separate, which may slow down search performance and search accuracy. The other modification is that we disabled the function that uses numbers as the delimiters. Because numbers have meaning to our users, the user may search for "student03" as opposed to using "student," which is differs from "student01."

3.3. Implementation detail and issues of text-based parsers

Extracting useful tags from metadata and filenames was a laborious process. One of the main efforts attempted to segment the words properly, another was to use Word Regulation. Here we discuss more detail about the String Segmentation and the Word Regulation process.

3.3.1. String segmentation

To provide as much information about an asset as possible, the digital assets were named with concatenated words, akin to the metadata embedded in Maya files. The search engine cannot find the words that are concatenated with others. For example, "HappyFeet" cannot be found by searching for "Happy" or "Feet" or will be

assigned a relative low rank. Thus, we developed a String Segmentator to extract the keywords and save them separately as tags.

The segmentation mechanism is built upon the following four principles:

a. Segment strings at delimiters: all symbols, uppercase letter, and numbers.For example, segment "femaleChild_Low" to "female," "child," and "low."
b. Ignore connective words such as "and," "to," "from," and "or."For example, "fromCourseProject" would only generate tags "course" and "project."
c. Special handling of uppercase letters.For example, preserve the capitalized abbreviations "USA", by not dividing the characters into "U," "S," and "A."
d. Special handling of numbers.For example, we generate tags "student" and "student003" from "student003."

The first two principles are straightforward and do not require special explanation. On the other hand, principles c and d are more complicated. There will always be an exception that does not fit the segmentation rules. The following sections discuss the problems encountered with the special handling of uppercase and numerical values.

I. Special handling of uppercase letters

Because we use uppercase letters as the delimiters, problems occur with capitalized abbreviation (in this case, CAPString stands for the string of capitalized abbreviations). We observed that there are different combinations of CAPString and regular words shown in the filename list. For example, AFLow stands for two abbreviation "Adult," "Female," and a regular word, "Low." IA06project stands for IA, year 06, and project. Dividing words with upper case letters will not work perfectly in all cases. Certain words or terms can be identified (such as: ICT, VW, LOD, US, etc.). These consecutive uppercase letters on the list are preserved to be recognized as a word. However, there are still various unrecognized words (such as FO, GB, ES) which are difficult to parse.In addition, sometimes CAPStrings are followed by a word that begins with a lower case letter, such as ICTproject and USArmy. The correct cut is either cutting before or after the last uppercase letter. For example, correct segmentation for USArmy is before the last uppercase letter (A in this example) to generate "US" and "Army"; however, the correct segmentation for ICTproject is after the last uppercase letter (p in this example) to generate "ICT" and "project."Thus, we applied a free Java-based Dictionary, API Suggester [41] to check whether a segmented word is a valid word recognized by the dictionary. In our implementation, we also added the list of ICT-defined acronyms (such as: ICT, USC) in the dictionary to help uppercase letters segmentation. With this dictionary, we check the following two conditions: (i) if the lower case letters forms a valid word; (ii) if combining the last uppercase letter and lower case letters forms a valid word. The current segmentation decision is made according to the following truth table illustrated by Table 4.Although several methods of special handling are applied, it is difficult to find a perfect solution that can accurately segment all filenames which were not initially named in a standard format. Some meaningful tags would still become meaningless after word segmentation. For example: "noAccum" is divided into two separate tags as "no," and "accumulation."

II. Special handling of numbers

The principle of handling numbers is to always associate the numbers with the tags in front of the number. Although numbers are used as delimiters in our parser, we do not want to discard these numbers because they are part of the information. Thus, if there are numbers within a filename, we generate an extra tag preserving the numbers, by saving the number along with the word it concatenated to. For example, the tags "student002" and "student" will both be preserved. "skyDay512simple" would be divided to "sky," "day," "day512," and "simple." Saving just the numbers alone creates a long, meaningless tag. For example, the tag 002 is meaningless if separated from student002.There are also exceptions in the case of some words; when 2D, 3D, or a single character, occur immediately after the numbers (e.g., final02b), the characters should not be separated. Solving this issue proves to be demanding.The preservation of the special number by generating extra tags requires additional memory space. This is, however, needed to enhance the accuracy of the search performance. During the system design phase, end-users at ICT indicated a desire to search for a particular 3D model from a set of selections that are differentiated numerically. For example, a set of 3D buildings are named with the word "building" followed by sequentially increased numbers, e.g., building01, building02, etc. When a user wants to retrieval a specific building from the set, he/she may not enter a query string with a number immediately after word building. Thus, we designed our system accordingly.

