Dynamic Context Extraction in Personal Communication Applications

Andreas Bergen, Nina Taherimakhsousi, Pratik Jain, Lorena Castañeda, Hausi A. Müller

University of Victoria, Victoria, Canada

Abstract

Data generated by video applications are rarely mined for context in order to augment and improve user experiences. In this paper, we propose an innovative approach to extract context from video streams dynamically. As a case study we apply this approach in a context based multimedia chat application. In particular, this paper discusses how we perform video stream analysis to recognize faces and logos; separate audio from video contents for further analysis; and mine text chat messages for keywords to infer contextual information at all levels. This allows us to recognize people, logos, as well as conversation topics and recommend videos on the fly. One key technique is that the chat application maintains different context spheres that can be selectively combined and intersected to provide a context sensitive and improved chat user experiences.

1 Introduction

Personalized communication applications have become ubiquitous in today's world. They are generally single purpose applications that facilitate communication. However, while using these applications users perform other activities simultaneously. Depending on the context of the conversation, users browse the web, share links, research topics of interest or even plan future activities which are related to the conversation. None of these actions are fully integrated with existing personal communication tools.

Copyright © 2013 Andreas Bergen, Nina Taherimakhsousi, Pratik Jain, Lorena Castañeda, Hausi A. Müller. Permission to copy is hereby granted provided the original copyright notice is reproduced in copies made.

To illustrate this further, imagine you are using a video chat application, such as PALTask [1], with someone who is located relatively close by, perhaps in another building. It is approximately noon, but your discussion is very productive so you are considering to continue this conversation while eating lunch together.

Throughout your conversation, the application is constantly extracting keywords from the audio and determines that the relevant keywords for this context are now "lunch" and "food". Consequently, the application automatically displays several suggestions for food outlets close to both users.

At this very moment, another coworker comes closer to the camera and joins the conversation. Through face recognition, the application is now aware of the additional participant in this conversation and is considering her contextual information as well. It quickly becomes apparent that the new user is also considering lunch options.

In addition to the chat's contents, public elements from the coworkers personal context sphere [2, 3] can be used on the fly to further personalize the resources list. A personal context sphere is a repository of context information relevant to the user and her personal goals hosted by a third party. Some of this information might include, gender, age, favourite locations and web sites of her preference. For instance, the coworker really likes Subway sandwiches, which is reflected in her context sphere. Neither of your personal context spheres indicate that you dislike Subway, so an application like PALTask suggests for you to have lunch at Subway nearby having automatically extracted the context of the conversation, as well as the related items of the personal context sphere.

How does PALTask know to display locations of food outlets close by? First, the conversation's

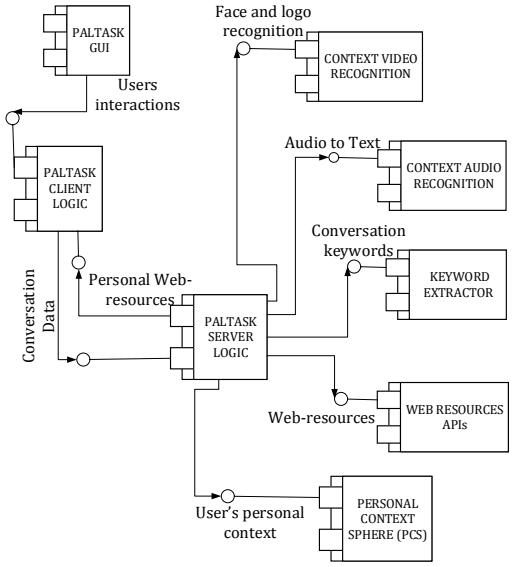

Figure 1: Component architecture of the integrated components

audio is converted to text in order to extract keywords, from here it obtains the keywords "food" and "lunch". Your location can be determined via IP, GPS or other means (if on a mobile platform), which can also be stored in the personal context sphere. Location context narrows down the suggestion to food outlets nearby. Assuming neither you nor the original chat partner have food preferences in the personal context sphere, the application uses the context sphere of the 3rd chat partner who has this information stored and available in her personal context sphere. The coworker is automatically recognized, when she enters the view of the camera which automatically integrates her context sphere to provide improved contextual information. Her context sphere on food items makes it clear that she likes "Subway". Subsequently PALTask displays the location of a subway restaurant close by, which saves time, effort and enhances the user's experience.

Due to the complexity of this system, we introduce the individual subsystems separately. Our evaluation of each system is also based on individually examining each context extraction sub-system. At this stage of development we present the components separately, full integration of these components will occur as future work. PALTask grew out of a preliminary prototype which was focussed solely on context extraction of text chat for automation of personalized web tasks [1]. Figure 1 illustrates the proposed component architecture of our chat system PALTask. This paper focusses on the keyword extractor, audio module and video module in detail. The remainder of this paper is organized as follows: Section 2 presents existing approaches and related work; Sections 3 and 4 describe the context gathering modules and their evaluation; Section 5 concludes the paper and Section 6 highlights future work.

