CONTEXTUAL SEARCH ARCHITECTURE

SRINATH VAIDYANATHAN

Masters Project Defense

January 10th, 2006

Committee:

Dr.Susan Gauch
Dr. Arvin Agah
Dr. Donna Haverkamp

Information and Telecommunication Technology Center

University of Kansas

Presentation Outline

- Introduction
- KeyConcept – A Brief Overview
- Goals
- Related Work
- Contextual Search Architecture – Overview
- Implementation
- Experiments and Results
- Conclusion and Future Work
- Questions???

Introduction

- Exponential growth of the World Wide Web since its inception in 1993.
- Users undergo increased difficulty in finding documents relevant to their interests.
- Current search engines return results based on key-word matches.
- Little relevance to the concepts in which the user may be interested.

Introduction contd…

- Personalization of search results – process of tailoring results to individual user's characteristics or preferences.

- Collect and represent information about the user – User Profile.

- Use the User Profile directly in the search process or use it to filter results returned from initial retrieval process.

- Profiles can be built by:
    - Collecting information from browsing history
    - Capturing content from open windows

Introduction contd…

- User Profiles can be either short-term or long-term.
- **Contextual Short-Term User Profiles** – Constructed by capturing what the user is working on at the moment they conduct the search.
- **Long-Term User Profiles** – Result from accumulation of user’s experiences over a longer time span.

Introduction contd…

Short-Term User Profiles

- Reflect user’s current interests
- Task Oriented
- More relevant results

Long-Term User Profiles

- Bring out user’s general interests over a period of time
- Not Task Oriented

KeyConcept - Overview

- Building a search engine that retrieves documents based on a combination of keyword and conceptual matching.
- System that includes information about the conceptual frame of the queries as well as its keywords.
- Documents are automatically classified to determine the concepts to which they belong.
- Earlier query concepts were determined by a small description of the query entered by the user.
- Currently query concepts are determined concepts by means of user profile. This helps in providing personalized search results.
- Hosted on UNIX platform at ITTC.

Goals

- Overall objective is to provide a working architecture for contextual search in windows environment.
- Major goal is to build contextual short-term user profile in the windows environment.
- Application for building user profile already existed in UNIX environment as part of KeyConcept source.
- Challenge was to port the existing application to WIN32 environment.
- Ensure that the profile is built efficiently.

Related Work

"Improving Ontology-Based User Profiles", Joana Trajkova and Susan Gauch, RIAO 2004, University of Avignon (Vaucluse), France, April 26-28, 2004, pp. 380-389.

- Worked on building long-term user profile
- Built profile by monitoring user's browsing over time

"Contextual Information Retrieval Using Ontology Based User Profiles", Vishnu Kanth Reddy Chellam, Masters Thesis, University Of Kansas 2004.

- Personalized search results using contextual short-term user profiles
- Studied effect of Conceptual ranking vs. Keyword based ranking

Related Work contd…

- **Microsoft's UNIX Application Migration Guide –** Excellent resource for looking at issues involved in code migration from UNIX to WIN32 environment.

- Major issues of difference are
  - File System Interaction and I/O
  - Memory Management
  - Process Management
  - Graphics Subsystem
  - Miscellaneous

Related Work contd…

- File System interaction
  - UNIX path separator(/) vs. Windows path separator(\)
  - Do not refer Windows based file in native form
    - “C:\dir\text.txt” translated as “C:dir ext.txt”
  - WIN32 file system is not case-sensitive unlike UNIX

- File I/O
  - WIN32 systems provide WIN32 APIs as counterparts to standard C and UNIX APIs
  - Mixing WIN32 APIs with Standard APIs may lead to problems
    - E.g.: open a file using *fopen()* and using *ReadFile()* to read the opened file
  - Restrict all I/O operations to standard APIs

Related Work contd…

- Process Model
  - UNIX has *fork*; WIN32 has *CreateProcess* & *CreateThread*
  - Our build profile application was not a system-level application. Hence did not have to deal with differences in the process model.

- Graphical model
  - UNIX uses X Window System GUI
  - Windows uses GDI
  - No simple mapping of X API to GDI API

Related Work contd…

- Memory Management
    - UNIX and WIN32 handle memory differently
    - UNIX separates out different types of memory
    - WIN32, which combines many memory types into a smaller set of APIs

- Miscellaneous
    - *strcasestr()* (header: string.h) available in standard C library is not supported by VC++ run-time library in WIN32
    - Calling of system commands from application
        - E.g.: “*cp*” in UNIX and “*copy*” in Windows

