MUDABlue: An Automatic Categorization System for Open Source Repositories

By Shinji Kawaguchi, Pankaj Garg, Makoto Matsushita, and Katsuro Inoue
In 11th Asia Pacific software engineering conference (APSEC 2004)
Problem: Identifying similar Open Source projects

- Volume of open source projects necessitates efficient and accurate categorization
  - SourceForge had over 70,000 software systems registered when the paper was published
- Most categorization systems for open source archives are manual, which is time-consuming, scales poorly, and prone to error
  - Determining categories is a key component of these systems
- Software should be able to belong to more than one category
  - Categories based on non-use similarities, such as underlying architecture or required libraries, would also be useful
Latent Semantic Analysis (LSA)

- Natural language processing technique
  - Works with a set of documents, typically written in natural language
  - Other applications include data clustering, synonymy and polysemy, human memory study
- Distributional hypothesis: words that are similar in meaning will appear in similar pieces of text with similar distribution
  - Originated from linguistics
- Creates high dimension matrix of word relations between documents
  - Word-word, word-passage, and passage-passage relationships
- Drawbacks
  - Some relations can be mathematically sound but make little sense in human language terms
  - Cannot handle words having multiple meanings
  - Bag of words model flaws, though other techniques can be applied to address this
  - Standard LSA assumes Gaussian model distribution, and a newer alternative called probabilistic latent semantic analysis gives better results with a multinomial model
Goal Characteristics for MUDABlue

1. Don’t need pre-defined categories
   a. Namely, automatic generation of categories from the texts themselves

2. Software systems can be members of multiple categories
   a. Aim to capture aspects of usage, architecture, and library dependencies

3. Rely only on source code
   a. Disinclude any design documents, READMEs, build scripts, etc.
   b. Quantity and quality of non-source code documents varies too much between systems
MUDABlue LSA

- Each column is a document
- Each row is a word
- Cell entries are a count
- One simple similarity definition is the cosine of two row vectors
- **This table is overly simplified**; single value decomposition is applied to the matrix

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Human machine interface for ABC computer applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>A survey of user opinion of computer system response time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c3</td>
<td>Relation of user perceived response time to error measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m1</td>
<td>The generation of random, binary, ordered trees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m2</td>
<td>Graph minors IV: Widths of trees and well-quasi-ordering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m3</td>
<td>Graph minors: A survey</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Example Input Documents

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>user</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>response</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>time</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>survey</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>trees</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>graph</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>minors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. An Example of LSA Matrix
MUDABlue Process

● Focus on identifiers, e.g. variable names and function names, for creating the categories
  ○ Exclude reserved words and comments
  ○ After initial matrix is created, useless identifiers are removed (appearing in only one software system or appearing in more than half of all systems)

● Categories contain highly related identifiers
  ○ This is where LSA and the relationship between different but similar words comes into play
  ○ Use the ten highest valued identifiers to name the category

● If a software system has identifiers belonging to a category, it belongs in that category
Figure 3. Algorithm of Retrieving Categories
MUDABlue System

- Web-based interface which allows users to explore the categorization derived from the MUDABlue process for a set of software systems
- Visual components of interface include
  - Unifiable Cluster Map view created to simplify the categories and allow users to combine or expand categories, which is based on Cluster Map method
  - Category hierarchy and category list
  - Keyword search box
- Selecting components in one view also selects them in the others
MUDABlue Interface

- Top: UCM
- Left: Hierarchy
- Right: List
Evaluation Process

- Sample data was 41 C programs from SourceForge
- Target goals:
  ○ Does the prototype categorize the systems properly compared with existing manual categorization?
  ○ Can the prototype categorize by the libraries used in a system?
- Measures:
  ○ Precision - how many of MUDA’s decisions were correct
  ○ Recall - how many of the correct decisions MUDA caught
- Comparison is to “ideal” categorization, if done by hand by experimenters
Categorization Output

- The 41 projects produced 40 categories
  - 18 categories are the same as SourceForge categories
  - 8 categories are based on library dependencies and architecture
    - YACC, GTK, regexp, JNI, getopt, Python/C

