Building a Speech Understanding System Using Word Spotting Techniques

by

Theresa K. Burianek

S.B., Massachusetts Institute of Technology, 1999

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

July 2000

Signature of Author .....
Department of Electrical Engineering and Computer Science
July 6, 2000

Certified by .....
Timothy J. Hazen
Research Scientist
Department of Electrical Engineering and Computer Science

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Building an Speech Understanding System Using Word Spotting Techniques

by

Theresa K. Burianek

Submitted to the Department of Electrical Engineering and Computer Science
on July 6, 2000 in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The thesis discusses the development and evaluation of a word spotting understanding system within a spoken language system. A word spotting understanding server was implemented within the GALAXY [4] architecture using the JUPITER [3] weather information domain. Word spotting was implemented using a simple context-free grammar that maps words to key-value classes. Word spotting was evaluated by comparing understanding error rates against those of the TINA [17] natural language understanding component. The results were encouraging, sustaining the hypothesis that word spotting can perform well as an understanding system. Further analysis was done using confidence scoring mechanisms [5] in both full-parse and word spotting understanding. The use of confidence scores improved performance over 25% for both understanding systems. These results, in addition to the ease of use and extensibility of a word spotting system, offer tremendous potential for enabling the rapid implemenation of understanding systems by naive developers.

Keywords: speech understanding, word spotting

Thesis Supervisor: Timothy J. Hazen
Title: Research Scientist

Acknowledgments

First and foremost I would like to thank TJ Hazen, my thesis advisor, who has guided, directed, and supported me from the beginning of my advanced undergraduate project to the completion of my masters thesis. Without his patient help in explaining the SLS systems I would have never made it this far.

I would also like to thank Joe Polifroni, who endlessly helped me setup, understand, manipulate, and debug the evaluation tools and everything else I asked about. To everyone else in the SLS group, thank you for providing a wonderful environment in which I learned so much.

I would also like to thank the friends who supported me throughout my time here at MIT. Especially Bree, Taslim, Kelly, Ken, Jason, Adeline and Sam who provided a friendly ear and smile whenever needed. And I would like to thank the soccer and track teams for keeping me sane by giving me a place to expend energy and frustrations and have an amazing time while doing it. Those memories will last a lifetime.

Lastly, but certainly not least, thanks go to my family, who have supported me in everything I do, from academics to athletics and everything in between. To my parents for instilling the faith I can do whatever I dream and my brothers for pushing me to achieve those dreams and so much more. The support and guidance I have received from my parents, my brothers, and their families has helped me get me where I am today.

This research was supported by DARPA under contract DAAN02-98-K-0003, monitored through U.S. Army Natick Research, Development and Engineering Center; and contract N66001-99-8904, monitored through Naval Command, Control, and Ocean Surveillance Center.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

The area of speech recognition continues to be on the forefront of research fields. Speech-based interfaces are being built into an increased number of systems that people interact with on a daily basis. These include airline information systems, directory assistance, weather information lines, and dictation programs for computers and personal digital assistants.

However, complete spoken language systems are still difficult to build, often requiring knowledge in many areas including signal processing, statistical pattern recognition, and linguistics. In order to reduce the complexity of the recognition and understanding processes, many systems are constrained to a particular domain of interest and vocabulary. Even with these limitations, building such a system can be a large undertaking. One area that poses a particularly difficult challenge is the development of the language understanding component of a spoken language system. Many state-of-the-art understanding components implemented today require complex knowledge

in speech understanding on the doctoral level [21].

To this end, this thesis explores the use of word and phrase spotting as an alternate method for the understanding component of the speech system. Word spotting understanding is analyzed in comparison to full parse understanding. Furthermore, both understanding systems are measured using various levels of recognition confidence scoring.

1.2 Background

This thesis focuses on using word and phrase spotting understanding within GALAXY, a client-server architecture developed by the Spoken Language Systems (SLS) at the MIT Laboratory for Computer Science [4]. The GALAXY architecture supports all functions needed to implement a full conversational system (recognition, understanding, information retrieval, language generation and speech synthesis). The work for this thesis primarily encompasses the understanding component of the GALAXY environment.

The use of word spotting techniques has been an active speech research topic. However this research has varied in where the actual use of these techniques occurs within the spoken language system. The use of word spotting within the recognition stage of a spoken language system is very prevalent in current systems. This line of research has produced many word spotting algorithms for recognition of key words within a specified domain [2, 9]. Many of these word spotting algorithms have been implemented in the context of audio document indexing and retrieval, or directory assistance and call routing algorithms [11, 16]. This type of implementation is often constrained to single word utterances and is completely recognition based. This the-

sis seeks to utilize word spotting within the understanding component of a spoken dialogue system, rather than within the recognizer.

Understanding components are key to conversational systems. This is where computers impart knowledge from a user input. Early language understanding systems relied solely on syntactic models to map the understanding of a word string. These systems are effective for textual input but often fail with spoken material. Issues such as recognition errors (substitutions, insertions and deletions), pauses, false starts, and background noise create problems that break down the syntactic model [21]. In the field today there is a range of approaches to speech recognition and understanding that address these issues. They range from tightly constrained finite state grammars to unconstrained word spotting. Each has their own strengths and weaknesses.

Many commercial applications use finite state grammars to constrain the conversational environment. These grammars are very limited and restrict users to only those word sequences defined in the grammar. Examples of such systems include command based systems used to instruct the computer to perform an action or an interactive system in which the computer asks specific, directed questions designed to constrain the user's response. While systems such as these provide low error rates for in-domain recognition, they are very limiting. Utterances outside the defined sequences perform very poorly, if at all. Dialogue management within these systems must be designed to direct the conversational interactions to stay within the grammar.

Within the Spoken Language Systems group, a mixed-initiative approach is taken. This idea for a conversational system is based on modeling actual human-human interactions, rather than having the computer control the dialogue. This approach offers more freedom to the user than the finite state approach. The only constraint

to the grammar is a fixed vocabulary. Recognition is performed using statistical n-gram models, returning an n-best list of recognized hypotheses. From there TINA, the language understanding component of GALAXY, attempts to parse the hypotheses using a semantically tagged context-free grammar. This approach gives the system the capability to understand utterances with complex sentence structures. However, this does not ensure complete coverage of all spoken sentences. It is difficult to build a grammar which can cover all possible sentence structures which might be spoken by a user.

Word spotting offers even more freedom than the syntactic and semantic structure of a system such as TINA. The user is virtually unconstrained and the grammar is very extensible. The rules are fewer and more simplistic than the hierarchical rules implemented in a full context-free system. The ease of implementation does come at the expense of increased understanding error rate. Word spotting may not perform as well in understanding as a full-parse context-free system or a finite state system. What this thesis hopes to determine is how much understanding degrades when a stand-alone word spotting understanding component is used in a conversational system.

Another interesting aspect to word spotting is analyzing circumstances in which full parse understanding might fail while word spotting is successful at interpreting the user input. One such case might be if there are many spontaneous speech effects (pauses, false starts, sentence fragments). A full-parse system might not be able to discern a parse for the utterance, where word spotting could key in on the only the important (high-content) words or phrases and return a parse. One possible implementation could use both systems in series, i.e., if full-parsing fails, fall back to a word spotting parse.

Word spotting offers many benefits to the implementation of a conversational system. First and foremost is the ease of implementation. As will be discussed in Chapter 3, writing the grammar for a word spotting parser can be done within a few hours. In contrast, writing a full-parse grammar can take many weeks of work by a knowledgeable linguist. Word spotting offers a "quick start", on which a more complex system could later be built. Word spotting also has the ability to focus on high content words in an utterance, ignoring those that are irrelevant to the query.

Also aiding the development of understanding systems is the implementation of confidence scoring in recognition and language understanding. Scores are assigned at either the utterance or word level, giving an indication of how much confidence the system has in its generated hypothesis. This information can then be used to decide whether to accept or reject the utterance. Decisions based on confidence scores aim to improve the conversational interactions of the systems by making more informed decisions on the actions taken by the system.

A large number of speech system implementations already use confidence scoring within the recognition and/or understanding process [1, 20]. By evaluating the results of the recognition process, a system could select a hypothesis it had the most confidence in. Recent work at the AT&T Labs implemented utterance level verification into both the recognition and understanding components of a call-routing application within the telephone network [16]. In these circumstances, understanding error rates were reduced by as much as 23%.

