Georgetown University
Content Similarity
(gucs)
Alpha Release

Department of Computer Science
Georgetown University
Georgetown University
Content Similarity
(gucs)
Alpha Release

Department of Computer Science
Georgetown University
sdtext
Alpha Release

Department of Computer Science
Georgetown University
Georgetown Team

• Clay Shields
• Lindsay Neubauer
• Ophir Frieder
• Mark Maloof
• Micah Sherr
Content-Based Fingerprinting

• Our approach is to create fingerprints that are based on the content of the files
  – Formatting shouldn’t disrupt matching

• Fingerprints are digests of the file contents
  – Can be matched against each other to determine similarity
  – Designed to be robust to errors and edits

• This is a novel application of information retrieval techniques
Fingerprint Creation

• Use a training set of documents
  – Set you want to match against later
  – Documents that are similar to those sought
  – General documents in correct language

• Extract statistically important terms

\[ idf_T = \log \frac{|\# D|}{1+|\# D_T|} \]

• Create a dictionary of terms within a range of IDFs
  – Low IDFs too common
  – High IDFs too distinct
Bit Vector Fingerprints

• A Bit Vector fingerprint shows which dictionary terms were present in a document
  – Process document
  – For each term in document in dictionary, mark that position
Matching Bit Vector Fingerprints

• Allows for similarity matching
• Compute cosine similarity
  – Treat fingerprint as vector of $d$ dimensions
  – Measure cosine of angle between two vectors
  – If within specified range, consider match
• Allows for range of comparisons well after fingerprint creation
  – Specify parameter to matcher
  – Vary to optimize precision or recall
Order of Operations

- Determine tokenizers
  - Developing standard sets for various languages
- Determine parameters
  - Optional, can improve accuracy
- Build dictionary
  - Trim to appropriate IDF range
- Fingerprint files
- Compare fingerprints
Tokenizers

- First tokenizer must read from data source
  - FileTokenizer
    Reads and tokenizes a file
  - GzippedFileTokenizer
    • Reads and tokenizes a compressed file
  - OutsideInFileTokenizer
    • Extracts text and tokenizes a non-text file using Oracle’s OutsideIn
- Parameters:
  • “lines” (split file by line)
  • “tokens” (split file by whitespace)
Foreign Language Support

– ArabicFileTokenizer
  • Reads and tokenizes an Arabic language file
  • Uses Apache Lucene’s ArabicAnalyzer

– ChineseFileTokenizer
  • Reads and tokenizes a Chinese language file
  • Uses Apache Lucene’s ChineseAnalyzer or SmartChineseAnalyzer
  • Parameters:
    – “individual” (split by individual characters)
    – “smart” (split by probabilistic word segmentation)
Tokenizers

• Other tokenizers provide filter
  – StripPunctuationTokenizer
    • Removes punctuation from each token
  – RemoveNumericTokensTokenizer
    • Removes numbers from each token
  – RemoveTokensWithNumbersTokenizer
    • Removes tokens containing numbers
Tokenizers

• Other tokenizers provide filter
  – MaximumLengthTokenizer
    • Removes tokens that are too long
    • Parameter: token length
  – MinimumLengthTokenizer
    • Removes tokens that are too short
    • Parameter: token length
  – StopWordRemoverTokenizer
    • Removes tokens if they are present in a given list of stop words
    • Parameter: file name containing stop words
Tokenizers

• Language specific stemming
  – Stemming removes word endings to recognize word roots
    • Plurals
    • Conjugations
    • Imperfect but useful
  – PorterTokenizer
    • Alters tokens via English stemming
Experiment

• Best accuracy comes from analysis of files to be matched
• This looks for best dictionary IDF range and group of tokenizers for the data set
• Computationally intensive
• Must select parameters based on output
• Work in progress
Experiment Configuration

- Number of Trials, Number of Threads
- Database backend option
- Dataset name and path
- Dictionary size
  - (count/percent),
- Sample size
  - (count/percent)
- Min & max IDF ranges
- Tokenizers/Groups of Tokenizers
  - Manglers
- Fingerprinters
- Matchers and their parameters
Creating Dictionary

• Once parameters have been selected can create dictionary
  – Extracts text and analyzes term frequency
  – Then trim dictionary by IDF range

• XML output file contains remaining terms and frequencies
  – Can be shared for others to use for creating fingerprints
Creating Digests

• Given dictionary, fingerprint creation is easy
  – Parse text
    • Provide hooks to Oracle OutsideIn for extraction
    • Adaptable to other tools
  – Process with tokenizers
  – Record term presence

• Output configuration options available
  – Verboseness, ease of sharing
Fingerprinter Output

• Universal:
  – Base64 encoded fingerprint, fingerprint's unique identifier (GUID), and fingerprinter name
  – Version, creation time, and system on which the fingerprint was created
  – Name, directory, GUID, and version of the dictionary used to create the fingerprint
  – If provided at creation: the creator and creating program of the fingerprint
Fingerprinter Output

- Option: dataSource
  - Filename of the document fingerprinted
  - File path/directory of the dictionary used to create the fingerprint
  - System on which that dictionary was created
  - If provided: volume, disk image, and byte run