<table>
<thead>
<tr>
<th>No.</th>
<th>Title of cluster</th>
<th>Software</th>
<th># of tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AOP, cinode, IC_RESULT, IC.LEFT, sep, aopGet, IC.RIGHT, pcl4, srcnode, icode, etype</td>
<td>compilers/gcc, compilers/sdcc</td>
<td>8597</td>
</tr>
<tr>
<td>2</td>
<td>CASE_IGNORE, CASE.GROUND_STATE, screen, CASE_PRINT, CASE.BYP_STATE, Widget, TScreen, CASE_IGNORE_STATE, CASE.PLL_VEC, CASE.PLL_POINT</td>
<td>xterm/R6.3, xterm/R6.4</td>
<td>2160</td>
</tr>
<tr>
<td>3</td>
<td>YY.BREAK, yyparse, yyval, DATA, yy.current.buffer, triple, yy_current_state, yy.c.bufp, yy_cp, unt32</td>
<td>compilers/gcc, database/mysq1-3.23.40, database/postgresql-7.2.1</td>
<td>223</td>
</tr>
<tr>
<td>4</td>
<td>AVL_cino, OUTLONG, avl, AVLTree, hdllite, OUT4CC, nbb, ERR.EXIT, et/along</td>
<td>videoconversion/dv2pg-1.1, videoconversion/libav30-1.0, videoconversion/mpgtools</td>
<td>177</td>
</tr>
<tr>
<td>5</td>
<td>domainname, msgid1, binding, msgid2, domainbinding, msgid, __builtlns, expect, transmem, int, codeset, codecsp</td>
<td>boardgame/gatanav-1.0.4, boardgame/gchcl-1.2.1</td>
<td>165</td>
</tr>
<tr>
<td>6</td>
<td>board, main, moves, ply, patron, file, apace, patron, moves, white, to, move, move, s, promoted</td>
<td>boardgame/Spring-10.8, boardgame/etanag-1.1.4, boardgame/flat1.1.4</td>
<td>154</td>
</tr>
<tr>
<td>7</td>
<td>xdrx, blob, DB, UCHAR, XDR, mates, key_length, lopsp, page_no, bdb</td>
<td>database/firebird-1.0.6.796, database/mysq1-3.23.49</td>
<td>118</td>
</tr>
<tr>
<td>8</td>
<td>domainname, N, binding, gchcl, GdkWidget, PARAMS, codeset, gpointer, loaded110file, sarg</td>
<td>boardgame/gatanav-1.0.4, boardgame/gchcl-1.2.1, editor/gnomepad-1.3.3, editor/peacock-0.4</td>
<td>118</td>
</tr>
<tr>
<td>9</td>
<td>GdkWidget, gchcl, gpointer, ginit, widget, gdk, widget.show, N, g_free, dialog, g_return_if_nnull</td>
<td>boardgame/gatanav-1.0.4, editor/gnomepad-1.3.3, editor/peacock-0.4</td>
<td>104</td>
</tr>
<tr>
<td>10</td>
<td>AOP, cinode, esp, IC_RESULT, IC.LEFT, obstacle, aop, mov, aopGet, IC.RIGHT</td>
<td>compilers/clip-2.39, compilers/gcc, compilers/sdcc</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4. MUDABlue Result (excerpt)
Precision-Recall Graph Comparison

- GURU and SVM are similar automatic categorization tools
  - Use information retrieval techniques
  - Applied to software documents
- MUDABlue’s F-value is .72
- GURU’s best F-value is .6591
- SVM’s best F-value is .6531
Related Work

- Latent semantic analysis in general, including linguistics and computational uses
  - Tweaks to equations, improvements, and so on
- Research that analyzes source code to glean information
  - Latent semantic analysis, self-organizing map, file structure or names, and call graphs
  - These look at one piece of software and internal relationships
  - Also software fault identification using Support Vector Machine
- Information retrieval methods for finding reusable components of software
  - Looks for highly related methods with a given query
  - CodeBroker listed as example, which uses LSA with the query and Javadocs of methods
- Using LSA to determine links between bug handlers and new bug tickets
Main Takeaways

- Automatic categorization of software would provide many benefits to large archives of software systems
- Semantic language analysis, a natural language processing technique, can be used to calculate similarities between bodies of text
- Even a simple interface with good interactivity goes a long way
- Tips to improve your own papers:
  - Include visuals of your UI, processes, and as necessary
  - Caption every figure and table, and explain them in text
  - Always be explicit about the choices you make (e.g., throwing out specific types of keywords) and state if it’s arbitrary
  - Always include as many details as possible about your experiments
  - Be consistent and fair between the various tools you’re comparing your own tool to; minimize as many differences as possible
  - Acknowledge the weaknesses of your system
Discussion Questions
Latent Semantic Analysis

