PALTask: An Automated Means to Retrieve Personalized Web Resources in a Multiuser Setting

by

Pratik Jain
B. Tech., Uttar Pradesh Technical University, India 2009

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science

PALTask: An Automated Means to Retrieve

Personalized Web Resources in a Multiuser Setting

by

Pratik Jain

B. Tech., Uttar Pradesh Technical University, India 2009

ABSTRACT

When performing web searches, users manually open a web browser, direct it to a search engine, input keywords, and finally manually filter and select relevant results. This repetitive task can negatively impact the user's experience, something the automation and personalization of web search can address.

This thesis presents PALTask, an Instant Messaging (IM) application that exploits context of both the user and their conversation in order to automate and personalize related web tasks such as web searches relevant to the conversation. PALTask dynamically gathers context and provides feedback from the user and the system at runtime including keywords from the conversation and running them through various search services such as YouTube and Google to retrieve relevant results. This thesis also explores various natural language processing (NLP) tasks such as keyword extraction, sentiment analysis, and stemming. These NLP tasks help in the collection of dynamic context at runtime, identifying personalized context, and analyzing it to improve the user's experience. We also present our keyword ranking algorithm which aims to improve accuracy when retrieving web resources.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Hausi Müller, my supervisor, for his support, encouragement, and guidance. I want to thank him for his ideas, being my mentor, and providing moral support during this research. I have learned a lot under his supervision and I am grateful to him for providing me the opportunity to work with him.

Dr. Alex Thomo, for being my committee member and mentor, and providing feedback on this research.

Andi and Lorena, for their support, ideas, and implementation. They were part of the PALTask team and I would like to share the credit of this work with them. This thesis would have not been possible without their help. Thanks for being my mentors, friends, and colleagues.

Nina, Ron, Przemek, Ishita, Atousa, and all Rigi group members, for their valuable feedback and discussions which generated ideas and leads to implementations, and all the fun we had inside and outside the Rigi Research Lab.

My parents and other family members, for their love and support.

Chapter 1

Introduction

1.1 Problem Definition and Motivation

The internet is a part of our daily lives. Users perform numerous activities with web services and applications using ubiquitous, connected devices to achieve personal and professional goals. For this purpose, users turn to the web, with its hundreds of millions of pages presenting information on an amazing variety of topics. However, web search, a very common and ordinary activity, often becomes an arduous task given the complexity and colossal size of the internet.

Searching for a web resource (e.g., video, audio, text, or images) involves a set of repetitive steps that increases the complexity of the task and diminishes a user's experience. Users have to manually input keywords into search engines or related websites and manually filter results. Writing a thesis and searching for synonyms of particular words, or communicating with friends and colleagues when sharing interesting resources, are both examples of multi-step, manual web searches performed by a user. However, these tasks could be simplified into fewer steps and automated by exploiting *context*. Context is defined as all relevant information gathered from the environment, users, web interactions, sensors, devices, and other systems that affect the situation of users [ADB$^+$99]. Contextual information gathered from numerous sources (e.g., users, devices, applications, and conversations) can be useful in enhancing the automation and *personalization* of context-driven web searches.

Ng et al. describe the purpose of "web browsing" as information retrieval in the web of the user's interactions, whereas "web tasking" as an action towards user goals using information cues in the web [NL13]. The authors identify that web browsing lacks context awareness as well as customization and personalization in returned HTML pages.

Web tasking can aid web browsing by concentrating on actions associated with a user's goals. Actions involved in web tasks can be mined for context to customize the task according to the user's needs and preferences. Web tasking can be conducted by users or machines acting on behalf of the user. The automation of a web task, which is a web task conducted by programmed code on behalf of a user, can simplify the task by reducing repetitive steps involved in web browsing [CnMV13]. Automating web tasks to achieve a personal goal can improve user experience. However, decomposing a personal web task into simpler tasks whose complexity is hidden to the user is challenging [CnMV13].

The recent proliferation of smart mobile devices with embedded sensors along with Big Data analytics has enabled the collection of huge amounts of contextual information. Although the information can be used to improve user experience, it has no value unless we analyze, interpret, and understand it. Most of the time, sufficient context is available to perform web searches, but it is not used to reduce the number of steps required to identify relevant information on the internet.

Previous work has shown that context is gathered during post processing (after chat session ends) rather than dynamically at runtime [HPK$^+$10]. However, understanding the dynamic context (those unpredictable changes) and responding to it at runtime remains an open challenge [BHCNM01].

According to Chignell et al., “The new generation of internet which can be termed as smart internet where web entities, represented by on-line services and content, are discovered, aggregated and delivered dynamically, automatically, and interactively according to users’ needs and situations” [CCNY10]. Therefore, a smart internet needs smarter applications that can retrieve web entities (e.g., web resources) dynamically, automatically, and interactively according to user needs and situations. This thesis intends to provide PALTask as an example of such an application.

Based on the above motivation, we formulated three Research Questions (RQ). In this thesis we aim to answer the following:

- *RQ 1 : How can we automate the web search task by exploiting context in an Instant Messaging (IM) application to improve user experience?*
- *RQ 2 : How can we gather dynamic context and provide feedback at runtime in a personalized chat application in which the user has control of their own web profile?*
- *RQ 3 : What are the natural language processing techniques that can be used in a collaborative environment to improve user experience?*

1.2 Research Methodology

During a manual web search task, users know beforehand what web resources need to be search and retrieved. Users filter the results according to their goals and select useful results. However, automation of web searching tasks is challenging when the context is dynamic. Dynamic situations such as an online conversation has dynamic topics and searching occurs while carrying out the conversation. Users have to navigate back and forth between their Instant Messaging (IM) tool and browser. Furthermore, users do not have personal goals to reach when retrieving web resources.

This thesis aims to provide context-aware resource retrieval in a personalized environment, employing techniques and processes used in an IM conversation. The thesis is intended as a proof of concept for automation of resource retrieval in a dynamic environment. The IM scenario can also be replaced with email conversation, website content, business communication, or resource retrieval in corporate repositories.

Instant Messaging (IM) is one of the most popular forms of daily communication because it is fast, cheap, convenient, and reliable. Initially designed for one-on-one personal chats, it has permeated the workplace. Many businesses are choosing text-based IM in concert with phone calls and email, preferring its immediacy and streamlined efficiency in getting real-time information from partners, suppliers, customers, and colleagues working remotely.[1] In workplaces, there can also be a huge repository that can be searched while communicating with colleagues regarding policies, ideas, actions, or codes.

When instant messaging is integrated with user context, fascinating results emerge. It can simplify many complex personalized tasks. Picture yourself in a conversation with a colleague or customer. You wish to break for lunch and find a good restaurant. The application, from the context of your messages, researches local restaurants that are specialized in items you like and displays them in the conversation window. Then you simply drag and drop web resources to share your personal interests with your colleague. The shared web resource might be interesting to them, too. Results can be affected if the application can interpret the location and conversation context, along with personal preferences.

[1]Microsoft, Instant messaging for business. Retrieved Jan 2015, http://www.microsoft.com/business/en-us/resources/technology/communications/10-tips-for-using-instant-messaging-for-business.aspx?fbid=5ayGWY8cHXw

The most popular applications using context over the internet are social networking sites and chat applications such as Facebook Messenger, Gtalk, Skype, and iMessage. These applications allow users to communicate with each other with little or no context to enhance user experience.

We took an approach to contextualize contents by building an IM application called PALTask (Personalized Automated web resources Listing Task), which provides context-aware, self-adaptive capabilities. It is an application that collects dynamic context (through context gathering at runtime) and retrieves web results dynamically. PALTask reduces repetitive and mundane tasks in retrieving personalized web resources in an IM conversation. We also developed a component called ConRank (Context Ranking), which performs various operations over text such as natural language processing. It is a component of our PALTask application, helping PALTask generate more accurate, context based, personalized web resources by prioritizing keywords and retrieving more personalized results. It improves the user experience by exploiting dynamic context in an IM conversation.

ConRank analyzes a conversation by performing various operations over text such as sentiment analysis, stemming, and integrating the Personal Context Sphere (PCS) [VM10][Vil13]. The PCS is a user's preference repository that can be controlled by the user. ConRank checks for sentiments in communication text in the form of positive, negative, and neutral sentences, and also performs stemming (reducing inflected words to their word stem, base or root form) operations on text in order to make context easier to process.

1.3 Thesis Outline

This chapter introduced our research area, goals, and motivation. The remaining sections of this thesis are organized as follows.

Chapter 2 provides the problem description and related research work, which gives us an idea of what work has been done already to increase user experience in IM and other scenarios.

Chapter 3 discusses dynamic context gathering and resource retrieval including the design and implementation of our IM application PALTask.

Chapter 4 discusses retrieval of more accurate and personalized web resources using ConRank. This chapter also presents the design and implementation of ConRank.

Chapter 5 presents the evaluation of PALTask based on efficiency, effectiveness, and user experience.

Chapter 6 summarizes the contributions of this thesis and proposes future work.

Chapter 2

Problem Description and Background

This chapter presents an overview of applications that exploit context and perform various operations on text to improve user experience in Instant Messaging (IM) applications. We also discuss various language processing libraries we are exploring for keyword extraction, sentiment analysis, and stemming of keywords as well as the Personal Context Sphere (SmarterContext).

2.1 Introduction

Web search, which is an ordinary and repetitive task, often frustrates users. The challenge in the automation of such tasks is to fully understand them and execute it efficiently using the information provided. The personal context of the user, location, and conversation can be used to infer contextual information needed for retrieving web resources, thereby enhancing user experience in IM applications. Due to the complexity in gathering, mining, and providing feedback for dynamic context, the challenge is to identify and retrieve web resources dynamically [VM10].

Context analysis for the purpose of providing personalized augmentation has been demonstrated before [ZSL05]. Ubiquitous computing and existing chat technology have used context gathering for personalized communications, but most related approaches have failed to provide dynamic feedback from the context they collect. In most of the related work described in this chapter, context gathering is done as a post-processing step, rather than dynamically at runtime. We aim to demonstrate our ideas with PALTask, an IM application that uses improved context extraction and mining techniques. This IM application gathers context from a variety of sources and mines it at runtime in order to improve user experience based on contextual information.

To gather and mine context from the conversation, natural language processing (NLP) is needed. NLP is a large research area in computer science. Combined with Artificial Intelligence (AI) (which involves understanding and analytics), they support natural language comprehension using various tasks such as morphological segmentation, named entity recognition, keyword extraction, and sentiment analysis [Cho03].

For our PALTask application, we explored a few open source libraries and APIs for keyword extraction, sentiment analysis, stemming tasks, and web services.

2.2 Context-Aware Personalized Applications

Personalized applications have become ubiquitous in today's world. These applications mainly focus on user context in order to enhance user experience. Mobile applications are the best example of applications that can be personalized with user context. For example, Google Maps[1] on mobile devices gather user context dynamically and provide improved results as we continue to use it. It uses the current

[1]Google Maps. Retrieved Jan 2015, https://www.google.ca/maps

location context for route searches and suggests routes to the user based on saved searches performed.

Learning from past searches and providing a space to store personalized destinations makes it a smart context-aware application. Context-aware applications are of great interest as they can adapt to different situations and become more responsive to user needs.

Another example of a personalized context-aware application is Google's email client Gmail,[2] which uses context to provide advertisements. To show relevant advertisements, Gmail uses account information, text from email conversations, and the user's Google search queries.[3] Google also extracts keywords from user emails that have context information related to the Google calendar application. It automates the steps needed to add an entry to the user's Google calendar (e.g., a meeting, dinner, or lecture). The multi-step process is reduced to a single click to add the calendar entry.

The important entity used by Google is the users' context information — what, how, and why the user searches or performs operations with Google applications. The disadvantage of Google's advertisement model for some is their profile, which is not under their direct control. This user profile is different from the one which the user would set up themselves, providing some information to Google such as name, address, and phone number. The profile used in the advertisement model is created automatically from the user's browsing habits; users cannot make direct changes to their preferences and interests and they cannot definitively update what they do or do not want to see in advertisements, which may lead to frustration.

In a web network, there is a need for a model in which users can control their own web profile. They should be able to update their preferences, likes, and dislikes, as well

[2]Gmail. Retrieved Jan 2015, https://mail.google.com/

[3]Ads in Gmail. Retrieved Jan 2015, https://support.google.com/mail/answer/6603?hl=en

as receive automated suggestions for a personalized experience. For example, Pratik visited Toronto, and was interested in flights from Toronto to Victoria. However, after returning home to Victoria, he still received information about flight deals in his Gmail account based on previous web searches. In this thesis, we assume such a web model to gather a user's context by employing the Personalized Context Sphere (PCS), which is a concept of the SmarterContext management system proposed by Villegas [Vil13].

2.3 Context-Aware IM Applications

This section discusses IM applications that exhibit functionality similar to PALTask. These applications use dynamic or static context extraction techniques. Our application is more efficient and effective than the applications listed below because of the way we handle dynamic context, apply context extraction techniques, and use natural language processing libraries.

GaChat, as described by Satoshi et al., is built to improve awareness among chat partners and augments the chat dialogue with related information [HIHO09]. GaChat extracts only proper nouns from communication and then searches for online images and articles in Wikipedia and Google Image Search as depicted in Figures 2.1-2.2.

The authors mention in their paper that the goal of their GaChat application is to avoid misunderstanding certain topics due to low awareness. GaChat demonstrated that by extracting the proper nouns from the conversations and synchronously displaying the image or article, the quality of the conversations improved, and new topics were often suggested. Chat partners retrieve the same kind of resources which also increases their knowledge and common understanding of the topic.

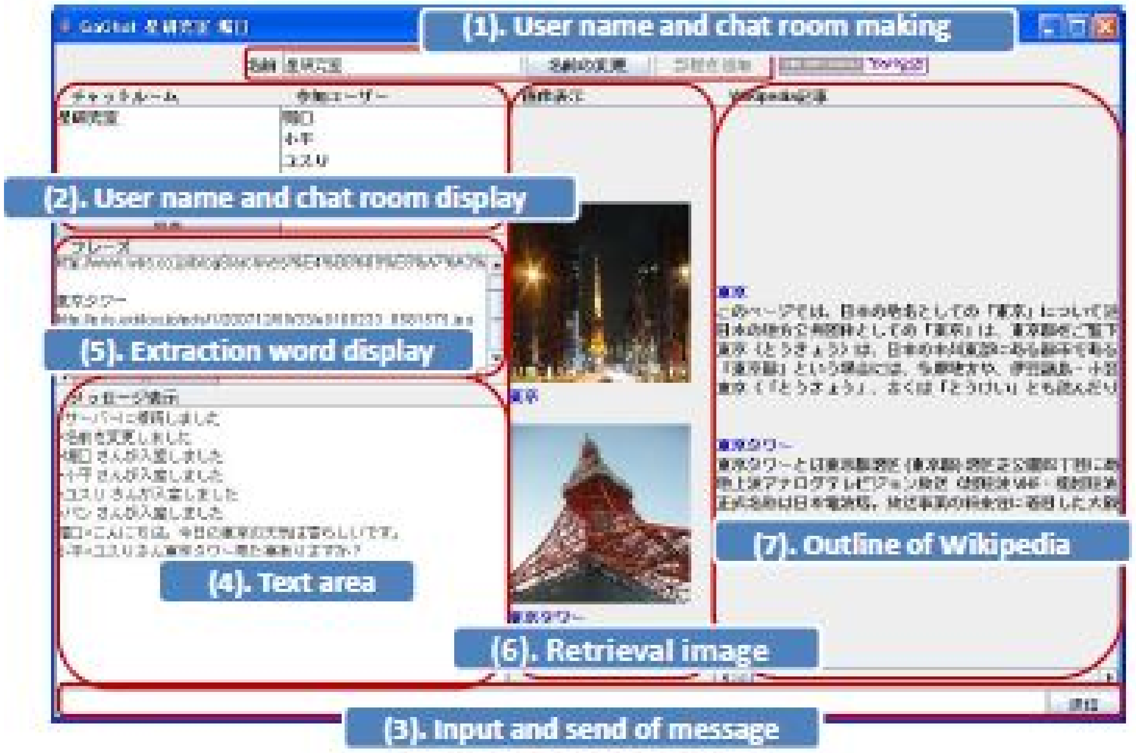

(1). User name and chat room making
(2). User name and chat room display
(5). Extraction word display
(4). Text area
(6). Retrieval image
(7). Outline of Wikipedia
(3). Input and send of message

Figure 2.1: GaChat [HIHO09]

Windows Live Messenger also has an integrated web search function and retrieval (search) button, which adds a URL to the associated proper noun. The disadvantage is that the user has to perform chat and search simultaneously as searching takes place in the users' browser, thus the user has to cut and paste between browser and IM application [HIHO09].

Another application that analyzes context in chat conversations is *ConChat* [RCRM02]. Rangnathan et al. demonstrate that chat messages can be augmented by collecting contextual information from the user to prevent semantic ambiguity between chat participants as depicted in Figure 2.3. ConChat resolves semantic ambiguities related to time, currency, date formats, and units of measurement. It collects context from various sensors such as location, lighting, and temperature. In their paper they illustrate the issue with a conversation between an American and a

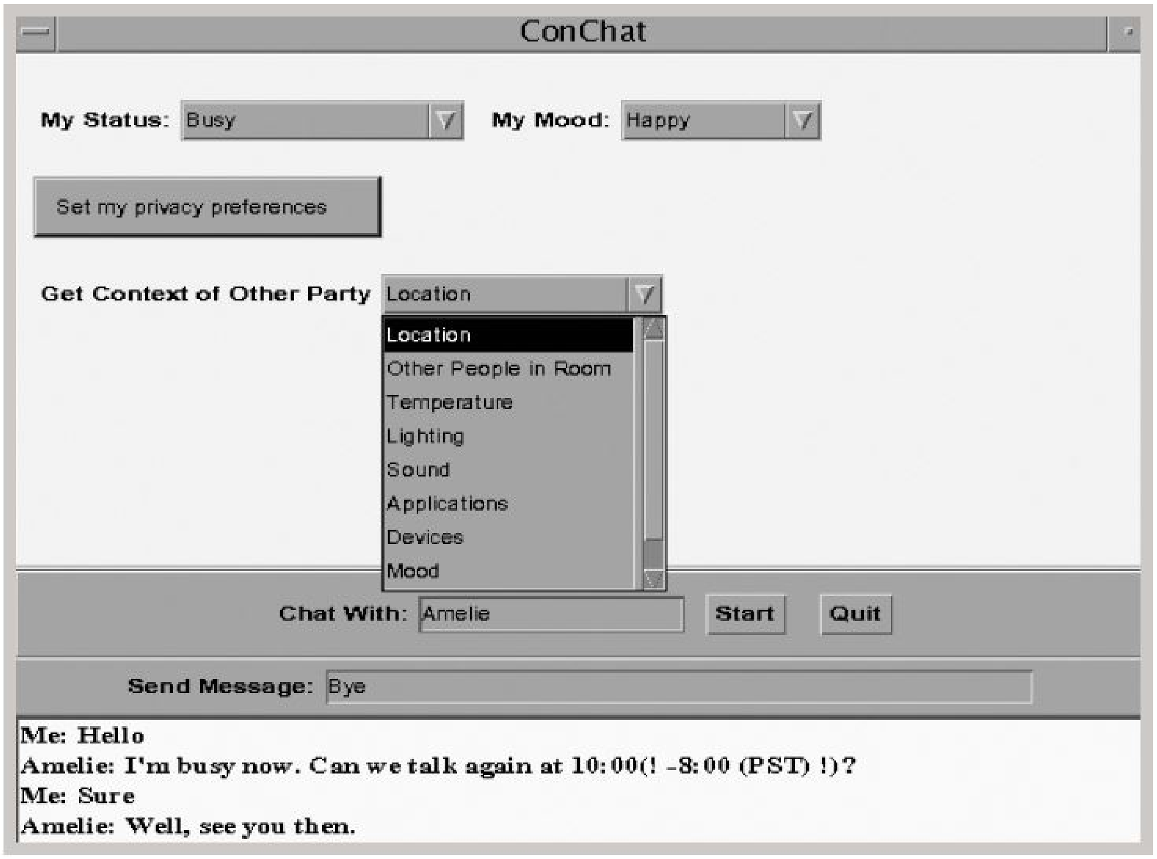
The explanation about Tokyo University of Technology	
Image	Textual information
	Established: 1947 Location: Hachioji,Tokyo,JP Website: http://www.teu.ac.jp/ etc...

Figure 2.2: GaChat [HIHO09]

Canadian. If one of them says “$10,” it is not clear whether CAD or USD is implied. ConChat resolves it using a location sensor and identifies the currency.

SemChat works with the notion of a social semantic desktop [DF04]. Semantic Desktop aims to tackle the difficulties in managing personal information in a social context. It focuses on strengthening Personal Information Management (PIM) using the contents of a user’s desktop by using semantic web standards and technologies.