MIT's Spoken Language Systems group has also implemented confidence scoring into its GALAXY architecture. Initial implementations focused on utterance level

scoring within the language understanding module [12]. The goal of implementing confidence scores was to reduce the number of misunderstood sentences processed by the system. Confidence scores could be used to help direct the actions of the system. For example, if the system was not fairly confident in the recognition, it would ask for clarification rather than produce a possibly lengthy incorrect response.

Recent work done in the SLS group has expanded this to word level confidence scoring [7, 8]. Using word level acoustic scoring allows a finer grained analysis of the recognition output. The understanding component can reject individual words in addition to the utterance level rejection previously implemented [5]. All of these improvements represent significant gains towards improving understanding and consequently the user experience.

Given the significance of these gains, it is desirable to evaluate the effect of incorporating confidence metrics into the word spotting understanding component. Word spotting provides an ideal environment to implement word confidence scoring within understanding. With word spotting only focusing on high content words and phrases (as defined in the grammar), confidence scoring can capitalize on this. An utterance may receive a low confidence score even if the recognizer has difficulty on only a few words. With word spotting, if these misrecognized words have low content value, they can be confidently overlooked. The word spotting understanding component will still produce a parse on the high content words (which have high confidence), rather than reject the entire utterance.

Confidence metrics offer another way to improve the performance of word spotting understanding. Because word spotting focuses solely on important words, it is desirable to only accept those words which the recognizer is fairly sure are accu-

rate. This will prevent extraneous noise from being interpreted as key words. All these cases present examples in which confidence scoring could help augment word spotting understanding.

1.3 Overview

The goal of this thesis is to implement and analyze word spotting within the context of the understanding component of a spoken language system. This will be done within the GALAXY architecture of the Spoken Language System group. Word spotting is analyzed against the current full-parse understanding component of the GALAXY architecture. This analysis is done with various levels of confidence scoring within both understanding components.

The background environment of these experiments will be explained in Chapter 2. This includes the GALAXY architecture, the JUPITER system and the TINA understanding component. TINA is a full-parse understanding server within GALAXY. TINA was used as the baseline for comparison with word spotting understanding.

Chapter 3 describes the implementation of the word spotting server. The ideas and benefits of word spotting are presented first. An explanation of the server within the GALAXY architecture is then presented.

There are many ways to evaluate the understanding of a speech understanding systems. The baseline analysis between word spotting and full-parse understanding is explained in Chapter 4.

The use of confidence metrics within understanding systems seeks to improve un-

derstanding within the system. These metrics have been implemented within GALAXY for both full parse understanding (using TINA) and word spotting understanding. Chapter 5 describes and compares the effects of confidence metrics on both understanding systems.

Conclusions and future work are presented at in chapter 6.

Chapter 2

Systems Overview

2.1 GALAXY

The Spoken Language Systems (SLS) group at the MIT Laboratory for Computer Science has developed a spoken language system platform, GALAXY, which incorporates all functions needed to establish a conversational system [4]. These functions include speech recognition, language understanding, information retrieval, language generation, and speech synthesis.

The GALAXY system uses a client-server architecture to manage the functions of the spoken language system. Figure 2.1 shows a typical configuration of the GALAXY environment for a generic spoken language system. The flow of control starts with the audio server which returns a waveform of spoken input. This signal is then sent to the speech recognition server (SUMMIT) which interprets this signal, and returns an n-best list of recognized hypotheses. The understanding server (TINA), along with the discourse component, takes these hypotheses as input and returns a meaning representation for the utterance. After understanding is complete, this representa-

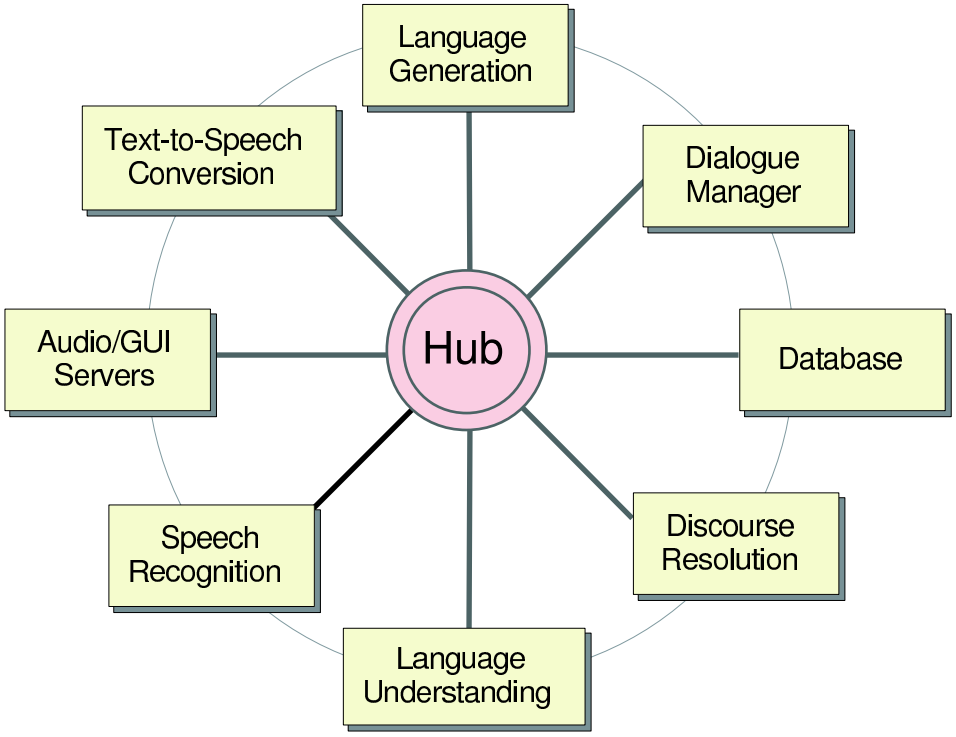

Figure 2.1: A typical configuration of a spoken dialogue system, showing the central hub and the various specialized servers [14].

tion is given to the dialogue management server where the meaning representation is formulated into a query and sent to the database server. Once an answer has been received from the database, the system uses language generation (GENESIS) and speech synthesis to return an answer to the user. This thesis explores a change in the understanding component of the GALAXY system, from full parse understanding to word spotting understanding.

The work for this thesis was done within the GALAXY framework [14]. This framework supports rapid configuration of multiple servers. Within the GALAXY framework a central hub controls all of the system servers through a scripting language. Through hub scripts, configurations and test conditions can be developed that include any or all of the GALAXY servers. This structure and flexibility leads to a "plug-and-play" capability within the system. For example, if a system developer wanted to try a new recognizer with any of SLS's systems, they could remove MIT's recognizer and simply substitute another recognizer. The flexibility of the GALAXY environment was highly useful in isolating the understanding components for evaluation. A word spotting understanding component was easily added to the system, replacing the current understanding module (TINA). This change was analyzed through the GALAXY framework which also allows for evaluation servers to be added to the standard set of speech system servers.

2.2 JUPITER

The system chosen for this analysis of word spotting has been SLS's JUPITER weather information system [22]. JUPITER is a telephone based conversational system available to the public through a toll free phone number. JUPITER can answer queries about the weather in approximately 500 cities worldwide.

The JUPITER system has been chosen for implementation of word spotting because of the minimal contextual requirements of its domain. Queries to JUPITER are very easily represented in a key-value format. For example if the user asked *"What is the weather in Seattle today?"*, the key-value representation would be *"WEATHER: weather, CITY: Seattle, DATE: today"*. In contrast, the VOYAGER system (which handles traffic and city information queries) is not as well suited for strict key value representation. VOYAGER queries are much more hierarchical. With a query such as *"What is the traffic at the interchange of I-90 east and Massachusetts Avenue north?"* there has to be a hierarchy relating the direction east to modify I-90, and the direction north to modify Massachusetts Avenue. These dependencies can easily be represented with hierarchical parsing done within a full-parse, context-free grammar understanding system. However, it would be much more difficult to understand these modifiers using only word spotting techniques.

The current JUPITER recognizer was used for the experiments in this thesis [22]. The JUPITER training set was used to develop and improve the word spotting grammar and evaluation tools. Only training utterances were used when building the grammar for the word spotting server. Evaluation was performed on a unique test set of 2388 JUPITER utterances.

2.3 TINA

2.3.1 Overview

The key understanding engine of the SLS GALAXY platform is the natural language system TINA [17]. TINA was first implemented in the late 1980s to complement SlS's

speech recognition research. This constituted a shift from pure speech recognition towards the implementation of complete conversational systems, where the computer must understand the utterances it sees, not just recognize their words.