Contextual Search Architecture – Overview

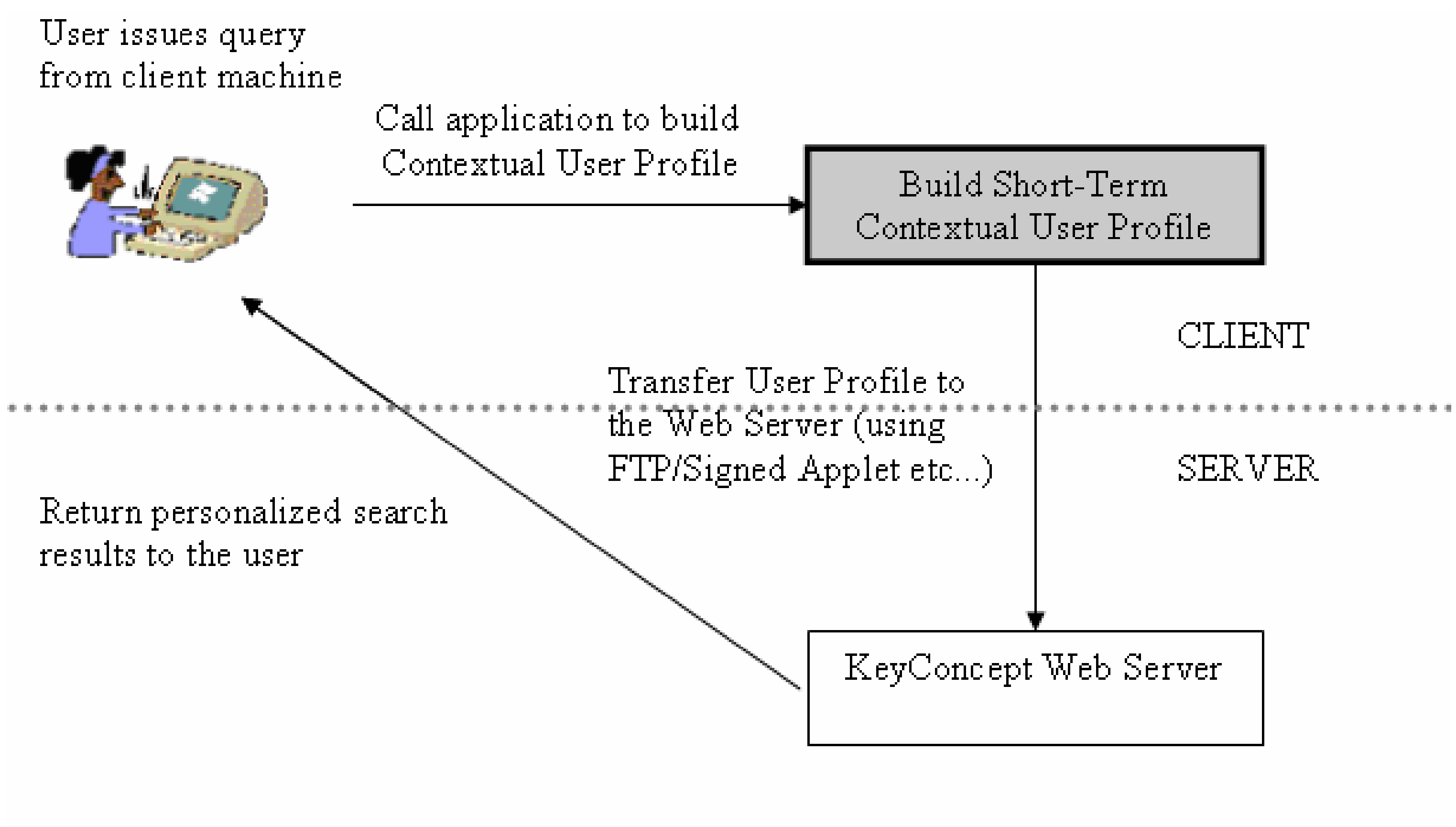

User issues query from client machine
Call application to build Contextual User Profile
Build Short-Term Contextual User Profile
CLIENT
Transfer User Profile to the Web Server (using FTP/Signed Applet etc...)
SERVER
Return personalized search results to the user
KeyConcept Web Server

Figure 1: Overall Contextual Search Picture

Contextual User Profile Application

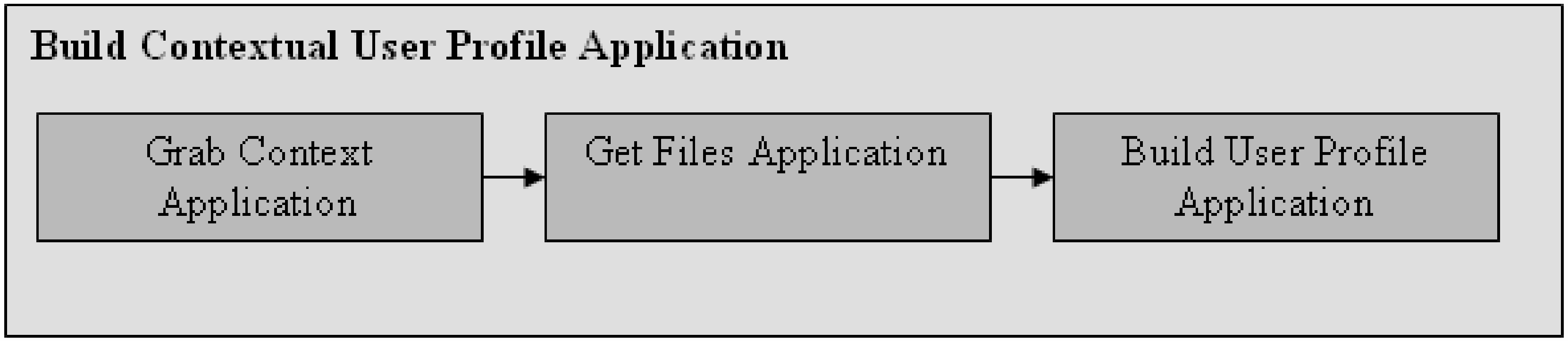

Build Contextual User Profile Application
Grab Context Application
Get Files Application
Build User Profile Application