2 Related Work

Context analysis for the purpose of providing personalized augmentation has been demonstrated before. Previous approaches can be found in the domain of computer supported collaborative work (CSCW), ubiquitous computing and in existing chat technology to name just a few areas. However, many of these existing approaches are often dedicated to improve context gathering for a rather narrow aspect of personalized communications (i.e., purely text). Many of these existing approaches gather context during post processing, rather than dynamically at runtime. Similarly, many approaches also aim to augment only one specific aspect of the user's experience. With improvements in computing resources and context extraction techniques, we aim to demonstrate a tool which can gather context from a variety of sources at runtime to improve users experience based on contextual information.

Schilit and Theimer coined the terms context and context awareness in the early 1990s [4]. Their early work at XEROX corporation introduces the concept of context aware applications and provides models to understand applications' behaviour and interactions; context is viewed as more than simply location data, instead it includes people who are nearby, available resources, and many more [4, 5]. Weiser, who illustrates the impact and penetration computing has reached to the point of achieving ubiquity, inspired their work [6].

Recent context aware communication applications include Ranganathan's ConChat [7]. Ranganathan illustrates that augmenting chat messages with contextual information, aimed at preventing semantic ambiguity between the chat partici-

pants, can improve the user experience [7]. Similarly, with the aim of improving context awareness, SemChat proposes a non-dynamic, offline, context analysis after the end of a chat session [8]. While static analysis, as well as, augmentation for a single purpose is valuable, we propose a tool which performs context analysis dynamically and, ultimately, can augment the user experience in more than just a single dimension by providing additional relevant information from a variety of sources and formats.

GaChat on the other hand takes a dynamic approach; it automatically annotates a chat conversation, by integrating contextually relevant information directly into the chat [9]. GaChat performs its context analysis of chat messages on the server side, rather than on the client side. This is similar to our approach as far as utilizing greater processing capabilities at the server-side. However, it again is limited to augmenting the user experience in a very specific way.

Abowd et al. provide a comprehensive resource of existing approaches on context and context aware computing at the end of the 1990s [10]. Their work also points out that the very definitions of context and context awareness are themselves different depending on the particular system, its scenario and its use case at any given moment [10].

Context extraction is also prominently discussed in document analysis. Bauer et al. introduced Word Sieve which uses external user information, such as search and access patterns, to augment individual documents or document groups with additional contextual information [11].

Similar to PALTask, Huang describes a system aimed at multi-media context extraction. Huang, however, developed a classification system aimed at dynamically building a hierarchy of news casts, commercials and news anchors by analysing text, and audio [12]. PALTask differs from this approach because we do not build a context hierarchy, but rather extract context dynamically from a time relevant window of information.

Aside from communication and document analysis, video context extraction has also other applications. Xing et al. introduce SafeVchat in collaboration with Chatroulette to automatically analyse video streams in order to dynamically identify specific types of behaviour of its users [13].

Hong et al. provide a comprehensive survey including 237 journal articles on context aware systems [14]. This survey highlights articles in various classifications, including context aware algorithms, context awareness at the level of network infrastructure, middleware, the user interface, the application level relating to uses at home, the classroom, health care applications, or information and communication systems.

PALTask is designed to improve upon the existing work by dynamically gathering a broad range of context in order to augment the user experience with useful information on the fly.

3 Context Gathering

This section aims to provide the reader with an understanding of the individual subsystems that compose the context gathering mechanism in a chat application. The server side subsystems and their interactions are depicted in Figure 1. We introduce technical details for each of the relevant subsystems to illustrate how context is extracted from text, audio, and video sources.

Context gathering, provides the underlying system with the ability to automate or predict behaviour. The system can automate its own behaviour based on the contextual information it received. This is particularly useful when contextual information can be mined to provide the users with additional information which otherwise would have been obtained manually or not at all. Context sensitive actions such as searching the web, displaying and sharing resources, or scheduling tasks can be automated.

Chat applications are rich in context: they contain video, audio, and in many cases additional textual information. These sources are largely untapped resources for context. Context in this case is not limited to the conversation content, but it includes: location; the people participating; number of people at each unique connection end; time; visual queues such as gestures, items or logos; spoken keywords, etc. Extracting this information can be beneficial in improving the user experience, and can automate tasks that currently are only performed manually by the users themselves. The initial prototype used extracted context information and displayed conversation relevant information, in the form of Youtube videos, on a pane next to the main chat. Further additions will automate other tasks such as browsing for web resources and the display of relevant educational information as

desribed in this paper. In this paper we focus on three distinct context extraction methods: keyword extraction from text and audio, as well as face and logo recognition from video.