Extending the semantic desktop in a social dimension, which can facilitate information distribution and collaboration, creates a social semantic desktop. SemChat extracts the relevant concepts for a particular user from conversations which are not present in the Personal Information Model (PIMO) and updates the PIMO for each user as depicted in Figure 2.4, an architecture of SemChat. It also identifies and extracts the events from chat conversations that can be annotated with a task/event scheduler. It provides a search facility for the chat-related concepts and events. The disadvantage of SemChat is that it monitors chat sessions but only analyzes data after the conversation ends. It extracts the keywords and uses ANNIE (a named



Me: Hello
Amelie: I'm busy now. Can we talk again at 10:00(! -8:00 (PST) !)?
Me: Sure
Amelie: Well, see you then.

Figure 2.3: ConChat [RCRM02]

entity recognizer) for recognizing entities like locations, people, organizations, and dates [AC10].

According to a SemChat usability study, the most exciting feature for all participants was extraction of concepts and events to provide information from Wikipedia after the chat session ended. Out of all related chat tools, SemChat is closest to our application PALTask. It contains chat analytics but does not analyze conversations at runtime. Instead, it extracts keywords, recognizes entities, and retrieves resources after the chat session has ended. PALTask has a keyword extractor (RAKE), which uses context such as location and also extracts user data from the Personal Context Sphere. It also provides feedback at runtime.

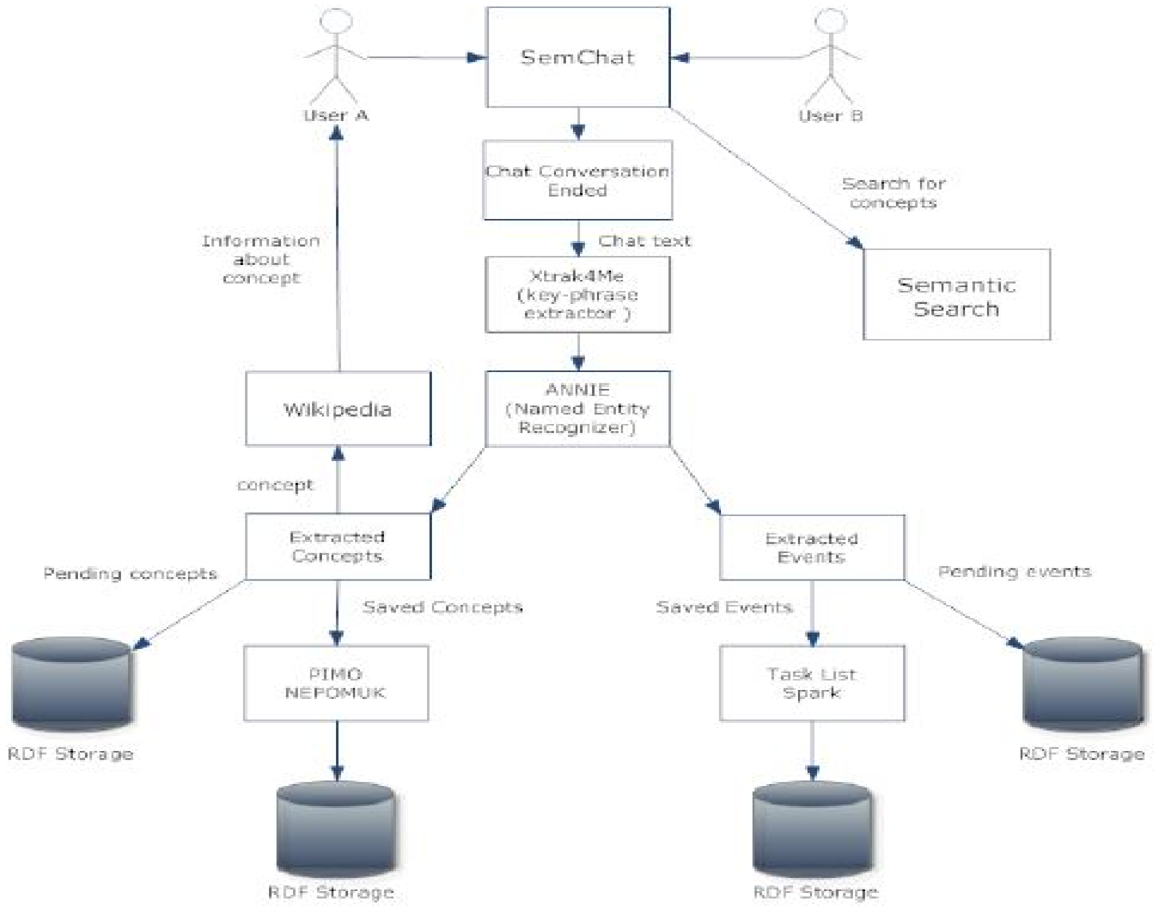


Figure 2.4: Architecture of SemChat [AC10]

2.4 Natural Language Processing Tasks

Natural Language Processing (NLP) tasks comprise information extraction and classification that are useful for context extraction and analysis. In particular, we can extract information from text or documents and label them using classifiers. In this section, we introduce the keyword extractor, sentiment analysis, and stemming libraries we explored for our research.

2.4.1 Keyword Extractor

Keywords are frequently used as a simple method of providing descriptive metadata about a collection of documents or conversations. Keywords are the essence of a conversation and can be used as search keys for finding relevant resources on the web. We evaluated several natural language keyword extractors based on various factors, such as quality of results, availability of a remote API and source code, cost, and license. The keyword extractors we investigated include RAKE,[4] Yahoo API Term Extractor,[5] World Finder Extractor,[6] Sketch Engine,[7] and Alchemy.[8] We decided to use a Python implementation of the Rapid Automatic Keyword Extraction (RAKE) algorithm for PALTask [RECC10]. It requires no training and the only input is a list of stop words. Its source code is freely available for use and the quality of results are high.

2.4.2 Sentiment Analysis and Stemming

The task of sentiment analysis is to identify the polarity of text as positive, negative, or neutral. Sentiment analysis is becoming a popular area of research in social media analysis, especially around user reviews and tweets. It is a special case of text mining generally focused on identifying opinion polarity. Its accuracy rate is approximately 80% using various algorithms [Liu12].

For simplicity, and because the training data is easily accessible, we looked at various open source text analytic tools for sentiment analysis and stemming of words. A few of the tools are Natural Language ToolKit (NLTK),[9] R Text Mining module (R

[4]RAKE implementation. Retrieved Jan 2015, https://github.com/aneesha/RAKE

[5]Yahoo term extractor. Retrieved Jan 2015,
http://developer.yahoo.com/search/content/V1/termExtraction.html

[6]World finder extractor. Retrieved Jan 2015, http://wordsfinder.com/api_Keyword_Extractor.php

[7]Sketch engine extractor API. Retrieved Jan 2015, http://trac.sketchengine.co.uk/wiki/SkE/KeywordsAPI

[8]Alchemy extractor API. Retrieved Jan 2015, http://www.alchemyapi.com/api/keyword/

[9]Natural Language Toolkit (NLTK). Retrieved Jan 2015, http://www.nltk.org/

TM),[10] General Architecture for Text Engineering (GATE),[11] and Sentiment classifiers for WEKA data mining workbench.[12] We chose NLTK as our sentiment analysis and stemming tool because it is a leading platform with built-in Python libraries. It allows us to modify code according to project needs, and all the data and datasets are freely available. It also has well structured documentation.

Stemming is a process that removes morphological affixes from words and leaves only the stem. There are various stemmers available in the NLTK Library. The Porter[13] stemmer works on various pluralized words. The Regexp[14] stemmer works on patterns provided to the stem, and the Snowball[15] stemmer is available for various languages.

2.5 Personal Context Sphere

The Personal Context Sphere (PCS) is a repository of context information relevant to users and their preferences; it is hosted by a third party and owned by users [Vil13]. Some of this information might include gender, age, favorite locations, and web sites. It is a concept of the Smart Internet, where users can integrate into dynamic context management processes of Situation-Aware Smart Software Systems (SASS) [NCCY10a]. PCS concepts are in compliance with the SmarterContext ontology, which is a model that represents the context entities proposed by Villegas [VM10].

[10]R TM (Text Mining). Retrieved Jan 2015, http://www.rdatamining.com/examples/text-mining

[11]GATE: Open source tool. Retrieved Jan 2015, https://gate.ac.uk/

[12]Weka: Data mining software. Retrieved Jan 2015, http://www.cs.waikato.ac.nz/ml/weka/

[13]Porter stemmer. Retrieved Jan 2015, http://tartarus.org/martin/PorterStemmer/

[14]NLTK stemmers. Retrieved Jan 2015, http://www.nltk.org/api/nltk.stem.html

[15]Snowball stemmer. Retrieved Jan 2015, http://snowball.tartarus.org/

2.6 Web Service APIs

Web service APIs are a method of connecting web applications via HTTP or another protocol. Currently, REpresentational State Transfer (REST) APIs are a preferred design when compared to traditional SOAP/WSDL XML based protocols. REST is an architectural style and SOAP is a standard XML based protocol communicated typically over HTTP. REST APIs are more dynamic in nature and are not restricted to XML formats like SOAP architecture. REST web services can send plain text, JSON, ATOM, and XML. In public APIs, REST is mostly used with the HTTP protocol and usually JSON is used for the structuring of data.

Retrieving web resources from a web service is an important task for our IM application. We explored many web service APIs to integrate into our tool. Out of all the APIs examined, Google web services APIs which includes Google Search,[16] Google Image Search, and the YouTube API,[17] were very well structured, efficient, and return results based on the users' needs. These Google web services are REST APIs which can send and receive text in JSON/ATOM formats. They are best suited for our purposes. Our retrieval of web resources in the IM application were keyword driven and these web service APIs include search functions based on those keywords. These web APIs provide functionalities for the retrieval of text, image, video, and audio resources. Additionally, use of these APIs is free for research purposes.

2.7 Summary

This chapter discussed context-aware IM applications, various APIs explored for keyword extraction, sentiment analysis, and stemming. Some of these IM applications

[16]Google custom search API. Retrieved Jan 2015, https://developers.google.com/custom-search/json-api/v1/overview

[17]YouTube search API. Retrieved Jan 2015, https://developers.google.com/youtube/1.0/developers_guide_python

provide feedback to users after processing or while performing the task. However, none of them provides feedback at runtime. We explored various web service APIs for the retrieval of web resources and found Google web services to be comparatively more structured and straightforward to use. We also discussed the Personal Context Sphere (PCS) which is a repository of context information relevant to users.

Chapter 3

Dynamic Context Gathering and Resource Retrieval

This chapter discusses the dynamic gathering of context and how resources can be retrieved, including a detailed picture of the design and implementation of PALTask. Further sections illustrate the components, architecture, and user experience of PALTask. This chapter also explains the Rapid Automatic Keyword Extraction (RAKE) algorithm used in our keyword extractor component [BK10].

3.1 PALTask

Personalized Automated web-resources Listing Task (PALTask) improves user experience through the automation of repetitive and ordinary tasks in order to fulfill personal goals when taking part in an IM conversation. It is a context-aware tool that gathers context from two resources: personal context spheres and the conversation itself [JBCnM13] as depicted in Figure 3.1.

First, user context is crucial. This includes aspects such as browsing history, search preferences, and interests stored in the personal context sphere. Second, the conversation between users can be analyzed dynamically to extract context.

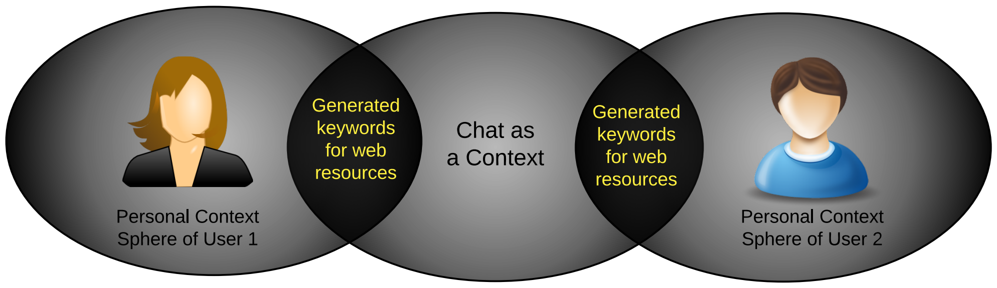


Figure 3.1: Gathering of Context

For example, during an online conversation, context analysis determines that one of the users is looking for restaurants nearby. Furthermore, the user's personal context sphere contains a preference list for restaurants (e.g., Mexican). The tool displays relevant restaurants from Google web search and other sources through context matching. This eliminates manual steps such as opening a browser, connecting to a website, searching for the preferred restaurants, and finally copying and pasting the URL into the chat.

Recommending web resources that satisfy users is challenging, as it is necessary to understand the personal interests of people. Furthermore, it is necessary to have a mechanism to identify the feelings of the user which influences to their attitude in a situation or event at a particular moment.

In order to recommend web resources of interest to users at runtime, we addressed the following research challenges. First, sentiment (i.e., conveying the attitude, opinion, or feelings of a user) is useful to determine the need for retrieval of web resources. Second, after analyzing the sentence and determining the need for retrieval of resources, keywords are extracted. Keywords are also matched from the context sphere of the user, which helps in retrieving more personalized keywords. Keywords are given to a different web service API in order to retrieve web resources.

In general, negative sentiment in a conversation indicates the user is less likely to be interested in retrieving resources, whereas positive sentiment indicates the opposite. For example, if the user does not like McDonald's, we should not retrieve resources related to McDonald's as it may lead to a higher degree of frustration. In this thesis, we are using sentiment analysis to filter out the results based on determining the polarity of positive and negative moods.

3.2 Components of PALTask

The architecture of PALTask comprises seven software components as depicted in Figure 3.2: Graphical User Interface (GUI), Server, Client, PCSManager, ConRank, Keyword Extractor, and Web Services APIs. Out of these seven components, PCSManager, ConRank, Keyword Extractor, and Web Service APIs are external services which are connected through APIs.

3.2.1 Graphical User Interface Component

The GUI's main function, as depicted in Figures 3.5-3.8, is to facilitate user interaction and display retrieved web resources automatically. The GUI provides the following widgets: chat console, web-resource list, and filtering buttons. The filtering buttons

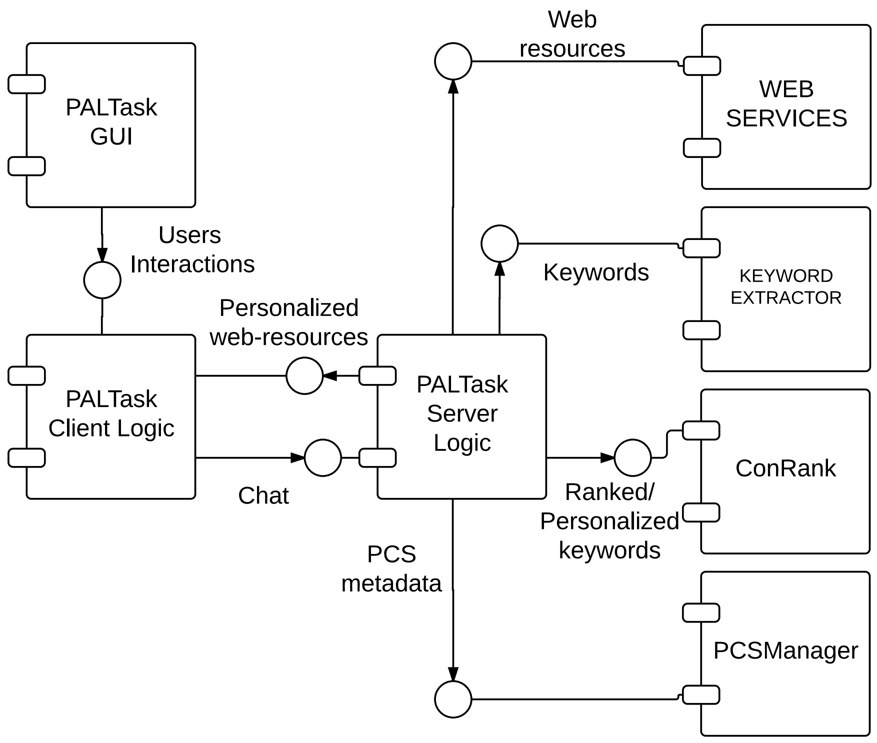
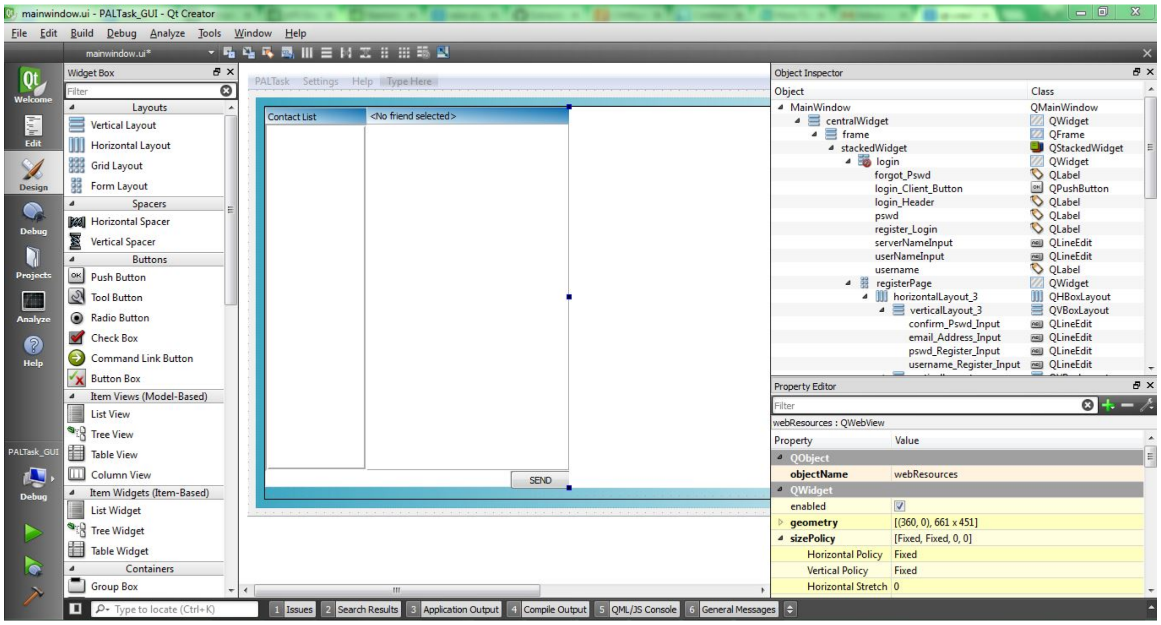


Figure 3.2: High Level Architecture of Components

provide the ability to like, delete, and share resources from different formats such as audio, video, text, and image. The menu provides, for example, chat and keyword history, and allows the user to turn off context information.

After two working prototypes for the client (built in JAVA and Python), we decided to build our GUI using QTCreator which simplifies prototype creation considerably. Selected components of QTCreator such as QT Designer, Widget box, Object inspector, and Property editor are depicted in Figure 3.3.



Object	Class
MainWindow	QMainWindow
centralWidget	QWidget
frame	QFrame
stackedWidget	QStackedWidget
login	QWidget
forgot_Pswd	QLabel
login_Client_Button	QPushButton
login_Header	QLabel
pswd	QLabel
register_Login	QLabel
serverNameInput	QLineEdit
userNameInput	QLineEdit
username	QLabel
registerPage	QWidget
horizontalLayout_3	QHBoxLayout
verticalLayout_3	QVBoxLayout
confirm_Pswd_Input	QLineEdit
email_Address_Input	QLineEdit
pswd_Register_Input	QLineEdit
username_Register_Input	QLineEdit

webResources : QWebView

Property	Value
QObject	
objectName	webResources
QWidget	
enabled	✓
geometry	[(360, 0), 661 x 451]
sizePolicy	[Fixed, Fixed, 0, 0]
Horizontal Policy	Fixed
Vertical Policy	Fixed
Horizontal Stretch	0

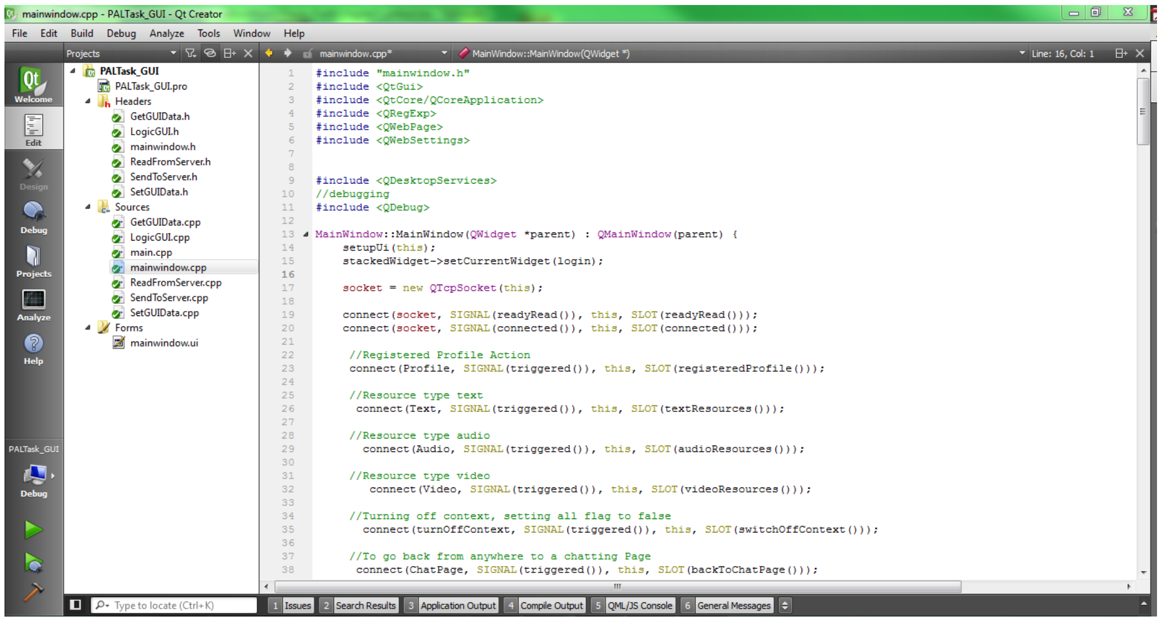
Figure 3.3: QTcreator components

GUI using QTCreator

QTCreator is a cross platform Integrated Development Environment (IDE) with an integrated code editor and QT designer. It uses the system's resources (e.g., draw windows and controls) to give the application a native look. Thus, the resulting applications look like native applications on their respective platforms (e.g., Mac, Windows, Linux, and Mobile platforms). The syntax-directed code editor of QT supports the C++ language. Similarly, QT designer is for designing and building graphical user interfaces from QT widgets.[1] The programmer can compose and customize widgets or dialogs and test them using different styles and resolutions. This all comes with no cost as QTCreator is licensed under the LGPL, which means it can also be used for commercial applications.