Figure 2: Build Contextual User Profile Application

Grab Context Application – Overview

- Non-invasively monitor user activity on a user’s machine
- Capturing content from open Internet Explorer (IE), MS-Office, MSN Messenger, and other active windows
- Store the captured content in a named folder for that source of information on the clients machine
- Content is assigned a time-stamp

- An application stub was developed that writes some static content (instead of the grabbed content) to files
- Content stored in a named folder for that source of information on the user's machine
- Content is assigned a time-stamp

Grab Context Application – Overview

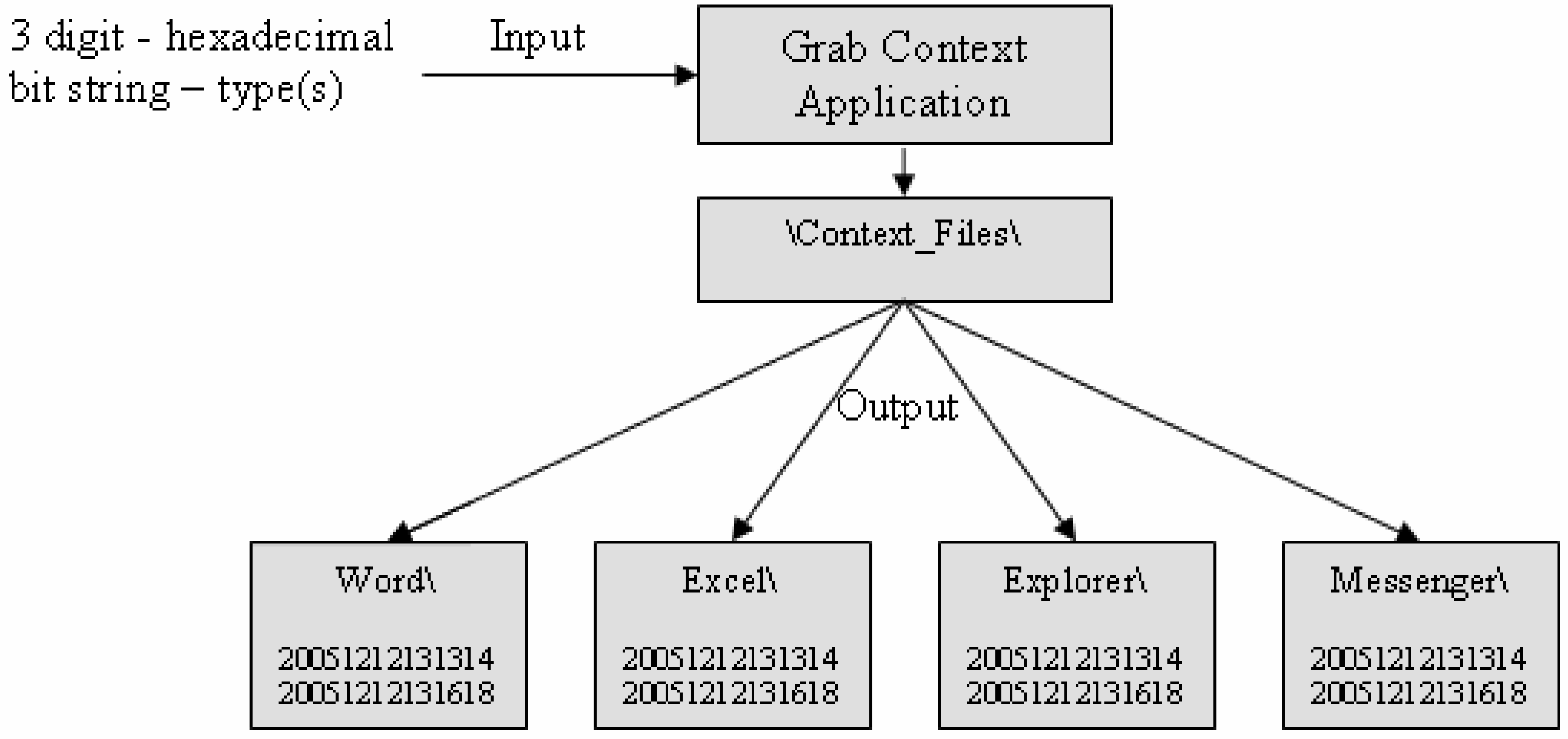
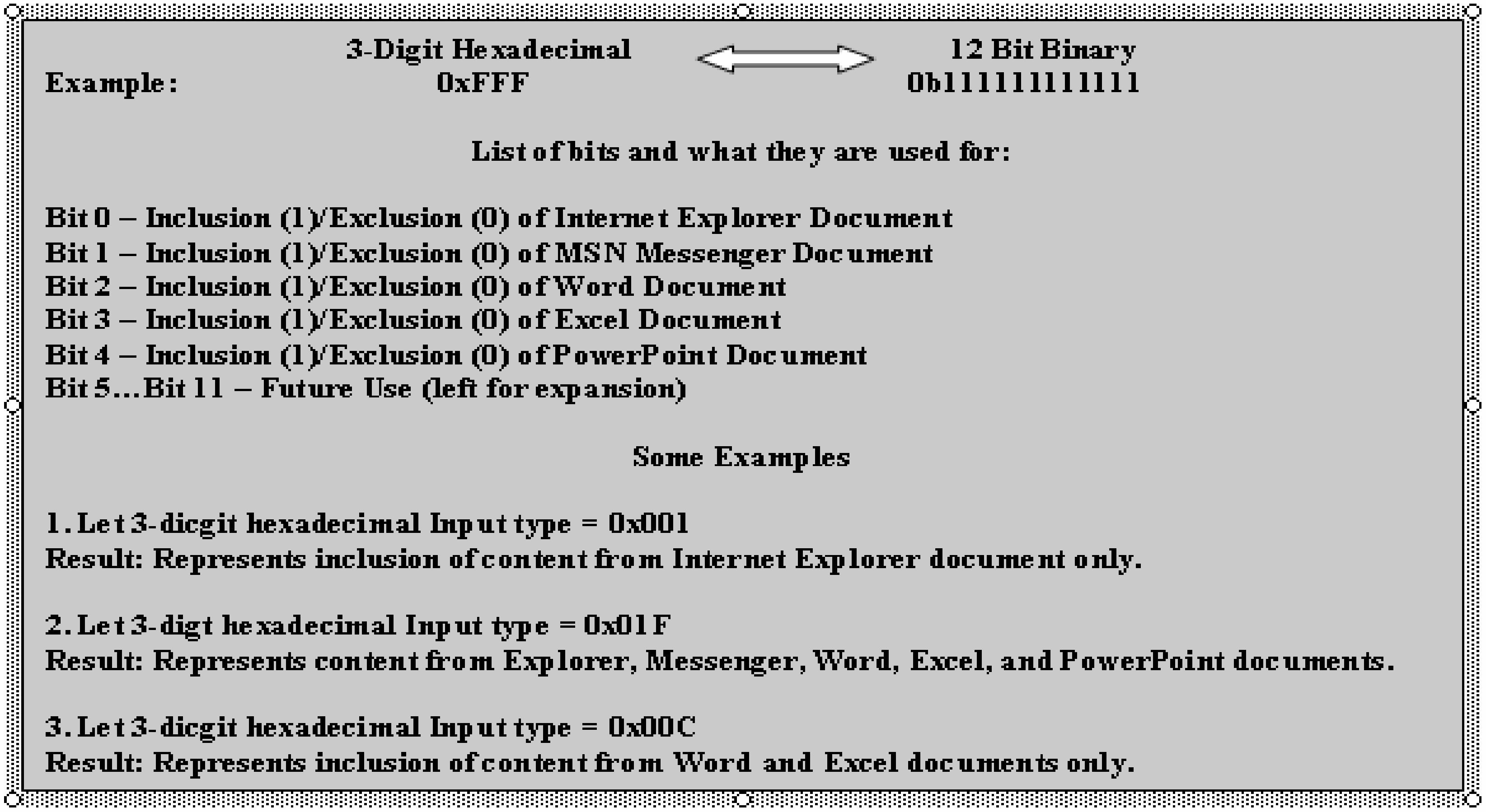