Table 4
The truth table for deciding segmentation.

Case i	Case ii	SegmentAt
True	True	Before
False	True	Before
True	False	After
False	False	Before

3.3.2. Word regulation

To increase search power, a word regulator is being developed to translate a tag into a consistent format. For example, both singular and plural forms of a noun can be saved, so it is necessary to search both forms with a keyword in such a way that one form is not excluded. In addition, many capitalized abbreviations are defined, for shorten the length of the file name in the ICT's naming convention. For example, "M" stands for "male," "A" stands for "Adult," "Chr" stands for "character." A word regulator is needed to translate these abbreviations. However, this is not universally true for filenames that do not follow the naming convention, or the metadata embedded in the Maya files and the AlienBrain files. These files should be handled carefully as part of semantic analysis; hopefully, the correct translation can be constructed based on the context. Ambiguous words are not presently being handled with this methodology, and are left as they are.

3.4. System performance comparison

The ICT database contains over half a million (571,533) art assets. Since most data have security restrictions, the evaluation given below was conducted by ICT professionals with security clearance. It aims to compare our system with the commercial DAM tool currently employed by ICT. From all asset files, our system generated 11.7 million (11,681,548) tags, including 11.1 million (11,117,168) tags extracted by the filename parser, 0.2 million (193,541) tags extracted by the Maya parser, and 0.4 million (370,839) tags extracted by the AlienBrain parser. In this evaluation, the 10 most common queries picked by the artists were used as test queries. The returned result of our system is reported in Table 5.

Table 5
System performance of the proposed AMIA search engine.

Search item	Search time (s)	Number of file returned	Relevancy (*n* out of 10)
Female civilian	1.16	10,812	9/10
Military truck	9.26	92,276	6/10
Military and truck	2.12	143	8/10
City building	2.79	28,575	8/10
Military tank	9.2	90,288	10/10
US soldier	2.89	33,791	9/10
Segment and star	0.11	51	9/10
50 cal	2.94	2757	9/10
Brick texture	1.8	22,611	10/10
Baghdad reference	7.45	78,809	7/10

The same set of statistics cannot be obtained with ICT's commercial DAM tool since the system repeatedly crashed when the queries were performed. The benchmark system also limits queries to be a single-word term. As a result, the system was not able to handle test queries in our first round of testing. Thus, we set-up another test set (i.e., a small database consisting of 19,422 files) and different queries for the commercial DAM tool. The returned result of the second test set is shown in Table 6.

In the test, ICT end users entered search queries in both systems and recorded the search time and the number of files returned. In addition, subjective scores of returned results' relevance were collected and averaged to obtain a relevancy score listed in the fourth column of both tables. The relevancy score ranges from 1 (being the lowest) to 10 (being the highest). Relevancy is an essential evaluation criterion recommended by end users at ICT to vote whether the first 10 returned results met their interest. For example, in search for "military truck," the user intended to find truck models for the military purpose, rather than finding arbitrary military scenes that contains a truck. The user can put in a more descriptive query to further refine search results. For example, in search for "military and truck," fewer results were returned and accuracy became higher.

By comparing results in Tables 5 and 6, we see that AMIA returns more results than the commercial DAM tool and achieves higher relevancy scores. Such performance improvement is attributed to additional tags generated automatically by our system as compared with the commercial DAM tool. Being with a huge amount of tags, the search time of our system is still bounded by 10 s and the average search time is about 4 s, which are comparable with those of the commercial DAM tool.