Designing the GUI is straightforward with QT Designer, as it integrates widgets and forms with the programmed code. QT Designer has a widget box with widgets

[1]QTCreator. Retrieved Jan 2015, http://doc.qt.digia.com/qtcreator-2.4/



```cpp
#include "mainwindow.h"
#include <QtGui>
#include <QtCore/QCoreApplication>
#include <QRegExp>
#include <QWebPage>
#include <QWebSettings>


#include <QDesktopServices>
//debugging
#include <QDebug>

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) {
    setupUi(this);
    stackedWidget->setCurrentWidget(login);

    socket = new QTcpSocket(this);

    connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()));
    connect(socket, SIGNAL(connected()), this, SLOT(connected()));

     //Registered Profile Action
     connect(Profile, SIGNAL(triggered()), this, SLOT(registeredProfile()));

     //Resource type text
      connect(Text, SIGNAL(triggered()), this, SLOT(textResources()));

     //Resource type audio
       connect(Audio, SIGNAL(triggered()), this, SLOT(audioResources()));

     //Resource type video
        connect(Video, SIGNAL(triggered()), this, SLOT(videoResources()));

     //Turning off context, setting all flag to false
       connect(turnOffContext, SIGNAL(triggered()), this, SLOT(switchOffContext()));

     //To go back from anywhere to a chatting Page
      connect(ChatPage, SIGNAL(triggered()), this, SLOT(backToChatPage()));
```

Figure 3.4: Code editor

such as Button, QTextEdit, QLineEdit, and QFrame. It also includes an object inspector which inspects object properties. As depicted in Figure 3.3, MainWindow Object from QMainWindow class contains all the graphical elements such as widgets, frame, label, textfields, buttons, and layouts. These graphical elements can be added easily using drag and drop from the widget box. Behavior of graphical elements can be assigned using the Signal and Slot mechanisms as depicted in Figure 3.4. The “connect” function is used to perform an action (Slot) on the selected menus, buttons, and forms that act as a Signal to widgets. Slots are implemented as functions that provide action on the QMainWindow class such as void MainWindow::switchOffContext(), void MainWindow::videoResources(), and void MainWindow::on_webResources_linkClicked(const QUrl &arg1).

User Interface

Figures 3.5-3.7 exhibit the GUI of PALTask with its two main pages. The first is the login page; its function is to register accounts, handle forgotten user names and

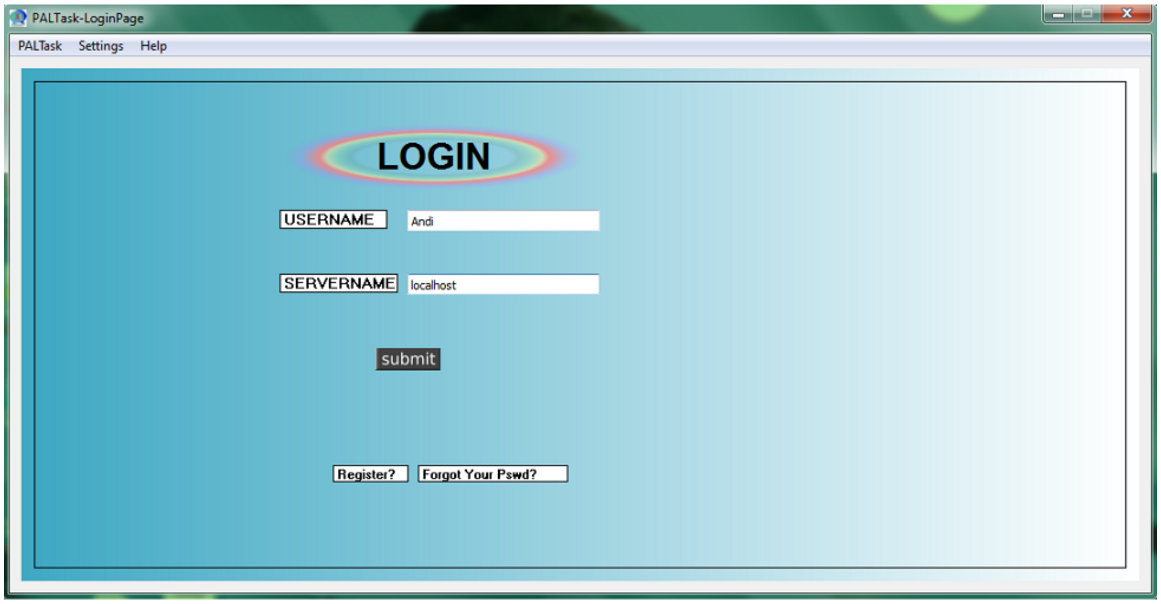


Figure 3.5: PALTask Login Screen

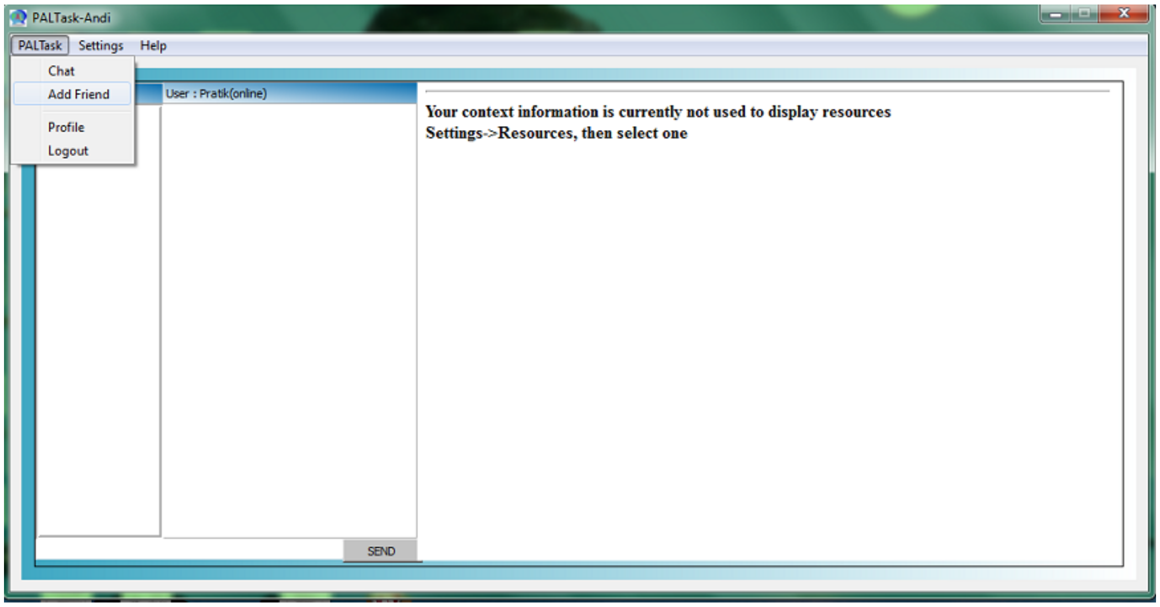
passwords, and log users in to registered accounts using the submit button. Second, the chat page has three main elements: *a*) the contact list; *b*) a conventional chat window; and *c*) the web resources list display.

The first element contains a list of friends (including their status) and a notification on the contact list if a message comes in from a friend. The second element contains the chat display window with a text input field and a send button. Finally, the last element has a tab navigation that represents the web resource format list (i.e., video, image, text, and/or audio).

As shown in Figures 3.6-3.7, PALTask also features a menu bar on all pages as follows:

PALTask

Chat: The menu button redirects to the conversation page, where people can chat and retrieve resources.



Your context information is currently not used to display resources
Settings->Resources, then select one

Figure 3.6: PALTask Menu

Add Friend: This button opens a new page where we can input details of a friend to be invited.

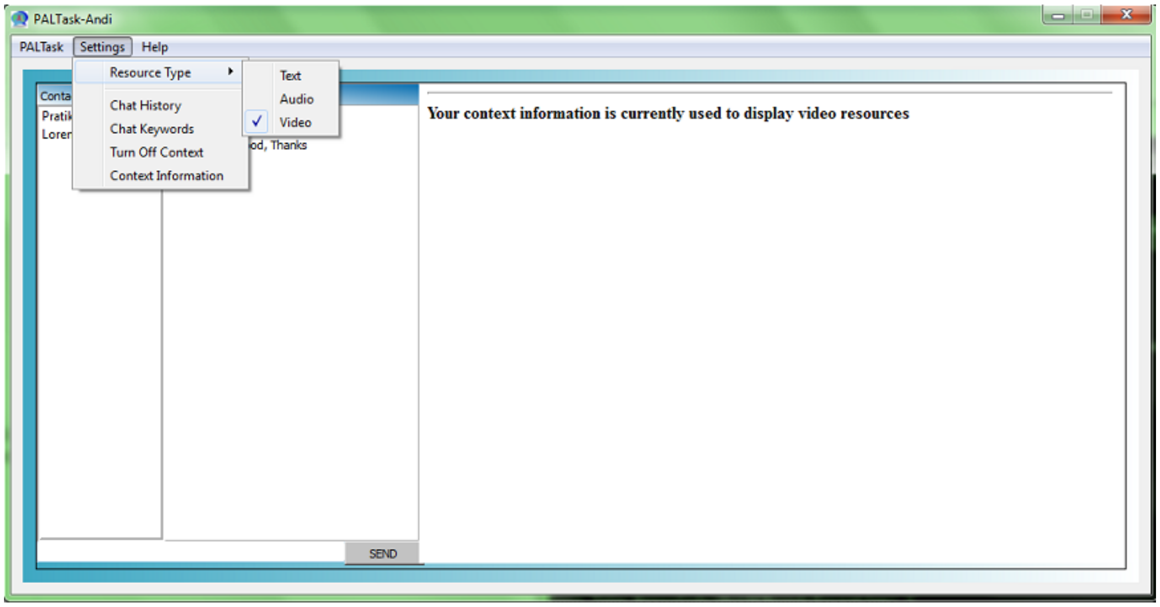
Profile: The profile page provides user details such as profile picture, name, and status. This information can also be stored and retrieved by accessing the users' personal context sphere (cf. Section 3.2.4).

Logout: To logout from the client, the logout button redirects to the login page.

Settings

Resource Type: Select the type of resource (i.e., text, video, audio and image). Enabling resource type will retrieve the resources from the respective API.

Chat History: Browse the chat history, which is stored at the client's end. Chat history, which is timestamped, is stored per chat partner. The functionality to delete chat history at any time from the user's clients is included.



Your context information is currently used to display video resources

Figure 3.7: PALTask Settings

Chat Keywords: Extracted keywords which are used for retrieval of resources are listed here. The user can modify listed keywords as acceptable or unacceptable based on their likes and dislikes. For example, if the user does not want a particular keyword to retrieve resources, the user can mark it as an unacceptable keyword. This keyword is added to the stoplist (cf. Section 3.2.5) and will not be used for retrieving resources.

Turn Off Context: Turns off collecting context information from the conversation. This feature is for users who feel that privacy/security is a concern. PALTask can act as a simple IM program instead of a context-aware one. Users can also disable the collection of location context.

Context Information: All the context information collected is stored in this page. We can also access and modify the user's personal context sphere from here. It contains the profile as set by the user in PALTask.

Help

About PALTask: All the information related to PALTask is provided here. It explains how to navigate and use the application.

Support: This displays the contact details for the PALTask support team.

3.2.2 Server

Our server component is a traditional chat management system to manage chat conversations, using sockets to connect to the client. The server includes functions that connect users, exchange messages, and control chat sessions. PALTask adheres to the traditional centralized client server architecture: clients are connected to a central server component via a network. All client messages pass through the central server, which controls all message passing. Furthermore, the server is responsible for relaying text that is to be analyzed by the ConRank and Keyword Extractor components.

The server component handles all web service APIs and has functions such as getVideoResources(), getTextResources(), and getAudioResources(). These functions return the web resources list from the respective web service APIs (e.g., YouTube,[2] Google custom search,[3] and Grooveshark[4]) when keywords are provided. The server component also handles functions such as adding friends, sending and storing messages, and keywords.

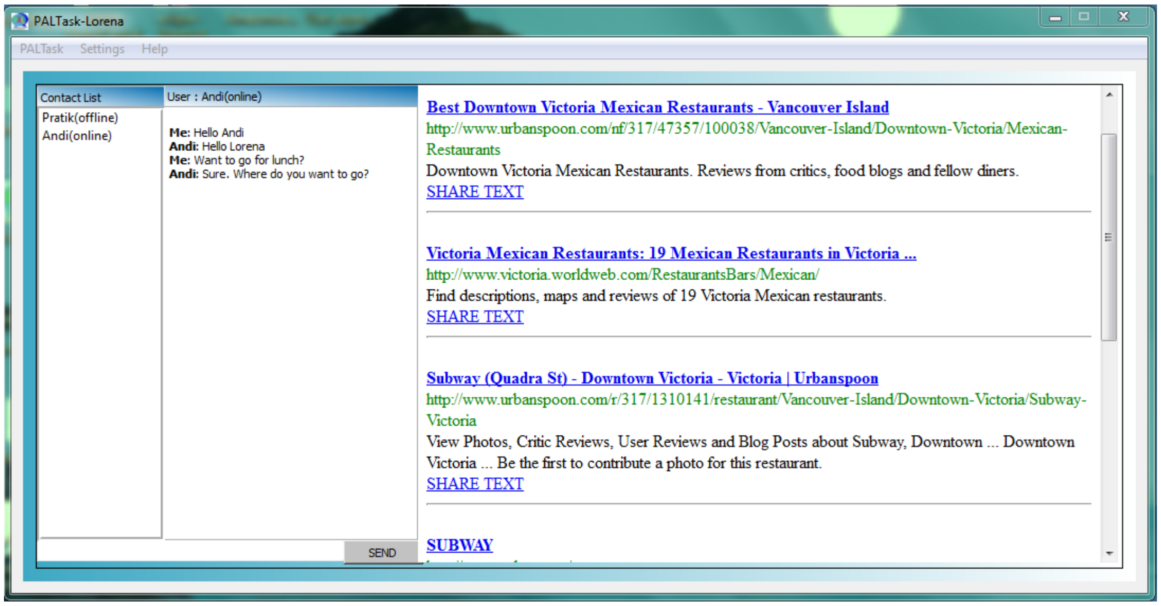
3.2.3 Client

A client connects to the server as an ordinary chat application, which includes login and communication interactions. To connect server and client, TCP sockets are used.

[2]YouTube search API. Retrieved Jan 2015, https://developers.google.com/youtube/

[3]Google custom search API. Retrieved Jan 2015, https://developers.google.com/custom-search/json-api/v1/overview

[4]Grooveshark API for songs. Retrieved Jan 2015, http://developers.grooveshark.com/



Pratik(offline)
Andi(online)

User : Andi(online)

Me: Hello Andi
Andi: Hello Lorena
Me: Want to go for lunch?
Andi: Sure. Where do you want to go?

Best Downtown Victoria Mexican Restaurants - Vancouver Island
http://www.urbanspoon.com/nf/317/47357/100038/Vancouver-Island/Downtown-Victoria/Mexican-Restaurants
Downtown Victoria Mexican Restaurants. Reviews from critics, food blogs and fellow diners.
SHARE TEXT

Victoria Mexican Restaurants: 19 Mexican Restaurants in Victoria ...
http://www.victoria.worldweb.com/RestaurantsBars/Mexican/
Find descriptions, maps and reviews of 19 Victoria Mexican restaurants.
SHARE TEXT

Subway (Quadra St) - Downtown Victoria - Victoria | Urbanspoon
http://www.urbanspoon.com/r/317/1310141/restaurant/Vancouver-Island/Downtown-Victoria/Subway-Victoria
View Photos, Critic Reviews, User Reviews and Blog Posts about Subway, Downtown ... Downtown Victoria ... Be the first to contribute a photo for this restaurant.
SHARE TEXT

SUBWAY

Figure 3.8: Personalized web resources displayed on the right

All PALTask interactions between the GUI and client logic are handled using QTCreator. Keywords, which represent the context of the conversation, are obtained from the server for displaying keyword history as a functionality. The client component accesses the user's PCS through PCSManager. It has functions which send various kinds of information to the server such as login and logout information, messages sent, and resource share requests. The client component also receives the personalized web resources list as depicted in Figure 3.8.

3.2.4 PCSManager

The PCSManager is a component that is responsible for requesting and updating the users' PCS into the SmarterContext Reasoning Engine (SCoRE) [Vil13]. SCoRE replies to the PCS in the form of an XML file representing RDF graphs. The PCSManager is comprised of two main modules: PCSReader and PCSUpdater.

The PCSReader module converts the XML file into a JSON string, which is used for context matching with conversation keywords. The PCSUpdate module updates the XML file whenever the user updates their personal context (which in turn updates the PCS). The PCSManager is accessed through the client component of PALTask. The user is identified by his/her name and email address, and the PCSManager sends requests for PCS to SCoRE for each user.

Listing 3.1 shows Pratik's PCS in XML format containing all likes, dislikes, and other personal information. These elements and values stored in the XML file are considered to be PCS keywords. Stemming is performed on these keywords for context matching with conversation keywords.

Listing 3.1: Pratik's PCS

```
   columns
<?xml version="1.0" encoding="UTF-8"?>
<pcs>
        <!-- From the Context Ontology by Villegas, 2013-->
        <pwc:user>Pratik</pwc:user>
        <gc:geoLocation type="country">Canada</gc:geoLocation>
        <gc:geoLocation type="city">Victoria</gc:geoLocation>
        <gc:geoLocation type="origin" >India</gc:geoLocation>

        <!-- From the Context Ontology for PALtask, 2013-->
        <!-- Personal Information-->
        <pi-lan language1="English" language2="Hindi">English</pi-lan>
        <pi-gender>M</pi-gender>
        <pi-age>Adult</pi-age>
        <!-- Topics of Interest (simplified version)-->
        <topicsInterest>
                <music>Bollywood</music>
                <music>Baba Sehgal</music>
                <music>Palash Sen</music>

                <sports>Badminton</sports>
                <sports>Cricket</sports>

                <technology>iOS</technology>
                <technology>Android</technology>
                <technology>Blackberry</technology>
                <development>C</development>
                <development>Java</development>
                <development>Ecplise</development>

                <food>Vegetarian</food>
                <food>Subway</food>
                <food>Tea</food>
                <food>Fairway Market</food>
        </topicsInterest>
</pcs>
```

3.2.5 Keyword Extractor

The essence of conversations can often be summarized in a few keywords. The keyword extractor component extracts those keywords from a textual representation of a conversation. It is an external service to the server component connected via an API. We used the Rapid Automatic Keyword Extraction (RAKE)[5] algorithm to extract keywords from chat messages. RAKE is document-oriented and thus does not rely on a corpus to identify keywords. Consequently, statistical analysis or frequency analysis is also unnecessary with RAKE. These aspects make RAKE attractive for use in an IM environment, where accuracy and speed are two crucial metrics.[6] Below is an overview of the RAKE algorithm used for extracting keywords.