Figure 3: Grab Context Application

Grab Context Application – Overview



	3-Digit Hexadecimal		12 Bit Binary
Example:	0xFFF	⟺	0b111111111111

List of bits and what they are used for:

Bit 0 – Inclusion (1)/Exclusion (0) of Internet Explorer Document
Bit 1 – Inclusion (1)/Exclusion (0) of MSN Messenger Document
Bit 2 – Inclusion (1)/Exclusion (0) of Word Document
Bit 3 – Inclusion (1)/Exclusion (0) of Excel Document
Bit 4 – Inclusion (1)/Exclusion (0) of PowerPoint Document
Bit 5…Bit 11 – Future Use (left for expansion)

Some Examples

1. Let 3-dicgit hexadecimal Input type = 0x001
Result: Represents inclusion of content from Internet Explorer document only.

2. Let 3-digt hexadecimal Input type = 0x01F
Result: Represents content from Explorer, Messenger, Word, Excel, and PowerPoint documents.

3. Let 3-dicgit hexadecimal Input type = 0x00C
Result: Represents inclusion of content from Word and Excel documents only.

Figure 4: 3-Digit Hexadecimal Bit String Values

Get Files Application – Overview

- Generates a list of filenames to be passed to the Profile Building application
- Application creates a *Filenamesfile* that holds the path to all files of the requested type(s) created between the start and end times
- E.g.: To request content from all Word and Excel documents created 10 minutes prior to the current time of 2:32 on Dec 31st 2005, call,

  GetFiles (0x00C, 20051231022200, 20051231023200).

