Augmenting Conversational Dialogue By Means Of Latent Semantic Googling

by

Robin James Senior

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
December 2004

Your file Votre référence
ISBN: 0-494-01065-7
Our file Notre référence
ISBN: 0-494-01065-7

Abstract

This thesis presents the concept of *Latent Semantic Googling*, a variant of Latent Semantic Indexing that uses the Google search engine to judge the semantic closeness of sets of words and phrases. This concept is implemented via *Ambient Google*, a system for augmenting conversations through the classification of discussed topics. Ambient Google uses a speech recognition engine to generate Google keyphrase queries directly from conversations. These queries are used to analyze the semantics of the conversation, and infer related topics that have been discussed. Conversations are visualized using a spring-model algorithm representing common topics. This allows users to browse their conversation as a contextual relationship between discussed topics, and augment their discussion through the use of related websites discovered by Google. An evaluation of Ambient Google is presented, discussing user reaction to the system.

Acknowledgments

I would like to thank my thesis supervisor, Dr. Roel Vertegaal, for his guidance and insight. I would also like to thank the other members of the HML for their comraderie, and my family and friends for their support and editing contributions. Many thanks to Dr. Purang Abolmaesumi for his help in editing this thesis and co-ordinating my defense.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The automated extraction of context from documents and communications is becoming an increasingly important topic of study. The progress toward ubiquitous computing has resulted in users relying on their computers for the majority of their daily tasks. Computers allow users to record and archive massive amounts of information, and serve as an extension to their own memory. However, this has resulted in information overload that may hinder or even negate the advantages of computer-assisted communication.

A recent example of how users are using technology to overcome overload is Google. It is frequently regarded as an oracle of sorts, granting users simple and accurate access to the wealth of information published on the Internet. However, Google requires users to manually perform explicit queries in order to find relevant information. This involves breaking from the task at hand, and switching to the act of querying.

This thesis aims to couple the power of Google with the notion of Implicit Querying. Implicit Querying augments user intelligence by allowing computers to become aware of the user's context. By monitoring what the user is typing, speaking or reading, the system builds an understanding of what they are interested in. This knowledge is used to present information relevant to the user's task. Implicit Querying systems work in the background and present information in the user's periphery. They avoid the user's focus on the task at hand. Implicit Querying is tightly coupled with the field of Augmented Intelligence, in which computers serve as an extension to human cognition. Augmented Intelligence sees the computer as a tool for assisting human reasoning. Doing so allows users to extend their own knowledge and reasoning abilities by having the computer perform tasks are either difficult or impossible for humans to accomplish. While this may seem a daunting challenge, humans have been using tools to augment the capacity of their minds for thousands of years, whether it be through the use of pen and paper, printing, or the internet. A simple filing cabinet relieves its owner of having to mentally organize and retrieve hundreds of documents; my thesis work aims to provide a similar service to organizing human conversation.

1.2 Problem

This thesis aims to assist and augment communication by putting forth a new concept for allowing computers to reason about contextual relationships amongst groups of words. I propose to achieve this through a process of Latent Semantic Googling (LSG), essentially an adaptation of existing Latent Semantic Indexing techniques to Google. By applying LSG to live conversations, we provide an alternative for understanding context based on knowledge structures represented in Google.

Current context-aware systems use simple keyphrase-matching algorithms to determine topics that are being discussed or are of interest to the user. This results in inaccurate context discovery, because it does little to filter useless inforatmation that causes information overload. More advanced systems for analyzing the semantics of large bodies of text, such as Latent Semantic Indexing (LSI), require a wide-ranging corpus of documents for accurate context discovery. These corpora tend to be domain-specific, and lead to poor performance when terminology is used that is unrelated to that specific domain.

Latent Semantic Googling is a variation of LSI that seeks to overcome some of these shortcomings. It does so by using Google search results as its corpus, thus removing both the requirement for a pre-existing corpus and the problem of domain impartiality. Instead of querying the entire Google search engine, LSG continually builds its open-ended corpus[1] from search results returned by Google. Semantic relationships between topics are gauged by the frequency of a topic's appearance in other topics' Google results.

As a demonstration of this concept, this thesis presents Ambient Google, an implementation of LSG. Ambient Google works by passively monitoring a user's conversation, using speech processing techniques to filter out the key topics that are being discussed. These topics are used to query Google, which returns related web sites. Each topic that has been discussed is then cross-referenced against previous topics. This allows for the discovery of context-overlap between topics, which can then be used to form relationships amongst the subjects that have been discussed in the user's conversation. Ambient Google visualizes topics in such a way that the relationships are evident to the user, allowing for simple navigation of the conversation's key points,

[1]An open-ended corpus is one that is growing or expanding during a sensemaking activity.

which further allows the exploration of each topic's related web sites.

1.3 Contributions

The main contribution of this thesis is Latent Semantic Googling. LSG provides a means for users to gauge the contextual relationships within a set of words, by using the Internet to generate its open-ended corpus. Standard Latent Semantic Indexing works on a static set of documents provided by the user, but LSG uses Google to increase the number of potential documents to over 4 billion [17]. By doing so, LSG offers a unique method of mining the data on the Internet by context as opposed to just keywords.

The second contribution of this thesis is Ambient Google, which uses LSG to contextually group the topics discussed in a conversation. Ambient Google extends visualization techniques used for conversation augmentation with a spring-model algorithm [11].

1.4 Overview

First, we will discuss background, covering implicit querying, augmented communication, Latent Semantic Indexing, and speech recognition systems. Next, we discuss how we implemented Latent Semantic Googling. Following that, we discuss the main components of the Ambient Google system: Speech input & parsing, Google querying, and information visualization. A user evaluation is then presented, followed by a summary and conclusions.

Chapter 2

Implicit Querying

Implicit Querying is a subset of the field of Augmented Intelligence. Augmented Intelligence takes advantage of a computer's ability to store and access huge amounts of information, saving the user from having to know everything about a situation. Systems focus on understanding the situation and presents the user with informed options that will assist them in performing their task.

Traditional searches for information on a computer are typically expressed in the form of an *explicit query* [7]. This means that the user has to manually state the parameters of a search, such as the keyphrase, document location, etc. To do so, the user must switch from the current task to the search task. The user must then recall the specifics of the information that is required, enter it, and wait for the engine to query the database. If the parameters were incorrect or not specific enough, the user must repeat the query until the desired file or information is found.

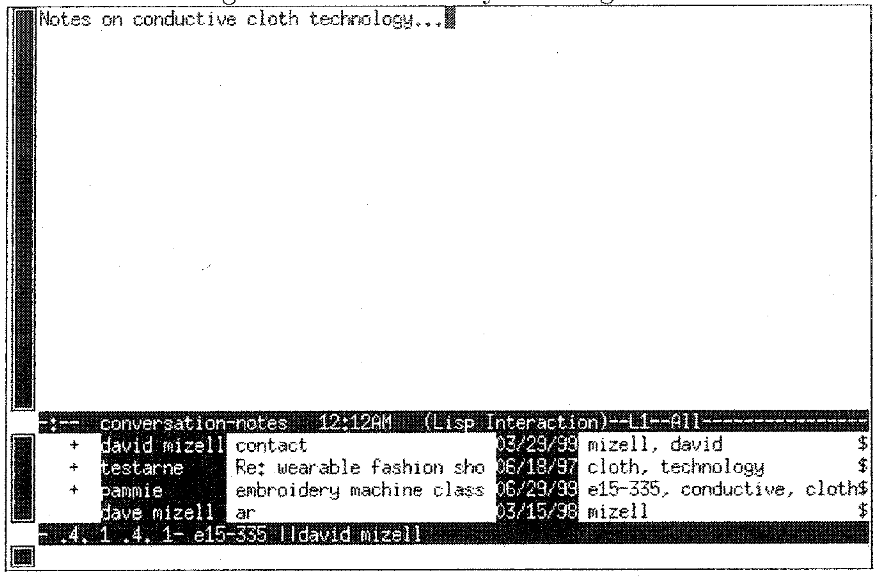
As opposed to explicit querying, *implicit querying* allows the user to augment his or her work environment with an automatic presentation of documents that are relevant to the current task. To achieve this, the system tracks the context of the user's task, and uses this information to gather related information in the background. Discovered information is presented in the user's periphery, and is easily moved to the foreground of user attention.

Fundamental to the concept of Implicit Querying is the notion of *context awareness*. The system must have knowledge of what the user is doing, and what documents relate to that work. Context awareness is typically accomplished through the use of *similarity metrics*, algorithms that judge word co-occurrence amongst documents. If the similarity metric is highly accurate, there is a better chance that the system will provide relevant feedback to the user.

2.1 Previous Implementations

Rhodes' *Just-in-time Information Retrieval* (JITIR) [26] was an early example of the use of implicit queries for document retrieval. Rhodes' system, *The Remembrance Agent*, was described as a "continuously running automated information retrieval system." The Remembrance Agent focused on retrieving the user's personal files stored on his or her computer. It did so by monitoring the text entered by the user into the EMACS text editor and by comparing it to a corpus of documents such as emails and text files (See Figure 2.1). Rhodes found that there is "a large difference between relevant suggestions (those that have a strong relation to the user's current context), and useful suggestions." Items may have been semantically related, but either contextually unimportant or hopelessly out of date [26]. Rhodes emphasized the importance of continuous monitoring, and of presenting information in the periphery. While the Remembrance Agent succeeded in its gathering of related information, its filtering was too insufficient to be of any use. A subsequent system developed by Rhodes, *Margin Notes*, expanded the JITR project to monitor web pages being viewed by the user in order to extract situational context.

Figure 2.1: Rembrance Agent: a continuously running automated retrieval system.



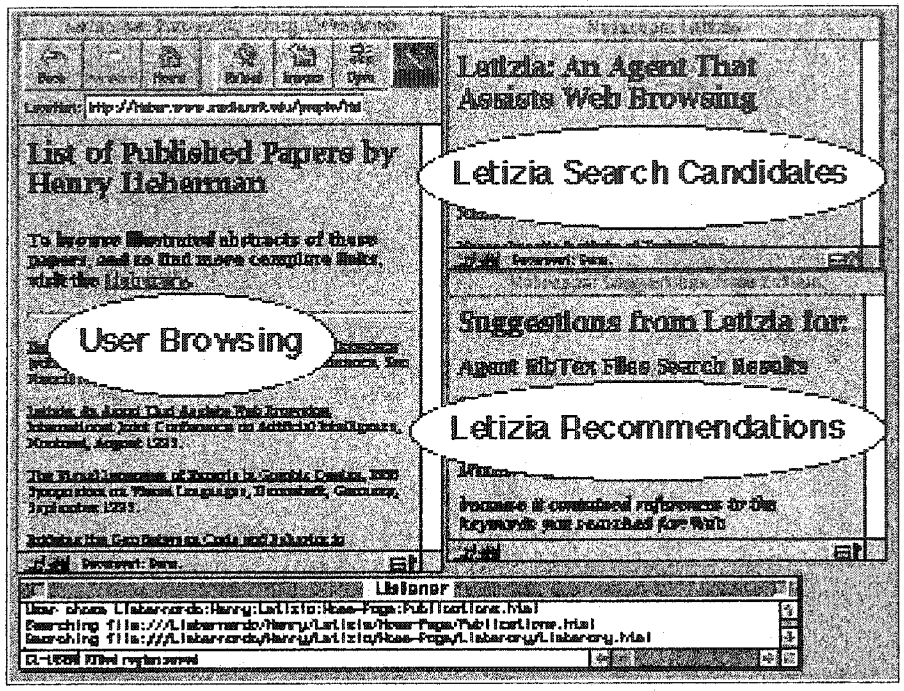
```
Notes on conductive cloth technology...

-:--  conversation-notes   12:12AM   (Lisp Interaction)--L1--All---------------
  +   david mizell contact                   03/29/98 mizell, david                $
  +   testarne     Re: wearable fashion sho  06/18/97 cloth, technology            $
  +   pammie       embroidery machine class  06/29/99 e15-335, conductive, cloth$
      dave mizell  ar                        03/15/98 mizell                       $
- .4. 1 .4. 1- e15-335 ||david mizell
```

Similarly, Lieberman's [22] *Letizia* augments a user's web browsing tasks by monitoring the pages currently viewed, and by recommending hyperlinked pages that are contextually related to what the user appears to be interested in (Figure 2.2). This is done by pre-fetching web pages that are hyperlinked from the current page, and by using heuristics that estimate how related that page is to what the user has previously viewed. If it is deemed relevant, the system shows the web page in the periphery of the user's screen, at no point taking control of browser navigation. By doing so, it allows the user to skip past irrelevant hyperlinks and instead see which links might be of use.