- Option: dictionary
  - Full dictionary included in the fingerprint file

- Option: digest (experimental)
  - File segment (by position in token stream)
  - Base64 encoded digest
  - Compression settings
  - Information about unknown tokens (compressed and Base64 encoded)
Fingerprint Comparison

• Two digests can be easily scored against each other
  – Output ranges from 0 to 99
  – Up to you to decide cut-off for appropriate match
    • Higher gives better precision
    • Lower give better recall
Ongoing work

• Automatic parameter selection
• Simple GUI for parameter selection
• Gnu Java Compiler testing
  – Native executable
• Creating fingerprints over multiple file segments
• Multiple parallel dictionaries
• Whatever we can do to help you all
Code available

• You can download and try the code at:

  www.cs.georgetown.edu/~clay/research/sdtext.html

• After DNS propagates (by tomorrow)

  www.sdtext.com
Where to Start

• Run ant build from the base directory
  – All command line programs can be run using sdtext.jar, located in the build directory
  – Each command line program is also built in the build/dist directory

• Some programs can have large memory requirements
  – Expand java heap size
    javam = java -d64 -Xms1g -Xmx8g
File Tokenization

TokenizeFile
• Tokenizes a file and prints the resulting tokens to the screen
  – Configuration file specifies a list of tokenizers to use on the file
• Usage:
  javam -jar sdtext.jar TokenizeFile
  -i <filename> -c <tokenizer config file>
• Example
  javam -jar build/sdtext.jar TokenizeFile
  -i doc/input_files/federalist/9 -c doc/configuration_files/tokenizers_config.xml
Experiment

• Usage
  javam -jar sdtext.jar Experiment -c <config file>

• Example
  javam -jar build/sdtext.jar Experiment -c doc/configuration_files/experiment_config.xml
Dictionary Creation

CreateDictionary

• Configuration file specifies a list of tokenizers to apply to the file

• Usage
  javam -jar sdtext.jar CreateDictionary -o <dictionary name> -p <dataset path> -c <tokenizer config file>

• Example
Dictionary Creation

TrimDictionary

- Creates a new dictionary without any tokens from the current dictionary that are outside the range of the given normalized IDFs.
  - English heuristic: min IDF ~ .3 and max IDF ~ .7

- Usage
  javam -jar sdtext.jar TrimDictionary -d <dictionary file> -b <minIDF> -t <maxIDF> -o <trimmed output filename>

- Example
  javam -jar build/sdtext.jar TrimDictionary -d doc/output_files/dictionary.xml -b .3 -t .8 -o doc/output_files/dictionary_trimmed.xml
Dictionary Creation

ShowDictionaryTokens

• Displays all the dictionary’s tokens with their frequencies, IDs, and normalized IDs.

• Usage
  javam -jar sdtext.jar
  ShowDictionaryTokens -d <dictionary file>

• Example
  javam -jar build/sdtext.jar
  ShowDictionaryTokens -d doc/output_files/dictionary.xml
Dictionary Creation

ShowDictionaryStatistics

- Displays the dictionary’s total number of documents, total number of tokens, maximum IDF, and tokenizers, and whether it has been trimmed.

- Usage
  javam -jar sdtext.jar
  ShowDictionaryStatistics -d <dictionary file>

- Example
  javam -jar build/sdtext.jar
  ShowDictionaryStatistics -d doc/
  output_files/dictionary.xml
Fingerprint Creation

BitVectorFingerprinter
• Creates a fingerprint for the given file using the given dictionary.
  – May specify the output’s destination filename and an output configuration file (specifies which types of output to include)
  – If no destination filename is specified, `fingerprint_<randomInteger>.xml` is used
  – If no output configuration filename is specified, uses full output

• Usage
  ```bash
  javam -jar sdtext.jar BitVectorFingerprinter -i <file to fingerprint> -d <dictionary file> -o <optional: destination file> -c <optional: output config file>
  ```

• Example
  ```bash
  javam -jar build/sdtext.jar BitVectorFingerprinter -i doc/input_files/federalist/9 -d doc/output_files/dictionary_trimmed.xml
  ```
Sharing Fingerprints

ExtractDictionary
• Given a fingerprint containing a dictionary, extracts the dictionary as an XML file.
  – Will clobber an existing dictionary file if different name not specified
• Usage
  javam -jar sdtext.jar ExtractDictionary -f <fingerprint file> -d <optional: dictionary filename>
• Example
Fingerprint Comparison

ScoreFingerprints
• Compares two fingerprints and scores their similarity based on the given matcher.
  • Range 0 to 99
• Usage
  javam -jar sdtext.jar ScoreFingerprints -m <matcher> -f <fingerprint file> -f <fingerprint file>
• Example
Fingerprint Comparison

CompareDirectory

• Compares a fingerprint to all files in a directory
  • Dictionary included in fingerprint or specified as a file
  • Output list of files ordered by score, high to low, above minimum

• Usage
  javam -jar sdtext.jar CompareDirectory -m <matcher> -f <fingerprint file> -p <directory> -s <optional: min score> <optional: dictionary file>

• Example