RAKE Algorithm

Input parameters of RAKE are stop words (or a stoplist), a set of phrase delimiters, and a set of word delimiters. RAKE uses stop words and phrase delimiters to partition a document into candidate keywords. The score of keywords is calculated based on

[5]RAKE implementation. Retrieved Jan 2015, https://github.com/aneesha/RAKE

[6]Keyword extraction tutorial. Retrieved Jan 2015, https://www.airpair.com/nlp/keyword-extraction-tutorial

co-occurrences within these candidate keywords. Frequency (freq(w)) and degree (deg(w)) of a word is calculated using co-occurrence graph of words [BK10]. Steps to extract keywords are as follows:

Identify Candidate Keywords

1. Create an array of words using word delimiters.
2. Remove standard punctuation and stop words.
3. The list of contiguous candidate keywords is ready.
4. Candidate keywords are divided into individual keywords for calculating scores. For example, a sentence given for keyword extraction is “System of linear Diophantine equations”. Here, candidate keywords are “System” and “linear Diophantine equations.”

Score the Keywords

1. Create a graph of word co-occurrences (e.g., system, linear, Diophantine, and equations are plotted on x and y axis and co-occurrences of these keyword are calculated in large document).
2. Calculate word frequency (freq(w)), word degree (deg(w)) and ratio of degree to frequency $(deg(w))/(freq(w))$.
3. Individual keyword score is ratio of degree to frequency.

Adjoin Keywords

1. Look for pairs of keywords that adjoin one another at least twice in the same document and in the same order. This is to identify keywords that contain interior stop words such as *axis of evil*.

2. A new candidate keyword is created as a combination of those keywords and their interior stop words.

3. The score of a new candidate keyword is the sum of its member keywords score.

Extract Keywords

1. Write down all candidate keywords with the new scores.

2. The top one third of the scoring candidates should be selected. For example, if the number of content keywords is 37, then select the top 12 as candidates.

3. Extracted keywords are ready to use.

Evaluation of RAKE in comparison to several comparable keyword-extraction methods on a benchmark dataset of short technical abstracts shows that RAKE achieves higher precision and recall in extracting keywords [BK10].

Keyword Extractor in PALTask

In PALTask, we analyze the most recent messages sent. Sentences are built up for analysis until they are 160 characters long. We define *sentence chunk* as a group of words that are of at least 160 characters in length. We have used 160 characters as our threshold value, which seem to be a sufficient character limit for effective communication [BV04][RS10]. We implemented the formula for our semantic analysis in the ConRank component: if a sentence is less than 160 characters, add one more sentence to it until the 160 or more characters are obtained. The last sentence in the sentence chunk is not truncated even if it exceeds the character limit. The three equations below define how a sentence chunk is formed [JBCnM13].

$$Length\ (Sentence)\ <= 160\ characters \tag{3.1}$$

$$Length\ (Sentence) = Length\ (Sentence)\ +\ Length\ (Last\ Sentence) \tag{3.2}$$

$$Sentence\ Chunk = Length\ (Sentence) \tag{3.3}$$

Each time a message is sent to the ConRank component, it forms a sentence chunk. ConRank analyzes this sentence chunk to determine sentiment polarity and sends it to the keyword extractor. To obtain significant information from a conversation, analysis is performed on more than one chunk at a time. Keyword extraction occurs on the last individual chunk that was formed, as well as on the several most recent chunks. This is necessary because chat messages are often short.

Table 3.1: Extracted keywords and stop words

Sentence	**keywords**	**stop words**
I like Subway	Subway	I, like
I don't like food in Victoria	food, Victoria	I, don't, like, in
Victoria is a great place for food	Victoria, place, food	is, a, great, for

Therefore, by keeping a record of approximately the 10 most recent chunks, we are able to gather keywords representing the conversation's context more accurately. For short messages, RAKE often returns no keywords. This is due to the high frequency of stop words. Stop words as depicted in Table 3.1, are common elements in text, yet do not aid in providing unique contextual information [BTJ$^+$13]. Examples of stop words are *the, a,* and *should.* In short phrases, stop words are very frequent and proper keyword candidates are not present. In personal chat applications, text communication often doesn't follow a standard language dictionary in terms of spelling and capitalization. Spelling mistakes are frequent and remain uncorrected, and abbreviations and acronyms or "chatspeak" (e.g., LOL, BRB, TTYL) are common. Consequently, we modified the stopword list to reflect this type of text. Without the modified stopword list, chatspeak is erroneously interpreted as a keyword [BTJ$^+$13].

Table 3.2 shows the sentence, keywords extracted, stop words, and modified stop words list.

Table 3.2: Modified stop words list

Sentence	**keywords**	**stop words**	**modified stop words list**
LOL, It's a hilarious movie	hillarious, movie	It's, a	LOL, It's, a

Currently, we alter the stoplist manually to include words commonly found in chat text. In the future, this will be replaced with an automatically generated stop words list that is also domain specific to chat. Techniques on how to generate these stop word lists are illustrated by Berry et al. [BK10].

3.2.6 ConRank

Context Ranking (ConRank), as the name suggests, handles the ranking and personalization of keywords. The ConRank component improves PALTask's results by reducing the number of resources retrieved and personalizing the results. PALTask can also work without the ConRank component, but results are not personalized. ConRank is an external service connected through an API to the Server, PCSManager, and Keyword extractor.

The ConRank component performs following activities to improve PALTask's results: *a*) sentiment analysis; *b*) stemming of words; *c*) integration of PCS and keywords from conversation; *d*) managing location context in the PCS; and *e*) provides a keyword ranking algorithm. ConRank uses natural language processing tasks such as sentiment analysis and stemming of words for personalization. Sentiment analysis filters the results by analyzing sentences as positive, negative, and neutral; stemming of words is useful for keyword matching with the users' PCS. The component has inputs from the PCSManager, Server, and Keyword Extractor, and outputs to the

Keyword Extractor and Web service API. ConRank first passes filtered chat to the keyword extractor, which then passes the keywords back to ConRank.

ConRank receives each user's PCS as input from the PCSManager. The PCS is context matched with the extracted keywords using a stemming technique, which in turn provides a more personalized keyword list. Analysis and ranking of keywords is done by calculating the score of candidate keywords and sentiment polarity (cf. Chapter 4).

3.2.7 Web Service API

PALTask has used various web service APIs for retrieving web resources. Here in this section we describe two of them: Google custom search[7] API and YouTube[8] API. These two APIs are used for retrieving text and video resources respectively.

Google Custom Search API

Google has deprecated its web search API, but we can still use its custom search to explore the entire web. Steps to create a Google custom search engine that searches the entire web or mentioned websites are:

- From the Google custom search homepage (`http://www.google.com/cse/`), click the link: Custom Search Engine (CSE).
- Type a name and description for your search engine.
- Under "define your search engine" (the sites to search box), enter at least one valid URL (e.g., www.google.com). We can also have other websites in the box. But to search the whole web, just enter any one to pass this screen.

[7]Google custom search API. Retrieved Jan 2015, https://developers.google.com/custom-search/json-api/v1/overview

[8]YouTube search API. Retrieved Jan 2015, https://developers.google.com/youtube/1.0/developers_guide_python

- Choose the CSE edition, accept the terms of service, and then click "next". Select the layout option and click "next".
- Click any of the links under the next step section to navigate to your control panel.
- In the left-hand menu, under control panel, click "Basics".
- In the search preferences section, select "search the entire web but emphasize the included sites".
- Click save changes.
- In the left-hand menu, under Control Panel, click "sites".
- Delete the site you entered during the initial setup process.
- Now your custom search engine will search the entire web.

Google custom search enables you to search the entire web or a collection of websites. We can create a search engine that searches only the contents of one website (site search), or one that focuses on a particular topic from multiple sites. The JSON/Atom Custom Search API helps in retrieving and displaying the search results from Google Custom Search programmatically. With this API, we can use RESTful requests to get either the web search or image search results in JSON or Atom format.[9]

JSON/Atom Custom Search API can return results in one of two formats (JSON is the default data format). JSON/Atom Custom Search API requires the use of an API key, which users can obtain from the Google cloud console. For experimental research purposes, we have used free CSE in which the API provides 100 search

[9]Google custom search API. Retrieved Jan 2015, https://developers.google.com/custom-search/json-api/v1/overview

queries per day for free. If we need more, we may sign up for billing in the Cloud Console. Additional requests cost $5 per 1,000 queries, up to 10k queries per day.

Representational State Transfer (REST) in the JSON/Atom Custom Search API is somewhat different from the traditional REST. Instead of providing access to resources, the API provides access to a service. As a result, the API provides a single URI that acts as the service endpoint. We can retrieve results for a particular search by sending an HTTP GET request to its URI or pass the details of the search request as query parameters. The format for the JSON/Atom Custom Search API URI is:

```
https://www.googleapis.com/customsearch/v1?{parameters}
```

Three query parameters are required with each search request:

- API key: Use the key query parameter to identify your application.
- Custom search engine ID: Use either cx or cref to specify the custom search engine we want to use to perform this search.
  - Use cx for a search engine created with the Control Panel.
  - Use cref for a linked custom search engine (does not apply for Google Site Search).
  - If both are specified, cx is used.
- Search query: Use the q query parameter to specify your search expression.

All other query parameters are optional. Here is an example of a request that searches a test Custom Search Engine for keyword "lectures":

GET `https://www.googleapis.com/customsearch/v1?key=INSERT_YOUR_API_KEY&cx=0123456789:omuauf_lfve&q=lectures`

In above GET request, API Key is INSERT_YOUR_API_KEY, cx is 0123456789, and search query (q) is lectures. Below is the Python code for web search from custom search API:

Listing 3.2: WebSearch.py

```
import httplib2
import sys
import pprint
import time
from apiclient import discovery
def main(argv):
  query = ["Mark Hamil"]
  if (len(argv) > 0) :
    if "," in argv[1] :
      query = argv[1].split(",")
    else :
      query = argv[1]
  # Create an httplib2.Http object to handle our HTTP requests .
  http = httplib2.Http()
  # Construct the service object for the interacting with the CustomSearch API.
  service = discovery.build('customsearch', 'v1',
  developerKey='abcdefghi123456789', http=http)
  results = ""
  for item in query[:-1] :
    test = item + ""
    res = service.cse().list( q=test, cx='0123456789:-oitaexu1tu', num=3,
    safe="high", gl="ca", start = 1, googlehost="google.ca").execute()
    time.sleep(1)
    for items in res['items']:
      try :
        url=items['link']
        title=items['title']
        snippet=items['snippet'].replace("\n", " ")
      except IndexError :
        pass
      results += url + "\n" + title + "\n" + snippet + "\n"
  searchresults = results.encode('utf-8')
  print searchresults
# For more information on the CustomSearch API you can visit:
# https://developers.google.com/custom-search/v1/using_rest
# For more information on the CustomSearch API Python library surface you can visit:
# https://developers.google.com/resources/api-libraries/documentation/customsearch/v1/python/latest/
if __name__ == '__main__':
  main(sys.argv)
```

YouTube API

YouTube API is also an API from Google services. We need to enable the service from Google Cloud Console and an API key is needed to access YouTube API.[10] It is a REST API similar to the Google Custom Search API. Below is the Python code for video search from YouTube API:

Listing 3.3: YouTube.py

```
#!/usr/bin/python
import sys
from apiclient.discovery import build
from optparse import OptionParser
# Set DEVELOPER_KEY to the "API key" value from the "Access" tab of the
# Google APIs Console http://code.google.com/apis/console#access
# Please ensure that you have enabled the YouTube Data API for your project.
DEVELOPER_KEY = "abcdefghi123456789"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
def youtube_search(options):
youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=DEVELOPER_KEY)
search_response = youtube.search().list(q=options.q, part="id,snippet",
maxResults=options.maxResults).execute()
videos = []
channels = []
```

[10]YouTube search API. Retrieved Jan 2015,
https://developers.google.com/youtube/1.0/developers_guide_python

```
playlists = []
for search_result in search_response.get("items", []):
  if search_result["id"]["kind"] == "youtube#video":
   videos.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["videoId"]))
  elif search_result["id"]["kind"] == "youtube#channel":
   channels.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["channelId"]))
  elif search_result["id"]["kind"] == "youtube#playlist":
   playlists.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["playlistId"]))
if __name__ == "__main__":
  my_keywords = ""
  for item in sys.argv[1:]:
        my_keywords += item + " "
  if len(my_keywords) == 0 :
    my_keywords = "Hausi Muller"
  parser = OptionParser()
  parser.add_option("--q", dest="q", help="Search term", default=my_keywords)
  parser.add_option("--max-results", dest="maxResults", help="Max results", default=5)
  (options, args) = parser.parse_args()
  youtube_search(options)
```

3.3 Architecture of PALTask

The architecture of PALTask consists of seven components as defined in Section 3.2. In Figure 3.9, we describe how all these components interact with each other.

The GUI component interacts with the client for functionalities such as chat conversations, retrieving resources, turning off context, web resources display, chat history, and keyword history. All functions of the context aware IM client are a click away from the GUI.

When the user is logged into PALTask, the Client component interacts with PCSManager to request the user's PCS. The server interacts with various clients and works as a simple standalone chat server. It also records chat data from the client component and sends it to the ConRank component for analysis. The ConRank com-

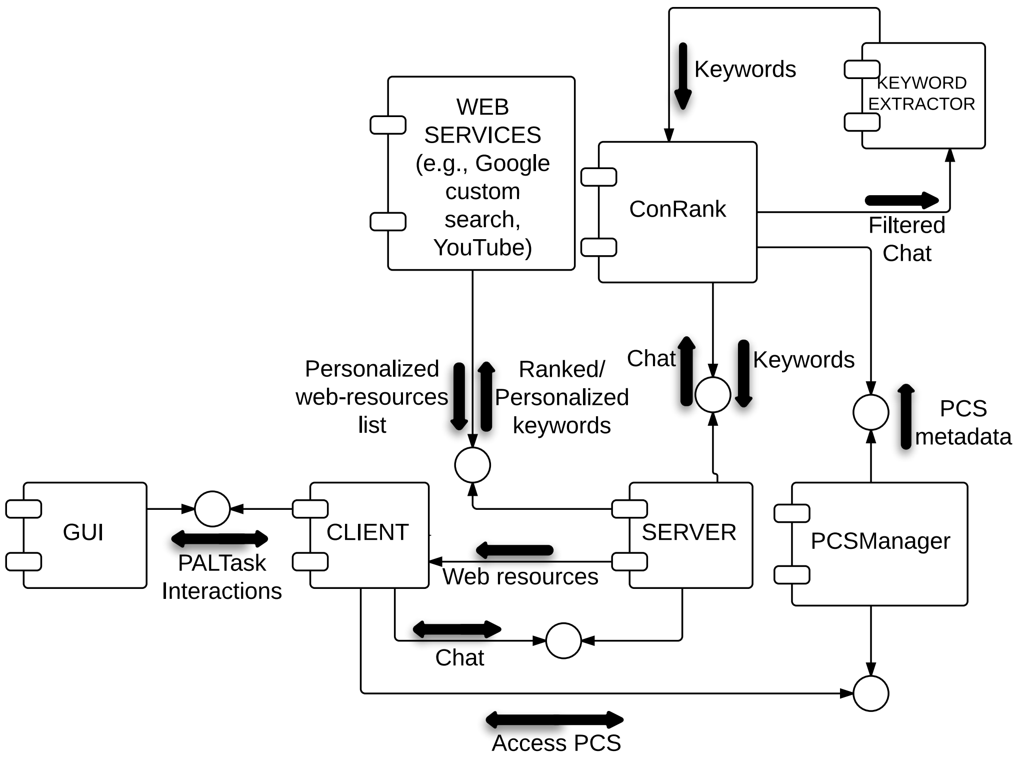


Figure 3.9: Detailed Component Architecture

ponent analyzes chat messages using sentiment analysis. After identifying the polarity of sentences, it sends the filtered chat to the keyword extractor. Keywords are extracted from the filtered chat and the extracted keywords are returned to the ConRank component for personalization. Personalization is achieved using PCS metadata by performing context matching. PCS metadata is obtained from the PCSManager in the form of a JSON string, which contains the PCS of each user. Keyword stemming is performed on keywords retrieved from chat and PCS metadata to perform context matching. If the context is matched, then more personalized keywords are retrieved. Ranking of keywords is performed using our keyword ranking algorithm. Personalized ranked keywords are passed to various web services, which provide the personalized web resources list to the server. The server sends the top five resources retrieved from each keyword to the client which are displayed in the resources pane of the user's GUI.

3.4 User Experience

First, we ensure that the user's experience is consistent across all devices as we have used QTCreator to create a cross platform application. Furthermore, using a concept like the PCS, which allows users to control their own web profile for personalized applications, greatly increases user experience. The application has the ability to retrieve personalized resources automatically and share it with the chat partner. The chat application is context aware and has self-adaptive capabilities. It continually configures and reconfigures itself, and provides feedback to the user while keeping its complexity hidden. Some of the features of self-adaptability include modifying the stop list, dynamically updating the user's PCS, and retrieving resources as the mood of the user changes dynamically using sentiment analysis. The users experience in this application is not confined to only chat with a partner, but includes context aware chat which saves time and automates web searching by exploiting context.

3.5 Summary

This chapter introduced PALTask, the Personalized Automated Listing web resources Task, which is a proof of concept for personalized automated applications in an IM scenario. It is an application that can automate the repetitive steps in a web search by exploiting context information. This application provides an overview of how the user experience can be increased by task automation, using the users' context and other contextual information gathered before or during an IM conversation. Feedback at runtime is provided in the form of retrieved resources based on the context provided. The architecture of PALTask, with all of its seven components, are explained in detail. This chapter also describes the dynamic context gathering algorithm RAKE, which is implemented as a keyword extractor. Further, it discusses how to create and use Google web services such as the Google Custom Search API and YouTube API.

Chapter 4

Personalization of Web Resources

This chapter focuses on the analysis of personalization techniques and discusses the design and implementation of Context Ranking (ConRank) in detail. ConRank improves personalization of web resources by providing ranked personalized keywords that can be fed into web services. To provide ranked personalized keywords, it exploits context gathered and performs various operations on text as depicted in Figure 4.1. Most importantly, it is an external service used as a component for PALTask.

ConRank performs the following activities:

- Sentiment analysis
- Stemming of words
- Integration of PCS and keywords from conversation
- Managing location context in the PCS
- Provides a keyword ranking algorithm

To succeed with task personalization, we can either look for factors involved in the success of tasks, or failures to be eliminated. In our IM scenario, a large number of

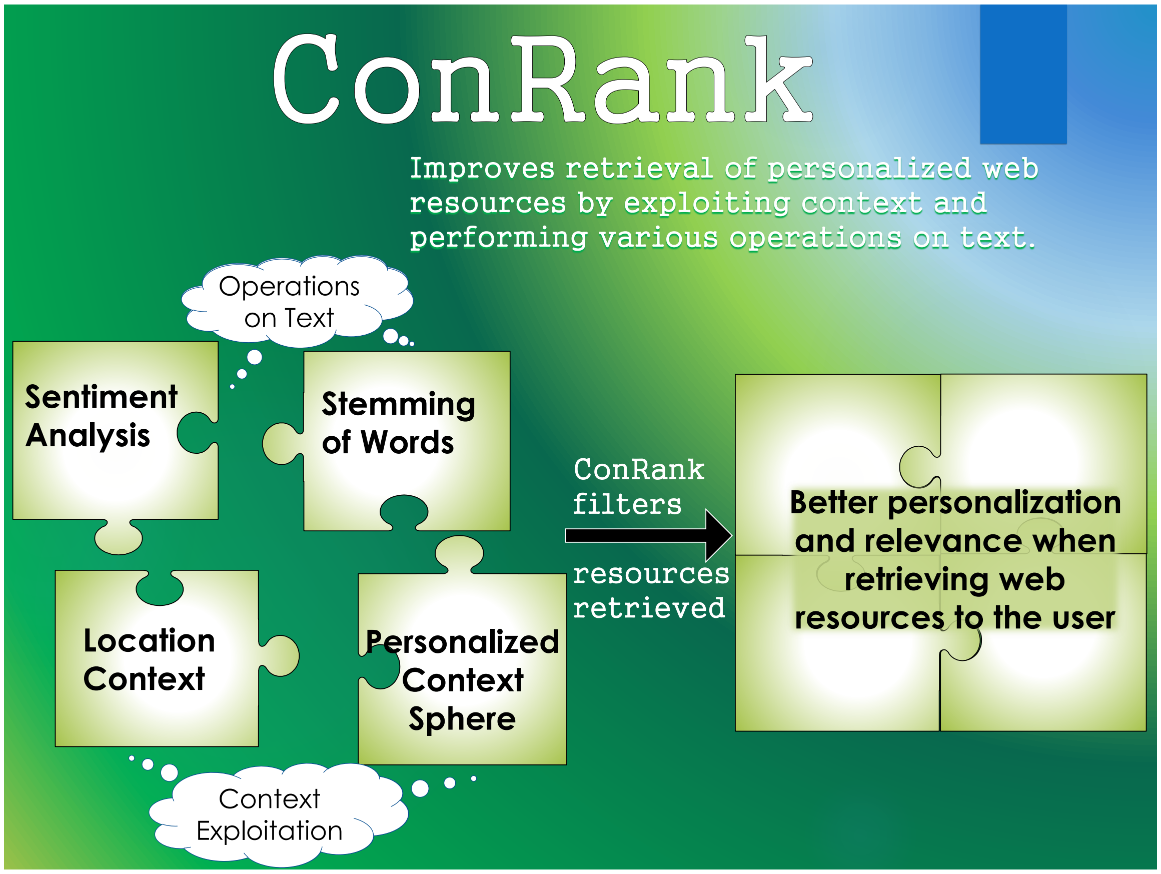

ConRank
Improves retrieval of personalized web resources by exploiting context and performing various operations on text.
Operations on Text
Sentiment Analysis
Stemming of Words
Location Context
Personalized Context Sphere
Context Exploitation
ConRank filters resources retrieved
Better personalization and relevance when retrieving web resources to the user

Figure 4.1: ConRank Overview

irrelevant web resources are failures and it decreases user experience. These failures can be eliminated by taking the users frame of mind into account. Otherwise, the user will become frustrated if PALTask retrieves a large and irrelevant number of resources. Personalization, properly implemented, brings focus to the task at hand and delivers an experience that is user-oriented, quick to inform, and relevant. Poorly implemented personalization complicates the user experience and orphans content.[1] We illustrate the implementation of personalization in a way that simplifies the complexity associated with delivering and consuming rich, dynamic, personalized content.