4. Ongoing and future work

3D mesh and associated textures are an extremely valuable resource to the ICT, and therefore necessitate an effective management tools for reuse. Our text-based indexing and retrieval tool is capable of managing assets that were named properly, or that have embedded attributes already linked to the file. However, there are still many digital assets, many created years ago, which lack suitable metadata or were named randomly (e.g., object1.ma, 123.jpg). These assets are inaccessible using the text-based indexing tool, and a content-based indexing retrieval tool is still a goal to access these assets.

Table 6
System performance of the commercial DAM tool.

Search item	Search time (s)	Number of file returned	Relevancy (*n* out of 10)
Iraqi woman	4	0	0/10
Woman	3	10	4/10
Truck	2	97	8/10
Civilian	5	156	9/10
Tank	3	18	6/10
Building	4	75	6/10

As text-based indexing/search is nearly completed, content-based retrieval for 3D Shapes is the next goal of this ongoing project. The Princeton 3D search engine [42] provided a system framework which meets our need, it appears proper to use it as a reference for 3D assets search. However, unlike our goal to organize and reuse existing art assets, their system searches for available assets on the Internet.

For a large database holding over 400,000 assets, like ICT's Digital Backlot (DB), developing an indexing data structure and shape matching function with fast speed performance is important. Tangelder and Veltkamp [43] presented an overview on the 3D match function. According to their comparison, the feature-based-spatial-map approach [44,45] and the weighted point method developed by Funkhouser et al. [46] performed relatively better than various other approaches in terms of efficiency, discriminative power and robustness. The geometry-based approach [46] is evaluated as the first powerful tool for part-in-whole matching.

The AMIA project will start working towards content-based retrieval for 3D shapes by implementing methods proposed in [44–46]. Then, by comparing their performance, an improved approach will be proposed in the future. In addition to 3D shape digital assets, the database contains many image, audio and video data as well. After text and 3D mesh retrieval, an automatic content-based approach for 2D image, audio and video retrieval are next priorities in future development.

5. Conclusion

The public effort currently focused on the automated analysis and indexing of art assets with an eye towards integration and reuse of said assets via a searchable repository, especially targeted towards game and simulation development in the fields of training and education, is limited. Most research efforts are in the private sector and are not freely available for military and/or government use. The AMIA project strives to enhance the Digital Backlot (DB) so it can continue to provide a wide range of Army training system developers with a database of content for simulation environments, including everything from terrain to 3D models of buildings, to characters and props, to graphic textures. The research results from AMIA, as well as DB content, will be made broadly available to all Army training system developers, with respect for security and license restrictions. The AMIA research group will seek out for additional opportunities to collaborate with the US Army Research, Development, and Engineering Command (RDECOM), TRADOC, STTC, Defense Advanced Research Projects Agency (DARPA), and Universities that support this area of research.

Based on the current text-based search engine, we anticipate extending our work with focus on the following tasks:

1. Provide recommendations and estimations for visual dictionaries, and provide a document that chronicles collaborative work on creating digital asset management standards (such as naming convention, Maya metadata standard) for describing animations and other visual assets used in game and simulation environments.
2. Improve semantic search function. "Props" should bring up Prop and Prp, "vehicle" should bring up car and automobile. The word "Girl" should bring up results for Child and Female.
3. Research and develop a content-based retrieval indexer for multimedia assets. We will continue our work starting from 3D shape retrieval, and then 2D image, and then audio to the video content retrieval.

4. Providing a "shopping cart"-style system with the ability to pick and choose multiple items to download from the search results. All reference textures or scene files associated to a Maya file must be able to automatically be downloaded as well when the associated file is downloaded. In addition, the classifying security will take precedence before the development of a shopping cart system. Assets in the DB are restricted to different levels of security. Assets restricted for governmental use should and cannot be downloaded by general system users.
5. Developing a filename generalized parser. Ensure that all filename can be renamed following the naming convention standard.

Advances made in the aforementioned tasks will be chronicled in the future.