TINA is based on understanding through syntactic and semantic analysis of an utterance. This blending of syntax (the structure of the utterance) and semantics (the meaning of the utterance) offers the power of both linguistic constraints and knowledge constraints.

TINA is built from the idea of a semantically tagged context-free grammar (CFG). Using this approach, nodes in a syntactic parse tree are tagged with semantic class information [17, 18]. From the semantically tagged parse tree emerges a meaning representation for a recognized utterance. In addition to the standard CFG, TINA is augmented to handle additional linguistic phenomena [19]. The additional features include a trace mechanism to handle movement and syntactic constraints for verb tense agreement, number agreement (singular, plural) and subject-verb agreement.

2.3.2 Implementation / Limitations

Building an understanding system such as TINA requires extensive knowledge of grammar and language structure. This structure is used to build a parsing scheme for the recognized utterances. In a top-down approach, TINA parses utterances into syntactic categories first (such as "subject" and "predicate".) As the parsing continues, a meaning representation emerges from the application of specific within-domain categories (such as "city" and "weather" for the weather information domain). This meaning is represented in a semantic frame, as shown in Figure 2.2, which is used as input to query the backend database.

```
clause wh_query
   :topic weather
      :quantifier def
      :pred month_date
         :topic date
            :name ‘‘tomorrow’’
      :pred in
         :topic city
            :name ‘‘boston’’
```

Figure 2.2: Example semantic frame for the utterance *"What is the weather going to be like tomorrow in Boston?"*

The grammar and rules for TINA were written by hand for each domain. Even with the limitations of a single domain this is a very time consuming process. In addition it is impossible to completely cover the full range of possible user inputs. The additional syntactic features of TINA adds a layer of complexity requiring the developer to be familiar with extensive linguistic knowledge. This complex linguistic knowledge requirement is one of the things that makes building an understanding system such as TINA so difficult.

Another key limitation of TINA is high confidence needed over the entire utterance. Because TINA is based on a full-parse of the recognized utterance, it needs fairly high recognition confidence over the entire utterance. While this threshold has improved over the development of TINA (including the addition of robust parsing), it is still impossible to correctly anticipate every user input and response. Spontaneous speech effects, such as false starts or partial words, may cause a parse failure within understanding. In addition to variations of user input, outside factors such as poor channel conditions may influence the recognition and understanding confidence of an

utterance. All of these issues may prevent TINA from accepting and understanding the utterance.

The TINA server is a well established understanding system. As such, it provides a baseline with which to evaluate the new implementation of a word spotting understanding system.

Chapter 3

Word Spotting

3.1 Theory

The use of word spotting for recognition and understanding seeks to simplify the grammatical knowledge required for implementing an understanding system. As explained previously, implementing a full-parse grammar such as TINA requires syntactic and semantic knowledge of the language model. Word spotting simply requires a grammar which allows for semantic labeling of words or simple phrases. No complex knowledge of language structure is required.

The idea of word and phrase spotting is to focus on key aspects of the input utterance, and only extract this important information. For example, in the utterance: "*Hello, umm, what is the weather in Boston please,*" the key words are *weather* and *Boston*. These words are very significant for understanding as they are high in content value. The other words are extraneous (low in content value) and not required to understand the request.

The idea of only needing to recognize the important words and phrases in an utterance is heavily used in implementing confidence metrics within word spotting recognition and understanding. With word spotting, high confidence is only needed on the key words and phrases. Thus an utterance which receives a low overall score, may be accepted when word spotting is applied, if there is high confidence on the relevant words and phrases. The implementation of this explained in more detail in Chapter 5.

3.2 Implementation

3.2.1 Building a Word Spotting Grammar

Word spotting analysis of user utterances is done through a finite state transducer (FST). The lexical access search component of the SUMMIT recognizer is implemented using weighted finite state tranducers [10]. The transducer used in this thesis is pre-computed, yielding a single tranducer to be applied to the search space. The transducer, T, is computed from $C \circ P \circ L \circ N \circ U$, where C maps context-dependent acoustic model labels to context-independent phones, P maps phones to phonemes, using context-dependent phonological rules, L is the lexicon mapping from phonemes to words, N is an n-gram statistical models mapping words to words, and U is the grammar which maps words to meaning representation [6].

This model offers flexibility in implementation. For example, the SUMMIT recognizer can implement the entire transducer T, thereby elimination the need for a separate understanding server. Figure 3.1 displays another way in which the FST model can be implemented, where the grammar component, U, can be separated from the rest of the recognition stages of the FST and run as a separate server. This

would allow an alternate recognizer to be easily inserted into the system model. It is this grammar, U, which represents the word spotting model, mapping words to key-value pairs.

$$T = \underbrace{C \circ P \circ L \circ N}_{\Downarrow \text{ Recognition}} \quad \underbrace{\circ U}_{\Downarrow \text{ Understanding}}$$

Figure 3.1: An FST separated between recognition and understanding

The recognizer, using the FST, takes wavefiles of the user utterances and transforms them into n-best lists of recognition hypotheses. With standard recognition the output of the FST is an n-best list of word sequences, as seen in Table 3.1. In the case of word spotting, the FST can create recognized lists with key words transposed into key-value format. This is also seen in Table 3.1. While it is possible to generate key-value n-best lists directly from the recognizer, we chose to generate standard n-best lists from the recognizer and apply the word spotting grammar, U, in a separate understanding server. This allows additional processing, such as confidence scoring, to be applied to the recognized output before understanding is performed.

The word spotting grammar was built from simple word and phrase classification of the JUPITER in-vocabulary words. These words were assigned to key classes based on content. For example, *Boston* is classified as a CITY, *rain* is classified as WEATHER, and words such as *what, is, the,* and *and* are classified as FILLER and do not generate any output. A subset of the grammar can be found in Appendix A. This classification is the basis of creating key-value pairs, which are later used in the understanding and evaluation components of the system. The classifications were

Standard n-best list:

```
what is the weather like in tianjin china
what is the weather like in beijing in china
what is the weather like in japan china
what is the weather like in shanghai in china
weather weather like in tianjin china
```

Word Spotting n-best list:

```
WEATHER: weather CITY: tianjin COUNTRY: china
WEATHER: weather CITY: beijing COUNTRY: china
WEATHER: weather COUNTRY: japan COUNTRY: china
WEATHER: weather CITY: shanghai COUNTRY: china
WEATHER: weather WEATHER: weather CITY: tianjin COUNTRY: china
```

Table 3.1: Example n-best lists from the recognizer. The first is the standard output of the recognizer. The second list is the output of the word spotting recognizer.

further refined as the WORDSPOT server was incorporated in the GALAXY system. This assignment of vocabulary words and basic phrases to classes is the only grammar development needed for a basic word spotter.

3.2.2 Building a Word Spotting Server

Once recognition is complete, control transfers to the understanding component of the spoken language system. This is where the new word spotting server, WORDSPOT, was incorporated into the GALAXY environment. In order to serve as the understanding component of a GALAXY system, WORDSPOT had to process its input (n-best recognized lists) into a form compatible with the GALAXY hub program. As explained in Chapter 2, the hub controls the server configurations. In the context of this thesis, the results of the WORDSPOT server are sent to the evaluation server. In a live system, the results of the understanding component would be relayed to the backend servers,

which handle discourse modeling, dialogue management and information retrieval.

In order to achieve this, the key-value n-best lists had to go through further cleansing within WORDSPOT. The FST word spotting algorithm returns a key-value list based solely on recognition. In order to correctly evaluate this result we had to make some key decisions on the format of the key-values used for evaluation. The additional formatting leads to more accurate analysis, which will be described in Chapter 4.

Key-value pairs are already implemented in the GALAXY architecture, when a TINA semantic frame is "flattened" into key-value pairs. This representation is then used to form the query for the backend database. Therefore, the use of key-value pairs for word spotting easily fits into the GALAXY model of a conversational system.

Because our entire analysis process was purely on the basis of keys, the issue of multiple keys in a hypothesis had to be addressed. How would the understanding component interpret a hypothesis string such as: *"WEATHER: rain WEATHER: crisis: CITY: boston"?* WORDSPOT was designed to just take the first occurrence of a key. This was implemented because most reasonable queries would not have duplicate keys. In analyzing the key-value n-best lists, it was observed that in most cases of duplicate keys, the first occurrence was the most specific. Take, for example, the utterance in Table 3.2. For the WEATHER key, *"crisis flood"* is a more specific value than *"crisis"*.