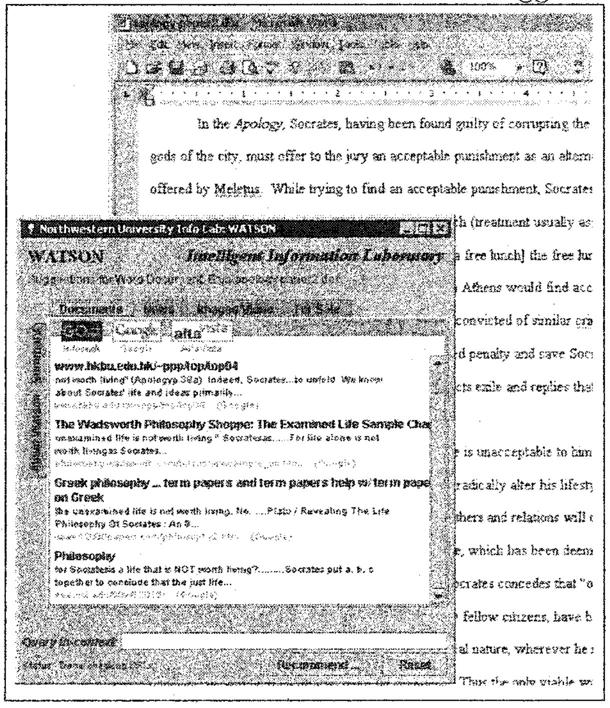
Budzik's *Watson* [4] is another system that provides forward-searching of web pages (Figure 2.3). Unlike earlier systems, Watson is not limited to web pages that

Figure 2.2: Letizia: a system for providing websites that are contextually related to the site being viewed.


List of Published Papers by Henry Lieberman
Letizia: An Agent That Assists Web Browsing
Letizia Search Candidates
User Browsing
Suggestions from Letizia for:
Agent BibTex Files Search Results
Letizia Recommendations
Listener

are linked from the current web page. In fact, the user need browse the web to receive suggestions. Watson monitors the user's word processing input, and extends the notion of Information Augmentation by expanding its knowledge base to include information on web sites that the user has never seen before. Watson uses the Altavista search engine to discover related documents on the Internet, providing an automated querying tool to the user.

Figure 2.3: Watson: monitors user interactions and suggests related websites.



In the *Apology*, Socrates, having been found guilty of corrupting the
gods of the city, must offer to the jury an acceptable punishment as an altern
offered by Meletus. While trying to find an acceptable punishment, Socrates
h (treatment usually as
a free lunch] the free lu
Athens would find acc
convicted of similar cr
d penalty and save Soc
cts exile and replies tha

is unacceptable to him
radically alter his lifest
thers and relations will
e, which has been deem
crates concedes that "o
fellow citizens, have b
al nature, wherever he
Thus the only viable w

Northwestern University Info Lab: WATSON

WATSON *Intelligent Information Laboratory*

Documents | News | Images/Video | For Sale

www.hkbu.edu.hk/~ppp/op/op04
not worth living" (Apology 38a). Indeed, Socrates...to unfold. We know about Socrates' life and ideas primarily...

The Wadsworth Philosophy Shoppe: The Examined Life Sample Cha
unexamined life is not worth living." Socrates.... For life alone is not worth living as Socrates...

Greek philosophy ... term papers and term papers help w/ term pape on Greek
the unexamined life is not worth living. No. ....Plato / Revealing The Life Philosophy Of Socrates : An 8...

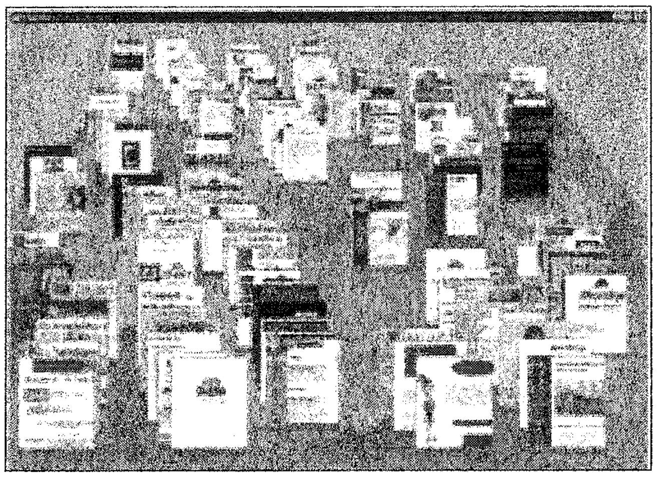
Philosophy
for Socrates is a life that is NOT worth living?.........Socrates put a, b, c together to conclude that the just life...

Query in-context:

Recommend ... Reset

Czerwinski dubbed this form of searching "Implicit Querying." Czerwinski's system is similar to that of Lieberman's, but uses the information from the currently viewed web page to organize and present related web pages from a set of bookmarked pages (Figure 2.4). It does so through simple content comparison algorithms. Czerwinski represents bookmarked pages by small screen captures of the actual web page. This allows the user to take advantage of distinguishable graphics on the page, allowing for visual recall of the page's content [7]. Unlike Lieberman's system, where the page's preview gives a suggestion of the page's content, Czerwinski's tool is used simply as a means to jog the user's memory.

Figure 2.4: Czerwinski's Implicit Querying tool; provides a visual representation of related bookmarks.



IBM has developed *Suitor*, a framework for creating customized monitoring agents [23]. The architecture of Suitor is such that several agents operate on a shared memory workspace known as the blackboard. There are three classes of agents: Investigators, which collect data and post it to the shared blackboard; Reflectors, which process the collected data and repost new information; and Actors, which perform an action on the collected data to present it to the user. Suitor has been used to create a broad range of attentive systems, due to its extensibility. The framework is designed such that different types of Investigators, Reflectors and Actors may be placed in the system, allowing for a broad range of applications.

Chapter 3

Augmented Communication

As computer networks become commonplace in today's work, academic and social environments, computer-assisted communication (such as email and instant messaging) is becoming increasingly popular. This form of textual communication has immediate advantages over hard-copy text, such as the speed of transmission, and the ability to archive and search conversation transcripts. These factors are changing the face of written communication, allowing for computers to further aid collaborative interaction.

While written text may lack the auditory cues that are present in voice conversations, and the non-verbal cues that exist in face-to-face conversations, it has the advantage that it is easy to process by computer algorithms. Speech input as a means of precise transcription still features an unacceptably high error rate. Textual communication is rapidly becoming a primary discourse medium, resulting in a prime opportunity to extend and augment this form of communication.

3.1 Simple Augmentation

In addition to the previous examples of archiving and searching, text-based conversation can also be augmented with simple content analysis.

Cockburn and Thimbleby created *Mona*; an "email system that provides an automatic hypertext representation of conversational context" [5]. Mona was an early attempt at classifying conversational structure, allowing for email collections to appear as related discussions, as opposed to being arranged in the order they are received. The context extraction is done entirely without user input, and is accomplished by means of heuristics gleaned from standard email headers. The heuristics are based on factors such as the names and addresses of the sender and recipients, the time the message was sent, and the time it was received. Mona combines this knowledge with a record of previous communications, and "infers probable relationships" [5] between messages.

A more recent example of email context extraction is *Gmail*, a free email service provided by Google [15]. Gmail provides context-relevant advertisements that appear alongside users' messages. Gmail analyzes the content of the current email, which is then used to select an appropriate advertisement. The entire system is completely automated, thus protecting the users' privacy. This form of augmentation not only benefits the advertisers, but also the users who may find Gmail's targeted advertisements more useful than those that are randomly selected. Gmail also provides a conversation-based approach to viewing emails, as opposed to the standard folder-based view. Messages are grouped by context, and when retrieving old emails, the user uses a contextual search engine, as opposed to sorting by sender or date. Gmail is still in limited beta-testing, but if it proves successful, it has the potential to further augmented communication.

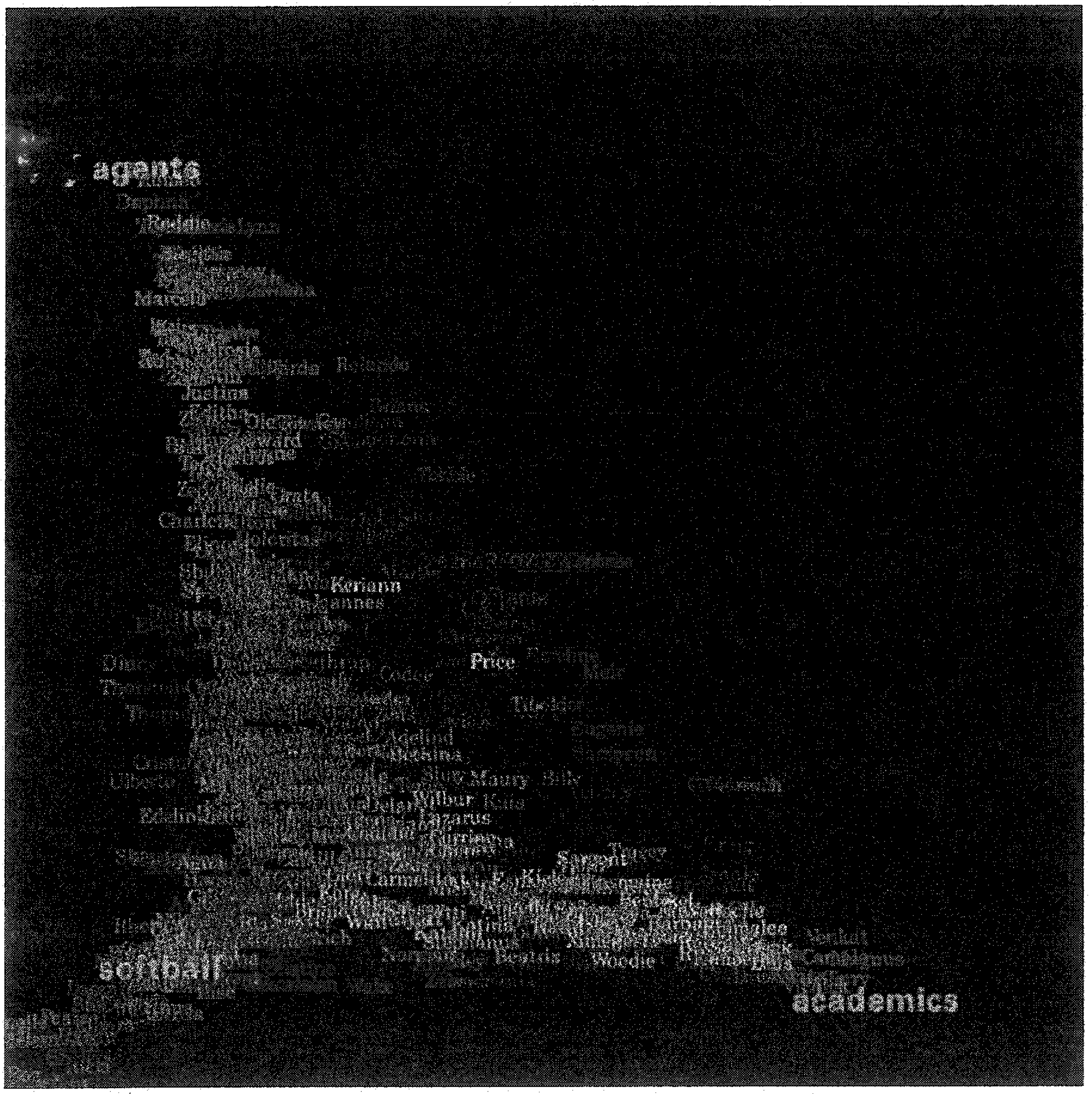
Mail.App, an email client for Mac OS X, also uses word relationships to help users filter out one of the negative side effects of computer-assisted communication. It deals with the increasing amount of Spam (unwanted bulk electronic mail) that arrives in inboxes by using Latent Semantic Indexing in its filtering algorithm. By analyzing the content of the user's emails, it can deduce which emails are legitimate.