3.1 Keyword Extraction from Text

Automatic keyword extraction from text has been studied extensively over the past decades. For our application, we selected the Rapid Automatic Keyword Extraction (RAKE) to extract keywords from chat messages [15, 16].

RAKE is document-oriented and thus does not rely on a reference corpus to identify key words. Consequently, statistical analysis or frequency analysis is also unnecessary with RAKE. These aspects make RAKE very attractive to use in a chat environment where accuracy and speed are two crucial metrics.

The purpose of the keyword extractor component is to dynamically rank and analyse keywords obtained from text. To identify meaningful contextual keywords in chat conversations, the analysis is performed on more than just one message at a time. Keyword extraction occurs on the last individual message which was sent, as well as on several most recent messages. This is necessary because chat messages are often short. Therefore, by keeping a record of approximately the 10 most recent messages 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 are common elements in text, yet do not aid in providing unique contextual information. Examples of stop words are *the*, *a*, *should*, In short phrases stop words are too frequent while proper keyword candidates are not present. Subsequently, a larger message body is constructed by using approximately the last 10 sent messages. This is similar to the 140-160 character length of SMS [17] and Twitter [18] statements to produce meaningful messages. Keywords obtained from the single most recent message are weighted as more importance and a combined average is returned as a ranked list of keywords using a custom API.

To leverage greater computational power and storage resources, the keyword extractor is deployed on the server side. The keyword extractor component is also a key component of the audio to text module (cf. Figure 1). The audio to text module uses the keyword extractor module to analyse the textual representation of its audio data.

3.1.1 Modified Use of RAKE

Messages available to us in chat applications are short, thus, forcing two distinct modifications in the way we use RAKE. First, each time a message is sent, our keyword extraction module processes this message for keywords. Then it processes a history of generally the 10 most recent messages for keywords. Any resulting keywords which are common between the two resulting sets are combined in their weight in order to signify that they represent the most recent context of the conversation. While this does not modify the implementation details of RAKE, it does modify the return results by using the RAKE component in a modified way.

The history of the past messages will be stored external to RAKE in the actual server threads which are responsible for facilitating communication between the clients. This is a design and implementation decision. From a design perspective this ensures that the data is kept close to the source which generates and uses it. Secondly, it allows us to only modify the usage of RAKE, not the actual implementation. Additionally, this also allows us to weigh the keywords of the last message differently based on criteria which are specific to each conversation rather than using one metric for all ongoing chat sessions. It is conceivable that each chat session is able to modify the weight it gives to the keywords of the last chat message based on content of a user's personal context sphere rather than applying a single metric to all users.

The second modification we make is even less intrusive in nature. Rose describes that only one-third of the keywords found by RAKE are usually used [15]. For completeness and to obtain a richer set of results we retain the full list of identified keywords for later processing in the chat application. This is necessary because in short input texts, where the number of keywords is often very small. By default, the system often returns zero or only one keyword when there are in fact 2 or 3 keywords present (a small number). In this case it is better to return 2 or 3 keywords than to return 0 or 1, which would correspond to the original third proposed by Rose. For a larger number of keywords the top third may be sufficient, though further analysis is needed

to determine when this threshold actually occurs in typical chat conversations.

One of the inputs to RAKE is a list of stop words, words that are commonly used but provide no additional information for context. In personal chat applications text communication often does not follow a standard language dictionary in terms of spelling and capitalisation of words. Spelling mistakes are frequent and remain uncorrected, abbreviations, accronyms or *chatspeak* (e.g., LOL, BRB, or AFK) are common. Consequently, we modified the stop word list to reflect this type of text. Table 1 shows the improved result when applying a different stoplist on even a very simple input text. Without the modified stop word list, *chatspeak* is erroneously interpreted as a keyword.

Table 1: Keywords of the phrase "lol we should go to San Fransisco and not Town xyz" using a modified stop word list.

Keywords	*chatspeak* in StopList
san fransisco, lol, town xyz	no
san fransisco, town xyz	yes

Currently, we alter the stoplist manually to include words commonly found in chat texts. In the future, this is 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. [19].

With these modifications in place, RAKE is an integral part of our keyword extraction component. RAKE's performance and accuracy are essential to the success of the keyword extraction from textual representations containing context information. With the modifications in place, we are able to use RAKE in a chat setting successfully; a setting which does not necessarily follow existing rules of grammar, spelling, or even uses words found in dictionaries.