One exception to this was the DATE key. The DATE key is critical in analysis. The FST computations generated individual keys such as month, day. For example, *"Give me the weather for August 8th"* would be recognized under word spotting

Reference:	do you have any flood warnings in the u s
WORDSPOT Hypothesis:	WEATHER: crisis flood WEATHER: crisis COUNTRY: united states

Table 3.2: Example of an utterance with duplicate keys.

as: *"WEATHER: weather MONTH: august DAY: 8"* However for understanding this was grouped under one heading (DATE) so the word spotting server had to address this date issue. The resolution was to transform all relevant date references to be under the DATE key. For the above example, the resulting key-value string would be *"WEATHER:weather, DATE: august 8"*.

The WORDSPOT server is completely integrated into the GALAXY architecture. It may be made available in future distributions of the GALAXY platform.

Chapter 4

Analysis of Understanding Systems

4.1 Overview

The ultimate goal of this thesis was to see if word spotting understanding could be used in place of full-parse understanding and still achieve similar results. Therefore, for the evaluation of the WORDSPOT server, a comparison was done between the results of the full-parse understanding (TINA) and the results of the WORDSPOT server.

4.2 Evaluation Environment

Evaluation of the word spotting understanding system was done within the GALAXY environment. The GALAXY architecture is designed to easily evaluate individual components of a spoken dialogue system [14]. Multiple servers are managed through the GALAXY hub and its corresponding hub scripts. The analysis is performed by an evaluation server which performs comparisons and accumulates statistics, outputting the results into a log file for each evaluation run.

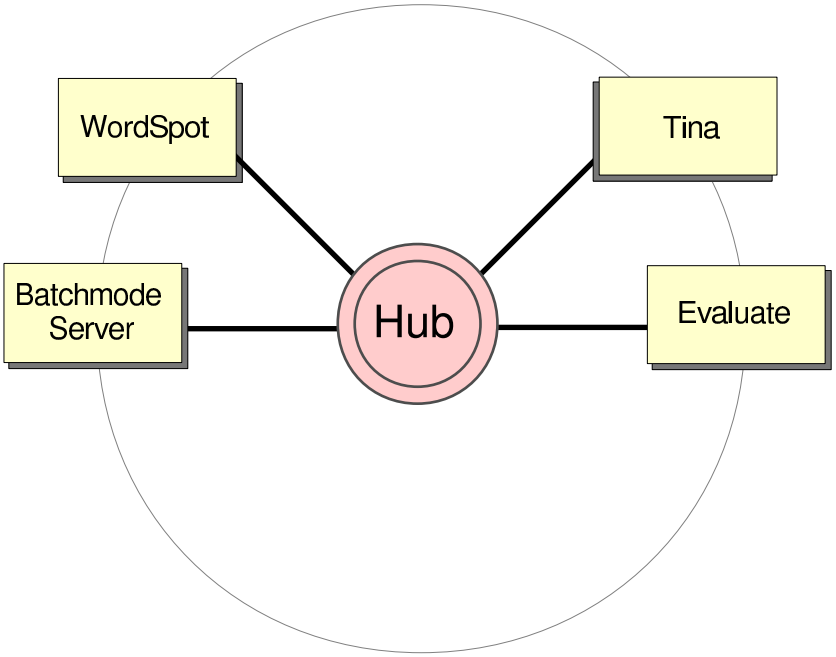

Figure 4.1: The GALAXY configuration for evaluation of the WORDSPOT server.

For this evaluation the servers involved are the HUB (flow control), BATCHMODE (input control), EVALUATE (evaluation tools), NL (TINA), and WORDSPOT (for word spotting analysis only). Figure 4.1 illustrates the hub configuration used throughout the evaluation of the word spotting server. This configuration allowed for individual evaluation of the understanding components.

The batchmode server component controls the input for off-line evaluation of the system. The batchmode server accepts large datasets of input for rapid evaluation of specific server components. It can handle input from any stage of the system, whether it is waveform files to test recognition, n-best lists to test understanding, parse frames for database queries or typed input to test synthesis. In the context of this thesis, the batchmode server was used to process n-best lists, effectively performing the duties of recognizer. As input, the batchmode server took logfiles of data which contained

the orthography (the manual transcription of the utterance) and the n-best recognized hypotheses for thousands of utterances. For each cycle through the system, the batchmode server supplied individual utterance data to the HUB, which then passed this input to the understanding module. This configuration made it easier to test the new word spotting server on large sets of collected data.

After receiving the input from the logfile (as returned by the batchmode server) the HUB passes the orthography to the NL server (which is TINA). There it was parsed, converted into key-value format, and returned the the HUB. Table 4.1 shows an example of a sentence parsed into key-value format. This key-value pair from the orthography serves as a reference (i.e. the correct answer) for the evaluation of the understanding components.

Orthography	Key-Value Pair
What is the weather in boston tomorrow	WEATHER: weather CITY: boston DATE: tomorrow

Table 4.1: Example of key-value pairs used in understanding evaluation.

After the orthography was parsed, the recognized n-best list of utterances was sent to the understanding server (either TINA or WORDSPOT.) The understanding server returned a key-value pairing, based on the recognized utterance. This key-value pairs serves as the hypothesized answer in the evaluation of the system. The evaluation server then used the reference key-value pairs and the hypothesized key-value pairs to produce statistics comparing the two results. Figure 4.2 shows the flow of data through the evaluation system.

One of the advantages of the GALAXY evaluation environment is the ability to

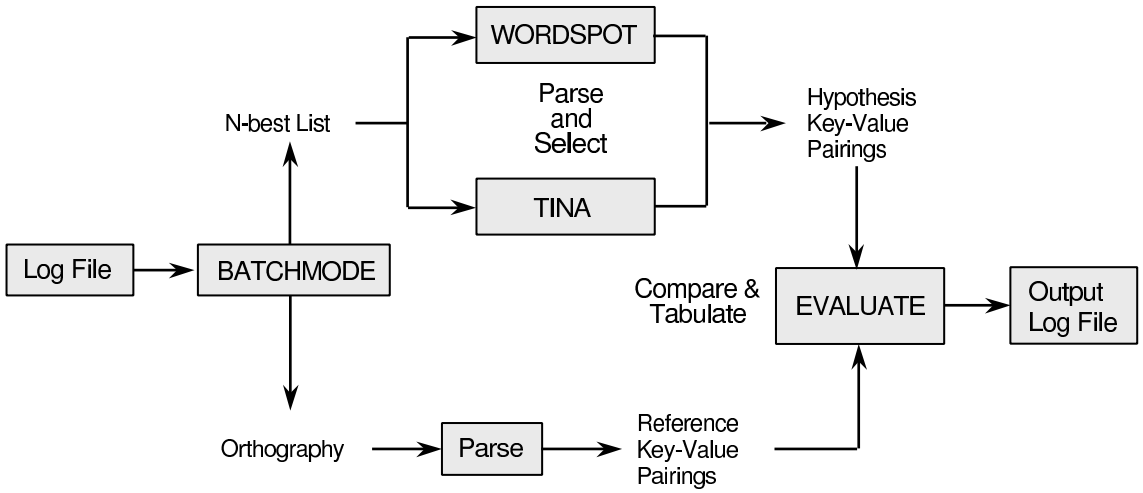

Figure 4.2: The flow of data through the evaluation system.

run individual servers in off-line (batch) mode. There is no difference in the servers in these evaluation runs, to those deployed in a real-time system. This environment allows a comparable environment in which to evaluate individual components of the system.

4.3 Experimental Conditions

4.3.1 Evaluation Metrics

As discussed earlier, the reference of this analysis is generated by parsing the orthography into key-value pairs using the TINA understanding system. This parse is taken to be the correct answer. The evaluation was performed by comparing the correct (or reference) key-value pair against the hypothesis key-value pair. The key-value representation captures the essential information of an utterance. In a live (non-evaluation) system, this information would be passed onto the backend servers to retrieve the requested information.

The comparison of key-value pairs in the reference to those in the hypothesis was used to compute a measure of understanding based on concept error rate [15]. The final measure of concept error rate is a percentage calculated from the total substitutions, insertions, and deletions against the reference key-values of the orthographies. A substitution occurs when both the reference (the orthography) and the hypothesis (from the understanding component) match the key portion of the key-value pair, but the value portion is different. An insertion occurs when the hypothesis has a key-value pair which is not found in the reference. Equivalently, a deletion occurs when the reference contains a key-value pair which is absent from the hypothesis. The concept error rate (CER) is expressed as follows:

$$\text{CER} = \frac{(\#\text{ of substitutions}) + (\#\text{ of insertions}) + (\#\text{ of deletions})}{\#\text{ of reference keys}} * 100\% \quad (4.1)$$

The evaluation server was programmed to gather statistics on the individual key level. Fourteen key classifications were used in the evaluation of the understanding servers. Examples of these classes include city, state, weather type, and date. The individual key statistics also help to analyze specific errors within understanding. For example, one experiment showed WORDSPOT much more likely to insert a city name, while TINA was more likely to delete a city name. The error rate was tabulated on individual keys and then summed to produce a total understanding error rate.