1. Is latent semantic analysis a valid method to use in this context?
2. Do you agree with the decision to analyse only source code while ignoring all external documentation?
3. Do you agree that comments should also be disincluded?
4. Do you agree with the removal of all reserved words from the source code? I.e., using only identifiers and not other code constructs.
5. Do you agree with the removal of “useless” words before the LSA step is computed? These were words that occur in only 1 project and words that occur in more than half of all projects.
Strengths and Weaknesses of the Paper

1. Did you understand it? Did you like it? Was it publish-worthy?
2. Is there anywhere you found yourself wanting more information?
3. Were provided examples meaningful, and did they help you understand the content being presented?
4. Was the UI implementation a strength or a weakness? How did you feel about its explanation?
How effective is this table?

<table>
<thead>
<tr>
<th>No</th>
<th>Title of cluster</th>
<th>Software</th>
<th># of tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AOP, emicode, IC_RESULT, IC_LEFT, aop, aopGet, IC_RIGHT, pic14_emicode, icode, etype</td>
<td>compilers/gdbk, compilers/sdcc</td>
<td>8597</td>
</tr>
<tr>
<td>2</td>
<td>CASE_IGNORE, CASE_GROUND_STATE, screen, CASE_PRINT, CASE_BYP_STATE, Widget, TS Screen, CASE_IGNORE_STATE, CASE_PLOT_VEC, CASE_PLOT_POINT</td>
<td>xterm/R6.3, xterm/R6.4</td>
<td>2160</td>
</tr>
<tr>
<td>3</td>
<td>YY_BREAK, yvysp, yvval, DATA, yy_current_buffer, tuple, yy_current_state, yy_c_btf_p, yy_cp, unt32</td>
<td>compilers/gdbk, database/mysql-3.23.49, database/postgresql-7.2.1</td>
<td>223</td>
</tr>
<tr>
<td>4</td>
<td>AVI, cinfo, OUTFLong, avl_t, AVI_errno, hidr_data, OUT4CC, nhb, ERR_EXIT, str2ulong</td>
<td>videoconversion/dv2jpg-1.1, videoconversion/libcu30-1.0, videoconversion/mpgTools</td>
<td>177</td>
</tr>
<tr>
<td>5</td>
<td>domainname, msgid1, binding, msgid2, domainbinding, pexp, _builtin_expect, transmem_list, codeset, codesetp</td>
<td>boardgame/gbNav-1.0.4, boardgame/gchch-1.2.1</td>
<td>165</td>
</tr>
<tr>
<td>6</td>
<td>board, num_moves, ply, pawn_file, npiece, pawns, moves, white_to_move, move_s, promoted</td>
<td>boardgame/Sjeng-10.0, boardgame/cnag-11.4, boardgame/taile-1.4.4</td>
<td>154</td>
</tr>
<tr>
<td>7</td>
<td>xdrs, blob, DB, UCHAR, XDR, mutex, key_length, logsp, page_no, bdb</td>
<td>database/firebird-1.0.0,96, database/mysql-3.23.49</td>
<td>118</td>
</tr>
<tr>
<td>8</td>
<td>domainname, N_binding, gchar, GtkWidget, PARAMS, codeset, gpointer, loaded_110nfie, argv</td>
<td>boardgame/gbNav-1.0.4, boardgame/gchch-1.2.1, editor/gnuplot+1.3.3, editor/peacock-0.4</td>
<td>118</td>
</tr>
<tr>
<td>9</td>
<td>GtkWidget, gchar, gpointer, gnt, widget, gtk_widget_show, N, g_free, dialog, g_return_if_fail</td>
<td>boardgame/gbNav-1.0.4, editor/gedit-1.1200, editor/gnuplot-1.10.0, editor/peacock-0.4</td>
<td>104</td>
</tr>
<tr>
<td>10</td>
<td>AOP, emicode, esp, IC_RESULT, IC_LEFT, obstack, aop, mov, aopGet, IC_RIGHT</td>
<td>compilers/clisp-2.30, compilers/gdbk, compilers/sdcc</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4. MUDABlue Result (excerpt)
Final Thoughts

1. Is complete automation desirable? If you had to put a human in the loop in this categorization process, where would you want them?
2. Are there advantages or disadvantages to MUDABlue’s method that we didn’t discuss?
3. Are there strengths or weaknesses of the paper we didn’t discuss?