3.1.2 Limitations of Using RAKE

There are few limitations when using RAKE in our keyword extraction module. However, these limitations are negligible when considering the rapid performance and accuracy it provides. The major drawback of using RAKE on text samples which are not guaranteed to adhere to proper spelling and grammar is the need to modify the stop word list accordingly. However, Berry et al. address this issue by describing methods to automatically generate stop word lists which are corpus and domain specific [19].

3.2 Audio as a Context Source

The audio to text component, as depicted in Figure 1, is not integrated with the other systems that compose the context extraction engine of this chat application. Isolating the implementation in such a way allows us to evaluate the system more thoroughly without external factors impacting the results.

3.2.1 The Technology

Audio to text applications are readily available on desktops and, in a limited fashion, are integrated in smartphones. Equivalent open source software, which can be modified and performs as well as commercial products, are rare. We selected *Pocketsphinx* released in 2006 by Carnegie Mellon University researchers [20]. *Pocketsphinx* satisfies our requirements regarding performance and accuracy; and was chosen because of its acceptance rate in the community and the quality of results.

Pocketsphinx is designed to provide real-time continuous speech recognition. Initially, as Huggins-Daines illustrates, *Pocketsphinx* performs at reasonable real-time speeds even on mobile platforms by achieving slight sub-real-time peak performance [20]. Architectural design decisions to leverage powerful computing resources at the server side prohibit use of *Pocketsphinx* in continuous mode. In our design, the transcription of audio to text occurs at the server side, rather than at the client side. This design decision forces the chat application to provide *Pocketsphinx* frequently with small input files containing segments of the conversation.

3.2.2 Modifications Specific to PALTask

As mentioned above our application makes two important changes to the originally proposed uses of *Pocketsphinx*. Most strikingly, *Pocketsphinx* is situated at the server component of the chat application where it receives a wav file containing a

transformed input source of the audio data. We obtain audio data by stripping audio information from the video data. Due to resource concerns this task is performed on the server rather than at each client. Because this relies on the existence of wav files there is a short lag needed to create the file and operate on it. A further delay is introduced by executing *Pocketsphinx* frequently, incurring startup and communication delays between it and our chat components. This potential loss of real-time performance needs to be mitigated regularly in order to prevent the processing delay from increasing. The exact extent of this delay and gravity is something that is currently unknown. However, it is anticipated that conversations with regular pauses will not experience this processing delay to the same extent as rapid content rich conversations.

Another significant change pertains to the actual words that this module is tasked to recognize. Initially we intended to use the provided dictionaries. When testing the system with a phrase containing thirteen words, eight of which were distinct, the system returned a result containing thirty-six words. None of the words in the returned result were found in the test phrase. Effectively *Pocketspinx* returned an incorrect result. Successively modifying the dictionary by only focussing on a specific range of contextual topics improved the results. Moreover, by providing alternate acoustic definitions of these words we were able to obtain a phrase containing seven words, all of which were also found in the original test phrase. Thus our modifications to the dictionary were successful and we are currently using a modified dictionary containing information on a variety of topics of interest which we expect the tool to recognize for context extraction.

3.2.3 Limitations

To further improve the results of *Pocketsphinx* we need to supply it with better acoustic models. This can take the form of training general acoustic models or even generating models which are trained specifically for each user. Prior to effecting these improvements *Pocketsphinx* will be limited to only return the keywords which we explicitly placed in the dictionary. Thus, limiting the number of words which we are able to recognize. In addition to having only a limited number of acoustic models, different pronunciations, regional dialects and slang can impede the results.

Another limitation is the fact that we currently rely on the existence of audio files. Generating and processing even small audio files is problematic and has its own challenges. First and foremost this approach introduces a time processing delay. However using files was an advantage for testing in order to ensure repeatability of the experiments.

Testing in isolation from other systems, under controlled conditions, creates limitations. In reality this system will be exposed to a variety of people who produce different audio content even when speaking the exact same phrase.

3.3 Recognizing Faces and Logos

Recognizing faces and logos in video chat will improve the user experience and provide contextual information to applications such as PALTask. As the other context extraction modules of this chat application we consider this module separately. Reviewing existing literature and implementations of face and logo recognition software we recognize the need for specific conditions to succeed. These approaches can also generate false results in their implementations. Our approach differs from these existing implementations.

Figure 2 depicts the major steps in our face and logo recognition algorithm. Face recognition requires two main steps: First Viola-Jones face detection using Haar-like features [21], and then face recognition by Principal Component Analysis (PCA) [22]. The Viola-Jones face detection is implemented using an OpenCV library [23].

Separate from face detection is logo detection. It consists of logo location detection within the frame followed by logo recognition.

3.3.1 Viola and Jones Face Detector

Viola and Jones face detection uses Haar-like features, by computing very simple masks over small regions of a gray-scale image. The crude quality of the features allows for an extremely fast classifier for detecting faces [21]. This type of classifier sacrifices accuracy over speed, resulting in the occasional failure to detect a face or false positive by classifying inanimate objects as faces.