4.3.2 Data Set

The evaluation of the understanding servers was done on a testing set of 2388 utterances from the JUPITER corpus. Using the finite state transducer described in

Chapter 3, this test set was run through off-line recognition modules, returning an n-best list of hypotheses. These n-best lists where then used as input to the evaluation environment presented above. The dataset used in this chapter did not any incorporate confidence scoring mechanisms. No rejection was performed on the hypotheses at either the utterance or word level. Understanding with rejection, through the incorporation of confidence scoring mechanisms, is presented in Chapter 5.

One of the key difficulties with this evaluation was the large number of insertions resulting from the failure of the orthography to parse with TINA. Within the 2388 utterances of the test set, the orthographies of 623 failed to parse. Since the orthography is the reference, if it fails to parse, every hypothesis key-value generated by the recognized utterance was counted as an insertion error. For example, if there are many extraneous words in the utterance, TINA may not be able to comprehend a reasonable sentence structure for the orthography, hence leading to a failed paraphrase attempt. However, the recognized utterance may parse. This occurs with the recognized utterance for both full parse understanding and word spotting understanding.

This was partially remedied by hand-parsing some of the orthographies which failed under TINA. If the orthography could be given a reasonable hand-labeled key-value, this was used as a reference, instead of an empty orthography key-value parse. Of the 638 orthographies which failed to parse, 181 were hand parsed. This helped to prevent artificial inflation of the insertion rate, when a meaning could be garnered from the reference. The were 255 empty orthographies and 187 which could not be hand parsed (they were either incomprehensible, or outside of the JUPITER domain.) This led to 1946 orthographies with a key-value parse.

The parsing of the orthographies led to a total of 3693 key-value "chances", which

would be evaluated against the hypothesis key-value pairings. The WEATHER, CITY and STATE keys had the most chances, representing over two-thirds of the key-value pairs.

Evaluation runs on this data set were performed using both the new WORDSPOT server and the TINA understanding component. Given that TINA is the understanding tool used in most of the systems developed by the Spoken Language Systems group, it was used as a baseline with which to compare the results of the WORDSPOT server. The results of the evaluation runs for both servers, and a comparison of the two, are presented in the following section.

4.4 Results

4.4.1 TINA

When evaluated on a standard set of data, the TINA understanding component has a concept error rate of 27.1%. Over half of these errors (54.5%) are from insertions. This means nearly 1 out of every 7 key-value pairs returned by TINA is an insertion. This propensity for inserting values can be explained by the methodology of the standard TINA algorithms. TINA will accept any hypothesis it is able to reasonably parse, taking no consideration to how likely it is that the hypothesis is correct. This leads to a high number of parses that contained extraneous key-value pairs, as evidenced by the high insertion rate in this experiment.

For keys with a significant amount of data (> 100 chances), TINA had the most trouble with the COUNTRY and CLAUSE keys. They had understanding error rates of 40.3% and 38.6%, respectively. CLAUSE encompasses such items as asking for

help, questions about time and closure statements, e.g. "*good bye*" or "*thank you*". Examination of these errors shows TINA has a tendency to insert closures, and delete help requests.

TINA was most successful with the WEATHER key. Of the 1063 chances, TINA correctly parsed 1014 key-pairs, while only returning 137 key-value pairs which were errors, for a concept error rate of 12.9%.

These evaluations provide insight to areas which TINA is successful and highlight areas which could be improved. Most importantly, in the context of this thesis, they provide a baseline metric against which the performance of word spotting for understanding can be evaluated.

4.4.2 WORDSPOT

Word spotting performed well in the standard experimentation. Initial implementation of the WORDSPOT server returned a concept error rate of 35.2%. As with TINA understanding, over half of the errors (52.4%) were insertion errors. This shows the propensity of the system to accept any recognized word contained in the grammar, even if the word is completely out of context of the domain. One example of this would be the sentence: "*What was the score of the Red Sox game in Boston yesterday?*" The orthography of this would not parse, as it is out of context for the weather domain. However, word spotting would return the parse "*CITY: boston DATE: yesterday*", resulting in two insertion errors for the utterance. This behavior, which leads to a high insertion rate, is one of the trade-offs when implementing a word spotting understanding system.

WORDSPOT was also most successful with the WEATHER key with a concept error rate of 17.7%. This key had a slightly higher percentage of substitutions than the overall system. Substitutions were 17.5% of the errors within the WEATHER key, while only 11% of the total error rate. When the grammar was first developed, there was a much higher substitution rate(over 50% of the errors for the key), which was lowered by creating more weather specific mappings within the grammar. In an example of the substitutions that remained, the hypothesis key-value was "*WEATHER: weather*" while the reference was much more specific, "*WEATHER: crisis*" or "*WEATHER: rain*". This suggests that word spotting is recognizing a weather event, but the grammar still does not differentiate between specific types of weather queries in all cases.

One key advantage to word and phrase spotting is that the grammar is very extensible. This was shown in our improvement of the WEATHER class. Initial grammar implementations grouped any reference to storms, blizzards, tropical storms, and severe weather to the value "storm". As the grammar was refined, these were reclassified in the grammar as "storm", "crisis blizzard", "crisis hurricane", and "crisis thunderstorm", respectively. As stated above, these quick changes in the grammar reduced error rates tremendously.

One key note on the CLAUSE key within word spotting is that deletions outnumber insertions. Within this key, out of 225 total errors, there were 131 deletions (58.2%) and 82 insertions (36.4%). Every other key with significant amounts of data had a higher insertion rate. Closer examination reveals that 77 of the 131 deletions were instances in which the reference parse contained a "help" value. The log files revealed that strings such as "*what else do you know?*" and "*what else can you tell me?*" returned empty strings or incorrect interpretations. The word spotting sys-

tem does not recognize these common phrases for help because there is no word like "*help*" within the utterance. It is performing understanding on individual words and phrases that are in the grammar. Utterances like this do convey a request for help when interpreted as a whole phrase, as done in TINA, but fail under word spotting. It should be noted, however, that the WORDSPOT server can handle the use of phrases within its grammar, so that phrases such as these could be added as needed.

Overall, word spotting performed to expectations. This implementation produces an understanding system with an acceptable understanding rate. Additional analysis of the type of errors can lead to quick improvements across the system, through the extensibility of the grammar. In the next section, word spotting will be contrasted with TINA, the baseline understanding system for these experiments.

4.5 TINA vs WORDSPOT

As expected, TINA performed better than word spotting. Table 4.2 shows the breakdown of the error rates for both systems. TINA achieved a concept error rate of 27.1% while WORDSPOT had a CER of 35.2%, a relative 30% degradation.

Experimental Conditions	Error Rates (%)			
	Sub.	Ins.	Del.	Total
TINA	2.6	13.8	9.0	27.1
WORDSPOT	3.9	18.5	12.8	35.2

Table 4.2: Understanding error rates using two understanding systems.

One note of interest is that the error rates are similarly distributed across error types. Figure 4.3 shows how close the distributions are. Word spotting may cause

more errors than standard TINA understanding, but it is making the same kind of errors as TINA. This shows that word and phrase spotting is performing well, but that the coverage is not as full as a semantically tagged context-free grammar.