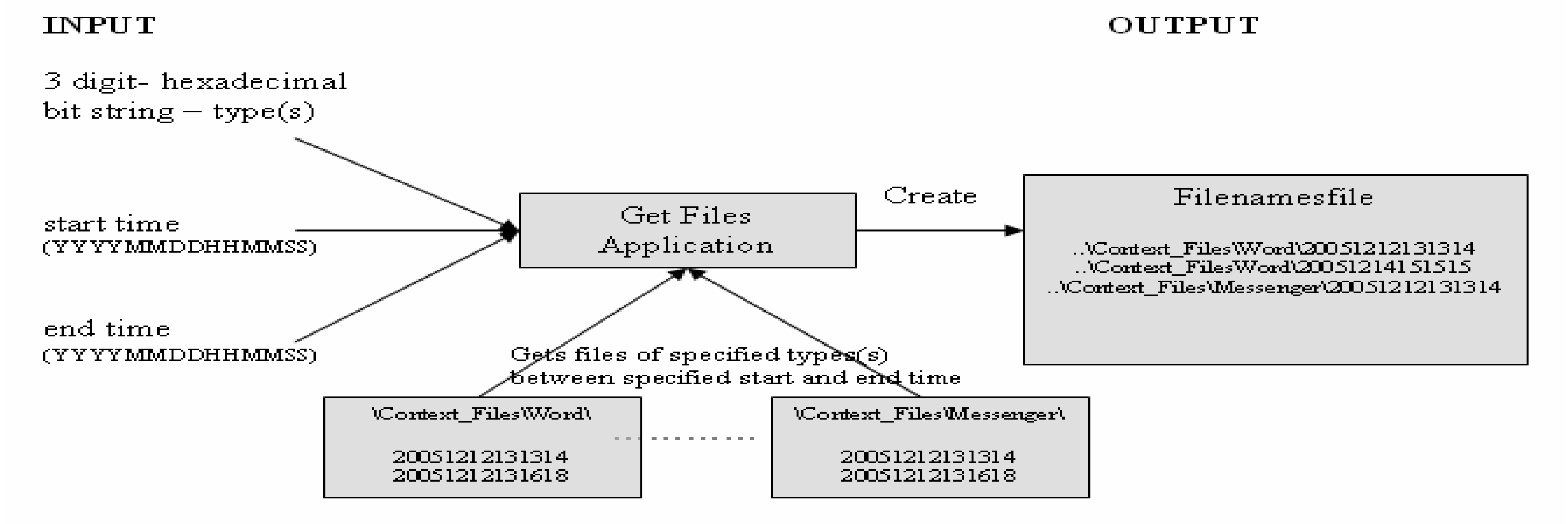

INPUT
OUTPUT
3 digit- hexadecimal bit string – type(s)
start time (YYYYMMDDHHMMSS)
end time (YYYYMMDDHHMMSS)
Get Files Application
Create
Filenamesfile
..\Context_Files\Word\20051212131314
..\Context_Files\Word\20051214151515
..\Context_Files\Messenger\20051212131314
Gets files of specified types(s) between specified start and end time
\Context_Files\Word\
20051212131314
20051212131618
\Context_Files\Messenger\
20051212131314
20051212131618