[1]Personalization is not Technology: Using Web Personalization to Promote your Business Goal. Retrieved Jan 2015, http://boxesandarrows.com/personalization-is-not-technology-using-web-personalization-to-promote-your-business-goal/

4.1 Sentiment Analysis

In the PALTask initial prototype without ConRank, each message sent by the user results in the retrieval of web resources. It frustrates users with the large number of irrelevant web resources. To overcome this problem, we needed techniques to analyze conversations and filter web resources based on personalization.

Sentiment analysis is the process of extracting opinions, emotions, and attitude from text. It is done by classifying the contextual polarity of a given text as positive, negative, or neutral. The accuracy of sentiment analysis is approximately 80%. Sentiment analysis on Twitter feeds, reviews from shopping sites, and analysis of social networks is an exciting approach in machine learning. It ranges from predicting the U.S. elections to movie sales from blogger opinions to analysis within social networks [BMZ11][CGR+11][MG06].

We used the sentences listed in Table 4.1 to experiment with our sentiment analysis technique.

Table 4.1: Experiments of Sentiment Analysis

Sentence	**Label**
Dr. Hausi Müller is a professor at the University	neutral
I am proud to be Dr. Hausi Müller's student	positive
Dr. Hausi Müller doesn't like sloppy work	negative

The first sentence is a fact and doesn't exhibit any sentiment. In the other two sentences, the words "proud" and "sloppy" indicate positive and negative sentiment, respectively.

In PALTask, we used sentiment analysis as a technique to make things personalized by sensing the current mood of the user and reducing the number of sentences to be processed by the keyword extractor. The sentence mood can be positive, negative, or neutral. We assume that the user prefers to retrieve resources based on extracted keywords when the mood of user is likely to be positive. We rank the keywords based on the polarity of a sentence, but retrieve resources based on the top five keywords.

4.1.1 Sentiment Analysis Using NLTK API

Natural Language Tool Kit (NLTK) is a leading Python library for performing sentiment analysis which uses the Naive Bayes classification method.[2] The Naive Bayes classifier is a probabilistic classifier based on the Bayes' theorem [TSK06]. The sentiment is determined by computing the classification probabilities of subjectivity and polarity of the sentence. The text under classification is classified as subjective or objective. If the text is objective then a label of *neutral* is returned. Otherwise, a polarity classifier is used to determine the users mood as *positive* or *negative*. We use sentence chunks (cf. Chapter 3) for sentiment analysis. To increase the user experience, we analyzed the users mood, since it can change dynamically.

Here is a synopsis on how we perform sentiment analysis using the NLTK API:

- NLTK API provides a corpus of movie reviews in which reviews are categorized as positive, negative, and neutral.
- The trainable classifier Naive Bayes is used to train the data.
- The feature extraction method returns a simple dictionary mapping a feature name to a feature value. In this method, a simplified "bag of words" model is used, if the word is found in a bag of words then it would be a feature with a value of "True". The feature extraction method is illustrated as follows:

[2]Natural Language Toolkit (NLTK). Retrieved Jan 2015, http://www.nltk.org/

$$def\ words\ feats(words):$$
$$return\ dict([(word, True)\ for\ word\ in\ words])$$

- The movie reviews corpus has 1,000 positive files and 1,000 negative files. Out of these 2,000 reviews, 75% are used as a training set and the remaining as a test set. In other words, 1,500 training instances and 500 test instances.
- The classifier training method involves token tuples of the form [(feats, label)], where *feats* is a feature dictionary and *label* is the classification label.
- In our case, feats is of the form {word: True} and label is either "*pos*" or "*neg*".

Below is the Python code for training and testing a Naive Bayes classifier on the movie review corpus. The accuracy was approximately 73%.[3]

Listing 4.1: SentimentAnalysis.py

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews

def word_feats(words):
 return dict([(word, True) for word in words])

negids = movie_reviews.fileids('neg')
posids = movie_reviews.fileids('pos')

negfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'neg') for f in negids]
posfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'pos') for f in posids]
```

[3]Sentiment analysis. Retrieved Jan 2015, http://text-processing.com/docs/sentiment.html

```
negcutoff = len(negfeats)*3/4
poscutoff = len(posfeats)*3/4

trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]
print 'train on %d instances, test on %d instances' % (len(trainfeats), len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
classifier.show_most_informative_features()
```

4.1.2 Challenges in Sentiment Analysis

Sentiment analysis is an active research field. Accuracy of around 80% has been achieved using various methods of processing in different domains such as movie reviews, and sentiment from blogs and twitter [Liu12][SC10]. Sentiment analysis is a domain specific problem which needs labeled training data within the same domain to build an accurate sentiment analysis system. For example, the words we used to describe positive context for chat do not necessarily describe movies in the same way. Different domains can have different meanings when describing the text, for example, a "cold" beverage is good but a "cold" politician is bad [OSH06].

Another challenge is to judge the dynamic nature of human beings. For instance, negative sentiments are uniquely expressed. Sometimes criticism is expressed in a polite manner, making it difficult to recognize negative sentiment.

In the realm of machine learning, there are no reliable methods to detect sarcasm, irony, or other forms of expression where the literal meaning is opposite to what is intended. We are not dealing with pragmatic competence in this thesis (which argues based on intended meaning). However, the accuracy of sentiment analysis is not perfect, as it requires a deeper understanding of context in which the sentence is said[Tur02]. User reviews or chat conversations are unstructured, natural language texts. Interesting information such as opinions have to be extracted from the reviews or IM conversations for further analysis which is usually done with the aid of various natural language processing (NLP) techniques. Chat conversations also have chat-speak (e.g., LOL, ROFL, and OMG), which expresses some opinion or sentiment. These words are hard to recognize as opinion if they are not in a database. Usually a database of opinion words (including the strength of those words) is maintained for sentiment analysis.

The Sentiment Orientation (SO) of an expressed opinion can be determined using a database of opinion words with their predicted SO. The SO is a real-number measure of positive or negative sentiment in a phrase. Our scenario does not require the SO (strength) of the opinion words. For instance, both "excellent" and "good" represent positive sentiment, but we know that the sentiment implied by "excellent" is much stronger. We need to retrieve web resources if the sentiment is positive, but are not very concerned about the positiveness of a word; we therefore do not have to take strength into account.[4]

4.1.3 Sentiment Analysis in the PALTask Implementation

We used NLTK in PALTask as one of our services. The primary analysis requirement is to do an HTTP post using the URL[5] with the "text" that needs to be recognized. In response, it returns two JSON objects: *label* and *probability.* The text of the returned label is either positive (pos), negative (neg), or neutral (neutral). In addition, the probability of a "pos" and "neg" label is given, which adds up to 1. Neutral has a standalone probability; if neutral is greater than 0.5, then the label will be neutral. Otherwise, the probability of pos and neg, whichever is higher, determines the text.

[4]Sentiment analysis is hard. Retrieved Jan 2015, http://idibon.com/why-is-sentiment-analysis-hard/

[5]URL for HTTP post. Retrieved Jan 2015, http://text-processing.com/api/sentiment/

The returned value is a JSON object. For example, it returns a “200 ok” response on success, which is as follows:

```
{
    ‘‘label’’: ‘‘pos’’,
    ‘‘probability’’: {
        ‘‘pos’’: 0.74,
        ‘‘neg’’: 0.26,
        ‘‘neutral’’: 0.3
    }
}
```

The “400 bad request” error message is returned if no text value is provided or the text exceeds 80,000 characters. We are using NLTK’s free API, and it has a daily request limit; if we exceed that daily request limit, then “503 throttled” is returned.[6]

Here are the parameters that are required with the HTTP post:

- The “text” that needs to be analyzed and is less than 80,000 characters; and
- The language of the text which needs to be analyzed (the default is English).

4.1.4 Analysis of Chat Conversation

For each sentence its polarity is returned from the NLTK API. The polarity reflects a positive sentiment if the probability value is between (0.5-1] or a negative sentiment if the value is between [0.0-0.5). A value of [0.5] shows the neutrality of a sentence.

The basic motivation of performing sentiment analysis in a chat tool is to improve user experience by retrieving personalized web resources based on the users current mood. Sentiment analysis, which provides a filter based on the polarity of a sentence,

[6]Sentiment analysis. Retrieved Jan 2015, http://text-processing.com/docs/sentiment.html

can be used to direct the retrieval of web resources. If the sentiment of the sentence is negative, the tendency to retrieve web resources is decreased and a low priority is assigned to the keywords. If the sentiments are positive, the tendency to retrieve resources is increased and a high priority is assigned to the keywords. If the mood is neutral, a high priority is still assigned to keywords to retrieve resources, because neutral behavior shows the sentence is basically a fact.

The table below shows the current analysis of a chat conversation and keyword retrieval.

Table 4.2: Keywords priority

Sentence	**Probability**	**Label**	**Keywords Priority**
Victoria is a great place for food	neutrality: 0.6, polarity: 0.4	neutral	High
I like Subway	pos: 0.7, neg: 0.3	positive	High
I don't like food in Victoria	pos: 0.3, neg: 0.7	negative	Low

4.2 Stemming of Words

Sentiment analysis, as explained in the previous section, works as a standalone application/service for injecting personalization into an IM application such as PALTask. To gather personalized keywords from a user's personal context sphere we need to use stemming of keywords. Otherwise, we would fail to hit sub personalized keywords from the conversation keywords.

Stemming is the linguistic normalization of a word mostly used in information retrieval (IR). Stemming reduces the inflected words to their word stem, base, or root form. For example, words such as *technologies* and *technology* can be reduced to *technolog* as a stem form. Two important types of algorithms are used in stemming; *lookup algorithms* and *suffix stripping algorithms.* Lookup algorithms are used in stemmers to look up the inflected form in a lookup table. All inflected forms must be explicitly present in the tables to match the word. Suffix stripping algorithms remove suffixes from the word and retrieve the stem. The disadvantage of suffix stripping algorithms is that the resultant word might not be a real word in that language.[7]

In the NLTK library, stemming can be done using various algorithms that remove and replace word suffixes to arrive at a common root form of the word. Stemming algorithms are present for different languages, such as English, French, German, Danish, or Dutch. In this thesis we only deal with the English language. The main stemming algorithms for English are Lovins,[8] Porter Stemming,[9] Lancaster Stemming,[10] and Snowball Algorithm.[11] All differ in accuracy, precision, and recall.

4.2.1 Porter Stemming Algorithm

We used the Porter stemming algorithm which removes the common morphological and inflectional endings from words in English [Por80]. The algorithm is similar to the normalization process usually performed when reducing the redundant letters from a word. The algorithm has been implemented in various programming languages (C, $C\sharp$, Java, Perl, Python, .Net, and many more), and has been widely adopted.

In any suffix stripping program for IR, two points must be kept in mind. First, the suffixes are being removed simply to improve IR performance, and not as a linguistic exercise. This means that it would not be at all obvious under what circumstances a suffix should be removed, even if we could exactly determine the suffixes of a word by automatic means. In our case, we don't care if suffix removal results in a non-English word, as we need this algorithm just for a context match from both

[7]Stemming Algorithms. Retrieved Jan 2015, http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[8]Lovins Algorithm. Retrieved Jan 2015, http://snowball.tartarus.org/algorithms/lovins/stemmer.html

[9]Porter stemmer. Retrieved Jan 2015, http://tartarus.org/martin/PorterStemmer/

[10]Lancaster background. Retrieved Jan 2015, http://www.comp.lancs.ac.uk/computing/research/stemming/Links/background.htm

[11]Snowball Algorithm. Retrieved Jan 2015, http://snowball.tartarus.org/

sides (PCS and keywords extracted from chat). Second, the success rate of a suffix stripping algorithm would be significantly less than 100%, regardless of the evaluation process. For example, if "sand" and "sander" are conflated, then probably "wand" and "wander" would be conflated as well. The problem here is that the "er" is treated as a suffix, but in "wander" it is part of the stem.[12]

Algorithm Overview

1. Identify vowels and consonants as "v" and "c", respectively, in a word that needs to be processed for stemming.

2. Recognize sequences of consecutive vowels and consonants with length greater than 0, such as vvv... as V and ccc... as C, respectively.

3. Transform all words or part of words into the following four expressions:

$$CVCV....C$$
$$CVCV....V$$
$$VCVC....C$$
$$VCVC....V$$

   The four expressions illustrated above can be represented in a single expression as [C]VCVC...[V], where [C] and [V] represent arbitrary presence of consonants and vowels.

[12]Suffix stripping algorithm. Retrieved Jan 2015, http://tartarus.org/martin/PorterStemmer/def.txt

4. Rebuild the expression [C]VCVC...[V] as [C](VC)m[V], where (VC)m (representing VC) is repeated m times. Some examples are:

$$m = 0 \quad TR, EE, TREE$$

$$m = 1 \quad TROUBLE, OATS, TREES$$

$$m = 2 \quad TROUBLES, OATEN$$

5. Consider the conditional substitution for removing a suffix as in the below rule:

$$(condition)S1- > S2$$

Where S1 and S2 are suffixes. This rule means that if the word ends with suffix S1, it can be replaced by S2 if the condition is satisfied.

6. Express the condition in terms of m, for example,

$$(m > 1)EMENT- >$$

Here S1 is EMENT and S2 is null. To satisfy the condition, m should be greater than 1. This would map REPLACEMENT to REPLAC, as it is a word part for m=4. The condition can also be *P, *v*, *d, (asterisk represents any character) which means the stem ends with P, contains a vowel (v), and ends with a double consonant, respectively.

7. Consider if there are multiple rules defined, then the rule which is chosen is the longest matching S1 for a given word.

The detailed description of this algorithm [Wil06], with all the rules and conditions is also available on the web.[13]

4.2.2 Stemming in the PALTask Implementation

Stemming in PALTask is similar to the HTTP post in sentiment analysis explained in Section 4.1. The stemming process performs the HTTP post in ConRank using the URL[14] with encoded data containing the "text." In response, a JSON object is returned and its text attribute contains the stemmed text. For example, a JSON object that returns a "200 ok" response on success is as follows:

```
{
        ''text'': ''stemmed text''
}
```

The "400 bad request" error message is returned if no text value is provided, the language is not compatible with the stemmer, or the text exceeds 60,000 characters. In addition, if the NLTK's free API daily request limit is exceeded, then "503 throttled" is returned.[15]

Here are the parameters that are required with the HTTP post:

- The "text" for the stem. It should be less than 60,000 characters;
- The language of the text to be analyzed (the default is English); and
- The stemmer algorithm used (the default is the Porter stemmer).

The table below shows an example of keywords extracted and transformed to its stem or root form.

[13]Porter Algorithm. Retrieved Jan 2015, http://tartarus.org/martin/PorterStemmer/
[14]URL of HTTP post. Retrieved Jan 2015, http://text-processing.com/api/stem/
[15]Sentiment analysis. Retrieved Jan 2015, http://text-processing.com/docs/sentiment.html

Table 4.3: Stemming of words

Keywords	**Stem or Root Form**
Development, Developed	Develop
Technologies, Technology	Technolog
Musical, Music	Music

4.3 Integration of PCS and Keywords from Conversation

The integration of personalized keywords from the PCS and the conversation is performed using context matching. Our approach mines the context sphere of each user and matches it to the keywords extracted from the conversation. The process in ConRank looks for the stemmed keyword in the PCS and returns personalized keywords. Personalized results are retrieved based on the context information associated with each users preferences, and will therefore return different results for the same type of query for different users.

As mentioned in Section 4.2, the Porter stemming algorithm using NLTK API is used to provide keywords in their stem or root form. The stemming process occurs in two places: *i*) keywords from the PCS; and *ii*) keywords from the conversation. First, stemming is performed over PCS keywords and then over the keywords extracted from the conversation. The stemmed keywords from the conversation are context matched with the stemmed PCS keywords. If matching takes place, then the PCS retrieves sub keywords that are personalized to the user.

For example, if the sentence "Which web technologies are you working on?" is used in PALTask, the keywords for this query would be "web" and "technologies." ConRank would return personalized results for different users for the same keywords. Although the keyword "web" is generic and will return the same result for all users, "technologies" can have context matching in the users PCS and can have more personalized keywords.

Therefore, the keyword "technologies" can have sub keywords in the PCS and could return different results for each user. For instance, a user's PCS keyword "technology" has sub keywords such as "Javascript," "PHP," and "HTML," and the keyword from conversation is "technologies." Stemming is performed on keyword from conversation returning "technolog" as the stem or root form. Similarly, stemming is applied to the PCS and the context is matched at "technolog," which returns new sub keywords such as "Javascript," "PHP," and "HTML."

4.4 Managing location context in the PCS

In Section 1.1 we mentioned an example of retrieving a list of nearby restaurants when colleagues are talking about eating lunch together. The user's location context helps to retrieve personalized resources related to location. Context is collected dynamically using GPS sensors on mobile devices and IP addresses on computers. Location context is stored in the user's PCS using the PCSUpdater module in the PCSManager of PALTask. If we already have stored the location information in the PCS, our PCSReader module can retrieve the location information from it and provide location keywords through context matching.

We also modified keywords in the PCS for PALTask to help identify location context. Previously, the PCS had only one keyword associated to location, but we manually added other similar keywords to help identify location context in chat. For example, keywords such as "place," "nearby," "location," "position," and "locality" can also return location context values from the PCS.

4.5 Keyword ranking algorithm

The keyword ranking algorithm ranks keywords based on the polarity of the sentence chunk and candidate keyword score calculated from the RAKE algorithm. Appendix A.1 contains the Python source code used for keyword extraction and calculation of candidate keyword score.

To gather accurate personalized web resources and to reduce irrelevant results, ConRank analyzes each sentence chunk and the last 10 sentence chunks from a conversation between two users. The last 10 sentences form the basis of the larger context structure that is derived from the whole conversation.

Each sentence chunk: The polarity of each sentence chunk is retrieved from NLTK API and the keywords are retrieved from the RAKE algorithm. If the polarity is positive or neutral, then the user is likely to retrieve web resources. We consider neutral because those sentence chunks are facts and do not show negative sentiments. The ranking of keywords from the same sentence chunk is performed by the candidate keyword score from the RAKE algorithm. Preference is given to personalized keywords retrieved from the PCS when retrieving web resources.

Last ten sentence chunks: The polarity of sentences (from which keywords are retrieved) and the candidate keyword score of each keyword are used to determine the rank of keywords for the last ten sentence chunks. The application recognizes the sentence to which a keyword belongs and identifies its polarity as positive, negative, or neutral. We define the Polarity factor as the weight assigned to the keywords based on the polarity of their sentence chunk. Higher weights are assigned to keywords of sentence chunks with positive polarity, as the user in this case is more likely to retrieve web resources. On the contrary, lower weights are assigned to sentence chunks with negative polarity as the user in that case is less likely to retrieve web resources. For example, if the sentence is negative and a keyword is extracted from a negative sentence, the polarity factor assigned to the keyword is -1. Table 4.4, exhibits the polarity factor for positive, negative and neutral sentences.

Table 4.4: Polarity Factor for Ranking

Sentence Polarity	**Polarity Factor**
Positive	1
Negative	-1
Neutral	0

We define ConFactor as the factor when ranking keywords in the last 10 sentence chunks. It is a summation of both the polarity factor and the candidate keyword score of a keyword. The higher the ConFactor, the higher the rank of a keyword. The ConFactor is calculated using the below equation to rank the keywords of the last 10 sentence chunks:

$$ConFactor = Polarity\ Factor + Candidate\ Keyword\ Score \tag{4.1}$$

Top 5 ConFactors exhibits top 5 keywords that can be given to web services for retrieving resources. We use top 5 keywords as they show higher precision and recall for web resources. The preference is given to personalized keywords retrieved from the PCS when retrieving web resources.

4.6 Example of PALTask using ConRank

The ConRank component has a collection of activities (mentioned at the beginning of this chapter), which are performed on sentence chunks. Given below is an example of a conversation on which an analysis has been made using the mentioned activities.

Conversation example:

User 1: Hello User 2

User 2: Hello User 1

User 1: What technologies are you working on?

User 2: Currently, I am working on Java, Ruby, and Scala.

User 1: ok, that sounds great. I have been using Java from quite long time now. Do you use any Java web frameworks?

User 2: Yes, I do a lot of web programming using Struts and Google Web Toolkit.

Activity 1 - Sentiment Analysis: The first activity that is performed on the sentence chunk is sentiment analysis. For the above mentioned conversation example, the analysis has been run on User 1 whose conversation is marked in red as depicted in Table 4.5. The sentiment analysis returns a JSON object from the NLTK API that provides label and probability of the chunk. The analysis on the sentence chunk of our chat example exhibits positive polarity. The returned JSON object is given below:

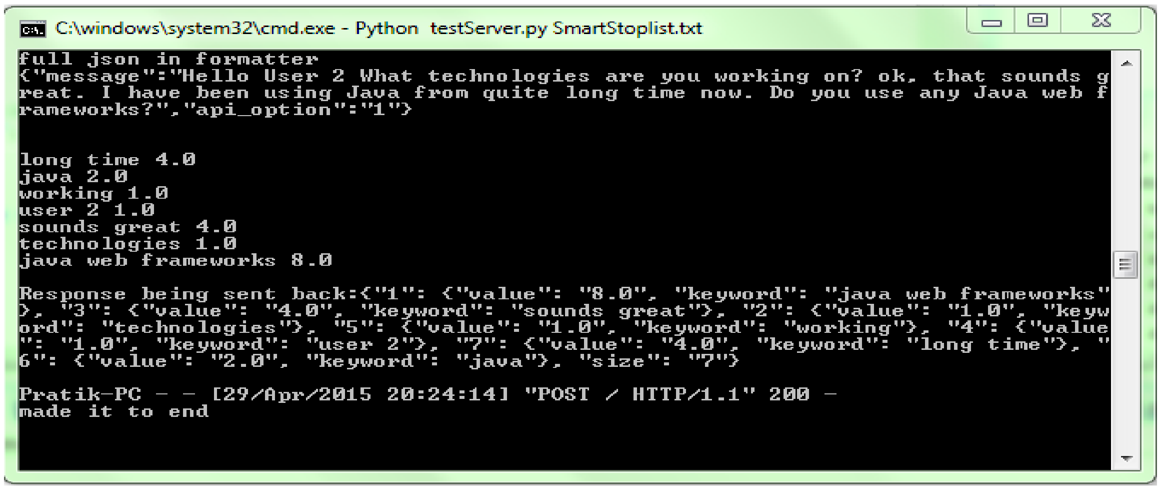
```
{
    ``label'': ``pos'',
    ``probability'': {
        ``pos'': 0.8,
        ``neg'': 0.2,
        ``neutral'': 0.2
    }
}
```

Activity 2 - Keywords Extraction: After performing sentiment analysis, keywords are extracted from the conversation using a RAKE algorithm. The candidate

Table 4.5: Sentiment analysis on chat example

Sentence	Probability	Label
Hello User 2 What technologies are you working on? ok, that sounds great. I have been using Java from quite long time now. Do you use any Java web frameworks?	pos: 0.8, neg: 0.2	positive

keyword score is calculated from all keywords extracted. The keywords retrieved from the chat example and their candidate keyword score is depicted in Figure 4.2 and Table 4.6


C:\windows\system32\cmd.exe - Python testServer.py SmartStoplist.txt