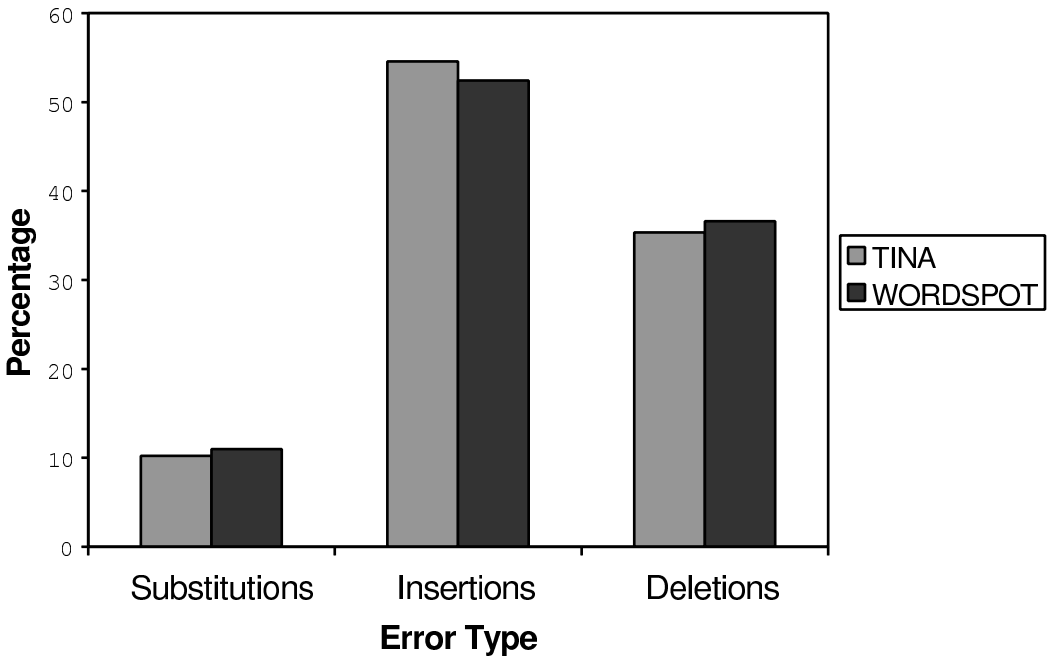

Figure 4.3: Types of errors, as a percentage of total concept error rate.

The evaluation environment allowed a closer inspection of where each system failed, and where each system performed better than the other. For 75% of the utterances, TINA and WORDSPOT had the same number of errors (including utterances with no errors). Of the subset of utterances with the same number of errors, 78% of the utterances had no errors within either TINA or WORDSPOT (which is 58% of the entire dataset). 22% had at least one error and the same number of errors in both systems (16% of the entire dataset). WORDSPOT generated more errors on 18% of

the entire dataset, while TINA had more errors on 7%.

These results help to expose some of the weaknesses of a word spotting understanding system. Of the utterances in which WORDSPOT had more errors than TINA some common themes emerged. The first was in relation to phrases and deletions. As discussed above, phrases such as "*what else do you know?*" and "*what else can you tell me?*" convey a meaning of help, but there is no specific word for word spotting to key in on. Full-parse understanding however, was able to extract a meaning representation.

Word spotting also tended to produce more relative errors when there was no parse for the reference utterance. The recognizer generated a hypothesis, but the reference key-value was an empty string. An empty reference key-value string can be generated two ways, (1) either the transcription failed to parse or (2) the human transcriber could not interpret the utterance, leading to no transcription. When the reference failed to parse, both TINA and WORDSPOT produced insertions errors, but WORDSPOT had many more utterances for which it produced a parsed hypothesis. TINA was more likely to fail in parsing the hypothesis, in agreement with the parsing of the reference string. Two scenarios are likely to describe the behavior of WORDSPOT. First, the non-constraining nature of word spotting is "grasping for straws" and generating extraneous hypotheses, which could result in the user receiving a spurious answer to his request. Or in contrast, word spotting is able to garner useful information from an otherwise unintelligible statement, thus fulfilling the users request without having to prompt for clarification (as would TINA would). With standard word spotting there is no way to discern between these cases, however, as discussed in the next chapter, confidence scoring offers a way to make a more informed decision on cases such as these.

Word spotting actually does better than full-parse understanding if the user is short and to the point. For example, with the reference string "*omaha friday*", TINA understanding preferred the hypothesis "*how about friday*", which is more aligned with grammatical and syntactical conventions, even though the recognizer preferred "*omaha friday*". TINA rescores the recognized hypotheses to give favor to those hypotheses that follow grammatical conventions. However, WORDSPOT was able to correctly parse both "*omaha*" and "*friday*" because it does not rescore the recognition hypotheses as TINA does. This is in contrast with a sentence such as "*Yes, I'd like to find out the weather for Omaha on Friday*". TINA is built to favor this structure, whereas WORDSPOT is more inclined to insert extraneous key-value pairs when recognition errors are present.

Overall, performance was fairly consistent across both systems. Word and phrase spotting had higher error rates than a semantically tagged context free grammar, but that was to be expected. Word spotting is less constraining and therefore will accept more extraneous information within utterances. However, this freedom does offer improvement in certain cases. With no hierarchical language structure, word spotting will understand utterances outside of this model. In utterances with large amounts of background noise or poor signal quality, the word spotter can pick up on the key words, ignoring the extraneous words. It is in many of these cases in which the word spotter has been shown to perform better than full-parse understanding. Word spotting was able to garner meaning from an utterance where full parse understanding fails. There are tradeoffs with both systems, but word spotting offers an acceptable, easy to implement, starting point for an understanding system.

Chapter 5

Incorporating Confidence Scores

5.1 Evaluation Metrics

The experiments presented in this chapter were run in the same environment as previously presented in Chapter 4. The only difference between the experiments is the augmentation of the n-best hypotheses with confidence scores (both utterance level and word level). The understanding of the servers is again analyzed using *concept error rate*. These errors rates are compared to a baseline system with no confidence metrics used. The following sections explain how the understanding servers use the confidence scores and the impact of confidence scoring on the understanding process. Comparison of TINA with confidence scores and WORDSPOT with confidence scores is presented at the end of this chapter.

5.2 TINA

5.2.1 Overview

Confidence scoring has been implemented at two levels within the TINA understanding server. Utterance level scoring has been incorporated within TINA since 1998 [12]. A single confidence score is produced from utterance level features such as acoustic score, purity, and score difference across the n-best hypotheses. Based on this single calculated score, the recognized utterance is either accepted or rejected. Tests on this method showed this implementation correctly rejecting 60% of misunderstood utterances and accepting 98% of correctly understood utterances [12]. In the experiments presented in this chapter, we will show the effects this rejection has on understanding error rates for the system.

Word level confidence scoring has been recently incorporated into TINA. Using only acoustic measures, each hypothesized word in an utterance is augmented with a score as to how much confidence the recognizer has in the individual word. Based on this score and a set confidence threshold, a hypothesized word is classified as rejected or accepted. Testing this method against the actual utterance resulted in a correct classification 90.6 percent of the time [5].

TINA required only a few modifications for the incorporation of word confidence scores into the understanding system. In the n-best lists of hypothesis each word was augmented with its score. The accept/reject classification is then performed on these hypotheses. Any word which is rejected is replaced with a "rejected word" marker. TINA then parses this new hypothesis, which includes the rejected word. TINA ignores the rejected words when flattening the resulting semantic frame into key-value

pairs, except in the case that the rejected word was a parsed as an unknown city name. If the rejected word is an unknown city name it is assigned to a key-value class of REJ_CITY which provides information useful for the dialogue manager.

In addition to improving understanding, this work aims to improve dialogue management. By rejecting at the word level, instead of utterance level, the system gains information that can be used to improve the user interactions. If word level confidence scoring produces a rejection on a city name, the system can prompt for clarification on the city, rather than outright rejection of the entire user input. While this is feature for classifying unknown words is currently limited to unknown city names, there is much potential for the concept to be expand to other word classes. This is made possible by the contextual information provided by the semantically tagged context-free grammar of the TINA understanding system.

5.2.2 Results

Evaluation of the TINA understanding server with confidence scoring was done under three experimental conditions: (1) using utterance rejection, (2) using utterance rejection plus word rejection, and (3) using word rejection by itself. The results of these experiments are presented in Table 5.1. Also included for reference is the results of TINA understanding with no confidence scoring and rejection (as presented in Chapter 4).

Examination of these results provides several key observations. Utterance level rejection reduces understanding error rate by 20% from using no rejection at all. Most prominent is the reduction of the insertion error rate, from 13.8% to 9.2%, a 33% improvement. The deletion rate did increase from 9.0% to 10.4%, but these results

Experimental Conditions	Error Rates (%)			
	Sub.	Ins.	Del.	Total
No rejection	2.6	13.8	9.0	27.1
Utterance rejection	2.2	9.2	10.4	21.7
Utterance + word rejection	1.8	4.1	14.0	19.8
Word rejection	1.8	4.3	13.6	19.7

Table 5.1: Understanding error rates for TINA using different levels of confidence scoring rejection.

are equivalent to removing nearly 5 insertion errors for every deletion error that is added.

