

Understanding Without Formality: augmenting speech recognition to understand informal verbal commands

Lee McCauley

Department of Computer
Science

University of Memphis
374 Dunn Hall

Memphis, TN 38152
+1 901 678 2486

mccauley@memphis.edu

Sidney D'Mello

Department of Computer
Science

University of Memphis
Memphis, TN 38152

+1 901 678 1690

sdmello@memphis.edu

Steve Daily

Department of Computer
Science

University of Memphis
Memphis, TN 38152

+1 901 678 1690

sdaily@memphis.edu

ABSTRACT

Verbal command and control systems are fairly common; almost all off-the-shelf speech recognition packages come with a way to perform various tasks through a voice command. Unfortunately, these systems require that the user utter the commands precisely in the format that it is expecting. These systems have a small number of grammar rules defined that are used to match against incoming utterances. Here, we present a method of using these same grammar rules to expand the capabilities of command and control engines to include semantically similar utterances. Latent Semantic Analysis (LSA) is used to connect specific grammar rules with the meanings underlying matching phrases resulting in utterances being matched to grammar rules even though the exact phrase did not match any specific rule. Experiments are described that determine the extent to which this method can be used and how accurate it is.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces – *Natural language, User-centered design, Voice I/O.*

I.2.7 [Artificial Intelligence]: Natural Language Processing – *Language parsing and understanding, Speech recognition and synthesis, Text analysis.*

General Terms

Reliability, Experimentation, Human Factors.

Keywords

natural language understanding, latent semantic analysis, speech recognition, command-and-control

1. INTRODUCTION

Speech recognition engines typically have two separate modes of operation. The first, called a dictation grammar, relies on

complex statistical models of a language, while the second, called a rule grammar, requires simplified regular expressions that can be matched against incoming utterances. The purpose of both of these grammars is to model the speech patterns of users in a way that narrows the search space thereby increasing the likelihood of accurate speech recognition. Dictation grammars are typically less accurate than rule grammars, but allow for free-form recognition. Rule grammars, on the other hand, are highly accurate given that the user utters a phrase that precisely matches one of the regular expressions in the grammar. Described below is a method for supplementing rule grammars with Latent Semantic Analysis (LSA) in order to remove the need for rigorous phraseology. Ultimately, this enhanced recognition is only useful if the recognized text can be understood by the system. “Understanding” here is used to mean that the system can match an utterance to an existing grammar rule and extract the appropriate information.

This work is being conducted within the framework of an Intelligent Environment (IE). An IE is a space within which a user or users interact with the computer system without the encumbrance of tactile interface devices such as a mouse or keyboard. The primary mode of interface will, therefore, be voice input. In addition to recognizing user commands and questions, this environment is intended to be conversational in nature.

2. BACKGROUND

In the system being described here, the natural language understanding module has one responsibility. That responsibility is to provide an analysis of the user’s utterance in order to determine what action or actions need to be taken to satisfy the user’s request.

When a system is employed in a limited domain and has only a limited number of choices of what to say or do next, it need only classify the user’s utterance according to what it needs to know in order to make that decision. For instance, if a simple travel advisor has just asked, “What airport will you be departing from?” then it might classify the user’s utterance according to airports, cities and other geographic regions. The main natural language understanding (NLU) technology utilized here is based on classification.

Notable among the various classification approaches are those that are based on word co-occurrence patterns such as LSA [8, 12] and HAL [6]. LSA is an attractive approach because it can be

trained on untagged texts and is a simple extension to keyword based techniques.

In general, methods for analyzing a large corpus of text, such as LSA, are described as generating "language models" rather than being applied to the specific task of speech recognition. It should be made clear that we are referring to corpus analysis based on large digital texts; this is quite different from the analysis of audio corpuses as is quite common in building dictation grammars for automatic speech recognition systems. What is being proposed here will not alter the audio models used within a speech recognition engine. Instead, a large corpus of text is analyzed in order to create a semantic representation. Our project attempts to use these semantic representations to categorize incoming utterances as if it were one of the existing grammar rules.