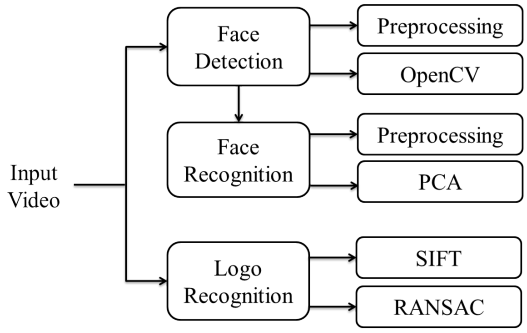

Figure 2: Block diagram of the face and logo recognition processing

3.3.2 Face Recognition

Once faces are detected, we attempt to identify them based on a predefined set of faces. This set can be context specific, and include coworkers, or team members depending on the contextual requirements. If faces are not recognized the system automatically rejects them for further classification.

To train the face recognition system we use 50 images, containing 10 persons with 5 images per class. The cropped faces captured by the face detector were used as the inputs of the recognition process; faces are centered and rescaled.

Even images for one person, can vary significantly in the color and direction of lighting, thereby bringing in more scatter within the same class on top of the variation of facial expressions and angle. The background could also vary, introducing differences between single images of the same class. To alleviate these effects, some preprocessing step were performed, including masking that crops out the face part only, color to gray transformation, and histogram equalization.

To complete the preparation steps, test images are projected onto eigenfaces using PCA. From this we have calculated the Euclidean distance between test images and training images to find the closest Euclidean distance which is recognized face.

3.3.3 Logo Recognition

While logo detection once focused on documents [24], the focus has recently shifted to video [25,26].

For logo detection we collected video of clearly visible logos from various orientations. Figure 4 illustrates the logo detection algorithm. Initially,

Figure 3: Preprocessed training set sample images

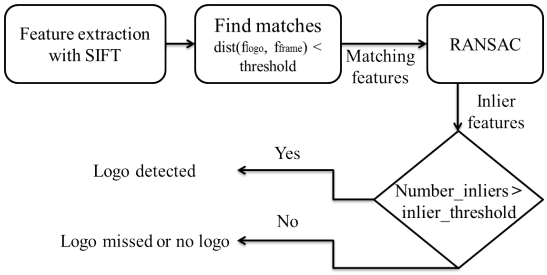

Figure 4: Illustration of the logo detection algorithm

Scale-Invariant Feature Transform (SIFT) features [27] are extracted from both the logo and the query frame. A nearest-neighbour matching between the features from both images is then performed, followed by a ratio test, which will filter the features by their distinctiveness. A set of matched features is forwarded for geometric consistency verification performed by Random Sample Consensus (RANSAC) [28]. As a result an affine transformation is produced which links the sets of spatial features from both logos.

4 Evaluation

In this section we evaluate the context extraction systems. Section 4.1 provides an evaluation of the component responsible for keyword extraction

from text. An evaluation of the audio to text analysis module is presented in Section 4.2. Last, we show an evaluation of our face and logo detection mechanism in Section 4.3.

4.1 Context Extraction from Text

An evaluation of RAKE has been illustrated by Rose [15]. Of note is that this evaluation was based on text sources which were almost guaranteed to follow proper grammar and spelling rules, two factors which we have identified as posing problems to using RAKE with standard input sources. Metrics showcased by Rose, such as precision, F-measure and recall, demonstrate that RAKE provides results comparable and often exceeding those obtained with existing state of the art techniques [15]. RAKE's evaluation has been shown in [15], which means the concept and algorithm of RAKE is demonstrated to be fast and accurate for document oriented analysis. Our use case is slightly different in the sense that input text is often short, and not guaranteed to follow accepted spelling and grammar rules.

Thus, we analysed RAKE in three distinct modes:

- RAKE without modified stoplist
- RAKE with modified stop list
- RAKE with text obtained from Audio to Text module

First, RAKE when operating with the standard stop list is evaluated on input texts commonly found in chat messages. The characteristics of these input texts are such that they contain *chatspeak* terminology, are relatively short (less than 20 words) and occasionally contain spelling mistakes. When tested with these input texts we found that RAKE evaluates these terms very often as keywords despite them actually being stop word candidates. This is observed when *chatspeak* is injected in the text. *Chatspeak* is often recognized as a keyword. Likewise misspelled stop words are erroneously identified as keywords. It total, this leads to incorrect results for the system.