Acknowledgments

The effort depicted herein is sponsored by the US Army Research, Development, and Engineering Command (RDECOM), and the content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] Institute for Creative Technologies. Available from: <http://ict.usc.edu/>.
[2] AlienBrain. Available from: <http://www.alienbrain.com/>.
[3] Adobe Bridge. Available from: <http://www.adobe.com/products/creativesuite/bridge/>.
[4] Apache Lucene. Available from: <http://lucene.apache.org/java/docs/>.
[5] D.A. Grossman, O. Frieder, Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, Norwell, MA, 1998.
[6] I.H. Witten, A. Moffat, T.C. Bell, Managing gigabytes, in: Compressing and Indexing Documents and Images, second ed., Morgan Kaufmann, San Francisco, CA, 1999.
[7] R.A. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley/Longman, Boston, MA/New York, 1999.
[8] T. Saracevic, P. Kantor, A.Y. Chamis, D. Trivison, A study of information seeking and retrieving. I. Background and methodology (1997) 175–190.
[9] M. Kobayashi, K. Takeda, Information retrieval on the web, ACM Computing Surveys 32 (2) (2000) 144–173. doi:http://doi.acm.org/10.1145/358923.358934.
[10] S.A. Macskassy, A. Banerjee, B.D. Davison, H. Hirsh, Human performance on clustering web pages, Tech. Rep., 1998.
[11] D. Soergel, Organizing Information: Principles of Data Base and Retrieval Systems, Academic Press, San Diego, CA, 1985.
[12] X. Lin, D. Soergel, G. Marchionini, A self-organizing semantic map for information retrieval, in: SIGIR'91: Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1991, pp. 262–269. doi:http://doi.acm.org/10.1145/122860.122887.
[13] S. Jones, G. Paynter, Topic-based browsing within a digital library using keyphrases, in: DL'99: Proceedings of the Fourth ACM Conference on Digital Libraries, ACM, New York, NY, 1999, pp. 114–121. doi:http://doi.acm.org/10.1145/313238.313279.
[14] G.W. Paynter, I.H. Witten, S.J. Cunningham, G. Buchanan, Scalable browsing for large collections: a case study, in: DL'00: Proceedings of the Fifth ACM Conference on Digital Libraries, ACM, New York, NY, 2000, pp. 215–223. doi:http://doi.acm.org/10.1145/336597.336666.
[15] A. Heydon, M. Najork, Mercator: a scalable, extensible web crawler, World Wide Web 2 (4) (1999) 219–229. doi:http://dx.doi.org/10.1023/A:1019213109274.
[16] R. Burke, Salticus: guided crawling for personal digital libraries, in: JCDL'01: Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries, ACM, New York, NY, 2001, pp. 88–89. doi:http://doi.acm.org/10.1145/379437.379455.
[17] S. Raghavan, H. Garcia-Molina, Crawling the hidden web, in: VLDB'01: Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann, San Francisco, CA, 2001, pp. 129–138.
[18] J. Cho, H. Garcia-Molina, Parallel crawlers, in: WWW'02: Proceedings of the 11th International Conference on World Wide Web, ACM, New York, NY, 2002, pp. 124–135. doi:http://doi.acm.org/10.1145/511446.511464.
[19] E. Rasmussen, Clustering Algorithms (1992) 419–442.
[20] M. Charikar, C. Chekuri, T. Feder, R. Motwani, Incremental clustering and dynamic information retrieval, in: STOC'97: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, 1997, pp. 626–635. doi:http://doi.acm.org/10.1145/258533.258657.
[21] O. Zamir, O. Etzioni, Web document clustering: a feasibility demonstration, in: SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1998, pp. 46–54. doi:http://doi.acm.org/10.1145/290941.290956.
[22] M. Perkowitz, O. Etzioni, Adaptive web sites: conceptual cluster mining, in: IJCAI'99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1999, pp. 264–269.
[23] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (3) (1999) 264–323. doi:http://doi.acm.org/10.1145/331499.331504.
[24] M. Uschold, M. Gruninger, M. Uschold, M. Gruninger, Ontologies: principles, methods and applications, Knowledge Engineering Review 11 (1996) 93–136.