There has been research conducted on the application of this type of corpus analysis to speech recognition, although it is generally assumed that the language models produced would replace or modify those in existing speech recognition engines. Some examples would include Siivola's technique of using a neural network to cluster semantically similar words based on their context [14] and a method created by Gotoh and Renals that automatically generates multiple topic-based language models using a statistical scheme related to singular value decomposition [9]. While the proposed research overlaps with this type of language modeling, it is more akin to information retrieval methods that do text classification. Aside from those already mentioned in the above section, some notable examples would be [3-5, 10, 11].

2.1 Latent Semantic Analysis

LSA has been remarkably successful at a number of natural language tasks. In the arena of query-based document retrieval [7, 8], LSA was compared to a large number of research prototypes and commercial systems. LSA was shown to perform as well as the best other method in some trials and as much as 30% better in others, with an average improvement of 16% over the competitors.

To use LSA, one must first develop an LSA space, which acts as a lexicon. The experiments described below were conducted using a combination of Grolier's Encyclopedia and the TASA corpus. The space represents the "meaning" of a word as a vector in a space of K dimensions (where K is typically 100 to 300). From the corpus, one computes a co-occurrence matrix that specifies the number of times that word W_i occurs in document D_j . A standard statistical method, called singular value decomposition, reduces the large $W \times D$ co-occurrence matrix to K dimensions. This assigns each word a K-dimensional vector in the space.

2.2 Task Domain

The task domain chosen for these tests was that of Microsoft™ Outlook's Calendar program. This was chosen for a number of reasons. First, the domain is narrow enough to have a tractable number of possible action or query types that might be presented. Second, the domain is broad enough to allow for a large number of possible utterances, thereby giving the system a sufficient test of semantic possibilities. A set of fourteen initial grammar rules, was expanded into a set of 14,792 instantiations of those rules even after not elaborating items such as dates. The third reason for this domain choice is the fact that it is a widely used program

that can be easily tested in real-world situations. Finally, Outlook has an easily accessible API.

3. CONNECTING LSA AND NLU

Understanding spoken speech really requires two very different capabilities. The first is the translation of the sound patterns into textual representations. This process is commonly referred to as speech recognition. This research does not delve into the area of audio to text conversion. We have chosen to use the commercially available Dragon Naturally Speaking™ software package as our speech recognition engine mainly due to the level of developer support available. Like most commercially available speech recognition products, Dragon Naturally Speaking™ can translate audio streams to text using two modes, dictation and grammar, discussed previously.

In a way, our system uses both modes simultaneously. The speech recognition engine is set up to use a list of grammar rules as its primary matching scheme; however, if no rule from this list is matched, the engine will still provide text based on the dictation grammar. For this reason, all the grammar rules used as the basis for semantic classification will still function exactly as originally intended. Our system comes into play only when an existing rule is not matched. In these instances, Latent Semantic Analysis [8, 12] is used to extract and match grammar rules to the text provided by the dictation grammar.

Instantiated grammar rules are represented internally as vectors in LSA space. Here, we need to map an arbitrary utterance to a preexisting rule that then states what knowledge needs to be updated and what actions need to be taken, if any. This is the same task as was performed by the rule-based approach described at the beginning of this section, but, with the LSA approach, the phrases spoken do not necessarily have to match precisely as long as their meanings, expressed as vectors in the LSA space, are similar enough. Unfortunately, there is not a one-to-one correspondence between a grammar rule and a vector in the LSA space. A given utterance may match to a combination of rules in the grammar. For example, the utterance "when is my next meeting," might match to a pair of rules such as the following:

(1) $\langle \text{whenQuestion} \rangle = \text{'when is my next } \langle \text{eventType} \rangle \text{'}$

(2) $\langle \text{eventType} \rangle = \text{'meeting' | 'conference' | 'luncheon'}$

Stated loosely, these two rules recognize the words "when is my next" followed by any of the events listed in (2). This very simple combination (not actually used in the research) recognizes three different sentences. LSA would be able to match the utterance against any of the instantiations of the rules, but there is no LSA vector that would match to (1) or (2) individually. Instead, a LSA vector is generated for each possible rule instantiation. Therefore, when an arbitrary utterance is received, its LSA vector is matched against the instantiations of all of the grammar rules currently in context. If a match is found then the system instantiates appropriate data extractors based on the types of variables contained within the matched rule. These extractors are then used to bind values to the necessary variables. These values would be things like a proper name, a date, etc. For the example given above, the $\langle \text{eventType} \rangle$ variable would be bound to "meeting." Since the grammar rule prescribes which variables need to be bound and their type, regular expressions specific to those types can be used to extract the necessary information from the text. A more complete version of the algorithm is provided in Figure 1.