Further improvement was seen when word level confidence scores were also used. There was an insertion error rate of 9.2% with utterance rejection. This dropped over 55% to 4.1% when word level rejection was implemented. This demonstrates that word level rejection is achieving its desired effect of removing extraneous words from the utterance. This tighter level of filtering does have its effects though. Deletions rose nearly 35% to 14.0% (from 10.4%) when word level confidence scores were used. These factors, along with the small drop in substitutions, lead to a 8.7% total improvement over utterance level scoring. Overall, word and utterance level rejection combined lead to a reduction of 27% from the baseline TINA implementation, which has no confidence scores.

Also striking is the fact that word rejection alone slightly beats utterance plus word rejection. The rational for this is the fact that many of statements rejected at the utterance level contain words with a low word level confidence score. These words, rejected through utterance level scoring (when the entire utterance was rejected), were also rejected with stand-alone word rejection. Where word rejection gains an edge is

in those utterances that have low utterance confidence, but not all words are rejected at the word level. When only the truly misrecognized words are rejected, some high confidence words are kept, reducing the number of deletion errors, as evidenced by the drop in deletion error percentage, from 14.0% for the system with both rejection mechanisms to 13.6% for word rejection alone.

5.3 WORDSPOT

5.3.1 Overview

Word level confidence scoring within word spotting was implemented using standard word rejection. Using the same methods described above, a standard recognized utterance was augmented with scores for each word. This augmented n-best list was input to the wordspot server. If a word received a score below the threshold it was replaced with a rejected word marker. The new utterance was then used as input to a finite state transducer (which was built using word spotting grammar.) This produces the key-value pairs used in evaluation.

5.3.2 Results

Confidence scoring greatly improved the word spotter. Word error rate dropped from 35% with standard word spotting to 26.5% when word confidence scores were used. This is a relative gain of 25%. The most significant gain with word confidence scores was in the insertion rate. Insertions dropped 66% (from 18.5% to 6.2%). Conversely deletions did increase 38% (from 12.9% to 17.8%.). Table 5.2 summarizes the errors rates for both word spotting and word spotting with word level rejection. The desirability of the 25% overall gain is relative to whether insertions or deletions

are more harmful to the system's response. In the context of the JUPITER system, the question is between possibly asking the user a clarifying question (for deletions) or returning unwanted data (insertions). The overall gain for improving user interactions in the weather information domain tends towards asking clarifying questions and avoiding a possible lengthy incorrect response.

Experimental Conditions	Error Rates (%)			
	Sub.	Ins.	Del.	Total
No rejection	3.9	18.5	12.9	35.2
Word rejection	2.6	6.2	17.8	26.6

Table 5.2: Understanding error rates for WORDSPOT with and without confidence scores.

The reduction of insertions and increase in deletions was most evident within the "CITY" key. Table 5.3 shows the different error rates for this key, for both standard and word spotting with confidence scores. Standard word spotting produced 246 insertions (52% of the errors for the CITY key) and 93 deletions (29% of errors) out of 1011 chances. When confidence scoring was implemented these number flipped, with 56 insertions and 171 deletions. The overall error rate for this key dropped 38% from 38.7% to 24.0%. This behavior is acceptable for the weather information domain because the system can ask for clarification on the city (when the city has been deleted) rather than return unwanted results (for an inserted city).

Experimental Conditions	Error Rates (%)			
	Sub.	Ins.	Del.	Total
No rejection	5.1	24.3	9.2	38.7
Word rejection	1.6	5.5	16.9	24.0

Table 5.3: Understanding error rates for the key "CITY" using WORDSPOT.

5.4 TINA vs. WORDSPOT

The incorporation of confidence scores leads to interesting comparisons between the two understanding systems evaluated in this thesis. Word spotting with confidence scores actually performs better than TINA understanding without confidence scores. Table 5.4 displays the error rates for TINA and WORDSPOT, both with and without the use of word level confidence scores. Examination of this data shows that both understanding systems have the same distribution of errors based on whether confidence metrics are utilized. Standard (no confidence scores) TINA has the same distribution as standard WORDSPOT, and word spotting with confidence scores follows the same trend as TINA with confidence scores.

Experimental Conditions	Error Rates (%)			
	Sub.	Ins.	Del.	Total
WORDSPOT - No rejection	3.9	18.5	12.9	35.2
TINA - No rejection	2.6	13.8	9	27.1
WORDSPOT + Word rejection	2.6	6.2	17.8	26.6
TINA + Word rejection	1.8	4.3	13.6	19.7

Table 5.4: Understanding error rates for TINA and WORDSPOT, with and without word confidence scoring.

In comparing number of errors within an utterance, (with both understanding systems incorporating word confidence scoring and rejection) the relative performance improved. 82% of utterances had the same number of errors. The overall percentage of utterances with no errors in either system improved to 67.1% (as compared to 58% with no confidence scoring). With confidence scoring, word spotting had more errors than TINA in 12.4% of the utterances. The percentage of utterances where TINA had more errors than word spotting dropped to 4.8%.

These results were common across levels of confidence scoring. Table 5.5 shows the percent of utterances which fall under each grouping. WORDSPOT with word rejection was compared against TINA with different rejection criteria. When TINA only incorporated utterance level rejection, the two systems were in less agreement on number of errors. Only 75% of utterance in this test set had the same number of errors. There was virtually no difference between TINA with word rejection alone and TINA with word and utterance rejection.

Experimental Conditions	WORDSPOT > TINA	WORDSPOT = TINA	WORDSPOT < TINA
WORDSPOT + Word rejection compared with TINA + Word rejection	12.4	82.0	4.8
WORDSPOT + Word rejection compared with TINA + Utterance rejection	15.1	75.1	9.8
WORDSPOT + Word rejection compared with TINA + Word & Utterance rejection	12.1	83.3	4.6

Table 5.5: Comparison of number of errors. The first column shows the percent of utterances for which WORDSPOT had more errors than TINA. Column 2 shows the percent with equal number of errors. And the third is the percent when TINA was worse than WORDSPOT.

One specific area that WORDSPOT is less robust than TINA was when an unknown city was asked about. Often the recognizer would have low confidence in this word, leading it to be rejected, and ignored in word spotting understanding. TINA used the contextual information of the sentence to infer that the rejected word was probably a city name (based on sentence structure). TINA then returned the "unknown

city" key, which was then used by the backend server to help guide the user towards a city that JUPITER knows about. This shows one limitation of confidence scores within word spotting. There is less ability to improve dialogue management through confidence scoring. This improvement is easily incorporated into TINA because the contextual information derived from full-parse understanding can predict where key classes belong within a parse. Word spotting does not have any mechanism in which to determine this kind of knowledge.

Confidence scoring offers great potential for improvement of understanding systems, as shown in this chapter. Both word spotting and full parse understanding improved over 25% from systems without confidence scoring. This understanding improvement, along with increased knowledge for dialogue management, represent significant gains towards a better spoken language system.

Chapter 6

Conclusions and Future Work

This thesis has shown that using word spotting within a spoken dialogue system is an acceptable method for a context-free understanding component. Understanding error rates, while worse than that of a full-parse context-free understanding system, are low enough to achieve reasonable performance in a spoken language system. Utilizing confidence scores with word spotting understanding actually results in better performance than a full-parse system which doesn't use confidence scores. These findings offer much support for the use of word spotting within understanding.

In addition to its performance, word spotting offers many benefits. First and foremost is the ease of implementation. A word spotting grammar can be developed in hours, rather than the weeks required for development of a full-parse understanding system. The implementation of a word spotting understanding system also does not require the work of a knowledgable linguist, as needed by a full-parse system.

A word spotting grammar is easily extensible. New words or phrases can simply be added to the current grammar. This ease of use was seen in the course of

this thesis, while improving the classification of *storms* within the WEATHER key. No additional work was needed to handle linguistic and grammatical constraints of adding modifiers (i.e. thunderstorm, hurricane, blizzard) to this class.

In addition to being used a "quick start" for an understanding component, word spotting can also be used as an alternate for a more complex understanding system. If the more advanced system fails to parse, it could fall back and utilize the less-constraining word spotting system.

There are many areas in which this method of understanding can still be refined. The area of confidence scoring offers tremendous potential for improving the system. The confidence scoring mechanisms implemented in this thesis used strict accept/reject thresholds. The possibility of an *uncertain* scoring classification offers more flexibility in understanding decisions. This has potential to help understanding rates by influencing the dialogue system to ask for more clarification on utterances with uncertain recognition.

The thesis explored word spotting in a limited domain, namely the JUPITER weather information system. Future work should encompass other domains in order to explore any limitations. Ideally, one would hope this system would be extensible across domains, but given its non-contextual nature, it is highly possible word spotting might perform significantly worse in domains which requires strict hierarchy.