3.2 Conversation Visualization

There are a number of systems that seek to remedy the social problems inherent in computer-assisted communication amongst groups. Donath designed **Visual Who**, a system for visualizing electronic communities by analyzing user membership across several *listservers*[1]. Visual Who allows users to discern people with related interests by looking at clusters formed in a window visualizing their interactions. Users can create anchor points from a list of listservers; users who are members are linked by means of a spring model (Figure 3.1). By using color to distinguish social groups (students, professors, etc.) and spring strength to discern listserver activity, the visualization helps the users understand the social interactions that are occurring.

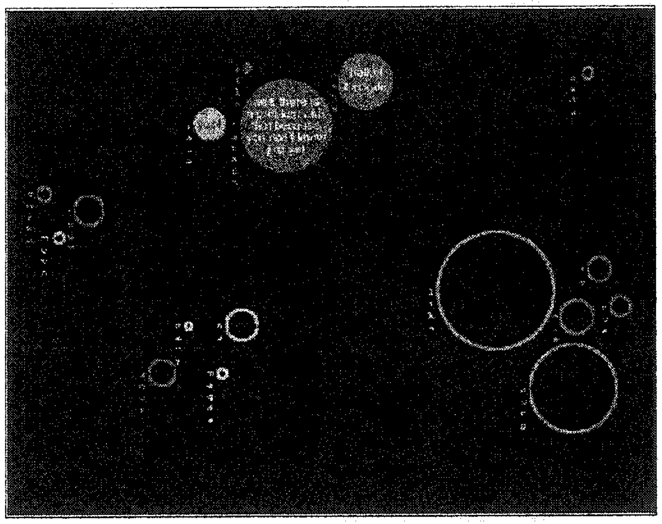
[1]A listserver is an automated tool for subscribing and posting to multi-party email lists.

Figure 3.1: Visual Who: anchors (in grey) distribute user names in accordance to their participation in listserv discussions.



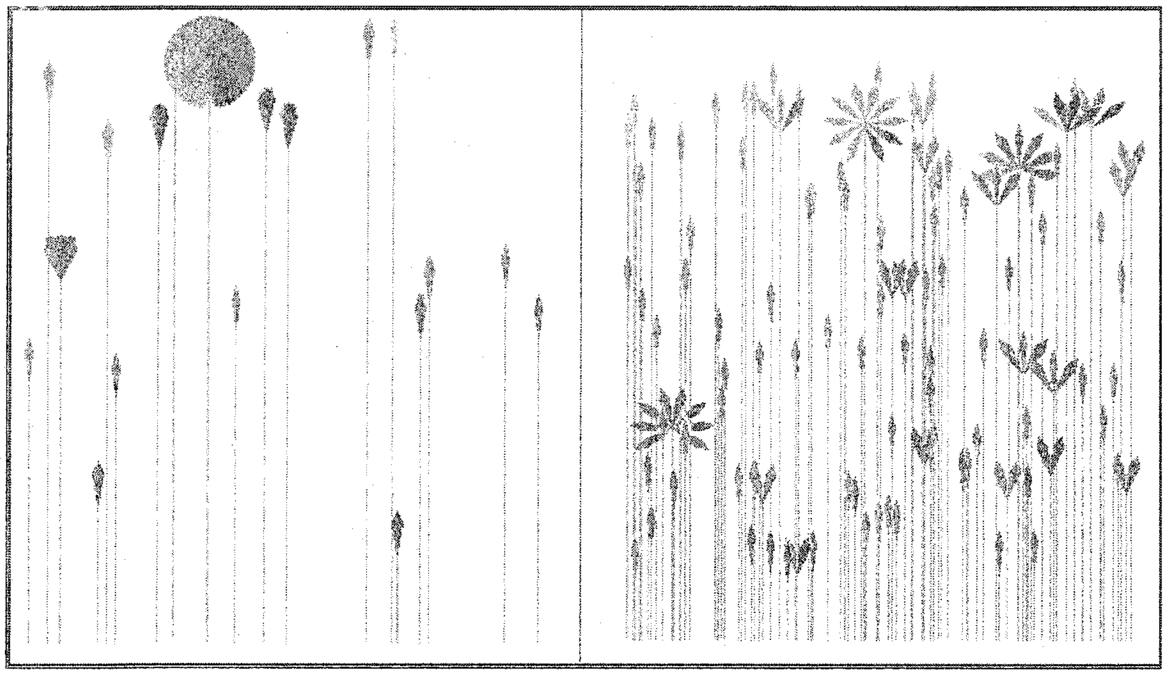
Donath also produced **Chat Circles** [12], a system that seeks to emulate face-to-face conversation in text-only chats (Figure 3.2). This is accomplished by using circular avatars that users can move to clusters of other avatars, allowing them to "overhear" each others conversation. Conversations in distant clusters cannot be heard, but the clusters can still be seen, showing an overview of all potential conversations.

Figure 3.2: Chat Circles: emulates face-to-face conversation by requiring the avatars of conversation participants to be close together.



Another related project is **People Garden** [12]. It visualizes the social dynamics of IRC (Internet Relay Chat) discussions. Each user in an IRC channel is given a "data portrait" in the form of a flower graphic (Figure 3.3). If a user is active in the discussion, his or her flower shows more petals. The power of PeopleGarden's visual metaphor allows for a quick understanding of the dominant voices in an IRC channel.

Figure 3.3: PeopleGarden: voices in an IRC channel are represented by a flower. More dominant voices result in a larger flower.



3.3 Augmented Speech

Simple systems for augmenting speech with transcription and command & control techniques have become quite common [see Chapter 5]. However, advanced augmentation for contextual awareness has yet to become popular in mainstream applications.

Context extraction for speech input allows for systems that can monitor discussions and provide assistance in real-time. Jebara et al. studied the use of tracking conversational context in machine-mediated discourse [20]. They designed a system that followed a conversation using speech recognition and topic modeling. The system used this information to query an archive of *Usenet* groups, to decide which group was being discussed. While this was a limited, closed-set experiment, it worked toward the idea of a machine acting as a third party to a conversation. However, as mediator, such machines must have some situational awareness or understand the conversational context such that they can intelligently assist the overall interaction. To perform this function completely, a system must be able to reason about an unlimited, open set of potential topics.

Systems that are aware of conversational context are likely to become more common as speech recognition systems become more ubiquitous. Interacting with machines as though they are humans enables them to understand social cues, allowing for proper turn-taking and limited interruptions [31].

Chapter 4

Latent Semantic Indexing

Deerwester et al. [8] conceived of Latent Semantic Indexing (LSI) as a method for extracting higher-level concepts from written human language. LSI allows a system to reason about semantic relationships in text, which can then be used to gauge the context of a document or discussion.

LSI is an approach for allowing computers to apply semantic reasoning to data: in this case, the words in the English language. LSI does not bestow an *understanding* of the data upon the computer; it is not artificial intelligence. Instead, it allows the computer to see patterns in unstructured data, and make these patterns understandable to the user.

Standard keyphrase searches can be unintuitive for humans to use, because they don't follow the same process that people use when searching for information. When a user researches information in a collection of documents, he or she doesn't simply look for the presence of single words, but also the relationships of the ideas behind the words. Relating concepts is a cognitive task that humans are much more adept at than machines.

LSI applies reasoning to the data set (or *corpus*) by calculating the *semantic distance* between words. When working with documents, it does this by looking at the content of each document, and comparing it with the contents of every other document in the corpus. The amount of content-overlap between documents results in the semantic distance between the two words. If the terms "usability", "accessibility", and "GUI" appear together in several documents, they are considered *semantically close*. Had they not appeared together, they would be considered *semantically distant.*

This concept of semantic distance is particularly useful when querying the entire corpus. Standard search engines use a literal search; if the user queries for "usability", only documents containing the exact phrase "usability" will be returned. Generally speaking, the documents with the highest occurrence of the query phrase are given a higher relevance ranking. This is a literal search, because the search engine cannot extrapolate the meaning of the query phrase to include related terms. Some search engines support stemming, which removes common suffixes of words, to allow for a broader search for varying forms of the query phrase. However, stemming only saves the user from having to repeatedly search for each variant of the root word[1], and does not increase semantic understanding.

Search engines that use LSI have the added benefit of returning documents that contain words that are semantically close to the query phrase, even though the document may not even contain the query phrase itself. A query for "usability" would additionally return pages that contain the word "accessibility", because the two words are semantically close. While the computer has no understanding of the words themselves, nor the concepts implied by them, it knows that they are somehow related due to their mutual appearances in several documents.

[1]Root Word: The form of a word after all affixes are removed.

Using LSI also gets around the problem of homographs [2] and multiple taxonomies for describing word semantics. This is due to the context that is provided through the LSI process; homographs are differentiated by the words they are each related to. The word "bow" could exist both in a grouping of words related to decorative knots, and in one related to canoes. Keyphrase searches would not be able to differentiate the two meanings; LSI solves this by using each group's context to separate them.

4.1 LSI Theory

The general concept behind LSI is to search for word co-occurrence in large set of documents. The words that will always have the highest co-occurrence are the ones that contribute the least to the content of a sentence, and are in fact the most commonly occurring words. These include functional words, conjunctions, prepositions, auxiliary verbs and others. These words are removed from the corpus, leaving only words that relate directly to the content of the documents. Yu et al. [37] present a standard algorithm for culling extraneous words from the corpus:

1. Make a complete list of all the words that appear anywhere in the collection
2. Discard articles, prepositions, and conjunctions
3. Discard common verbs (know, see, do, be)
4. Discard pronouns
5. Discard common adjectives (big, late, high)
6. Discard frilly words (therefore, thus, however, albeit, etc.)

[2]Homograph: A word that has the same spelling as another but different meanings or derivations

7. Discard any words that appear in every document

8. Discard any words that appear in only one document

The words that are left in the corpus are considered to have semantic value, which can then be used in the LSI process. The resulting corpus is used to create the *Term-Document Matrix*.

4.2 Term-Document Matrix

To calculate the frequency of occurrence scores, the corpus is arranged into a matrix, with the contents of each document along one axis, and the each individual word in the corpus along the other axis. For each individual word, the system iterates through each document and checks if the word appears in its contents. A binary flag is inserted at the intersection point of that particular row and column, indicating whether or not the word appeared. Once this has been repeated for each individual word, the Term-Document Matrix is complete.