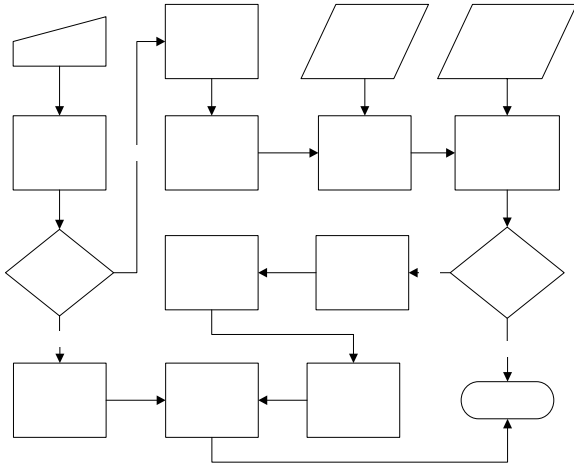


Figure 1. Algorithm flow for understanding spoken commands

The result is that the appropriate grammar rule is “fired” with the correct variables bound just as if the user had uttered the phrase exactly as prescribed in the grammar rule.

4. EXPERIMENTS

Initial testing using LSA has been both encouraging and enlightening. The system has demonstrated its ability to locate semantically appropriate rules even when a given rule phraseology does not match.

4.1 Setup

The general population of test instances is considered to be the set of utterances that, in principle, can be mapped to a grammar rule, but are not direct matches for any rule. In other words, all test instance phrases would have been missed by the speech recognition engine as being a match for any of the grammar rules it had been provided. For a given context, this set can be obtained by expanding every grammar rule (assuming that all feature sets are complete) within that context. However, this would lead to a combinatorial explosion because some features have extremely large feature sets. For example a complete feature set for the feature <first-name> would be every possible first name. In order to compensate for the intractability of the general population, a sampling frame (working population) was obtained for all the grammar rules, but with incomplete feature sets. Expansion of these grammar rules within the Microsoft™ Outlook context, lead to approximately nine million rule instances. A statistically significant sample size for the sampling frame was determined to be 700 by a method specified by [13]. The confidence interval was set at 95% with a margin of error of 3.705%. Each of seven tests attempted to match 100 instances to grammar rules. Each test modified the instances from the target rule in some prescribed way.

The samples were selected by a simplified stratified random sampling strategy [13] in which the population is divided into

three broad categories, namely semantic equivalent, feature, and gibberish instances. A handful of sample test instances, (presented as examples here), are derived from the *create* grammar rule presented below.

```
<create> = <creation-command> a new <event-type> with
           (the | that) <informal-actor-reference> <connector>
           <organizational-entity> at <time> on <month> <date>
```

An example rule instantiation of this grammar rule would be “begin a new meeting with the person from Microsoft at 3:00 on March 3rd.” The elements within the <...> are generally references to lists of alternative strings that could be placed in these positions. These will be referred to as features of a rule instance or variables.

4.1.1 Gibberish

The gibberish test instances are divided into three categories based on the amount of gibberish present in the samples. Gibberish is considered to be any out of context text or random noise. The gibberish test instances were randomly selected snippets of text taken from CNN.com [1]. In the first category, all test samples are completely constituted of out of context text. “defended security efforts amid questions about” is an example of a test instance of complete gibberish. The second set of gibberish instances address the situation where the speech recognition engine has accurately recognized a complete spoken utterance, but has introduced some gibberish due to noise. An example would be “begin a new meeting with the man of Microsoft at 3:00 on March 3rd *on his responsibility country was proud*”. The italicized text is the gibberish appended to the spoken utterance. Finally the third category addresses situations where a part of the spoken utterance has been recognized along with some gibberish. This is a common phenomenon observed in speech recognition. Consider the example, “*on his computer to extend* vice president of Microsoft at 3:00 on March 3rd”, where some gibberish (in italics) has been added before a partially recognized utterance of the <create> grammar rule.