```
full json in formatter
{"message":"Hello User 2 What technologies are you working on? ok, that sounds g
reat. I have been using Java from quite long time now. Do you use any Java web f
rameworks?","api_option":"1"}


long time 4.0
java 2.0
working 1.0
user 2 1.0
sounds great 4.0
technologies 1.0
java web frameworks 8.0

Response being sent back:{"1": {"value": "8.0", "keyword": "java web frameworks"
}, "3": {"value": "4.0", "keyword": "sounds great"}, "2": {"value": "1.0", "keyw
ord": "technologies"}, "5": {"value": "1.0", "keyword": "working"}, "4": {"value
": "1.0", "keyword": "user 2"}, "7": {"value": "4.0", "keyword": "long time"}, "
6": {"value": "2.0", "keyword": "java"}, "size": "7"}

Pratik-PC - - [29/Apr/2015 20:24:14] "POST / HTTP/1.1" 200 -
made it to end
```

Figure 4.2: Keywords extracted with their candidate scores

Table 4.6: Candidate keywords scores

Keyword extracted	**Candidate keyword score**
Java web frameworks	8.0
long time	4.0
sounds great	4.0
java	2.0
technologies	1.0
working	1.0
user 2	1.0

Activity 3 - Stemming: Extracted keywords are given to the stemmer. In this stage the root form of conversation keywords and PCS keywords are retrieved. As depicted in Table 4.7, the keyword “technologies” from the mentioned chat conversation example is context matched on *technologies* from the conversation and *technology* from the PCS. Personalized keywords for User 1 that are returned from User 1's PCS keyword "technology" are: Java, Spring MVC, and Python.

Table 4.7: Stemming on chat example

Keyword from the conversation	**Keyword from the PCS**	**Stem or Root Form**	**Personalized keywords retrieved**
Technologies	Technology	Technolog	Java, Spring MVC, and Python

Activity 4 - Ranking of Keywords: The sentence chunk analyzed from the chat example is of positive polarity. Therefore, the top 5 keywords for retrieval of web resources are selected based on their candidate keyword score. A keyword ranking algorithm gives preference to personalized keywords retrieved from the PCS.

Activity 5 - Web Service APIs: Finally the following personalized keywords are given to web services such as Google web search:

- Java
- Spring MVC
- Python
- Java web frameworks
- Long time
- Sounds great

The analysis described above is only for one sentence chunk. If analysis has to be done on 10 sentence chunks, the polarity of all sentence chunks are identified. Following above mentioned activities, the top 5 keywords from each sentence chunk with their candidate keyword scores are retrieved. The ConFactor is calculated for each keyword based on the keyword candidate score and polarity factor associated with a keyword which is described in the keyword ranking algorithm 4.5.

The higher the ConFactor for the keyword, the higher the rank of the keyword. Top 5 keywords with the higher ConFactor are chosen and given to web service APIs for retrieval of web resources. An example of calculating ConFactor is described in Table 4.8 by considering the sentence chunk 1, 3, and 4 whose contextual polarities are positive, negative, and neutral, respectively.

Table 4.8: Example of analysis on three sentence chunks using ConFactor

Sentence Chunk	**Keyword number**	**Sentence Polarity**	**Polarity Factor**	**Candidate score of keywords**	**ConFactor**
1	1	Positive	1	5	6
1	2	Positive	1	1	2
3	1	Negative	-1	8	7
3	2	Negative	-1	2	1
3	3	Negative	-1	4	3
4	1	Neutral	0	3	3
4	2	Neutral	0	1	1

Top five ConFactors in the Table 4.8 exhibits top 5 keywords that are given to web services for retrieval of web resources from larger context gathered.

4.7 Summary

This chapter presented the ConRank component of PALTask for ranking personalized context results. It provided an overview of how we can implement natural language processing techniques such as sentiment analysis, stemming of words, and PCS integration to increase user experience. The above mentioned techniques can also be applied in any context-aware resource retrieval tool. It also presents a keyword ranking algorithm for ranking of keywords in sentence chunks.

Chapter 5

Evaluation

The evaluation of our application and the techniques used are based on the accuracy of the application's results. Our evaluation shows that the techniques used in ConRank improve the retrieval of web resources, making them more personalized to the user. As a result, ConRank demonstrably improved user experience in PALTask. The evaluation of our application is completed using multiple factors of a usability study that measures an applications ability to achieve specific goals such as retrieval of resources. The factors involved in this study are:

1. Efficiency: the time and number of steps taken to reach the goal.
2. Effectiveness: the degree of accuracy in the attained goal compared to the planned goal.
3. User Experience: the user's level of satisfaction with the application.

5.1 Efficiency

Efficiency is based on the number of steps required to automate web searches and retrieve personalized resources. Another parameter is the time saved searching for those results. This evaluation factor shows how many steps are required to retrieve the preferred result while working through various operations on text such as keyword extraction, sentiment analysis, and stemming.

A user involved in a conversation in a collaborative environment knows what their preferences are and what they want to share. The user usually opens a browser to a particular website such as YouTube or Google Search and enters their keywords. They may or may not share the retrieved results with the other party. Our application helps by extracting the resources dynamically and automatically. As a result, the application reduces manual steps taken and a great deal of time is saved.

5.2 Effectiveness

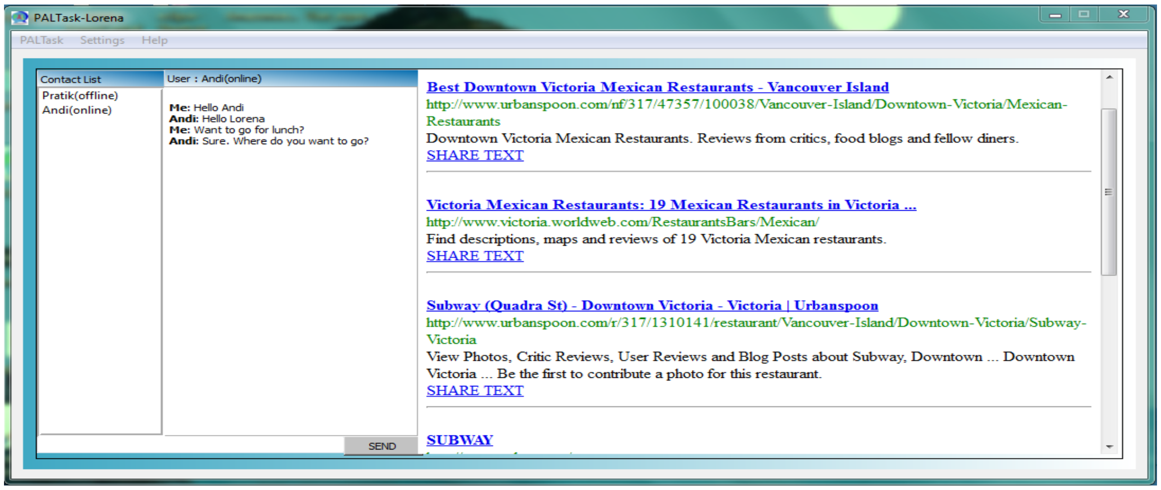
The effectiveness of the application is evaluated based on the quality of web resources retrieved. The evaluation is performed on PALTask by comparing the tool's results with and without ConRank, qualitatively analyzing the accuracy and personalization of the results. The planned set of user preferences and conversation was developed in advance. The keywords retrieved based on preferences and conversation are therefore known beforehand, as is the personalized web resource list.

We can therefore analyze the results by evaluating PALTask with and without ConRank, comparing the planned and actual results. The accuracy of the results can be divided into three categories: low, moderate, and high, thereby evaluating the effectiveness of our application.

For example, Andi is conversing with Lorena about having lunch together. From Lorena's Personal Context Sphere, we know her preferences towards food, and we have gathered other information from the planned conversation. Any results retrieved in the application could be highly, moderately, or barely accurate, comparing to the planned results.

5.3 User Experience

The level of satisfaction can be used to measure user experience. Is the user frustrated, satisfied, or dissatisfied with the search results? A scale from one to ten has been used to measure user experience, in addition to open-ended feedback, which can be used to improve PALTask.



Me: Hello Andi
Andi: Hello Lorena
Me: Want to go for lunch?
Andi: Sure. Where do you want to go?

Best Downtown Victoria Mexican Restaurants - Vancouver Island
http://www.urbanspoon.com/nf/317/47357/100038/Vancouver-Island/Downtown-Victoria/Mexican-Restaurants
Downtown Victoria Mexican Restaurants. Reviews from critics, food blogs and fellow diners.
SHARE TEXT

Victoria Mexican Restaurants: 19 Mexican Restaurants in Victoria ...
http://www.victoria.worldweb.com/RestaurantsBars/Mexican/
Find descriptions, maps and reviews of 19 Victoria Mexican restaurants.
SHARE TEXT

Subway (Quadra St) - Downtown Victoria - Victoria | Urbanspoon
http://www.urbanspoon.com/r/317/1310141/restaurant/Vancouver-Island/Downtown-Victoria/Subway-Victoria
View Photos, Critic Reviews, User Reviews and Blog Posts about Subway, Downtown ... Downtown Victoria ... Be the first to contribute a photo for this restaurant.
SHARE TEXT

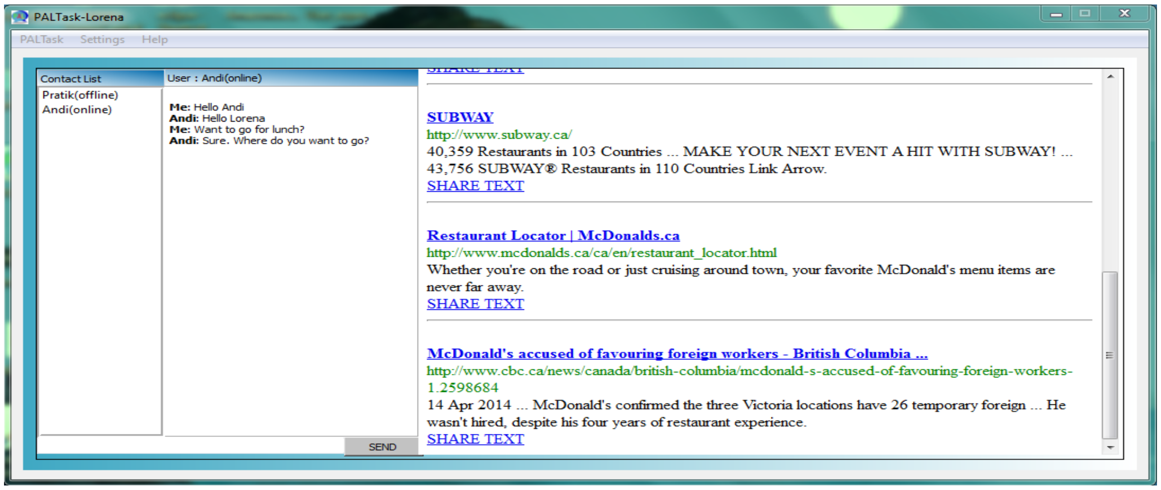
SUBWAY

Figure 5.1: Participant 1's screen, chat, and retrieved resources

5.4 Experiment 1

Our first participant, whose context sphere is filled with her preferences, talked about having lunch together with the second participant. Her context sphere has personalized keywords associated with lunch and food (Subway, Mexican food, and McDonald's). PALTask, in conjunction with ConRank, retrieves personalized resources, which includes Subway, Mexican food, and McDonald's (cf. Figures 5.1-5.2).

Figure 5.3 shows the second participant's screen, in which Participant 1 shared the resources reflecting her interests. On the other hand, PALTasks results without ConRank are merely a listing of restaurants in Victoria (using location context).



Pratik(offline)
Andi(online)

Me: Hello Andi
Andi: Hello Lorena
Me: Want to go for lunch?
Andi: Sure. Where do you want to go?

SUBWAY
http://www.subway.ca/
40,359 Restaurants in 103 Countries ... MAKE YOUR NEXT EVENT A HIT WITH SUBWAY! ... 43,756 SUBWAY® Restaurants in 110 Countries Link Arrow.
SHARE TEXT

Restaurant Locator | McDonalds.ca
http://www.mcdonalds.ca/ca/en/restaurant_locator.html
Whether you're on the road or just cruising around town, your favorite McDonald's menu items are never far away.
SHARE TEXT

McDonald's accused of favouring foreign workers - British Columbia ...
http://www.cbc.ca/news/canada/british-columbia/mcdonald-s-accused-of-favouring-foreign-workers-1.2598684
14 Apr 2014 ... McDonald's confirmed the three Victoria locations have 26 temporary foreign ... He wasn't hired, despite his four years of restaurant experience.
SHARE TEXT

Figure 5.2: Participant 1's screen, chat, and retrieved resources

In Figure 5.4, Participant 1 talked about how much she hates the weather in Toronto. No results are retrieved as the sentence carries negative sentiment. In addition, she also mentioned that she loves the weather in Victoria and she is looking forward to a job there. Certainly, this sentence was positive and retrieved two resources from each of the Victoria keywords: tourism, jobs, and weather (Figures 5.5-5.6). The results are based on sentiment analysis as depicted in Table 5.1, which is a part of ConRank. Otherwise, results are retrieved every time a message is sent and no personalization takes place.

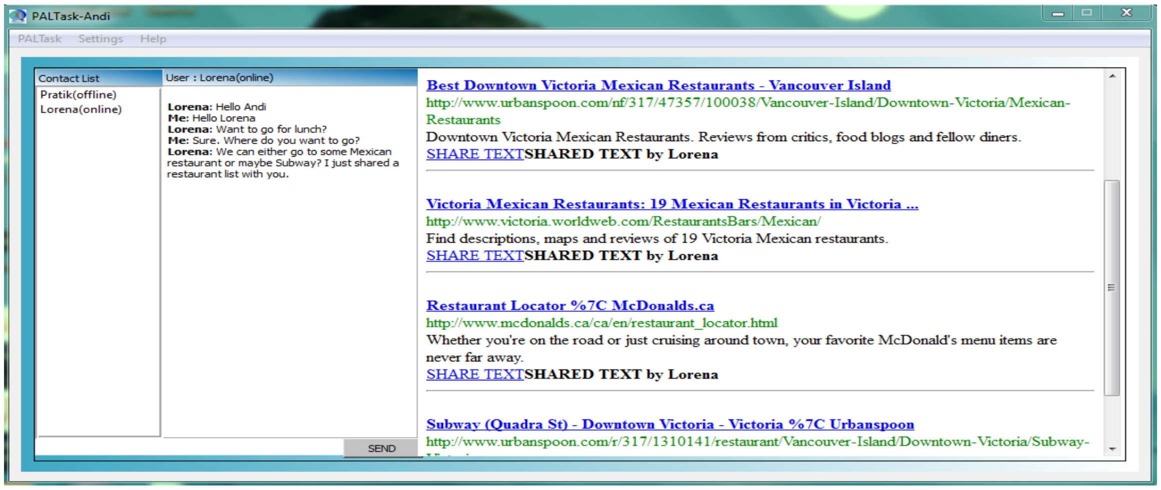
5.4.1 Evaluation by Participant 1

Efficiency : 0-1 steps (i.e., selection of resources)

- Ordinarily, a user opens a web browser and then searches Google, Yelp, or some other site to look for preferred restaurants. Here, web resource retrieval is automated and only the desired type of restaurant is returned. Results are returned based on user preferences (Subway, Mexican food, and McDonald's) as depicted in Figures 5.1-5.2 without any manual search steps taken (i.e., 0 steps).

- A great deal of time is saved in the search task. An automated web search took only 5 seconds to retrieve a list of restaurants without any manual steps taken. Participant 1 also had a choice of completely ignoring or sharing resources with the other participant.

- The selection of preferred choices was easily available, and therefore more efficient. The user has to take only one step in the sharing or selection of resources.



Lorena: Hello Andi
Me: Hello Lorena
Lorena: Want to go for lunch?
Me: Sure. Where do you want to go?
Lorena: We can either go to some Mexican restaurant or maybe Subway? I just shared a restaurant list with you.

Best Downtown Victoria Mexican Restaurants - Vancouver Island
http://www.urbanspoon.com/nf/317/47357/100038/Vancouver-Island/Downtown-Victoria/Mexican-Restaurants
Downtown Victoria Mexican Restaurants. Reviews from critics, food blogs and fellow diners.
SHARE TEXT**SHARED TEXT by Lorena**

Victoria Mexican Restaurants: 19 Mexican Restaurants in Victoria ...
http://www.victoria.worldweb.com/RestaurantsBars/Mexican/
Find descriptions, maps and reviews of 19 Victoria Mexican restaurants.
SHARE TEXT**SHARED TEXT by Lorena**

Restaurant Locator %7C McDonalds.ca
http://www.mcdonalds.ca/ca/en/restaurant_locator.html
Whether you're on the road or just cruising around town, your favorite McDonald's menu items are never far away.
SHARE TEXT**SHARED TEXT by Lorena**

Subway (Quadra St) - Downtown Victoria - Victoria %7C Urbanspoon
http://www.urbanspoon.com/r/317/1310141/restaurant/Vancouver-Island/Downtown-Victoria/Subway-

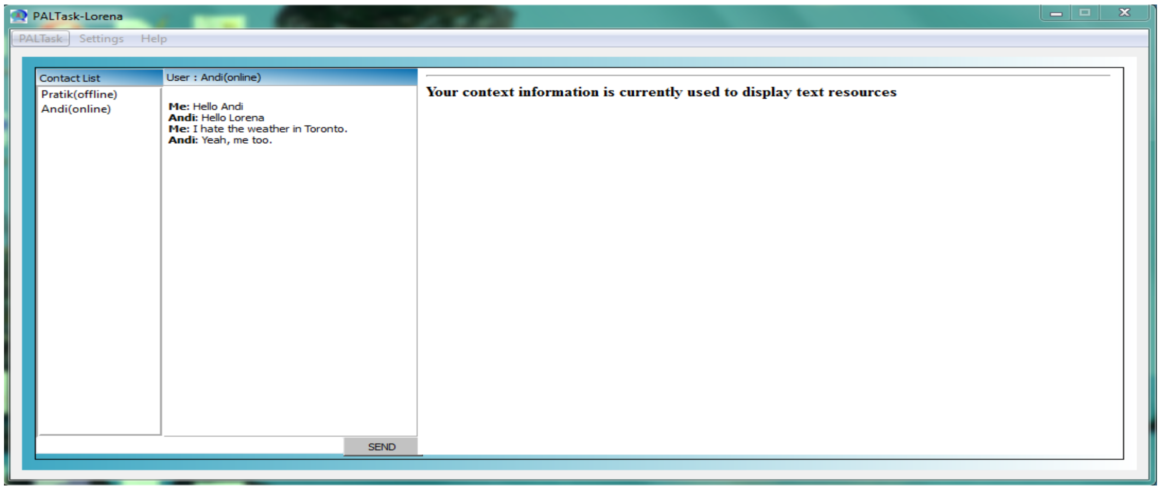
Figure 5.3: Participant 2's screen, chat, and resources shared by Participant 1

Table 5.1: Sentiment analysis with probabilities and label

Sentence	**Probability**	**Label**	**Resources Retrieved**
I hate the weather in Toronto	pos:0.1, neg:0.9	negative	No
Yeah, I have started looking jobs in Victoria because of awesome weather	pos:0.6, neg:0.4	positive	Yes

Effectiveness: High with ConRank; moderate without ConRank

- PALTask without ConRank retrieves eight resources, out of which six are related to restaurants. These restaurants do not match the preferred choices of the user. However, restaurants retrieved are from Victoria using the users location context. Her user experience was moderate; the results were acceptable to her but she had to spend more time looking for the specific restaurants of her choice.



Me: Hello Andi
Andi: Hello Lorena
Me: I hate the weather in Toronto.
Andi: Yeah, me too.

Your context information is currently used to display text resources

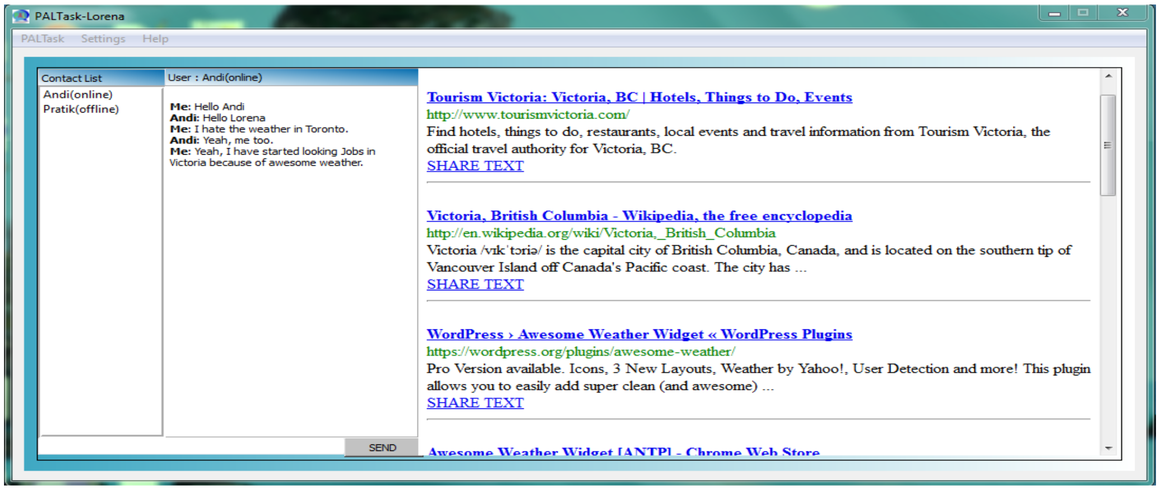
Figure 5.4: Participant 1's screen, showing negative sentiments

- When using PALTask with ConRank, we noted user preferences (Subway, Mexican food, and McDonald's) which were also reflected in the user's PCS. Knowing the preferred choices, it was not hard to determine the resources that met her requirements. Comparing the planned results (those the participant would have searched for) and the actual results from PALTask, Participant 1 was very happy with the restaurants provided.

- Another aspect of evaluating effectiveness is looking for results based on sentiment in the conversation. We planned to retrieve resources based on positive sentences. Participant 1 was happy with the results retrieved based on those positive sentences, and was pleased that she did not have to bother with results based on negative emotions.

User's Experience: Highly satisfied with the results retrieved

- After the experiment, we asked her opinion of our application. Based on her answers, we concluded that she was pleased with the experience, and would prefer to use PALTask over other applications. She understands that PALTask is not only providing chat functionality, but also a better experience by providing feedback at runtime using her dynamic context and PCS. She is also aware that she can switch off her context gathering whenever she wants.



Andi(online)
Pratik(offline)

User : Andi(online)

Me: Hello Andi
Andi: Hello Lorena
Me: I hate the weather in Toronto.
Andi: Yeah, me too.
Me: Yeah, I have started looking Jobs in Victoria because of awesome weather.

Tourism Victoria: Victoria, BC | Hotels, Things to Do, Events
http://www.tourismvictoria.com/
Find hotels, things to do, restaurants, local events and travel information from Tourism Victoria, the official travel authority for Victoria, BC.
SHARE TEXT

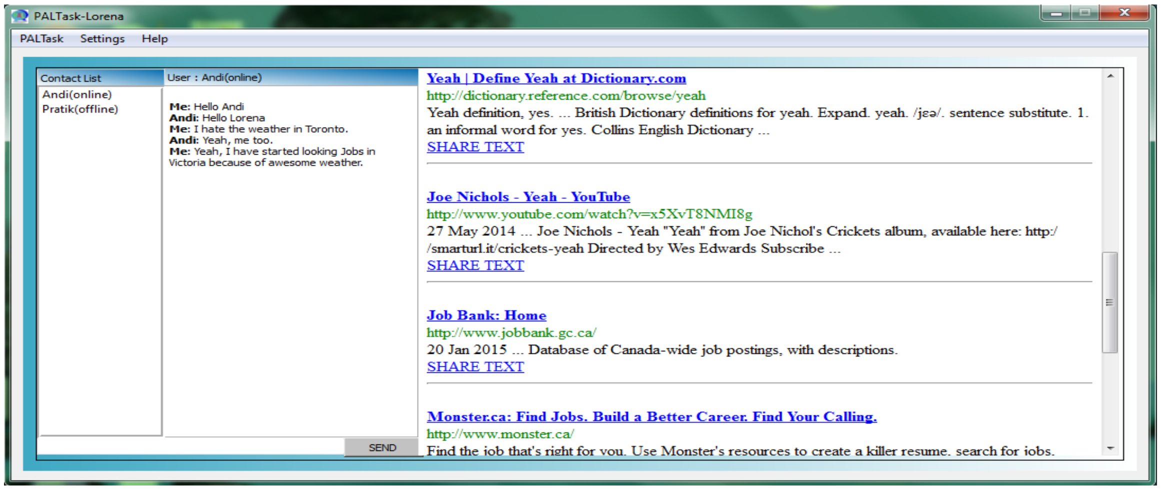
Victoria, British Columbia - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Victoria,_British_Columbia
Victoria /vɪkˈtɔriə/ is the capital city of British Columbia, Canada, and is located on the southern tip of Vancouver Island off Canada's Pacific coast. The city has ...
SHARE TEXT

WordPress › Awesome Weather Widget « WordPress Plugins
https://wordpress.org/plugins/awesome-weather/
Pro Version available. Icons, 3 New Layouts, Weather by Yahoo!, User Detection and more! This plugin allows you to easily add super clean (and awesome) ...
SHARE TEXT

Awesome Weather Widget [ANTP] - Chrome Web Store

Figure 5.5: Participant 1's screen, showing positive sentiments

However, she did not like when some irrelevant resources popped up, such as the last resource in Figure 5.2 (CBC News: McDonald's accused of favoring foreign workers-British Columbia) and resources in Figure 5.6 (Yeah, and Joe Nichols-Yeah). We learned that the former resource (from CBC news) retrieved from "McDonald's" keyword, and the resources in Figure 5.6 was included because of chatspeak (Yeah) as a keyword. For chatspeak, as we already mentioned, we are currently modifying our stoplist (cf. Section 3.2.5) manually. In future work, the stoplist will be updated automatically from context. Overall however, she rated PALTask with ConRank as an eight on a scale of one to ten.



Andi(online)
Pratik(offline)

User : Andi(online)

Me: Hello Andi
Andi: Hello Lorena
Me: I hate the weather in Toronto.
Andi: Yeah, me too.
Me: Yeah, I have started looking Jobs in Victoria because of awesome weather.

Yeah | Define Yeah at Dictionary.com
http://dictionary.reference.com/browse/yeah
Yeah definition, yes. ... British Dictionary definitions for yeah. Expand. yeah. /jɛə/. sentence substitute. 1. an informal word for yes. Collins English Dictionary ...
SHARE TEXT

Joe Nichols - Yeah - YouTube
http://www.youtube.com/watch?v=x5XvT8NMI8g
27 May 2014 ... Joe Nichols - Yeah "Yeah" from Joe Nichol's Crickets album, available here: http:/ /smarturl.it/crickets-yeah Directed by Wes Edwards Subscribe ...
SHARE TEXT

Job Bank: Home
http://www.jobbank.gc.ca/
20 Jan 2015 ... Database of Canada-wide job postings, with descriptions.
SHARE TEXT

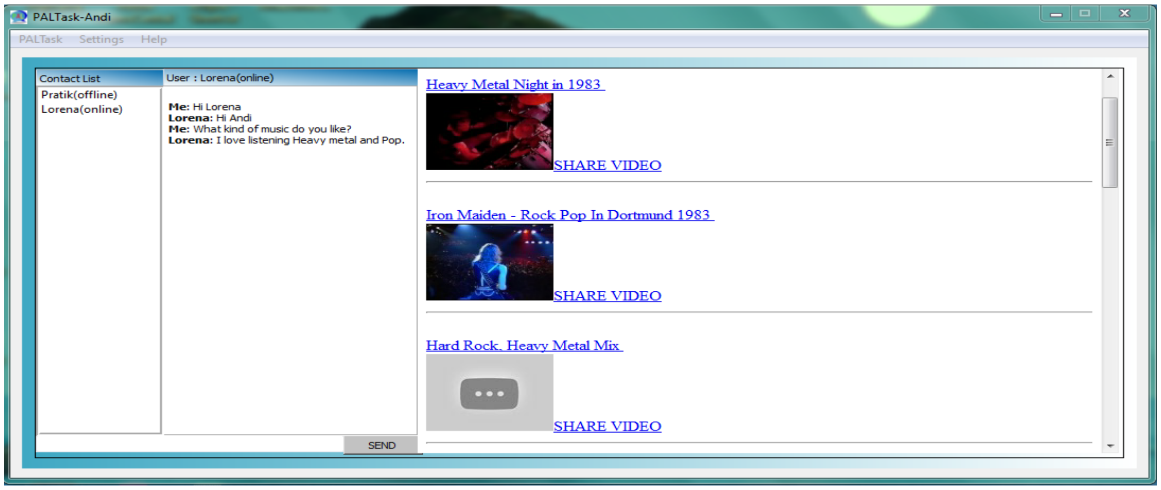
Monster.ca: Find Jobs. Build a Better Career. Find Your Calling.
http://www.monster.ca/
Find the job that's right for you. Use Monster's resources to create a killer resume, search for jobs.

Figure 5.6: Participant 1's screen, showing positive sentiments

5.5 Experiment 2

The experiment was conducted using our second participant, whose PCS is also filled with his preferences. He was more interested in video resources instead of text, therefore he updated his preferences accordingly. He started a conversation with his new school-mate, asking about her interests in music as depicted in Figures 5.7-5.8. He wrote, "What kind of music do you like?" She replied with Hip Hop and Heavy Metal. He was amazed to see similar interests.

Participant 2 has personalized keywords associated with "music" in his PCS (Heavy Metal, Rock, and Hip Hop). He retrieves video resources from YouTube on Heavy Metal, Rock, and Hip Hop (Figures 5.7-5.8) and shares a heavy metal song from Iron Maiden with his colleague (Figure 5.9). However, PALTask's results without ConRank does not include personalization, and is therefore a little frustrating because resources are retrieved every time he sends a message. He retrieves results as music videos from the YouTube API in PALTask using “music” as a keyword.