4.3 Term Weighting

Term weighting is an extension of Yu's algorithm for reducing the corpus to just meaningful words, but instead of eliminating certain stopwords, term weighting gives each word a score of its value in determining the relationship between words. This involves two steps:

Local Weighting Words that appear frequently in a single document are likely to be of more value than words that appear only once. This is calculated after the initial stopword removal, so high-frequency words such as "the" would not skew this score. A formulation known as Logarithmic Local Weighting results in the value that is used for each word[37].

Global Weighting Words that appear infrequently over the entire set of documents are considered to be of high interest. This value is calculated using an Inverse Document Frequency (IDF) formulation. The IDF formulation is based upon the notion that words with a low occurrence frequency tend to have a higher probability of occurring in documents that are relevant to a query. This is because when discussing a concept, it is described by using general words and phrases, allowing the audience to comprehend the concept in familiar terminology[2].

4.4 Normalization

Normalization allows the system to accommodate documents of varying length in the corpus. Without normalization, documents of a few hundred words would not be able to compete with documents that are tens of thousands of words long, due to the amount of content that they contain. Longer documents simply have a greater chance of containing many keyphrases. The normalization process penalizes large documents and favors small documents, allowing them to compete on a level field.

This normalization score, multiplied by the local and global weighting scores, results in a final value that appears in the term/document matrix.

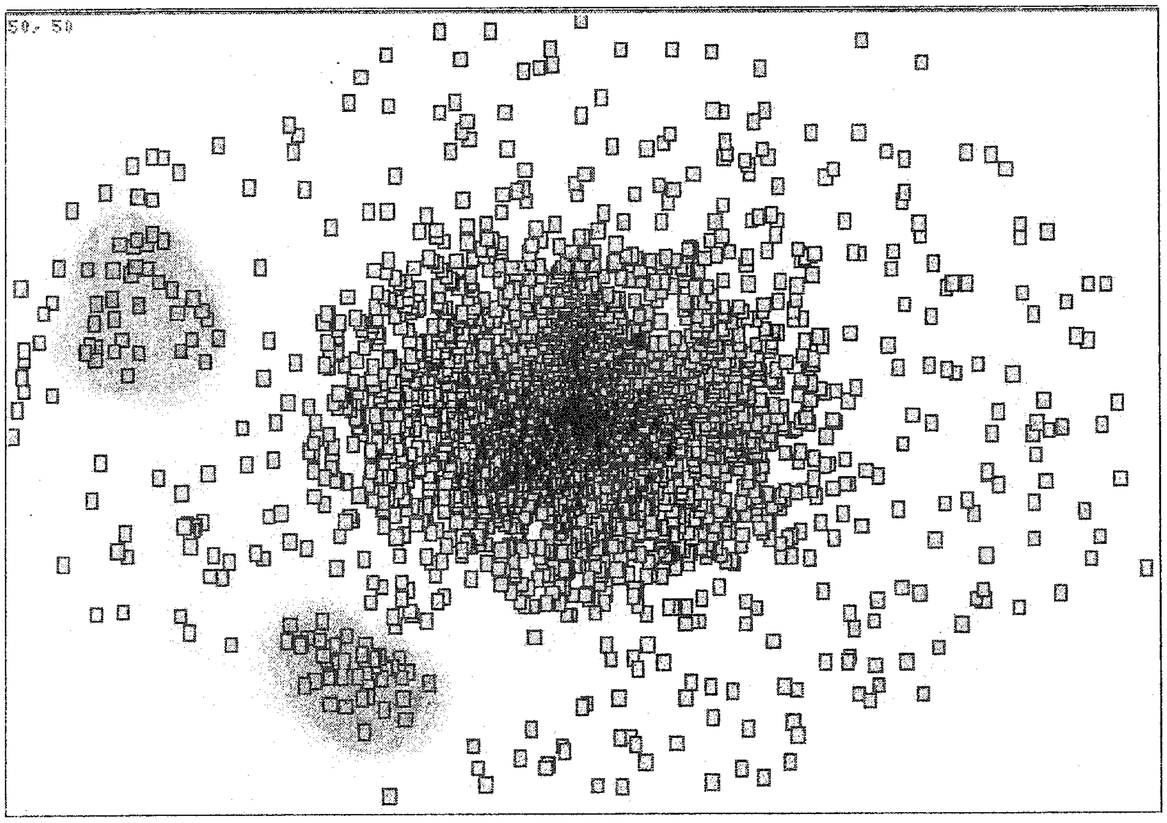
4.5 Singular Value Decomposition

A further application of LSI is the automated analysis of the term-document matrix. In order to distill the TDM into reasonable amounts of data, we must reduce its scope through the use of *Singular Value Decomposition* (SVD). The general purpose behind its use is that it allows us to compress hundreds of dimensions into just a handful. This optimal mapping of data points from one dimensionality to another compresses the matrix into a manageable set of overlapping word occurrences. These overlapping word occurrences are what are used to calculate the semantic closeness score. By applying SVD, systems can aggregate the semantic information to present documents that are related to terms that the user may be interested in.

4.6 Multi-Dimensional Scaling

LSI output can also be plotted for visual analysis. Once the scores for all terms are calculated, the semantic clusters are visualized using a *Multi-Dimensional Scaling* (MDS) process. MDS creates a two or three dimensional visual representation of the term-document matrix. This approach allows for humans to use their natural visual pattern recognition ability to detect clusters of related terms. Whereas applying singular value decomposition results in a perfect projection of the TDM onto a reduced space, multi-dimensional scaling uses an iterative approach to produce a visualization that is an approximate representation of the TDM. All terms in the TDM are initially scattered randomly around the visualization space. As the relationship scores are calculated, terms are drawn toward each other, depending on their semantic closeness. This results in clusters of terms, as shown in Figure 4.1. Clusters in the data become apparent, as shown in this example by Yu, et al. [37] The data being plotted is a number of Associated Press news stories; zooming in the colored clusters shows the individual data points to be contextually related.

Figure 4.1: MDS Visualization of News Articles [37]. Distinct clusters (shown in red and blue) indicate contextually related documents.



Chapter 5

Speech Recognition Systems

While speech recognition systems are not yet fully accurate, they are rapidly approaching that point [6]. Because of this, work on the applications of speech recognition must continue while the input systems are still evolving. The potential uses for speech input give rise to the need for continued development. According to Cohen and Oviatt [6], voice input is advantageous in the following situations:

1. When the user's hands or eyes are busy;
2. When only a limited keyboard and/or screen is available;
3. When the user is disabled;
4. When pronunciation is the subject matter of computer use; and
5. When natural language interaction is preferred

The first and fifth situations are most relevant to augmented intelligence and implicit querying. Both allow the user to perform one task while the computer performs another in the periphery.

5.1 Speech Recognition Methods

Speech input falls into two primary categories: *command & control* and *passive*. Each has its advantages and disadvantages, and is appropriate for different situations.

5.1.1 Command & Control

Command-driven systems are always passively listening, but to interact with the system, the user must first speak a word that is defined as a command word in the system's dictionary. Upon recognition of this command, the system processes subsequent input as arguments to the command. An example of this exists in modern mobile phones; if the user wants the phone to dial a number in its database associated with the name Jeff, he would speak the phrase "Call Jeff." The phone recognizes the command *Call* and processes the argument *Jeff*. It then looks up the phone number associated with Jeff and dials it.

The problem with command-driven interaction is that it creates unnatural conversational interaction for two reasons: the user must refer to the device directly, and also use a non-conversational set of words.

Referring to the device directly causes the user to treat the device as a tool, and not as a conversational partner. This may not seem like a problem when interacting with only one device, but the problem becomes evident as the number of devices increases. Interaction scenarios with multiple devices listening for commands require the user to first verbally identify which device they are speaking to, e.g. "Phone: Call Jeff" or "VCR: Record channel 3." This defies traditional turn-taking techniques that normally rely on non-verbal cues to identify dialogue initiative. In real-world multiparty conversations, speakers do not issue the name of the person they are addressing each time they speak. Simply casting one's gaze toward the intended recipient signifies who should be listening.

The second problem forces the user to perform a context switch, and recall the appropriate grammar required for interacting with a particular device. This is analogous to the use of command-line interfaces vs graphical user interfaces. While users may think that command-line interfaces are faster, it is only because their brain does not register the time that it takes to recall a command from memory. Loading this alternate grammar also causes the user to break from any conversations that are in progress.

5.1.2 Dictation

Dictation systems are generally used in word-processing applications to save the user from typing in large amounts of data. The first class of dictation systems uses a technique known as *discrete dictation*. The speaker must pause for a quarter of a second between words to detect boundaries, which causes a recognition event. The information from the recognition event is compared to a dictionary of words and their corresponding auditory features (this dictionary is known as a *grammar*). The system returns the dictionary item that best matches the spoken word. While command & control systems may have a very limited grammar to match their list of commands and arguments, a dictation system aims to have a grammar that is as large as possible. Some dictation systems use grammatical rules to choose the best match for recognized words, whereas some simply choose the word that most closely matches the features. A second class of dictation systems uses *continuous dictation*, which allows for more natural speech patterns. Continuous dictation can be less accurate and more processor intensive, but the benefits for usability are large. However, because users are not required to pause between words, it is harder to detect boundaries between spoken words. For example, when spoken, the phrases "the good can decay many ways" and "the good candy came anyways" are almost indistinguishable from one another. A continuous dictation system doesn't recognize word boundaries by silence, but instead continuously creates hypotheses as to what word may be spoken, and prunes off unacceptable options using a language model. When one of the word hypotheses reaches an acceptable degree of likeness with the spoken word, that hypothesis is chosen as the correct match. Advanced systems may actually go back and change a selected word if later evidence decreases its probability of correctness. [33]

The primary advantage of using a dictation system is that it does not force the user to break from his normal conversational routines. The system must rely on its own ability to understand the context of the speech, and use that context to perform the appropriate functions when required. Because of this, it is quite rare to see a dictation system used for anything other than transcribing speech input. Schneiderman comments that "Speech is slow for presenting information, is transient and therefore difficult to review or edit, and interferes significantly with other cognitive tasks." [29] However, dictation suits the purposes of Ambient Google (described in Chapter 7) because there is only one input command during dictation: that of performing a Google query. The control over when to perform this query is handled automatically by the system. Because of this, the user does not need to worry about when to execute a query; this allows for an uninterrupted conversation and a more natural speech pattern.

Chapter 6

Latent Semantic Googling

To extract context requires a wide-ranging corpus of documents. However, corpus documents are rarely impartial. To achieve impartiality, one must merge a large set of documents. We treat the information present on the Internet as a large corpus.

Latent Semantic Googling (LSG) is a variant of Latent Semantic Indexing (LSI) that uses the web sites indexed by Google to generate its corpus, or data set. The primary advantage of LSG over LSI is that the user is not required to have a large, domain-specific, pre-compiled corpus available ahead of time.

An overview of the LSG algorithm is presented later in this chapter. The exact implementation details will be discussed in chapter 7.

6.1 Overview of the LSG Algorithm

Sentences entered into the LSG system are processed in a fashion similar to that of Yu, et al. [[37]]. Instead of a complex English sentence full of low-value words, a processed sentence contains only words that are perceived to have some semantic value. Once a sentence is processed, it is sent to Google as a query string. Google in turn returns a result set containing the top n results (n=10 in the initial implementation). For each result in the set, the following information is included:

- The title of the web page;
- The URL of the web page; and
- A brief summary, or "snippet" of the web page's contents

Figure 6.1: Sample Google Result

Query String skiing swiss alps

Title Swiss Chalet family holidays rental Villars Alps Switzerland

URL www.swiss-chalet.net

Snippet ... resort of Villars sits on a sunny plateau at 1300m and is undoubtedly one of the best, and most easily accessible, dual season ski resorts in the Swiss Alps. ...