Figure 5: Get Files Application

Build Profile Application – Overview

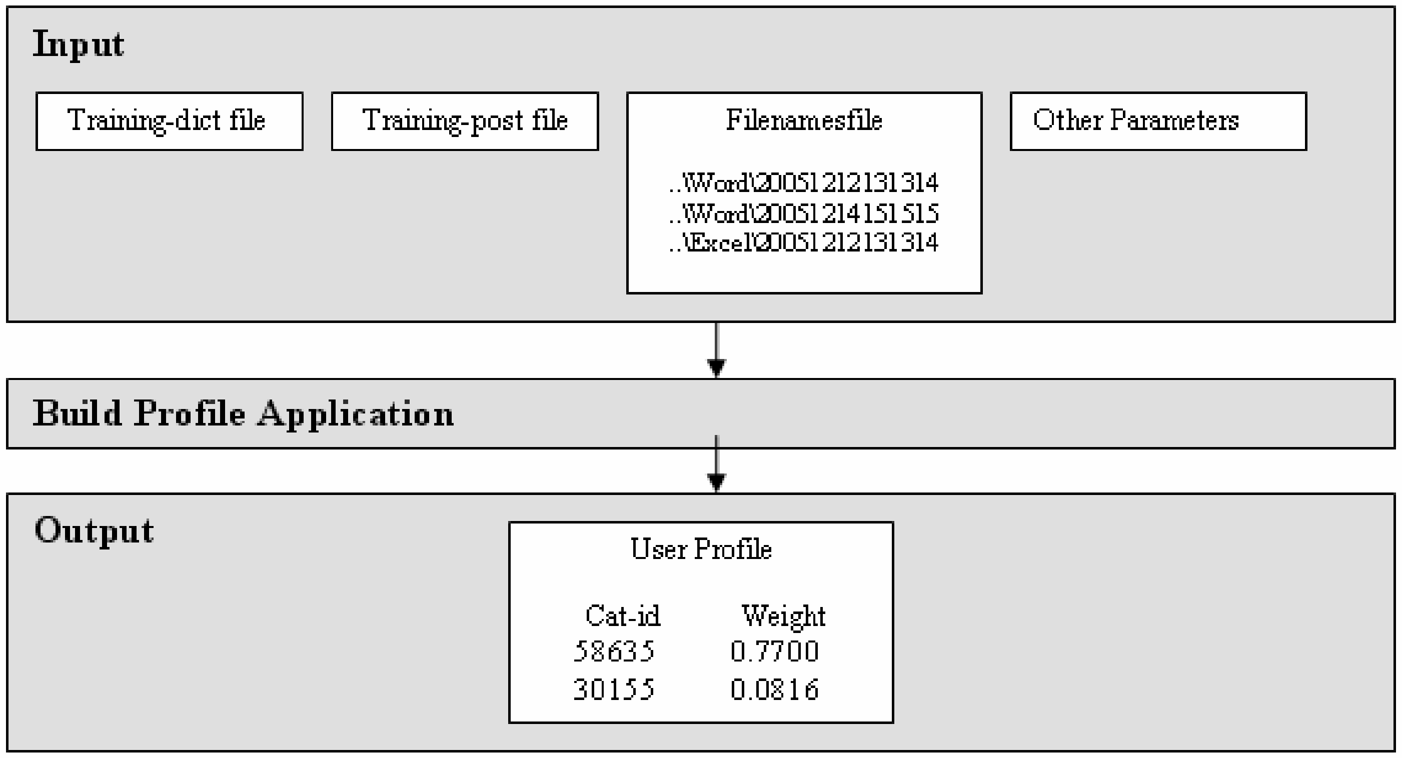


Figure 6: Build Profile Application

How to Build Contextual Short-Term User Profile?

- Monitor the activity of the user on his/her Windows machine. Capture content from open Word documents, Excel Documents, Web pages, Chat transcripts etc.. → **[Grab Context Application]**
- Create a *Filenamesfile* that holds the path to all files of the requested type(s) created between the start and end times. → **[Get Files Application]**
- Categorize the captured content to build a contextual profile
  - Consists of 2 phases
  - **Training Phase**: Classifier creates inverted index for collection of training documents.
  - Training documents are those data, that are manually assigned into categories of the ODP standard tree.

How to Build Contextual Short-Term User Profile?

- All documents that belong to a category are merged to create super document. This super document is indexed and hence for each category a vector of vocabulary terms and weights associated with the category are stored in inverted files. →[**Training-dict and Training-post files]**

- **Categorizing Phase:** Each captured content file pointed by the ***Filenamesfile*** is classified and the top N categories are stored for each document. When a particular category appears in multiple documents the weights of the documents in that category are added to show greater degree of user interest in that particular category. --> **[Build Profile Application]**

- The final user profile is a text file with a list of category id's and the weight assigned for each category. The category id's represent the concepts the user is interested in. The category weights give a measure of the degree of interest the user has in that particular category.

Implementation

- Development Environment
  - WindowsXP workstation with Visual Studio .NET 2003
  - .NET 1.1 framework, which is a platform for building, deploying, and running web services and applications
  - Visual Studio supports development in multiple languages including Visual C++ 7.1
  - C/C++ optimizing compiler and preprocessor
  - Compiler warning level was set to *level 3* - Severe

Implementation

- The Grab Context & Get Files application, was developed for the Windows environment.

- Build Profile Application – Implementation
  - Build Profile application, was ported to the Windows environment, from the UNIX environment.
  - Makes calls to the *categorize* and *retrieve* modules of the KeyConcept Source. Hence those modules were ported too.
  - *retrieve* module, for a given query retrieves all documents that contain the query words.
  - *categorize* module, for a given document, or set of documents, retrieves the top categories the document(s) match.

Build Profile Application – Function API

- void create_profile(prefilename, dict, post, inputType, input, wtflag, subjectTree, lcaHTMLsDir, output, numWords, numCat, prune_threshold, min_weightThold, update, date_processed)
    - dict - Path of the training dictionary file
    - post - Path of the training postings file
    - prefilename - Path to file specifying document pre-processing options
    - inputType - Specifies the format of the input
      1--input is a file containing a list of content files
    - input - Location of input (its type specified by inputType parameter)
    - wtflag - Relative weight flag. Use absolute or relative weight. (0 or 1) (UNIX only)
    - subjectTree - Location of the Standard tree (UNIX only)
    - lcaHTMLsDir - For UNIX version only
    - output - Location where User Profile is stored
    - numCats - Max number of categories to put one page in
    - numWords - Max number of words used for categorizing
    - prune_threshold - Minimum weight threshold for categories to be recorded in the output weighted std tree file
    - min_weightThold - Minimum weight threshold (expressed as % to the weight of the most important category)
    - update - 1 - Update existing profile, 0 - Create new profile. (UNIX only)
    - date_processed - String specifying date when profile was created

Porting Index Module

- File-based vs. Memory-based
- Ported Index module from KeyConcept source
- The index module when ran successfully and created a dictionary of words and a posting file with word locations

Porting Issues

1. File paths for configuration files and other input and output parameters.
2. UNIX “*sort*” utility vs. DOS “*sort*” utility
   - UNIX “*sort*” utility can perform alphabetical and numeric sort. It can also merge sorted files. It can also sort with respect to a specific token in the file. E.g.: *sort –brn +1 –o file1 file2*
   - DOS based *“sort”* utility is quite primitive. It can perform only alphabetical sort.
   - generic insertion sort routine and a merge sort routine in memory was developed for both the UNIX and WIN32 environments.
3. Difference in new line character in UNIX and Windows, which affects text files.