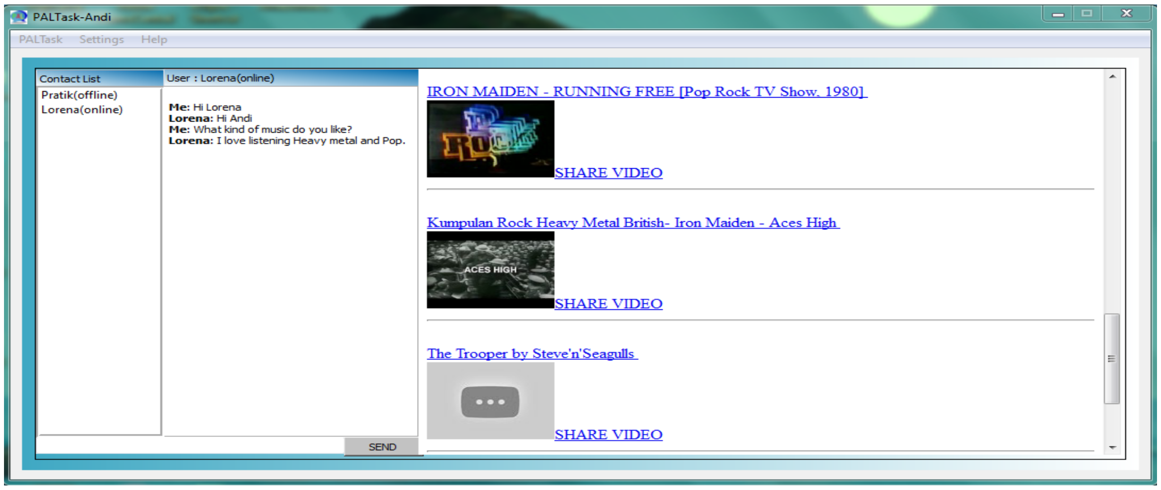
Me: Hi Lorena
Lorena: Hi Andi
Me: What kind of music do you like?
Lorena: I love listening Heavy metal and Pop.

Figure 5.7: Participant 2’s screen, chat, and retrieved resources

We also demonstrated sentiment analysis in the evaluation by Participant 2. In Figures 5.10-5.11, Participant 2 started talking about how much he loves listening to motivational TED talks. As the sentence was positive, three motivational TED talks were retrieved. He shared some of these at the request of his chat partner. The chat partner asked Participant 2: “Do you watch Hollywood movies?” Participant 2 replied, “No, I don’t like to spend much time watching movies.” Since Participant 2’s interest was not in watching movies, and as sentence showed negative sentiments, no resources were retrieved as depicted in Table 5.2.

Table 5.2: Sentiment analysis with probabilities and label

Sentence	**Probability**	**Label**	**Resources Retrieved**
I love to listen motivational TED talks	pos:0.6, neg:0.4	positive	Yes
Ya, sure. No, I don’t like to spend much time on watching movies	pos:0.3, neg:0.7	negative	No



Me: Hi Lorena
Lorena: Hi Andi
Me: What kind of music do you like?
Lorena: I love listening Heavy metal and Pop.

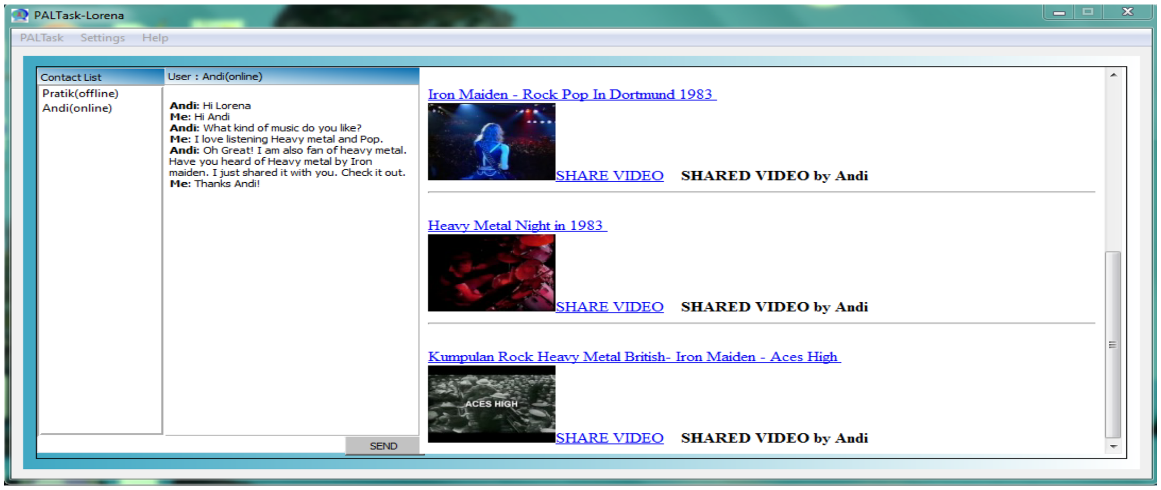
Figure 5.8: Participant 2's screen, chat, and retrieved resources

5.5.1 Evaluation by Participant 2

Efficiency : 0-1 steps (i.e., selection of resources)

- Usually, a user would open a web browser and YouTube or another website to look for preferred videos. Here, our application retrieve preferred videos from YouTube automatically (i.e., 0 steps).

- Time is saved in searching for videos related to the music of his preferred choice. Resources retrieved automatically in 6 seconds.



Andi: Hi Lorena
Me: Hi Andi
Andi: What kind of music do you like?
Me: I love listening Heavy metal and Pop.
Andi: Oh Great! I am also fan of heavy metal. Have you heard of Heavy metal by Iron maiden. I just shared it with you. Check it out.
Me: Thanks Andi!

Iron Maiden - Rock Pop In Dortmund 1983
SHARE VIDEO **SHARED VIDEO by Andi**

Heavy Metal Night in 1983
SHARE VIDEO **SHARED VIDEO by Andi**

Kumpulan Rock Heavy Metal British- Iron Maiden - Aces High
SHARE VIDEO **SHARED VIDEO by Andi**

Figure 5.9: Participant 1's screen, chat, and resources shared by Participant 2

Effectiveness: Moderate with ConRank; low without ConRank

- PALTask without ConRank retrieved videos of all kinds of music based on the keyword "music". However, the resources were retrieved automatically without opening any particular website. Participant 2 was not very happy with so many retrieved results, and his user experience was low.

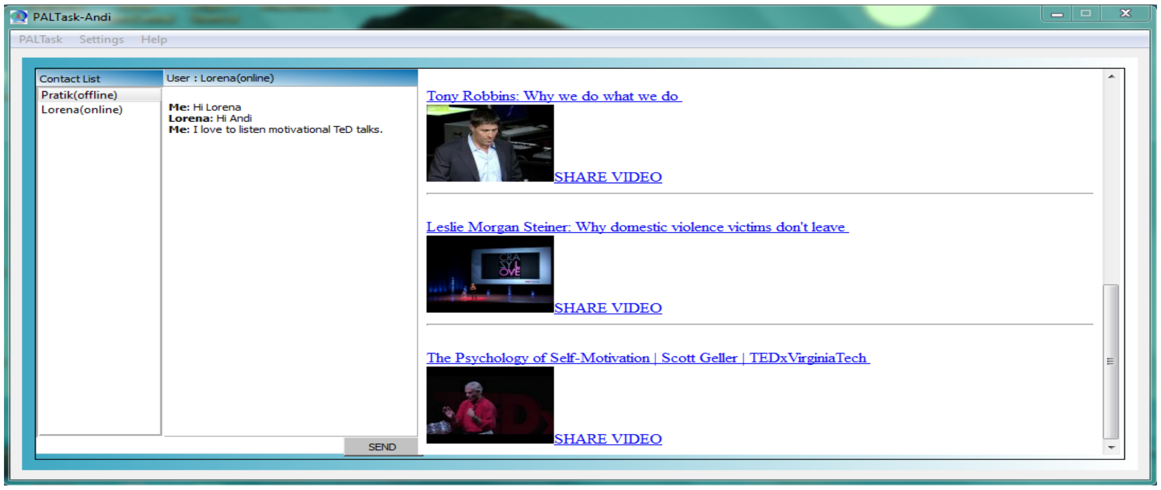
- Participant 2's preferences were known from his PCS, and the context is matched at "music." Resources from various music types were retrieved (Rock, Heavy

Metal, and Hip Hop). He was very happy with this, as he wanted to watch and share some of his favorite videos. Based on a comparison of the planned and actual results, the effectiveness was moderate. The degree of accuracy was not high, as Participant 2 was really looking for a specific video from Iron Maiden, which he didn't find in the retrieved resources.

- As planned, resources were retrieved only on positive sentences, and negative sentences were ignored.

User's Experience: Satisfied

- After the experiment, Participant 2 was very pleased with the video resource retrieval functionality, since he really enjoys listening to music and watching videos while working. His biggest frustration was that he was not able to play some of the resources, and two video links did not include thumbnails (Figures 5.7-5.8). We learned that two videos retrieved from YouTube were restricted, and could only be played on an embedded YouTube player or on YouTube's site itself.



Me: Hi Lorena
Lorena: Hi Andi
Me: I love to listen motivational TeD talks.

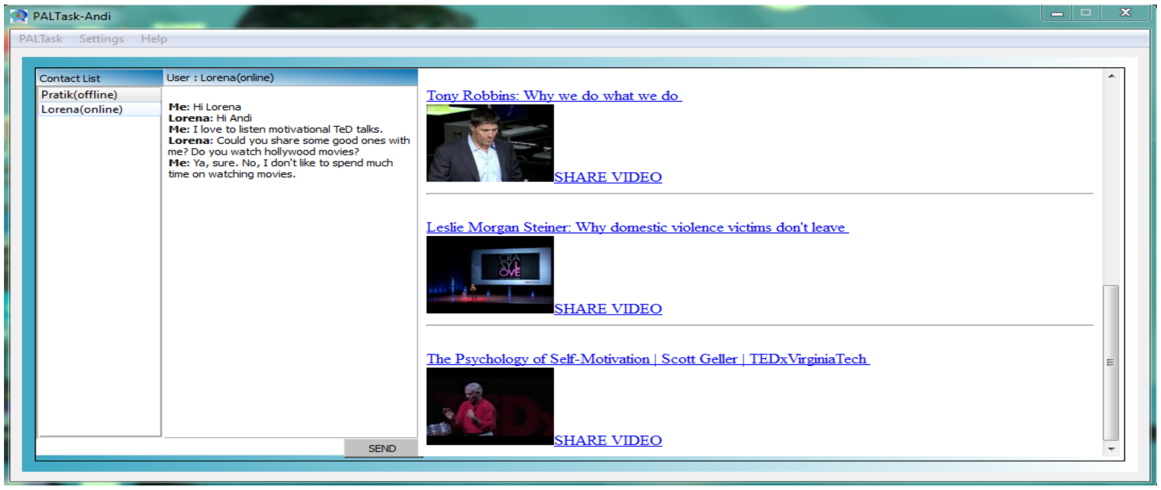
Tony Robbins: Why we do what we do

Leslie Morgan Steiner: Why domestic violence victims don't leave

The Psychology of Self-Motivation | Scott Geller | TEDxVirginiaTech

Figure 5.10: Participant 2's screen, showing positive sentiments

Thumbnails of some videos were not available from the YouTube API (which is a problem in video retrieval from the API). In future work, we can examine how we can eliminate such videos from the retrieval. Participant 2 also suggested the ability to listen to or watch videos while in conversation. This functionality is not currently present however, due to the fact that when a new chat message is delivered, new resources update the resource pane. Overall, Participant 2 had a good experience and rated our application as seven on the scale of one to ten.



Me: Hi Lorena
Lorena: Hi Andi
Me: I love to listen motivational TeD talks.
Lorena: Could you share some good ones with me? Do you watch hollywood movies?
Me: Ya, sure. No, I don't like to spend much time on watching movies.

Tony Robbins: Why we do what we do

Leslie Morgan Steiner: Why domestic violence victims don't leave

The Psychology of Self-Motivation | Scott Geller | TEDxVirginiaTech

Figure 5.11: Participant 2's screen showing negative sentiments (no resources retrieved)

5.6 Summary

This chapter evaluated our PALTask application with ConRank. Two users helped with the experiment, which used their daily dialog over the PALTask client. We evaluated the usability of our application based on three factors: efficiency, effectiveness, and user experience. The feedback helped in identifying pros and cons in the application. For example, Participant 1 was keen on looking at text web resources, while

Participant 2 was happy to retrieve videos. They were pleased to have the context gathering facility and to leverage it for an increased user experience.

Chapter 6

Conclusions

This chapter summarizes this thesis, outlining our contributions and ideas for future improvement in our application model.

6.1 Summary

This thesis investigated context extraction methods and services for providing feedback at runtime for dynamic context. We concentrated on natural language processing techniques such as keyword extraction, sentiment analysis, and stemming of words for analyzing gathered context. Services such as YouTube and Google web search are explored and used for providing feedback in the form of web resources at runtime. We have identified that web searching can be improved by leveraging context, and automating web tasks associated with users' goals can improve user experience.

We discussed how personalized applications are everywhere in today's world, especially in mobile applications. We illustrated various examples of Google's personalized applications, but the disadvantage of those applications is that the user's web profile is owned and managed by Google. As a consequence, the user can't modify their own preferences for advertisements or any other recommendations by Google. We used the Personal Context Sphere approach, which solves that problem by letting the user control their own context.

In Chapter 2, we focused on related work and identified that none of the reviewed applications in an IM scenario provide dynamic feedback to the user. We also discussed various libraries and APIs, and explored the keyword extractor, sentiment analysis, stemming techniques, and web service APIs. In Chapter 3, we discussed PALTask, an instant messaging tool that retrieves personalized web resources by automating the task of searching for those resources on the web.

In Chapter 4, we discussed ConRank, our personalization component to refine PALTask results and provide better personalized results. This chapter discussed the design and implementation of ConRank, including the implementation of sentiment analysis, stemming, PCS integration with keywords, and a keyword ranking algorithm. We evaluated PALTask in Chapter 5 based on the experiment conducted with the help of two participants. These experiments assessed our application based on three factors: efficiency, effectiveness, and user satisfaction. The participant's feedback was positive, and they also suggested few improvements in the tool which we can examine in our future work.

6.2 Contributions

The following summarize the contributions of this thesis:

- Concept and implementation of web search automation by exploiting context. PALTask, as a personalized automated web search application, retrieves web resources based on personal context, location context, and conversations. The application gathers dynamic context and provides feedback at runtime. This application is an example of a context-aware resource gathering model which acquires relevant context, then interprets it and provides relevant information to the user. This model could be applied to different scenarios.

- We identified RAKE, a document-oriented keyword extractor that can identify keywords from documents. We evaluated it based on factors such as quality of results, availability of a remote API and source code, cost, and license.

- Using the PCS in our IM application, we demonstrated how personalizing the application increases user experience. This also demonstrates the value of the Personal Context Sphere (PCS) for personalizing applications.

- ConRank, a component of PALTask that performs various operations on text such as sentiment analysis, stemming of keywords, integration of PCS and keywords, and ranking of keywords to provide more relevant results.

- We explored the NLTK API for sentiment analysis and stemming of words. The API consists of various operations for processing natural language text. It is a very rich API, well structured and documented.

- We identified that for designing GUI (front end) applications, QTCreator is straightforward. It is a cross platform IDE with an integrated code editor and QT designer, giving the application a native look. The application looks like all other applications on the various platforms (e.g., Mac, Windows, Linux, and Mobile platforms).

6.3 Future Work

Smart applications can leverage contextual information to increase users' experience as we demonstrated in this thesis. Our application PALTask was designed to demonstrate the approach adopted in this research. There is a great deal of work that can still be done in this resource gathering model:

- PALTasks future includes the ability to exploit context from audio and video conversations. Recognizing audio and video conversation is also proposed in our research paper [BTJ+13]. Audio conversation detection such as Siri by Apple [Aro11], could also be done via our application model. Video conversation can detect faces, tagging the people involved with their names or displaying their profile automatically in a conference call [BTJ+13].

- The PCSUpdate module can be enhanced in an application. Currently, the PCS cannot be updated based on the feedback of users engaging with resources. Acceptance or rejection of resources offered should update the PCS dynamically.

- We don't have a database in our application, something which could be very helpful for storing keywords, chat history, and even some resource links for future use.

- Currently, PALTask is a desktop application, but it can be implemented in mobile operating systems. However, turning this application into a web application could attract more users still.

- We have used sentiment analysis using a movie corpus from NLTK API, in which data is gathered from movie reviews. A corpus of chat data which is labeled as positive, negative, or neutral could improve accuracy.

- We can include smiley emoticons in sentiment analysis for detecting sentiment in text that can provide enhanced accuracy.

Bibliography

[AC10] C. Abela and K. Cortis. Semchat: Extracting personal information from chat conversations. In *Workshop on Personal Semantic Data, Knowledge Engineering and Knowledge Management (EKAW 2010)*, page 10, 2010.

[ADB$^+$99] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC 1999)*, pages 304–307. Springer, 1999.

[Aro11] J. Aron. How innovative is Apple's new voice assistant, Siri? *New Scientist*, 212(2836):24, 2011.

[BHCNM01] C. Basu, H. Hirsh, W. W. Cohen, and C. G. Nevill-Manning. Technical paper recommendation: A study in combining multiple information sources. *Journal of Artificial Intelligence Research (JAIR 2001)*, pages 231–252, 2001.

[BK10] M. W. Berry and J. Kogan. Text mining. *Applications and Theory. John Wiley & Sons*, 2010.

[BL01] T. Bauer and D. B. Leake. Word sieve: A method for real-time context extraction. In *Proceedings of the 3rd International Interdisciplinary Conference, Context : Modeling and Using Context*, pages 30–44. Springer, 2001.

[BMZ11] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[BTJ$^+$13] A. Bergen, N. Taherimakhsousi, P. Jain, L. Castañeda, and H. A. Müller. Dynamic context extraction in personal communication applications. In *Centre for Advanced Studies Conference (CASCON 2013)*, pages 261–273, 2013.

[BV04] L. Barkhuus and A. Vallgårda. Saying it all in 160 characters: Four classes of sms conversations. *The IT University of Copenhagen, Technical Report: TR-45*, 2004.

[CCNY10] M. Chignell, J. Cordy, J. Ng, and Y. Yesha. *The Smart Internet: Current Research and Future Applications*, volume 6400 of *Lecture Notes in Computer Science*. Springer, 2010.

[CGR$^+$11] M. D. Conover, B. Gonçalves, J. Ratkiewicz, A. Flammini, and F. Menczer. Predicting the political alignment of twitter users. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom)*, pages 192–199. IEEE, 2011.

[Cho03] G. G. Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[CnMV13] L. Castañeda, H. A. Müller, and N. M. Villegas. Towards Personalized Web-Tasking: Task Simplification Challenges. In *2013 IEEE Ninth World Congress on Services (SERVICES 2013)*, pages 147–153. IEEE, 2013.

[DF04] S. Decker and M. Frank. The social semantic desktop. In *Workshop Application Design, Development and Implementation Issues in the Semantic Web on World Wide Web (WWW 2004)*, volume 9, page 10, 2004.

[DJ11] M. Dostál and K. Jezek. Automatic Keyphrase Extraction based on NLP and Statistical Methods. In *Databases Texts Specifications and Objects (DATESO 2011)*, pages 140–145, 2011.

[Eva11] D. Evans. The Internet Of Things. How The Next Evolution Of The Internet Is Changing Everything (Whitepaper). *Cisco Internet Business Solutions Group (IBSG)*, 2011.

[FD06] J. Forlizzi and C. DiSalvo. Service robots in the domestic environment: A study of the Roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human Robot Interaction (HRI 2006)*, pages 258–265. ACM, 2006.

[GBL98] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: An Automatic Citation Indexing System. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 89–98. ACM, 1998.

[HFH$^+$09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[HIHO09] S. Horiguchi, A. Inoue, T. Hoshi, and K. Okada. Gachat: A chat system that displays online retrieval information in dialogue text. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW 2009)*, pages 1–5, 2009.

[HPK$^+$10] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles. Context-aware citation recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*, pages 421–430. ACM, 2010.

[HSK09] J. Y. Hong, E. H. Suh, and S. J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009.

[JBCnM13] P. Jain, A. Bergen, L. Castañeda, and H. A. Müller. PALTask Chat: A Personalized Automated Context Aware Web Resources Listing Tool. In *2013 IEEE Ninth World Congress on Services (SERVICES 2013)*, pages 154–157. IEEE, 2013.

[LHC05] B. Liu, M. Hu, and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web (WWW 2005)*, pages 342–351, 2005.

[Liu12] B. Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[LMJ03] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Advances in neural information processing systems*, page None, 2003.

[MG06] G. Mishne and N. S. Glance. Predicting movie sales from blogger sentiment. In *Association for the Advancement of Artificial Intelligence*

(AAAI) Spring Symposium: Computational Approaches to Analyzing Weblogs, pages 155–158, 2006.

[NCCY10a] J. W. Ng, M. Chignell, J. R. Cordy, and Y. Yesha. Overview of the Smart Internet. In *The Smart Internet*, volume 6400 of *Lecture Notes in Computer Science*, pages 49–56. Springer, 2010.

[NCCY10b] J. W. Ng, M. Chignell, J. R. Cordy, and Y. Yesha. Smart Interactions. In *The Smart Internet*, volume 6400 of *Lecture Notes in Computer Science*, pages 59–64. Springer, 2010.

[NL13] J. W. Ng and D. H. Lau. Going beyond web browsing to web tasking: Transforming web users from web operators to web supervisors. In *2013 IEEE Ninth World Congress on Services (SERVICES 2013)*, pages 126–130. IEEE, 2013.

[OSH06] S. Owsley, S. Sood, and K. J. Hammond. Domain specific affective classification of documents. In *Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 181–183, 2006.

[Por80] M. Porter. An algorithm for suffix stripping. Program, 3 (14): 130–137, 1980.

[PZCG14] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[RCRM02] A. Ranganathan, R. H. Campbell, A. Ravi, and A. Mahajan. Conchat: A context-aware chat program. *IEEE Pervasive Computing*, 1(3):51–57, 2002.

[RECC10] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. *Text Mining*, pages 1–20, 2010.

[RS10] S. Rybalko and T. Seltzer. Dialogic communication in 140 characters or less: How fortune 500 companies engage stakeholders using twitter. *Public Relations Review*, 36(4):336–341, 2010.

[Sat01] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17, 2001.

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *First Workshop on Mobile Computing Systems and Applications (WMCSA 1994)*, pages 85–90. IEEE, 1994.

[SC10] S. O. Sood and E. F. Churchill. Anger management: Using sentiment analysis to manage online communities. *Grace Hopper Celebration*, 2010.

[TSK06] P. N. Tan, M. Steinbach, and V. Kumar. In *Introduction to Data Mining*, volume 1, pages 231–258. Addison Wesley, 2006.

[Tur02] P. D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[Vil13] N. M. Villegas. *Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems.* PhD Thesis, Department of Computer Science, University of Victoria, 2013.

[VM10] N. M. Villegas and H. A. Müller. Managing dynamic context to optimize smart interactions and services. In *The Smart Internet*, volume 6400 of *Lecture Notes in Computer Science*, pages 289–318. Springer, 2010.

[Wei93] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.

[Wil06] P. Willett. The Porter Stemming Algorithm: Then and Now. *Program: Electronic Library and Information Systems*, 40(3):219–223, 2006.

[WWH05] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, 2005.

[ZSL05] A. Zimmermann, M. Specht, and A. Lorenz. Personalization and context management. *User Modeling and User-Adapted Interaction*, 15(3-4):275–302, 2005.

Appendix A

Source Code

Listing A.1: RAKE Keyword Extraction Algorithm