Google does not return the entire textual content of the pages. This content can be retrieved via a small script, but instead the current LSG implementation relies on the snippet alone to provide a glimpse at the page's content. This is to reduce the effects of document length on the LSG algorithm. Because the variance in snippet length is very small compared to the variance in the length of entire websites, we can eliminate some of the weighting steps required by LSI. Future implementations may fetch and search the entire HTML document.

Each time a query is performed, an object is created that contains the query data and the result set. These results are added to the initially empty corpus as they arrive. This results in a continually expanding (open-ended) corpus for cross-referencing query occurrences.

After a query is completed and its result added to the corpus, the system iterates through each Google result in the corpus and counts the number of times that the query string occurs within the text of each result. The result set that was generated by that particular query string is not included in the frequency count however, because the system is not interested in how often a topic refers to itself, only how often it refers to other topics. A query cannot access its own Google results, resulting in varying access to the corpus across queries.

The string matching is performed by using *regular expression* techniques[1]. Multi-word queries are handled as follows:

- If the words are surrounded by quotation marks, they are treated as one word, and pattern-matched as such. This is how Google itself functions, with quoted groups of words forced to appear next to each other, in that order. This allows for greater precision when querying. For example, there are currently 842,000 Google results for pages that contain both the words "Robin" and "Senior", but only 453 results for pages that contain "Robin Senior".

- If the words are not surrounded by quotation marks, each word is individually queried against the corpus. Each occurrence of each word results in an additional tally added to the reference count.

[1]A Regular Expression is a standard format for describing patterns that match various text strings.

6.1.1 Term-Document Matrix

The counting of references, rather than scoring binary presence only, is a distinguishing feature of LSG. LSI stores a binary record of the query strings appearance in the Term-Document Matrix, and then performs the term-weighting in a subsequent step. LSG uses the reference count as an alternative to LSI's term-weighting.

Once the frequency count for the new query string has been completed, the system iterates through all *previous* query strings, and does a frequency count for occurrences in the latest addition to the corpus. This allows for early entries into the system to cross-reference later additions to the corpus. Table 6.1 shows an example of the Term-Document Matrix once all cross-referencing has been completed. In this example, q_0 and q_1 are semantically close to another, as are q_3 and q_4. Other permutations, however, are semantically distant.

Table 6.1: Final Term-Document Matrix

	q_0's results	q_1's results	q_2's results	q_3's results
q_0	NA	3	0	0
q_1	4	NA	0	0
q_2	0	0	NA	2
q_3	0	0	2	NA

The primary difference between LSG and LSI at this stage in the algorithm is that the Term-Document Matrix (TDM) is both created and calculated as the corpus of query strings and Google results arrive. Traditional LSI calls for the corpus and TDM to be pre-computed, due to the overhead required in computing the Singular-Value Decomposition and/or the Multi-Dimensional Scaling. While LSG scores may be computed in real-time, the amount of computation required increases as queries are added. This puts a constraint on the system's ability to continually update the LSG scores. However, LSG has a significant advantage over standard LSI when it comes to approximating real-time score calculation: reduction of the search space.

For LSI to be effective, it must have a large enough corpus to draw from when calculating semantic distance. This is because each word in the corpus has no semantic meaning other than how often it appears relative to each other word. While LSG uses an extremely large dataset to draw from (the Google database), the actual corpus that is stored on the user's computer is relatively small, generally less than 300 words per query. Each query has its own 300 words that it is associated with, thus giving it semantic context. When other queries check for word co-occurrence, they need to only search 300n words, for a corpus constructed from n queries. This allows us to limit the document search space by an order of magnitude.

The Multi-Dimensional Scaling visualization is accomplished by presenting clusters relative to multiple anchor points (representing query strings of interest to the user) instead of a standard scatter-plot relative to one central anchor. This technique will be discussed at length in the discussion of Ambient Google, our first implementation of LSG.

Chapter 7

Ambient Google

7.1 Introduction

Ambient Google is an implementation of the Latent Semantic Googling technique, used for augmenting conversations with automated Google queries. Ambient Google uses a speech recognition engine to generate keyphrase queries directly from conversations. By extracting keyphrases rather than perfect transcriptions, Ambient Google overcomes typical accuracy constraints of speech recognition systems.

The system uses extracted keyphrases as Google query strings. The information returned from the Google query is used to create a knowledge base for analyzing the structural semantics and pragmatics of the speech input, using the Latent Semantic Googling algorithm.

As an implementation of LSG, Ambient Google seeks to visualize the contextual relationships between topics uttered by the user. This is accomplished by means of a two-dimensional visualization, in which users can create "anchors" from previously discussed topics. When a user has created two or more anchors, the other topics discussed in the conversation are placed in the visualization field, in relation to the anchors. If a topic frequently occurs in an anchor's Google results, the topic will be placed near to that anchor, because they are "semantically close." The user is free to add and remove anchors at will, or move them around the visualization to better understand the patterns expressed in the data.

Ambient Google further augments the conversation by creating a chronology of the conversation.

During one-on-one meetings, it is often difficult to take notes, because the participants are always either speaking or listening. Ambient Google frees the user from having to multi-task by keeping a chronological record of the conversation. Through the use of a list control, users can navigate through a linear representation of the chronology of their conversation. This interface allows incorporation of time as an organizational and navigational aid.

Even more distracting than taking notes during a conversation is the act of querying Google for more information about the current discussion point. This act is becoming more and more common as people learn to understand the power of Google. However, breaking from the conversation to perform a query causes a context switch to the user. Ambient Google prevents this by automatically querying Google with the discussed topics. As such, each topic has a subset of Google query results associated with it. To access this Google information, we incorporated a drill-down approach [27]. When the user enters a topic cluster with his mouse cursor, a contextual menu appears that contains the Google summaries of relevant websites. Selection of the summary in the menu causes the corresponding page to be loaded in an external web browser.

7.1.1 Design Rationale

According to Weiser and Brown [36], a ubiquitous computing system designed to complement a user's knowledge base should work on the periphery of the user's activities, possibly requiring only passive or implicit forms of input. By acting in the periphery, the user may continue his primary task, yet consult an interface seamlessly and without disruption of the focus. Traditional Google searching employs an interface that requires the user to manually input a search term and filter results. While users have become accustomed to this style of input, it requires them to stop working on their primary task and switch to the act of querying.

Figure 7.1 shows an early interaction storyboard that we developed for the system. The final interaction style is largely as it was depicted in the storyboard, with the only significant change being the removal of the fisheye view of the topic chronology. Figure 7.2 shows the final implementation of the Ambient Google user interface.

Figure 7.1: Interaction Storyboard. Users progress from speech input (1) to a conversational chronology (2), to a spring model placement of topics (3), concluding with drill-down web references (4).

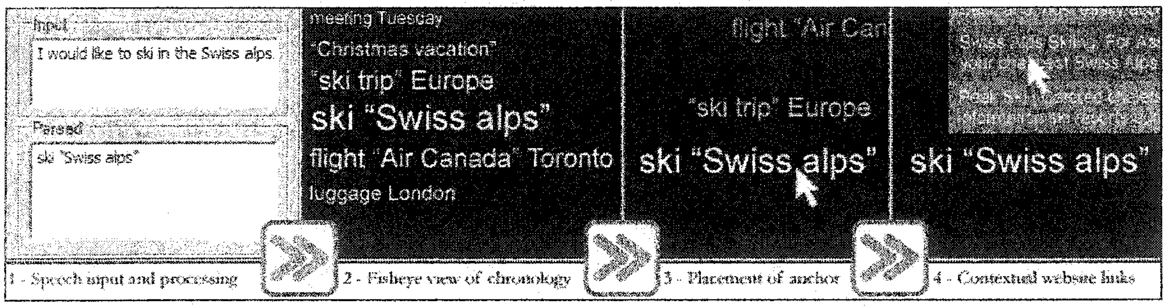
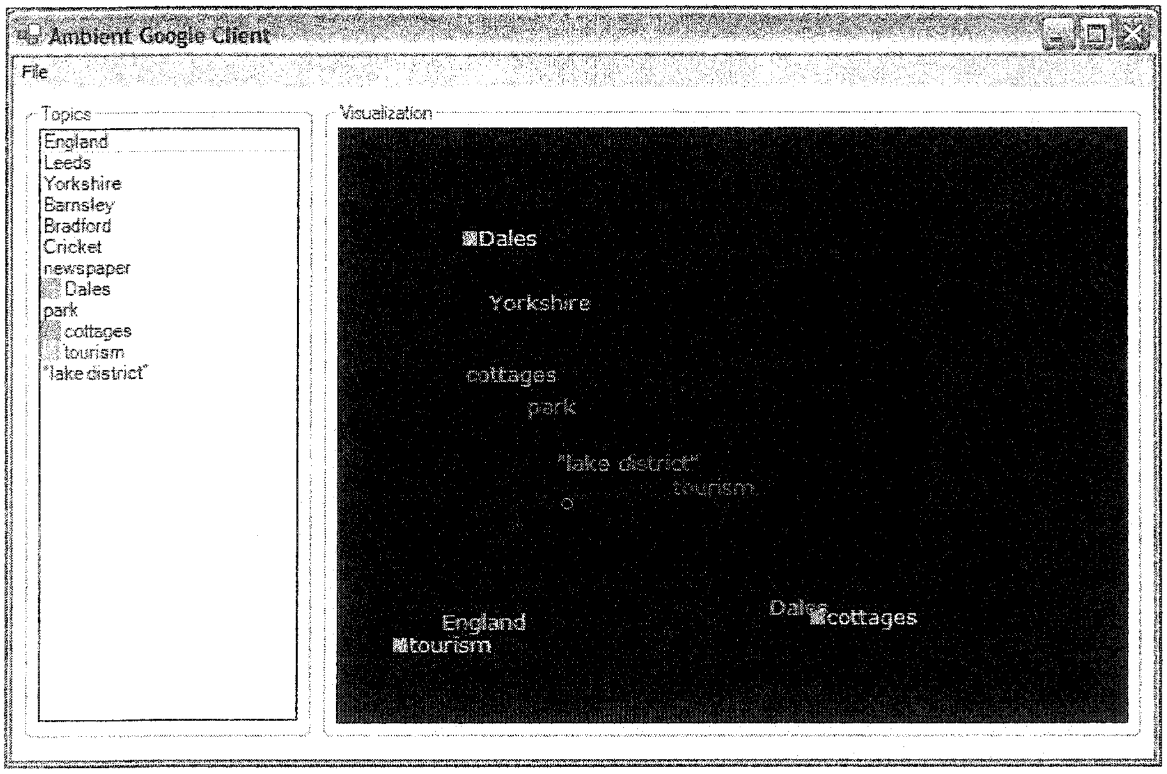


Figure 7.2: Ambient Google User Interface. Topic chronology appears at the left. Topics are dragged from the chronology to the visualization area on the right. Anchor colors relate topics in the chronology to their place in the visualization window.



7.2 Speech Parsing

Ambient Google augments conversations by recognizing and summarizing what is said, and submitting this information to Google. Due to the informal nature of natural language input, the error rate of speech recognition of normal conversational content is between 30-40% [9]. To increase robustness of our system, each Ambient Google user is required to wear a lapel microphone. We further improved robustness by summarizing the recognized phrases into noun phrases through a state-of-the-art language recognition system built on top of the Microsoft Speech API [24].

Ambient Google is only interested in the words in the speech input that can be used to query Google. Sentences are processed through the use of a keyphrase extractor; this allows the speech to be distilled to words that are considered relevant to our purposes.