Finally, a strong test for word spotting within understanding would be to have a naive user implement such a system. Recent work within the SLS group at MIT has resulted in the development of SLS-Lite [13], a web-based system which allows naive users to quickly create a spoken dialogue system, without having to completely

develop the backend servers. Currently SLS-Lite uses a web application to prompt for information to build a finite state understanding system. It is feasible that a word spotting understanding system could be incorporated into this model. Word spotting would offer less constraints on vocabulary and has the potential to be more flexible in recognition and understanding.

The potential use of word spotting as the method for understanding within a spoken language system is extensive. As interfaces to computing system continue to evolve, speech is playing an increasingly important role. More systems are being created by developers with little or no knowledge of the linguistics which are essential to an full-parse understanding system built on syntactic and semantic constraints. Word spotting offers a reasonable starting point for such work.

Appendix A

WORDSPOT Grammar

A.1 Word Classifications

Keys			
domain	clause	current	boolean
close_off	help	temp_type	island
city	unknown_city	state	country
region	direction	month	nth
day_of_week	holiday	time_of_day	digit
teen	ten	clock_time	weather

A.2 Word Spotting Grammar

The following is a subset of the grammar structure. Included are examples of classes from the weather information domain, which was used in this thesis. Words in an utterance are mapped as "*filler*" or "*key*". Each is given a score, 0.2 for "*filler*" and 0.1 for "*key*", so that when comparing hypotheses in an n-best list, the hypothesis with the lowest score will be selected. This insures that key words are preferred over filler words. The keys are defined at the top of the grammar. For each key, there is a mapping of words to key-value pairs, while "*filler*" produces no value. In the example below, within the key definitions, the word or phrase is on the left, and the value it is mapped to on the right in curly braces, {}.

```
grammar jupiter;

public <utterance> = <word>* ;

<word>  = ( /0.2/<filler> | /0.1/<key> ) ;

<key>
        = <domain>
        | <clause>
        | <current>
        | <boolean>
        | <close_off>
        | <help>
        | <temp_type>
        | <island>
        | <city>
        | <unknown_city>
        | <state>
        | <country>
        | <region>
        | <direction>
        | <month>
        | <nth>
        | <day_of_week>
        | <holiday>
        | <time_of_day>
        | <digit>
        | <teen>
        | <ten>
        | <clock_time>
        | <weather>

<clause>
       = what time      { CLAUSE: time }
        | repeat        { CLAUSE: repeat }
        | when will     { CLAUSE: time }
        | when is       { CLAUSE: time }

<close_off>
        = good_bye      { action: close_off }
```

```
        | bye_bye       { action: close_off }
        | bye           { action: close_off }
        | see_you       { action: close_off }
        | see_you_later { action: close_off }
        | thanks        { action: close_off }
        | thank_you     { action: close_off }

<help>
        = cities        { CLAUSE: help city }
        | countries     { CLAUSE: help city }
        | places        { CLAUSE: help city }
        | towns         { CLAUSE: help city }
        | what ( can | do ) i ( do | say )      { CLAUSE: help }

<weather>
        = weather               { WEATHER: weather }
        | conditions            { WEATHER: weather }
        | rain                  { WEATHER: rain }
        | monsoons              { WEATHER: rain }
        | flurries              { WEATHER: snow }
        | snow                  { WEATHER: snow }
        | storm                 { WEATHER: storm }
        | storms                { WEATHER: storm }
        | stormy                { WEATHER: storm }
        | lightning             { WEATHER: crisis thunderstorm }
        | thunder               { WEATHER: crisis thunderstorm }
        | thunderstorm          { WEATHER: crisis thunderstorm }
        | thunderstorms         { WEATHER: storm }
        | sundown               { WEATHER: sun_updown }
        | sun rise              { WEATHER: sun_updown }
        | sun set               { WEATHER: sun_updown }
        | sun come up           { WEATHER: sun_updown }
        | sun come out          { WEATHER: sun_updown }
        | sun going to come up  { WEATHER: sun_updown }
        | sun go down           { WEATHER: sun_updown }
        | sun going to go down  { WEATHER: sun_updown }
        | hurricane advisory    { WEATHER: crisis advisory }
        | marine                { WEATHER: marine }
        | tropical storms       { WEATHER: crisis hurricane }
        | tropical storm        { WEATHER: crisis hurricane }
```

```
        | storm warnings        { WEATHER: crisis blizzard }
        | storm advisories      { WEATHER: crisis blizzard }
        | severe weather        { WEATHER: crisis thunderstorm }

<city>
        = | boston                { CITY: boston }
        | cambridge             { CITY: cambridge }
        | groton                { CITY: groton }
        | washington_d_c        { CITY: washington d c }
        | orlando               { CITY: orlando }
        | new_orleans           { CITY: new orleans }
        | san_antonio           { CITY: san antonio }
        | florence              { CITY: florence }
        | los_angeles           { CITY: los angeles }
        | san_diego             { CITY: san diego }
        | las_vegas             { CITY: las vegas }
        | san_francisco         { CITY: san francisco }
        | eugene                { CITY: eugene }
        | redmond               { CITY: redmond }
        | seattle               { CITY: seattle }

<filler>
        = a
        | able
        | about
        | accent
        .
        .
        .
        | m_i_t
        | made
        | make
        | man
        .
        .
        .
        | "you+ve"
        | you_are
        | your
        | zero
```

Bibliography

[1] M.C Benitez, A. Rubio, P. Garcia, and J. Diaz Werdejo. Word verification using confidence measures in speech recognition. In *Proc. International Conference on Spoken Language Processing*, 1998.

[2] S. Dhanranipragada and S. Roukos. A fast vocabulary independent algorithm for spotting words in speech. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, 1998.

[3] J. Glass, T. Hazen, and L. Hetherington. Real-time telephone-based speech recognition in the JUPITER domain. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Phoenix, AZ, 1999.

[4] D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue. GALAXY: A human-language interface to on-line travel information. In *Proc. International Conference on Spoken Language Processing*, Yokohama, 1994.

[5] T. Hazen, T. Burianek, J. Polifroni, and S. Seneff. Recognition confidence scoring for use in speech understanding systems. In *Proc. International Workshop on Automatic Speech Recognition*, Paris, France, September 2000.

[6] L. Hetherington. Finite state transducers. *MIT Spoken Language Systems Group, Summary of Research*, July 1998 - June 1999.

[7] S. Kamppari. Word and phone level acoustic confidence scoring for speech understanding systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[8] S. Kamppari and T. Hazen. Word and phone level acoustic confidence scoring. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Istanbul, Turkey, 2000.

[9] A. Manos. A study on out-of-vocabulary word modelling for a segment-based keyword spotting system. Master's thesis, Massachusetts Institute of Technology, 1996.

[10] M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Pereira. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, 1998.

[11] K. Ng. *Subword-based Approaches for Spoken Document Retrieval.* PhD thesis, Massachusetts Institute of Technology, 2000.

[12] C. Pao, P. Schmid, and J. Glass. Confidence scoring for speech understanding systems. In *Proc. International Conference on Spoken Language Processing*, 1998.

[13] J. Pearlman. SLS-LITE: Enabling spoken language systems design for non-experts. Master's thesis, Massachusetts Institute of Technology, to be complete August 2000.

[14] J. Polifroni and S. Seneff. GALAXY-II as an architecture for spoken dialogue evaluation. In *Proc. International Conference on Language Resources and Evaluation*, Athens, Greece, 2000.

[15] J. Polifroni, S. Seneff, J. Glass, and T. Hazen. Evaluation methodology for a telephone-based conversational system. In *Proc. International Conference on Language Resources and Evaluation*, Granada, Spain, 1998.

[16] R. Rose, H. Yao, G. Riccardi, and J. Wright. Integration of utterance verification with statistical language modeling and spoken language understanding. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, 1998.

[17] S. Seneff. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86, March 1992.

[18] S. Seneff. The use of linguistic hierarchies in speech understanding. In *Proc. International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

[19] S. Seneff, M. McCandless, and V. Zue. Integrating natural language into the word graph search for simultaneous speech recognition and understanding. In *Proc. Eurospeech '95*, Madrid, Spain, 1995.

[20] A. Setlur, R. Sukkar, and J. Jacob. Correcting recognition error via discriminative utterance verification. In *Proc. International Conference on Spoken Language Processing*, 1996.

[21] V. Zue. Converstational interfaces: Advances and challenges. In *Proc. Eurospeech '97*, Rhodes, Greece, 1997.

[22] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington. JUPITER: a telephone-based converstaional nterface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1), January 2000.