Secondly, we evaluate RAKE with a modified stop word list. We manually expanded the stop word list, words which are common and generally not relevant to be uniquely identified as a meaningful keyword. The expansion included *chatspeak* terminology and some misspelled variations of existing regular stop words. Applying RAKE to these same input texts results in a much improved performance. Now only actual keywords were successfully identified. Misspelled words found in the stop word list and *chatspeak* was correctly ignored and not presented as keywords. This approach required us to manually append the stop word list, but we see no reason to believe that this can not be achieved automatically in the future.

Lastly, we tested the usability to RAKE when the Audio to Text module provided the input text to the keyword extractor module. A typical input text in this scenario does not contain spelling mistakes because the input text is obtained from a dictionary file. However, the sentence structure is unreliable and sometimes erratic. This causes words to be in different positions in a sentence simply because of the audio to text module's performance. Additionally, in the overall system, the lack of 100 percent accuracy in audio to text transformations means that while RAKE performs correctly, the results actually do not contain contextually relevant keywords. This however is a limitation of the Audio to Text module. Until this module improves, we give lower weight to keywords obtained via this module and instead promote the results obtained via text input.

Overall the keyword extraction module's did not require major modifications to be used by PALTask and the existing prototype. Small modifications in usage and input parameters were carried out to ensure accurate performance within a chat application. Challenges arising from pure text input in chat applications are overcome by modifying stop word lists, anticipating and internally correcting spelling errors, and by keeping a history of recent chat messages to present RAKE with a larger body of text. Input from the Audio to Text module is problematic purely because of the unreliability of the Audio to Text module. Poor quality results from the Audio to Text module, subsequently result in incorrect results from the keyword extractor. To compensate for this, results are ultimately weighed differently to address their high degree of inaccuracy in the current implementation.

Lastly, since pure text input is relatively short. RAKE is often unable to discern keywords due to the high ratio of stop words. We find that keeping a history of recent past messages improves performance significantly since it increases the input

space on which RAKE can perform analysis. It is beneficial to additionally weigh keywords higher when they are found in the most recent message provides further benefits to obtaining relevant contextual keywords.

4.2 Context Extraction from Audio

As mentioned in Section 3.2, we are using CMU's *Pocketsphinx* software as a foundation for the Audio to Text module. *Pocketsphinx* was designed to work with a medium size dictionary and much of the original evaluation is based on execution time performance [20]. While execution time is a critical criteria, we focus on accuracy of the returned results.

Initially we tested *Pocketsphinx* in its unmodified state. Further we did not create custom acoustic models, grammars, or train the system to a particular speaker. The result was unusable in our application. We provide a wav file containing a phrase composed of thirteen words to *Pocktsphinx*. At first the application returns thirty-six words, none of which are in the supplied phrase. Altering acoustic models or training the system to particular speakers is not realistic for applications which are used by diverse groups of people. As an alternate approach to this method, we modified the dictionary file. This file contains words which the system is capable of recognizing. Altering this file allowed us to provide a variety of distinct definitions for the words we were interested in. We also reduced the dictionary size considerably to less than one-hundred. With these modifications we were able to obtain reasonable results for the test phrase. Now the system returned seven words, all of which were in the input phrase.

Since this component of our chat application is tasked with obtaining keywords from the audio, we forward the extracted text to our keyword extractor. The keyword extractor's results depend on the quality of the input text provided by the audio module. Thus the overall quality of extracted keywords is dependant on the quality of the Audio to Text module. Poor transcriptions from audio to text will ultimately result in poor quality of the returned keywords from this module. Improving the Audio to Text module is expected to increase the quality of results significantly. At the current moment, the results obtained from the audio to text module are treated with less significance or weight in comparison from keywords obtained solely from text. This is done to prevent incorrect results from tainting the overall user experience by augmenting irrelevant information to the chat application.

4.3 Context Extraction from Video

To evaluate the video face and logo recognition module, we need to regulate all thresholds and parameters that impact this systems performance. Different criteria settings affect positive identification of faces and logos, but are often coupled with an increased number of false positives in the detection phase. Consequently, we adopt thresholds designed to avoid false positives.

Our implementation was tested contained 120 face images of 12 different people, two of which were not in our database, and a set of 20 University of Victoria logos, 5 of which contained additional logos as well. The video was captured at 720p high definition and 30 frames per second with total duration of 3 minutes. The Euclidean distances between the test image coordinates and the training image coordinates are calculated and the closest training image is picked out. A threshold is set such that if the closest distance is above the threshold, the test face is considered unrecognized. If the value is below the threshold the face is associated with the identity of the closest match. By tuning the threshold value, we are able to achieve a total correct rate for positive matches of up to 94%.

Performance in terms of execution time is also an important evaluation criterion. The current implementation uses the Matlab-based VLfeat [29] implementation to implement a scale invariant feature transform algorithm. Further, OpenCV is used for face detection [23]. The complete face and logo recognition system was implemented and tested under MATLAB. While the performance of this implementation is acceptable, speed improvements are expected when transitioning this code to a C++ implementation once the prototype evaluations are concluded.