7.2.1 Grammatical Noun-Phrase Extraction

Grammatical text parsers work by using the grammatical structure of a sentence to create a hierarchical representation (known as a parse tree) of its components. From this, contextually-relevant information can be extracted, such as noun or verb phrases. Grammatical extractors are suited to command & control SR applications because they allow the computer to gain an understanding of the input by recognizing commands and then backtracking through the parse tree to identify the subject that it is referring to.

The grammatical extractor used in Ambient Google is the Link Parser API, developed at Carnegie-Mellon University [32].

"The parser has a dictionary of about 60000 word forms. It covers a

> wide variety of syntactic constructions, including many rare and idiomatic ones. The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and assign some structure to the rest of the sentence. It is able to handle unknown vocabulary, and make intelligent guesses, from context and spelling, about the syntactic categories of unknown words. It has knowledge of capitalization, numerical expressions, and a variety of punctuation symbols."[32]

Despite my early concerns of using a grammatical parser for conversational speech, the Link Parser API is robust enough to handle all but the most incoherent of sentences. While it would be optimal on perfectly formatted text, it served well enough to work as the basis for the system.

The use of a speech parser was initially motivated by Barker et al. in their work on the use of noun-phrases for document keyword extraction. Through evaluation of their system, they found that "the simple noun phrase-based system performs roughly as well as a state-of-the-art, corpus-trained keyphrase extractor" [1]. Although they were discussing their own system, it is accepted that the Link Parser system is an equally robust implementation that uses similar techniques [1].

A noun-phrase is a part of a sentence that consists of a noun and its modifiers. The following examples were taken from a University of Calgary website [25]:

Noun phrase as subject

Input Sentence: The misty, eerie night cast a spell on us all.

Noun Phrase: The misty, eerie night

Noun phrase as object

Input Sentence I would love a nice, cold, vanilla shake right about now.

Noun Phrase a nice, cold, vanilla shake

Noun phrase as complement

Input Sentence: Calgary is a sunny location.

Noun Phrase: a sunny location

These examples do not include the simple nouns that exist in the sentence, such as "spell" in the first example, and "Calgary" in the third example, because they are intended to show the specific types of noun-phrases. The Link Parser API returns the following parse tree for each sentence (noun phrases are marked with NP):

Input The misty, eerie night cast a spell on us all.

Output
```
(S (NP The
       (ADJP misty ,)
       eerie night)
   (VP cast
       (NP a spell)
       (PP on
           (NP us all)))
   .)
```

Input I would love a nice, cold, vanilla shake right about now.

Output
```
(S (NP I)
   (VP would
       (VP love
           (NP (NP a
                   (ADJP nice ,)
                   cold)
               ,
               (NP vanilla shake
                   (ADJP right about
                         (PP now))
                   .)))))
```

Input Calgary is a sunny location.

Output
```
(S (NP Calgary)
   (VP is
       (NP a sunny location))
   .)
```

The Link Parser API performed admirably for these grammatically correct sentences. Here are some examples of its performance on grammatically malformed input:

Input This sentence contains puppy a superfluous word.

Output The system notes that word 'puppy' should not be there, and notes it as a *null link*: `this.d sentence.n contains.v [puppy] a superfluous.a word.n`

```
(S (NP This sentence)
   (VP contains puppy
       (NP a superfluous word)))
```

Input Well, you know, England is a very rainy country.

Output The system handles natural speech input quite well, identifying "England" and "country" as the noun-phrases.

```
(S (VP (ADVP well)
       , you know ,
       (SBAR (S (NP England)
                (VP is
                    (NP a
                        (ADJP (ADVP very)
                              rainy)
                        country)))))
   .)
```

7.2.2 Statistical Keyphrase Extraction

Statistical keyphrase extractors work by first determining the noun and/or verb phrases in a document, then analyzing their frequency and placement to create a ranking of keyphrase relevance.

The statistical extractor that we initially examined is known as *Extractor*, and was created by Peter Turney at the National Research Council of Canada.

> "The Extractor algorithm works as follows. For each phrase (i.e. a sequence of 1 to 3 consecutive words) in a document, Extractor computes a score representing the systems confidence that it is a keyphrase. The score is computed based on a series of features of the phrase such as: frequency of occurrence, position of the first occurrence in the text and length of the phrase. A series of 12 parameters determines how those features are combined to obtain the final confidence score. These parameters are tuned automatically by a genetic algorithm whose goal is to maximize the overlap between machine and human extracted keyphrases. Evaluation of the various possible tunings by the genetic algorithm is done using a training corpus for which humans have already extracted keyphrases." [10]

Unlike thoughtfully constructed text input, speech input tends to result in poorly-formed sentences due to stutters, pauses and train-of-thought errors, as well as the standard problems inherent in SR, such as poor transcription due to microphone or dictionary limitations. Because of this, a purely statistical extractor may have been better suited for Ambient Gooogle than a grammatical extractor. However, Extractor has only been tested on transcripts of SR sessions, and not on live SR. This is due to its algorithm being tuned for entire documents, rather than individual sentences. A major component of the algorithm looks at word frequency, and this is rarely a factor in single sentences. That said, it still functions quite reasonably. To use Extractor in Ambient Google, we would have had to buffer sentences until a significant amount of text had been accumulated before doing keyphrase extraction. This would not have allowed real-time querying, and would have negated one of the main goals of the Ambient Google project. Instead, we chose to use the Link Parser API on individual sentences, relying on its superior ability to extract noun-phrases.

7.2.3 Implementation

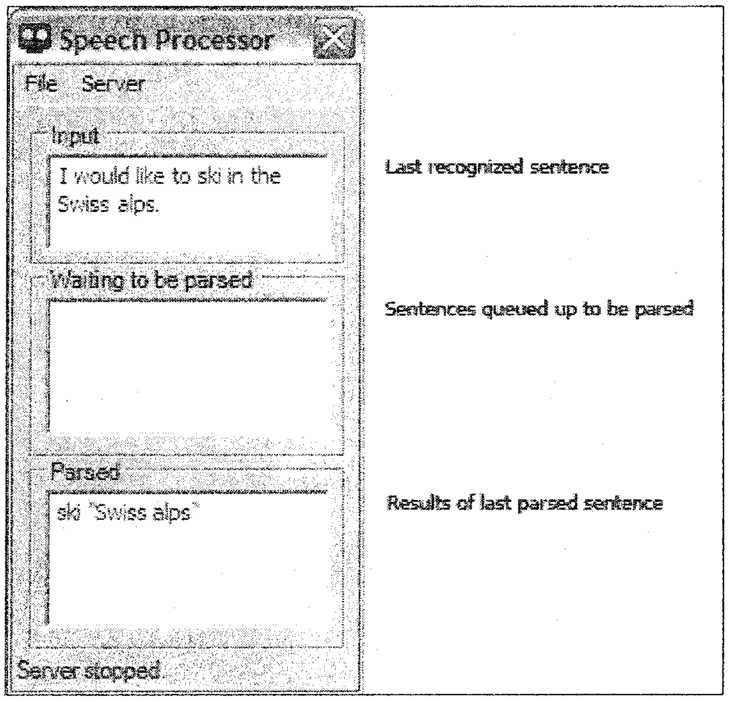
Voice input is processed by a C# application through the Microsoft Speech API v.5.1(SAPI) [24]. SAPI is a free, closed-source API that allows Microsoft Windows developers to incorporate speech functionality (command & control or dictation) to their applications. Our speech recognition engine functions at a rate that is consistent with other engines that we tested, including IBM ViaVoice [19] and Dragon NaturallySpeaking [28]. The speech processor runs in a separate window so that it doesn't interfere with the Ambient Google interface (as shown in Figure 7.3).

During development and testing, an inexpensive Plantronics microphone was used. As is the case with all speech recognition applications, the quality of microphone affects the consistency of recognition. The SAPI engine can be calibrated by having a user speak a number of sentences. Again, the more calibration that was performed, the better the recognition.

While our system uses continuous dictation for judging word boundaries, it uses discrete dictation for judging sentence boundaries. Simple heuristics are used to determine sentence breaks; if no SAPI recognition event occurs for 1.5 seconds, it is deemed to be the end of a sentence [30]. The recognition event occurs even for natural "umm" and "ahh" noises that occur in normal speech, so if a user simply stumbles over his words, the system will not assume it is the end of a sentence. After the 1.5 seconds has passed, the text in the sentence buffer is queued to be passed to the speech parsing function. Because the parser takes a moment to run, sentences must be queued so as to not interrupt the speech recognition process. Once the sentence reaches the head of the queue, parsing may begin.

As previously mentioned, we opted to use the Link Parser API [32] for extracting noun-phrases from the speech input. Link Parser uses a grammatical approach which parses sentences into their constituent parts. We adapted the algorithm to return only noun-phrases. When the noun-phrase contains more than one word, the sub-phrase is automatically surrounded by quotation marks by the link parser. When a query is submitted to Google, this summons the words to appear alongside one another, thus increasing the accuracy of the search [14]. The Link Parser API is robust enough to handle malformed sentences and unknown words [32], allowing for, or even correcting, imperfect sentence structures. The resulting noun-phrase groups are submitted to Google using the Google API [16].

Figure 7.3: Speech Processing GUI. Inputted speech is processed in a separate application before being sent to Ambient Google via TCP/IP.



I would like to ski in the Swiss alps.

ski "Swiss alps"

7.3 Querying Using the Google Search Engine

Since its creation in 1998 [3], Google has evolved as the dominant search engine on the Internet. At the time of writing, Google indexes over 4 billion web pages and receives over 200 million hits per day [17].

7.3.1 Google Theory

Google is regarded as the leader in search engine technology for two major reasons: Google indexes the largest number of web pages, and Google's PageRank algorithm returns the most accurate search results. These two factors are why we chose to use Google as opposed to another Internet search engines. The details behind the PageRank algorithm are described below.

From the Google Technology Overview [18]:

PageRank Technology: PageRank performs an objective measurement of the importance of web pages and is calculated by solving an equation of 500 million variables and more than 3 billion terms. Google does not count links; instead PageRank uses the vast link structure of the web as an organizational tool. In essence, Google interprets a link from Page A to Page B as a "vote" by Page A for Page B. Google assesses a page's importance by the votes it receives.

Google also analyzes the pages that cast the votes. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages important. Important, high-quality pages receive a higher PageRank and are ordered or ranked higher in the results. Google's technology uses the collective intelligence of the web to determine a page's importance. Google does not use editors or its own employees to judge a page's importance.

Hypertext-Matching Analysis: Unlike conventional search engines, Google is hypertext-based. It analyzes all the content on each web page and factors in fonts, subdivisions, and the precise positions of all terms on the page. Google also analyzes the content of neighboring web pages. All of this data enables Google to return results that are more relevant to user queries. As a result, millions of users worldwide look to Google as the fastest, easiest way to find exactly the information they're looking for on the web the first time.

7.3.2 Google Implementation

Google offers a free API [16] for accessing its database. The API uses SOAP (Simple Object Access Protocol) and WSDL (Web Service Definition Language) standards to allow an easy and efficient method for performing queries.

SOAP "Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses." [13]

WSDL "Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint." [13]

Using the Google SOAP implementation eliminated the need to write a HTTP parser for performing queries. The API identifies each user by a developer key, which limits them to 1000 queries per day. Full real-world use of Ambient Google would require more queries, but 1000 was sufficient for testing purposes.