Porting Issues

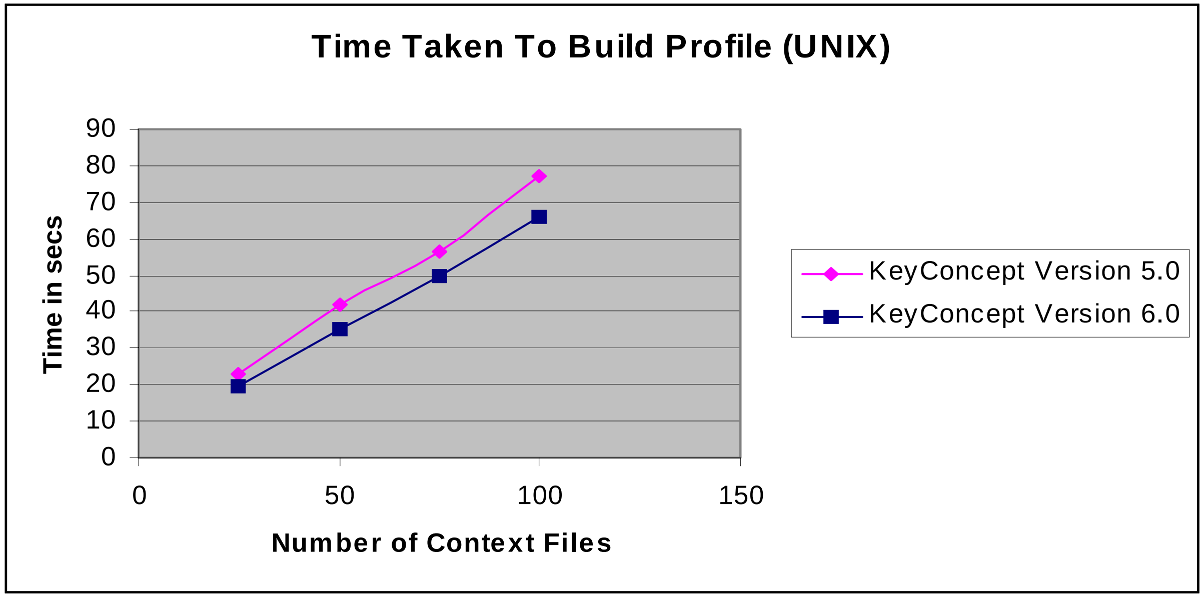
4. UNIX function *mkdir (dirpath, mode)* vs. WIN32 *_mkdir (dirpath)*
5. UNIX function *chmod (filename, mode)* vs. WIN32 *_chmod (filename, mode)*
6. Differing function names
   - *mkstemp()* vs _mktemp()
   - *snprintf() vs _snprintf()*
   - *vsnprintf() vs _vsnprintf()*
7. Use of *getopt()* function for command line processing
8. UNIX *dirent.h* header file used for directory access operations
   - Used a freely available wrapper for dirent interface in WIN32 environment.

Experiment & Results

- Executable created for all 3 applications
- Executable is capable of running on any Windows workstation that has the .NET Framework 1.1 installed.
- The workstation does not require Visual Studio or Visual C++ software
- Windows batch files were created that calls the executable with necessary input parameters