5 Conclusions

Using the keyword extraction modules as well as the face and logo recognition engines we can use the obtained information to further augment the chat environment for each user. When done cor-

rectly, by extracting and detecting the correct context, we hope to be able to improve the user's experience by automatically providing a rich set of additional information.

Evaluating the individual context extraction components has highlighted that several aspects of them perform well even at a prototype stage. However, this evaluation has also shown that some aspects, particularly audio to text conversions, present challenges which negatively impact the perceived results. By separating the evaluation of these systems we were able to isolate problem areas in terms of quality of results and other performance metrics. From this analysis it became apparent that the audio to text module requires significant improvements, while the text keyword extraction module only requires minor alterations to satisfy our use case.

Obtaining contextual information from sources available in chat applications, video, audio and text, will enable us to selectively augment the user experience with useful information. This paper has highlighted several aspects of such as system and provided an overview of how context extraction can be carried out. Combining the obtained context information with a users personal context sphere is also expected to further increase the usefulness of such a chat application.

6 Future Work

Personal communications applications have a long history. Their basic functionality has remained throughout the years. Recently, improved technology such as network bandwidth and processing power have propelled the technology to expand beyond purely facilitating communication. Context awareness and augmentation of user experiences with additional information has become available. In the future we plan to fully integrate the components described in this paper to a fully functioning chat application. Several aspects of this are already integrated in a prototype [1]. Future work also includes an attempt to improve the quality of the results of each of the individual components. Primarily the audio to text module requires changes to provide improved results. At the moment the entire system is in an early experimental stage. Likewise the keyword extraction from text will be improved by automatically configuring the system to use smart stop word lists which will not depend on correct spelling and usage of words found solely in dictionaries.

Additionally, to provide personalized information augmentation we plan to incorporate the users personal context sphere [2, 3]. Integrating the personal context sphere enables the chat application to customize not only the results, but also to influence the importance of selected aspects of extracted contextual keywords to improve user experience.

Finally, context extraction, as described here, from video and audio is not limited to chat applications. Use of these techniques in other domains may provide viable applications. This includes providing this kind of context gathering to classroom settings or similar information exchanges.

We have chosen to apply context extraction of text, video and audio to personal communication applications. However, the techniques described in this paper can be applied to a wide range of scenarios which are not necessarily limited to chat applications. The method for object and face recognition in PALTask can be applied generally in other context based detection and recognition tasks. Likewise, the text extraction offers an alternative for statistical analysis which often require reference texts as a means of obtaining valuable results. Speech recognition work on static context has been done before with Siri and Google Now, but dynamic context extraction of audio can be expanded further to near real time levels.

Acknowledgements

This work is funded in part by University of Victoria (Canada), the National Sciences and Engineering Research Council (NSERC) of Canada under the NSERC Strategic Research Network for Smart Applications on Virtual Infrastructure (SAVI - NETGP 397724-10).

About the Authors

Andreas Bergen is a PhD student in the Department of Computer Science at the University of Victoria. His background includes distributed systems, operating system kernels and modularity of software. His interests are self-adaptive systems and adaptive software components. Email: andib@cs.uvic.ca.

Nina Taherimakhsousi was born in Iran, in March 1982. She received the B.Sc. degree in computer engineering-hardware from IAU Najafabad branch,Iran and the M.Sc. degree in Computer Engineering-software from Ferdowsi University of Mashhad, Iran, in 2006 and 2008,respectively. She started her PhD degree in January 2012 in University of Victoria, where she work on Context-based face recognition system. Her research interests include human and machine vision, self-adaptive systems, Intelligent system,pattern recognition, data mining. Email: ninata@cs.uvic.ca.

Pratik Jain is a Masters Student in the Department of Computer Science at the University of Victoria, Canada. He has completed his Bachelors degree in computer science and engineering from UPTU university, India in 2009. He also has industry experience of three years in Infosys Technologies, Bangalore, India and has worked on projects related to IBM DB2, sql, Java, mainframes etc. His research interests includes self-adaptive systems, context gathering, data mining, application development. Email: pratik@cs.uvic.ca.

Lorena Castañeda is a PhD Student in the Department of Computer Science at the University of Victoria, Canada. She is a CAS student at the Center for Advanced Studies at the IBM Toronto Laboratory. She holds a double degree in Engineering: Computer Engineer (2007) and Telecommunications Engineer (2007), and a Master in Information and Communications Management (2012) from Icesi University, Colombia. Her research interests focus on smart applications, self-adaptive situation- and context-aware systems. Email: lcastane@cs.uvic.ca.