7.4 Visualization

Speakers generate large numbers of potential candidates for query submission during a conversation. In order to improve querying results, our system filters speech into appropriate search terms. For example, the sentence "I would like to ski in the Swiss Alps" would not yield appropriate results as a Google query. Expert users familiar with search strategies intuitively know how to construct search terms; in this case "Ski Swiss Alps" would perhaps be a more appropriate candidate. Effective Google queries are essentially a collection of noun-phrases, with extraneous grammatical information removed [21]. For novice users, Google automatically filters out some of this information. Google ignores common words and characters such as "where" and "how", as well as certain single digits and single letters, as these tend not to improve results [14]. To summarize speech into noun-phrases, our system processes each spoken sentence with a speech parser. This results in a keyphrase grouping [1], which can subsequently be submitted as a Google query string. After keyphrases are submitted to Google, the system returns with a list of candidate website addresses.

Due to the continuous nature of our submission process, a large number of potential links are generated by the system. We will now discuss our technique for filtering this information via contextual relationships.

7.4.1 Contextual Topic Clustering

Like Visual Who [11], our visualization algorithm groups query results into topics through a spring-mass system. Our clustering engine allows users to choose items from a list of keyphrases, designating them as topic anchors that other keyphrases may drift toward, depending on their relevance to the topic. Users can organize clusters by placing topic anchors freely within the 2D visualization environment. This not only allows for a filtering of keyphrases into categories, but also creates an understanding of the connections between topic areas.

7.4.2 Relevance Ranking

The LSG algorithm works in a similar fashion to Google. If an anchor's text is "Swiss Alps", then the algorithm iterates through all other topics, and checks their Google results to see if they contain the phrase "Swiss Alps". Figure 7.4 shows how a score is calculated for topic T and anchor A, according to how frequently T appears in the summaries of A's Google Results R. This score allows us to determine the overlap between topic pairs.

Figure 7.4: LSG Scoring Algorithm

For topics T, anchors A, Google Results R:

$$\sum_{a=1}^{n} \text{score}(T, A_a) = \sum_{r=1}^{m} \text{count of } T \text{ in } R_r$$

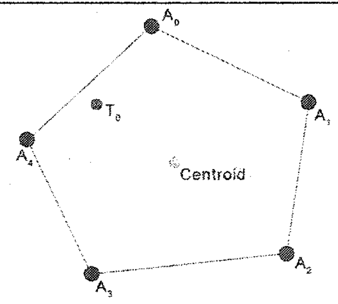
Due to the computational overhead generated by this algorithm, it is only used when an anchor is created, as opposed to every time a new topic is added. The overhead increases proportionally with the number of topics; an anchor created from the nth topic has to look at n-1 other topics, thus scaling with linear complexity.

When adding subsequent topics, Ambient Google updates each anchor's list of top topics to see where this new topic fits in. For a visualization of m anchors, this process executes m times.

With multiple anchors in the visualization, Ambient Google must decide where to place each topic relative to each anchor. This is accomplished by Ambient Google's spring-model visualization, known as contextual anchor placement.

If a topic is deemed by Ambient Google to be semantically-close to topic A, then it will appear near A's anchor in the visualization. If it is completely unrelated to A, or semantically-distant, it will appear further away from A's anchor. Placement of topics relative to the anchors is calculated through a weighting scheme derived from the topics' LSG scores (Figure 7.5). The distance from an anchor to the center of mass for all anchors represents the line upon which a topic's position is calculated. The system sums the position for all anchors, and then divides by the number of anchors. The resulting co-ordinates determine the final position for the topic. This layout is demonstrated in Figure 7.6 and Figure 7.7. In this scenario, the user initially has two anchors, "syllogism" and "semantic", with the related topics floating between the two anchors in relation to their semantic closeness. The user then adds a third anchor, "web", which has a strong influence over the topic "semantic". This is due to a large number of websites related to a project known as the semantic web, giving the two words a high level of semantic closeness.

Figure 7.5: Topic Placement Algorithm



(x, y) coordinates for Topic T:

$$T(x,y) = \sum_{i=0}^{n} A_i\left(\frac{x}{n} \times score(T, A_i), \frac{y}{n} \times score(T, A_i)\right)$$

All topic scores are rated from a minimum score of 0 (no occurrences in an anchor's Google results) to a maximum of 5 (5 or more occurrences in an anchor's Google results). In the rare occurrence of a very high score, this technique stops the score from overwhelming the positional scaling of the more common low scores. Normalizing the scores[1] was considered, but it would have more than doubled the number of operations required, which would have slowed down the system considerably. It would also not have solved the problem of data skew, which is solved with simple thresholding.

[1]Normalization is calculated by subtracting the minimum score from all scores, and dividing by the range of the data. This resulting scores have a similarly shaped histogram but with values between 0 and 1.

Figure 7.6: Topic Placement With 2 Anchors. Topics are linearly distributed between the 2 anchors.

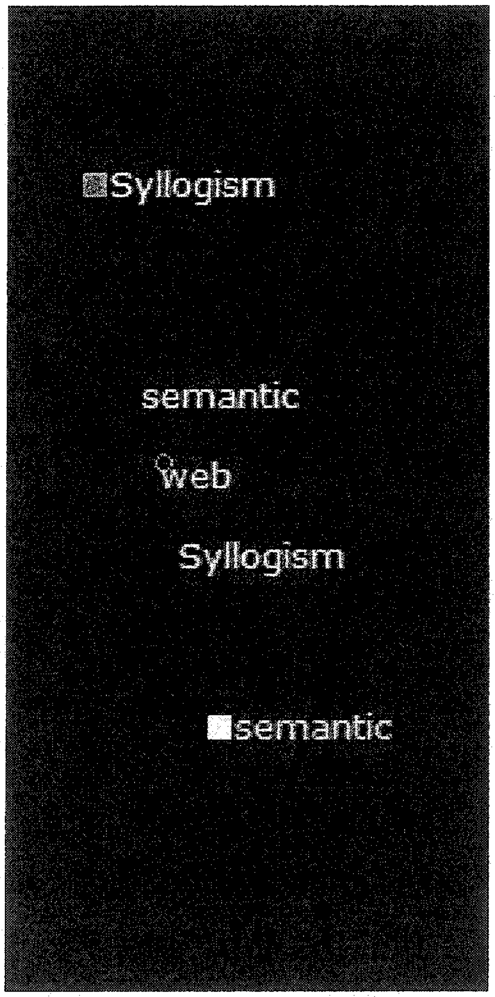
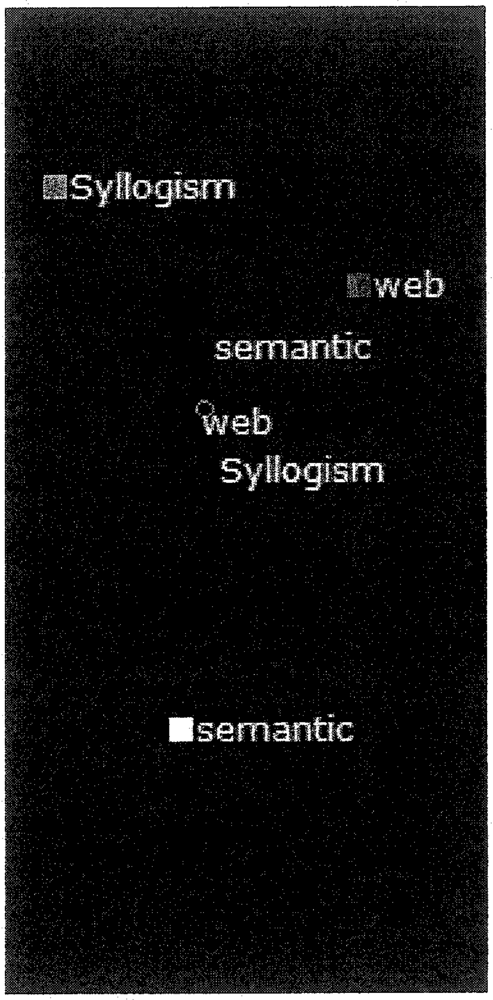


Figure 7.7: Topic Placement With 3 Anchors. The addition of the third anchor has caused the topics to be repositioned in accordance to their relationship with the anchors.



7.4.3 Access to Google Results

Ambient Google further augments conversations by giving the user access to the automated Google query results. Each topic has a subset of Google query results associated with it. To access this Google information, we incorporated a drill-down approach [27]. When the user enters a topic cluster with his mouse cursor, a contextual menu appears that contains the Google summaries of relevant websites. Selection of the summary in the menu causes the corresponding page to be loaded in an external web browser. By continuously providing links in the user's periphery, Ambient Google allows the user to continue his primary task, yet consult an interface seamlessly and without disruption of focus.

7.5 Interaction Scenarios

Augmenting a user's conversations with contextual information may be useful in many situations. We present two possible application scenarios, one illustrating the use of Ambient Google during a meeting, and one during a presentation.

7.5.1 Meetings

User Jeff is having a progress meeting with his thesis advisor, Alex. Previous meetings had resulted in a large number of notes, mostly on scrap paper. After the meetings, Jeff would piece together the notes into a legible transcript of their meeting. Once this was completed, Jeff would sit down behind his web browser and start researching the topics that had been discussed. Instead, today's meeting is augmented with Ambient Google. When Alex arrives for the meeting, he and Jeff put on a wireless headset and begin their discussion. As Alex discusses his ideas for a new direction of research, Ambient Google quietly listens, parsing his speech into a series of keyphrases which are arranged chronologically. When Alex pauses to think of the name of a project two of his colleagues had collaborated on, Jeff glances at the system and notices the names of the colleagues on the screen. Jeff selects the name of one of the researchers, and chooses his personal website as suggested by Google in the contextual menu. The website opens in a new window, quickly uncovering the name of the project. Jeff goes back to the visualization to click on the name of the researcher, causing it to become an anchor in the visualization. After the meeting concludes, Jeff is left with a chronology of the conversation's topics. To further research topics, he simply clicks on them, creating a new anchor. From there, he is free to navigate websites that Google deems relevant to the topic.

7.5.2 Presentations

User David is attending a conference with a wireless network installed in the presentation room. The Ambient Google server is running over this network, allowing it to process the speaker's voice. The topic of the first presentation is exactly within David's area of research. As David listens to the talk, it becomes apparent that the work is extremely relevant to his thesis. David does not want to use Google during the presentation, as this would cause him to miss part of it. Instead, his Ambient Google client automatically searches for websites that are relevant to the words recognized during the speaker's talk. During the break, David clicks on relevant topics that Ambient Google has extracted for him. Creating new anchors in the visualization, David begins to see the true extent of the presenter's work and decides to approach her after the next session. The websites featured in Ambient Google contain all copies of the speaker's papers, allowing David to read up on her work prior to the meeting.

Chapter 8

User Evaluations

User evaluations were performed to understand the effectiveness of Ambient Google. The participants consisted of four males and two females, aged between 24 and 29. All participants rated their computer literacy as above average to high.

8.1 Task

Each participant engaged in a short dialogue with the test coordinator. The participant wore a headset microphone, which was connected to Ambient Google; the coordinator did not wear a microphone. The speech recognition was not specifically trained for each individual user.

To obtain a standardized set of results, the coordinator prompted the participant to discuss a series of five topics:

- baseball
- hockey

- American elections
- Switzerland
- beer

Having pre-defined topics avoided the common problem of participants "freezing up" when asked to speak on a topic of their choosing.

The participants were allowed to interact with Ambient Google during the conversation, or they could wait until afterwards to operate the visualization.

8.2 Questionnaire

Section 1: Speech Recognition After completing the dialogue and Ambient Google interaction, each participant was asked to rate the following 9 statements, using a 5-point Likert scale. Statements between varied positive and negative affirmations, following standard techniques for avoiding bias on this type of questionnaire.