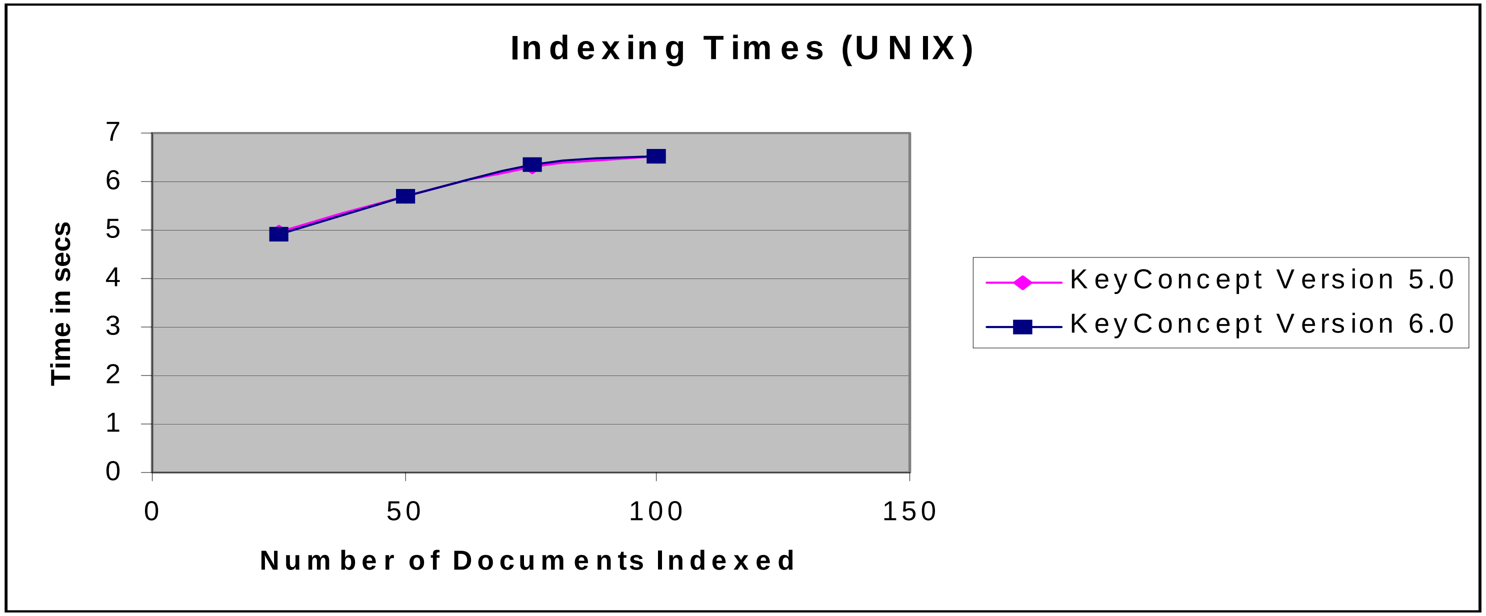
Experiment & Results

- Ported build profile module is faster than the older version.
- Comparison was carried out on the same UNIX box with the same input parameters.



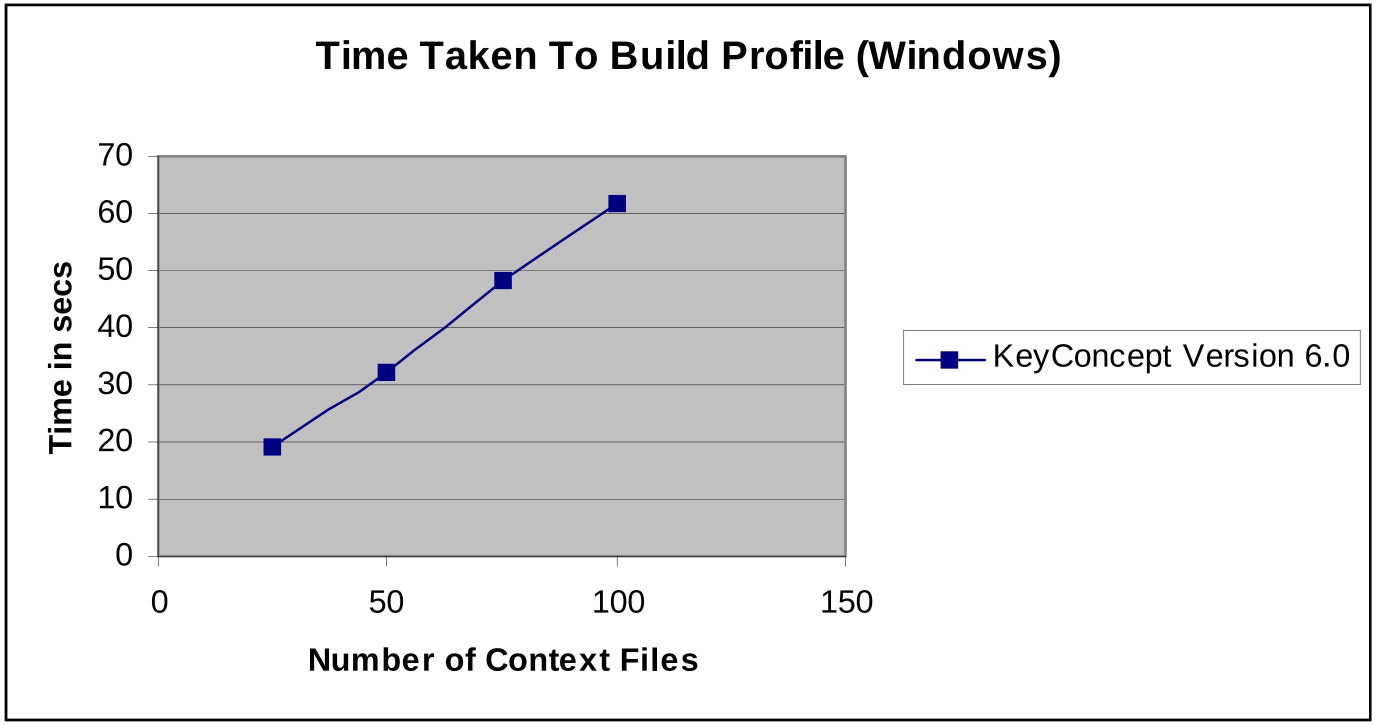
Experiment & Results

- Earlier version and ported version perform quite similarly in terms of indexing times
- Comparison was carried out on the same UNIX box with the same input parameters.



Experiment & Results

- Performance of the build profile module, version 6.0, in the Windows environment is illustrated using a graph below.



Conclusion

- Main goal of building a short-term user profile in the Windows environment was achieved.
- The performance of the profile builder application was also improved by making suitable revisions.
- Application was built with the required libraries statically loaded such that the application can be fired up from any Windows workstation successfully.
- KeyConcept source was suitably modified to make it work in both Windows and UNIX environment.

Future Work

- Application that non-invasively monitors user activity and grabs content from open windows can be further developed.
- Studies need to be done to determine the best time window within which documents captured should be included in the contextual profile.
- Effect of using long-term user profiles instead of short-term user profiles can be investigated.

Questions ???