Hausi A. Müller is a Professor, Department of Computer Science and Associate Dean of Research, Faculty of Engineering at University of Victoria, Canada. He is a Visiting Scientist at the Center for Advanced Studies at the IBM Toronto Laboratory (CAS). Dr. Müller's research interests include software engineering, self-adaptive and self-managing systems, context-aware systems, and service-oriented systems. He serves on the Editorial Board of Software Maintenance and Evolution and Software Process: Improvement and Practice (JSME). He served on the Editorial Board of IEEE Transactions on Software Engineering (TSE) 1994-2000, 2005-2009). He is Chair of the IEEE Technical Council on Software Engineering (TCSE). Dr. Müller received a Diploma Degree in Electrical Engineering in 1979 from the Swiss Federal Institute of Technology (ETH), Zürich, Switzerland and MSc and PhD degrees in Computer Science in 1984 and 1986 from Rice University in Houston, Texas, USA. Email: hausi@cs.uvic.ca.

References

[1] P. Jain, A. Bergen, L. Castaneda, and H. A. Müller, "PALTask Chat: A Personalized Automated Context-Aware Web-Resources Listing Tool," in *1st International Workshop on Personalized Web Tasking (PWT 2013)*, 2013 - To appear.

[2] N. M. Villegas, H. A. Müller, J. C. Muñoz, A. Lau, J. Ng, and C. Brealey, "A dynamic context management infrastructure for supporting user-driven web integration in the personal web," in *2011 Conference of the Center for Advanced Studies on Collaborative Research (CASCON 2011)*, pp. 200–214, 2011.

[3] N. M. Villegas, *Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems*. PhD thesis, University of Victoria, Canada, February 2013.

[4] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Network, IEEE*, vol. 8, no. 5, pp. 22–32, 1994.

[5] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *First Workshop onMobile Computing Systems and Applications (WMCSA 1994)*, pp. 85–90, IEEE, 1994.

[6] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[7] A. Ranganathan, R. H. Campbell, A. Ravi, and A. Mahajan, "ConChat: A context-aware chat program," *Pervasive Computing, IEEE*, vol. 1, no. 3, pp. 51–57, 2002.

[8] C. Abela and K. Cortis, "SemChat: Extracting Personal Information from Chat Conversations," in *Workshop on Personal Semantic Data, EKAW*, 2010.

[9] 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)*, 2009.

[10] 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 *Handheld and ubiquitous computing*, pp. 304–307, Springer, 1999.

[11] T. Bauer and D. B. Leake, "Word sieve: A method for real-time context extraction," in *Modeling and Using Context*, pp. 30–44, Springer, 2001.

[12] Q. Huang, Z. Liu, A. Rosenberg, D. Gibbon, and B. Shahraray, "Automated generation of news content hierarchy by integrating audio, video, and text information," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 6, pp. 3025–3028, IEEE, 1999.

[13] X. Xing, Y.-L. Liang, H. Cheng, J. Dang, S. Huang, R. Han, X. Liu, Q. Lv, and S. Mishra, "Safevchat: Detecting obscene content and misbehaving users in online video chat services," in *Proceedings of the 20th international conference on World wide web*, pp. 685–694, ACM, 2011.

[14] J.-y. Hong, E.-h. Suh, and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

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

[16] Aneesha, "Rake implementation." https://github.com/aneesha/RAKE. Retrieved April 2013.

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

[18] S. Rybalko and T. Seltzer, "Dialogic communication in 140 characters or less: How fortune 500 companies engage stakeholders using twitter," *Public Relations Review*, vol. 36, no. 4, pp. 336 – 341, 2010.

[19] M. W. Berry and J. Kogan, *Text mining: applications and theory*. Wiley. com, 2010.

[20] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, vol. 1, pp. I–I, IEEE, 2006.

[21] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, vol. 1, pp. I–511, IEEE, 2001.

[22] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Proceedings (CVPR 1991)*, pp. 586–591, IEEE, 1991.

[23] G. Bradski, "The OpenCV library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[24] D. S. Doermann, E. Rivlin, and I. Weiss, "Logo recognition using geometric invariants," in *Proceedings of the Second International Conference on Document Analysis and Recognition.*, pp. 894–897, IEEE, 1993.

[25] M. George, N. Kehtarnavaz, M. Rahman, and M. Carlsohn, "Real-time logo detection and tracking in video," in *SPIE Photonics Europe*, pp. 77240B–77240B, International Society for Optics and Photonics, 2010.

[26] S. Y. Arafat, S. A. Husain, I. A. Niaz, and M. Saleem, "Logo detection and recognition in video stream," in *Fifth International Conference on Digital Information Management (ICDIM 2010)*, pp. 163–168, IEEE, 2010.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[29] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings of the International Conference on Multimedia*, pp. 1469–1472, ACM, 2010.