1. The system accurately recognized my speech

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

2. The system produced proper Google queries from speech input

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

3. The system kept up with my conversations

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

Section 2: Visualization

4. Topics related by meaning appeared close to each other

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

5. When I moved topic anchors around, topic groupings responded as I expected

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

6. It was difficult to find the topic clusters

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

Section 3: Interaction

7. The system interfered with my conversations

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

8. It was easy to access suggested websites from the system

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

9. I would use this system for my daily tasks

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

8.3 Statistical Analysis

The null hypothesis for our experiment was that for each statement, test subjects did not have a positive, neutral or negative attitude to the statement. This was rejected for three of the statements (speed of response, meaningfulness of topics clustering, and ease of access to suggested URLs) with $\chi^2(2) > 7, p < .05$, as shown in table 8.1.

Table 8.1: Chi Square

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Chi-Square	4	3	12	7	12	1	1	12	1
df	2	2	2	2	2	2	2	2	2
Asymp. Sig.	.135	.223	.002	.030	.002	.223	.223	.002	.067

Median rank scores were 3 for recognition accuracy; 3.5 for quality of Google queries; 4.5 for speed of response; 4 for meaningfulness of topics clustering; 1.5 for interference with conversations; 4 for ease of access to suggested URLs, as shown in table 8.2.

Table 8.2: Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Statement 1	6	2.6667	.51640	2.00	3.00
Statement 2	6	3.5000	.54772	3.00	4.00
Statement 3	6	4.5000	.54772	4.00	5.00
Statement 4	6	3.8333	.40825	3.00	4.00
Statement 5	6	4.6667	.51640	4.00	5.00
Statement 6	6	2.5000	.54772	2.00	3.00
Statement 7	6	2.0000	1.26491	1.00	4.00
Statement 8	6	4.1667	.40825	4.00	5.00
Statement 9	6	3.1667	.75277	2.00	4.00

8.4 Participant Comments

All participants complained about the speech recognition accuracy, which was expected, given the current state of SR. It was also noted that while the recognition was poor, the system did a good job of ignoring nonsensical words that were placed in the transcribed sentences. Transcription accuracy could be improved with extended training for each user. Due to the buffering of transcribed sentences before they get parsed, the system had no problem keeping up with the users constant speech. For cases where the speech recognition was particularly poor (namely, participant 6) we allowed the user to type in keyphrases manually.

Clustering accuracy was deemed to be appropriate, especially given the poor input. While the clusters moved and responded as the users anticipated, there were complaints regarding the overlap of topic labels in the visualization. This could be overcome by devising a deconfliction algorithm, but possibly at the expense of clustering accuracy.

Accessing related websites was rated to be straightforward, but one user requested that information be shown when brushing over each topic, as opposed to right-clicking to bring up a context menu. All users agreed that Ambient Google in some shape or form would be useful for daily interactions, but was hampered by the speech recognition problems.

8.5 LSG Evaluation

In the future, we plan to also conduct an evaluation of the accuracy of the LSG algorithm. A survey of previous evaluations has shown what would be required for

LSG to undergo a similar evaluation. Van Rijsbergen [34] stated that the following criteria were foremost in assessing the performance of an information retrieval system:

- The coverage of a collection
- The response time between entering a query and receiving a response from the IR system
- The effectiveness of the output display
- The effort required by a user to obtain answers to their search request
- The proportion of relevant material actually retrieved in the system output
- The proportion of retrieved material that is relevant

Conducting an experimental evaluation would give quantitative values with which we compare LSG to other information retrieval techniques.

Chapter 9

Summary and Conclusion

9.1 Summary

The contributions made by this thesis apply knowledge from the domains of HCI & speech input, and linguistics & information retrieval.

9.1.1 Contributions to HCI & Speech Input

The initial concept behind this thesis, and the one most relevant to HCI, is Ambient Google. Ambient Google allows for high-level conversation augmentation by contextually grouping the topics discussed being discussed. The form of ambient information is heavily rooted in the works of Czerwinski and Rhodes, who aim to use Implicit Querying techniques as a passive extension to the human mind. Ambient Google also applies spring-model algorithms to the visualization techniques employed in previous textual conversation augmentation applications [11]. The system employs a natural interaction method, based upon passive speech input. By not relying on command

and control for speech input, the user is freed from being consciously aware of the computer, and instead only needs to call upon it when information is required. This allows the computer to serve as a mediator or third party in a discussion. Most systems employing speech recognition for input require 100% accuracy. Ambient Google does not require such accuracy, because it is based upon an aggregation of keyphrases that are uttered by the user, as opposed to pure dictation.

9.1.2 Contributions to Linguistics & Information Retrieval

Latent Semantic Googling provides a reasoning system for describing the contextual relationships within sets of words. Similar to Latent Semantic Indexing, it uses the co-occurrence of words across many documents to gauge the semantic distance between sets of words. The key difference lies in the corpus that is used to count the occurrences. LSI requires a wide-ranging corpus to create an accurate model of the semantic relationships. This is useful when one wants to understand the relationship between every pair of words in the corpus, but is inefficient when only examining a few pairs.

Latent Semantic Googling populates its corpus as it progresses, only ever adding documents that will contribute to word pairs that the user is interested in. This is done by pairing a word with its Google search results, essentially giving it semantic meaning.

LSG's continually expanding corpus allows for a much smaller search space than that of LSI, drastically reducing the search overhead, and allowing for real-time indexing and querying.

The words in the Google results are considered to be related to the search term, and are later used for evaluating semantic closeness. If a word appears frequently in another word's associated Google results, the two words are considered semantically close. This type of association is what lies at the heart of the Google PageRank algorithm, which is used to drive the most accurate and popular search engine on the Internet. Applying this approach to an information retrieval system is a novel idea, but is rooted in previously evaluated and respected principles.

9.2 Future Work

The concepts and contributions put forth in this thesis may be readily applied to several other domains and applications. The development of Latent Semantic Googling has shown that it is possible to mine an extremely large corpus, such as Google, and harness it to assist in augmenting user interactions.

The augmentation of human intelligence and communication is burgeoning research field. Early in my research, the choice was made to develop a system for augmenting conversational speech. While this proved fruitful, many challenges occurred along the way that would not exist when applying LSG to other input modalities. A system such as Rhodes' Remembrance Agent [26] is a project that takes advantage of the increasing amount of information that is typed directly into a personal computer. Archived emails and documents could assist in the process of identifying relevant topics.

As a standalone speech processing tool, LSG could be used for passive voice input where 100% accuracy isn't required. Conversation monitoring is becoming more prevalent in modern intelligence agencies. Monitoring of terrorist activities is already being done via phone-taps; the 2004 Olympics security team uses a system that can monitor in several different languages, alerting agents to potential terrorist plots [35].

9.3 Conclusion

This thesis presented the concept of *Latent Semantic Googling*, a variant of Latent Semantic Indexing that uses the Google search engine to judge the semantic closeness of sets of words and phrases. This concept is implemented via *Ambient Google*, a system for augmenting conversations through the classification of discussed topics. Ambient Google uses a speech recognition engine to generate Google keyphrase queries directly from conversations. These queries are used to analyze the semantics of the conversation, and infer related topics that have been discussed. Conversations are visualized using a spring-model algorithm representing common topics. This allows users to browse their conversation as a contextual relationship between discussed topics, and augment their discussion through the use of related websites discovered by Google.

We evaluated the use of Ambient Google as a means of augmenting conversation through use of a qualitative user study. This evaluation has demonstrated the potential and need for a system to help users overcome the information overload that is present in a technological society.

Bibliography

[1] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In *Thirteenth Canadian Conference on Artificial Intelligence*, 2000.

[2] R.K. Belew. Inverse document frequency. Internet, September 2000.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, 1998.

[4] J. Budzik, K. Hammond, and L. Birnbaum. Information access in context. *Knowledge Based Systems*, 14(1-2):pp 37–53, 2001.

[5] Andy Cockburn and Harold Thimbleby. Automatic conversational context: Avoiding dependency on user effort in groupware. In M. J. Rees and R. Iannella, editors, *Proceedings of OZCHI'92, Interface Technology: Advancing Human-Computer Communication*, pages pp142–149, 1992.

[6] P.R. Cohen and S.L. Oviatt. The role of voice input for human-machine communication. In *National Academy of Sciences*, 1995.

[7] Mary Czerwinski, Susan Dumais, George Robertson, Susan Dziadosz, Scott Tiernan, and Maarten van Dantzich. Visualizing implicit queries for information management and retrieval. In *Proceedings of CHI '99*, 1999.

[8] Deerwester, Dumais, Landauer, Furnas, and Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[9] Li Deng and Xuedong Huang. Challenges in adopting speech recognition. *Communications of the ACM*, 47(1), 2004.

[10] Alain Désilets, Berry de Bruijn, and Joel Martin. Extracting keyphrases from spoken audio documents. In *SIGIR'01 Workshop on Information Retrieval Techniques for Speech Applications*, 2001.

[11] Judith Donath. Visual who. In *ACM Multimedia '95*, 1995.

[12] Judith Donath. A semantic approach to visualizing online conversations. *Communications of the ACM*, 45(1), 2002.

[13] Factory3x5. Glossary of terms - factory3x5. Web, 2004.

[14] Google. The basics of google search, 2003.

[15] Google. About gmail. Web, May 2004.

[16] Google. Develop your own applications using google. Web, 2004.

[17] Google. Google press center: Fun facts. Web, 2004.

[18] Google. Google press center: Technology overview. Web, 2004.

[19] IBM. Ibm viavoice - product overview - ibm software. Web, 2004.

[20] Tony Jebara, Yuri Ivanov, Ali Rahimi, and Alex Pentland. Tracking conversational context for machine mediation of human discourse. In *AAAI Fall 2000 Symposium - Socially Intelligent Agents - The Human in the Loop*, 2000.

[21] M. Kelly. Developing a search strategy, 2003.

[22] Henry Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal, 1995.

[23] Paul P. Maglio and Christopher S. Campbell. Attentive agents. *Communications of the ACM*, 46(3):47–51, November 2003.

[24] Microsoft. Microsoft speech. Web, 2004.

[25] University of Calgary. Noun phrases. Web, 2004.

[26] Bradley Rhodes and Thad Starner. The remembrance agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*, pages pp. 487–495, London, UK, April 1996.

[27] Steven Roth, Peter Lucas, and others. Visage: A user interface environment for exploring information. In *Information Visualization '96*, 1996.

[28] ScanSoft. Dragon naturallyspeaking 7. Web, 2004.

[29] Ben Schneiderman. The limits of speech recognition. *Communications of the ACM*, 2000.

[30] Abigail Sellen. Speech patterns in video-mediated conversations. In *CHI '92*, 1992.

[31] Jeffrey S. Shell, Ted Selker, and Roel Vertegaal. Interacting with groups of computers. *Communications of the ACM*, 46(3):40–46, November 2003.

[32] Davy Temperley, Daniel Sleator, and John Lafferty. Link grammar. Web, 2003.

[33] Roberto Togneri. How speeech recognition works. Web, 2004.

[34] C. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[35] Leslie Walker. High-tech security's olympic moment. Internet, August 2004.

[36] M. Weiser and J.S. Brown. The coming age of calm technology. Technical report, Xerox PARC, 1996.

[37] Clara Yu, John Cuadrado, Maciej Ceglowski, and J. Scott Payne. Patterns in unstructured data: Discovery, aggregation, and visualization. A Presentation to the Andrew W. Mellon Foundation, 2002.

Stop Words

The following is a list of stop words that were used for filtering semantically unimportant words from speech input:

a about after all an and are as at be been but by can could did do down each find first for from had has have he her him his how I if in into is it its just know like many may more most my no not now of on one only or other out over people said see she so some than that the their them then there these they this time to two up use very was way we were what when where which who will with would you your