```
# Implementation of RAKE - Rapid Automtic Keyword Exraction algorithm
# as described in:
# Rose, S., D. Engel, N. Cramer, and W. Cowley (2010).
# Automatic keyword extraction from indi-vidual documents.
# In M. W. Berry and J. Kogan (Eds.), Text Mining: Applications and Theory.unknown: John Wiley and Sons, Ltd.

import re
import operator
import math

debug = False
test = True

def isnum (s):
    try:
        float(s) if '.' in s else int(s)
        return True
    except ValueError:
        return False

# Utility function to load stop words from a file and return as a list of words
# @param stopWordFile Path and file name of a file containing stop words.
# @return list A list of stop words.
```

```
def loadStopWords(stopWordFile):
    stopWords = []
    for line in open(stopWordFile):
        if (line.strip()[0:1] != "#"):
            for word in line.split( ): #in case more than one per line
                stopWords.append(word)
    return stopWords

# Utility function to return a list of all words that are have a length greater than a specified number of characters.
# @param text The text that must be split in to words.
# @param minWordReturnSize The minimum no of characters a word must have to be included.
def separatewords(text,minWordReturnSize):
        splitter=re.compile('[^a-zA-Z0-9_\\+\\-/]')
        words = []
        for singleWord in splitter.split(text):
                currWord = singleWord.strip().lower()
                #leave numbers in phrase, but don't count as words, since they tend to invlate scores of their phrases
                if len(currWord)>minWordReturnSize and currWord != '' and not isnum(currWord):
                        words.append(currWord)
        return words

# Utility function to return a list of sentences.
# @param text The text that must be split in to sentences.
def splitSentences(text):
        sentenceDelimiters = re.compile(u'[.!?,;:\t\\-\\"\\(\\)\\\'\u2019\u2013]')
        sentenceList = sentenceDelimiters.split(text)
        return sentenceList

def buildStopwordRegExPattern(pathtostopwordsfile):
        stopwordlist = loadStopWords(pathtostopwordsfile)
        stopwordregexlist = []
        for wrd in stopwordlist:
                wrdregex = '\\b' + wrd + '\\b'
                stopwordregexlist.append(wrdregex)
        stopwordpattern = re.compile('|'.join(stopwordregexlist), re.IGNORECASE)
        return stopwordpattern

def generateCandidateKeywords(sentenceList, stopwordpattern):
        phraseList = []
        for s in sentenceList:
                tmp = re.sub(stopwordpattern, '|', s.strip())
                phrases = tmp.split("|")
                for phrase in phrases:
                        phrase = phrase.strip().lower()
                        if (phrase!=""):
                                phraseList.append(phrase)
        return phraseList

def calculateWordScores(phraseList):
        wordfreq = {}
        worddegree = {}
        for phrase in phraseList:
                wordlist = separatewords(phrase,0)
                wordlistlength = len(wordlist)
                wordlistdegree = wordlistlength - 1
                #if wordlistdegree > 3: wordlistdegree = 3 #exp.
                for word in wordlist:
                        wordfreq.setdefault(word,0)
                        wordfreq[word] += 1
                        worddegree.setdefault(word,0)
                        worddegree[word] += wordlistdegree #orig.
                        #worddegree[word] += 1/(wordlistlength*1.0) #exp.
        for item in wordfreq:
                worddegree[item] = worddegree[item]+wordfreq[item]

        # Calculate Word scores = deg(w)/frew(w)
        wordscore = {}
        for item in wordfreq:
                wordscore.setdefault(item,0)
                wordscore[item] = worddegree[item]/(wordfreq[item] * 1.0) #orig.
                #wordscore[item] = wordfreq[item]/(worddegree[item] * 1.0) #exp.
        return wordscore

def generateCandidateKeywordScores(phraseList, wordscore):
        keywordcandidates = {}
        for phrase in phraseList:
                keywordcandidates.setdefault(phrase,0)
                wordlist = separatewords(phrase,0)
                candidatescore = 0
                for word in wordlist:
```

```python
            candidatescore += wordscore[word]
        keywordcandidates[phrase] = candidatescore
    return keywordcandidates

def test (message, api_option):
    #text = "I don't think so, we shold go to San Fransisco and not Chicago"
    msg = message
    if len(message) == 0 :
        import sys
        print sys.argv[1]
        msg = ""
        for item in sys.argv[1:]:
            msg += item + " "
    text = msg
    # Split text into sentences
    sentenceList = splitSentences(text)

    stoppath = "SmartStoplist.txt"
    #SMART stoplist misses some of the lower-scoring keywords in Figure 1.5,
    #which means that the top 1/3 cuts off one of the 4.0 score words in Table 1.1

    stopwordpattern = buildStopwordRegExPattern(stoppath)

    # generate candidate keywords
    phraseList = generateCandidateKeywords(sentenceList, stopwordpattern)

    # calculate individual word scores
    wordscores = calculateWordScores(phraseList)

    # generate candidate keyword scores
    keywordcandidates = generateCandidateKeywordScores(phraseList, wordscores)
    if debug: print keywordcandidates
    sortedKeywords = sorted(keywordcandidates.iteritems(), key=operator.itemgetter(1), reverse=True)
    if debug: print sortedKeywords
    totalKeywords = len(sortedKeywords)
    if debug: print totalKeywords
    print sortedKeywords[0:(totalKeywords/3)]
    print sortedKeywords[0:(totalKeywords/1)]
    return sortedKeywords[0:(totalKeywords/1)]
```