[25] M. Uschold, M. King, Towards a methodology for building ontologies, in: In Workshop on Basic Ontological Issues in Knowledge Sharing, Held in Conjunction With IJCAI-95, 1995.
[26] B. Chandrasekaran, J.R. Josephson, V.R. Benjamins, What are ontologies, and why do we need them?, IEEE Intelligent Systems 14 (1) (1999) 20–26
[27] M. Gruninger, M.S. Fox, Methodology for the design and evaluation of ontologies, 1995.
[28] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York, NY, 1986.
[29] A. Moffat, J. Zobel, Self-indexing inverted files for fast text retrieval, ACM Transactions on Information System 14 (4) (1996) 349–379. doi:http://doi.acm.org/10.1145/237496.237497.
[30] J. Zobel, A. Moffat, K. Ramamohanarao, Inverted files versus signature files for text indexing, ACM Transactions on Database System 23 (4) (1998) 453–490. doi:http://doi.acm.org/10.1145/296854.277632.
[31] M. Kaszkiel, J. Zobel, Term-ordered query evaluation versus document-ordered query evaluation for large document databases, in: SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1998, pp. 343–344. doi:http://doi.acm.org/10.1145/290941.291031.
[32] D. Cutting, J. Pedersen, Optimization for dynamic inverted index maintenance, in: SIGIR'90: Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1990, pp. 405–411. doi:http://doi.acm.org/10.1145/96749.98245.
[33] R. Guo, X. Cheng, H. Xu, B. Wang, Efficient on-line index maintenance for dynamic text collections by using dynamic balancing tree, in: CIKM'07: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, ACM, New York, NY, 2007, pp. 751–760. doi:http://doi.acm.org/10.1145/1321440.1321545.
[34] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems 30 (1–7) (1998) 107–117. doi:http://dx.doi.org/10.1016/S0169-7552(98)00110-X.
[35] P.A. Chirita, C.S. Firan, W. Nejdl, Personalized query expansion for the web, in: SIGIR'07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 2007, pp. 7–14. doi:http://doi.acm.org/10.1145/1277741.1277746.
[36] D. Fensel, Ontology-based knowledge management, Computer 35 (11) (2002) 56–59. doi:http://dx.doi.org/10.1109/MC.2002.1046975.
[37] L. Ballesteros, W.B. Croft, Resolving ambiguity for cross-language retrieval, in: SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1998, pp. 64–71. doi:http://doi.acm.org/10.1145/290941.290958.
[38] J.-Y. Nie, M. Simard, P. Isabelle, R. Durand, Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web, in: SIGIR'99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 1999, pp. 74–81. doi:http://doi.acm.org/10.1145/312624.312656.
[39] J.S. McCarley, Should we translate the documents or the queries in cross-language information retrieval?, in: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 1999, pp. 208–214. doi:http://dx.doi.org/10.3115/1034678.1034716.
[40] V. Lavrenko, M. Choquette, W.B. Croft, Cross-lingual relevance models, in: SIGIR'02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, 2002, pp. 175–182. doi:http://doi.acm.org/10.1145/564376.564408.
[41] Java Suggester. Available from: <http://softcorporation.com/products/suggester/>.
[42] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, D. Jacobsé, A search engine for 3D models, in: ACM Transactions on Graphics, ACM, New York, NY, 2003, pp. 83–105.
[43] J.W. Tangelder, R.C. Veltkamp, A survey of content based 3D shape retrieval methods, in: Multimedia Tools and Applications, Kluwer Academic Publishers, Hingham, MA, 2008, pp. 441–471.
[44] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation invariant spherical harmonic representation of 3D shape descriptors, in: Proceeding of Symposium on Geometry Processing, 2003.
[45] M. Novotni, R. Klein, 3D zernike descriptors for content based shape retrieval, in: Proceeding of Solid Modeling, 2003.
[46] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by example, in: Proceeding of SIGGRAPH 2004, 2004.