4.1.2 Feature Replacement

The feature instance test samples are a type of controlled semantic equivalence instances in which the utterance replaces one or more features in the rule instantiation. These samples are also divided into three sub-categories in the replacement feature selected. In the first category, the replacement is chosen from a small, but frequently used set of synonyms (taken from [2]) of the replaced feature. An example would be, “begin a new *conference* with the person of Microsoft at 3:00 on March 3rd”. (In the grammar rule instantiation *meeting* was bound to the <event-type> feature and has now been replaced by *conference*). In the second category, the replacement is chosen from a larger set of synonyms. An example would be replacing the original binding of the <event-type> feature *meeting* with *parley*. Finally, the third category consists of test instances where a binding is replaced with a highly specific instance of itself. For example, *shareholder* would replace *person* as the binding of the <informal-actor-reference> feature in the <create> grammar rule, because a *shareholder* is a very specific type of person.

LSA Module (created from corpus)

NLU module calculates variables for modification

Original text matched rule passed to extractor

NLU module passes matched rule and extracted variable bindings

Audio Input

Speech Recognition

Input matches directly to a grammar rule?

Yes

SR engine passes matched rule and extracted variable bindings

SR engine passes recognized text (from dictation grammar)

Feature Extractors remove known semantically neutral elements

No

Variable bindings are extracted

Associated action is performed

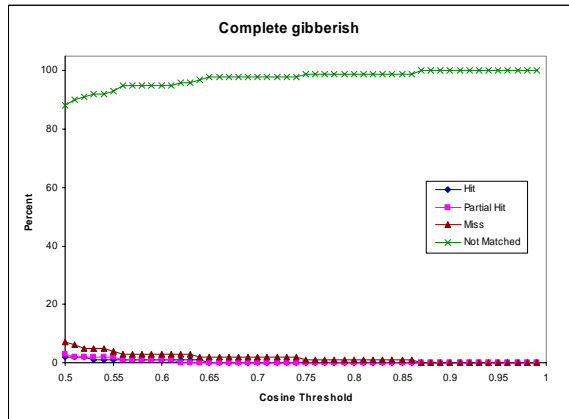


Figure 2: Results of experiments using complete gibberish instances. The ideal result would show 0 hits, 0 misses, and 100 unknown.

4.1.3 Semantic Instances

The semantic equivalence test instances address situations where the spoken utterance means the same as a rule instantiation, but is specified with significantly different wording. The test instances for this experiment were manually created by human volunteers. Each volunteer was given the text for several rule instantiations and was asked to rewrite each instance in a way that preserved its meaning while being as structurally dissimilar as possible. An appropriate example based on the *create* rule specified above, would be, “let’s set up a meeting to meet the gentleman from Microsoft at 3:00 am on the 3rd of March”.

An LSA vector for each test instance was obtained and its cosine similarity to the vectors for all rule instantiations within context was obtained. The rule instantiation with the highest cosine similarity to the spoken utterance is considered to be the *best match* for the utterance, and the rule instantiation is said to have *fired*. The cosine between the fired rule and the spoken utterance is the confidence of the match. A false positive occurs when an incorrect rule has been fired, sometimes with an alarmingly high confidence level. For example, “The Attaché coming from Cuba needs to meet with me at 11:00 am on March the 16th” matching to the <where-event> rule instead of the <create> rule. In order to prevent false positives, a rejection threshold is introduced so that if the confidence of the best match is under the rejection threshold, the fired rule is rejected and the spoken utterance is considered to be unclassified. Selecting an appropriate threshold for rejection is a non-trivial problem because it involves minimizing the frequency of false positives while maximizing the number of true positives. Therefore, this study experiments with a rejection cosine thresholds ranging from 0.5 to 1.0 with increments of 0.01. All 700 test instances were run in the search space and the cosine value of the best match along with the *fired rule* was recorded for each test instance.

4.2 Results

First off, it should be reiterated that the system described here is a supplement to the standard rule grammar systems already in use for command and control. For this reason, all of the phrases that the command and control grammars are designed to recognize

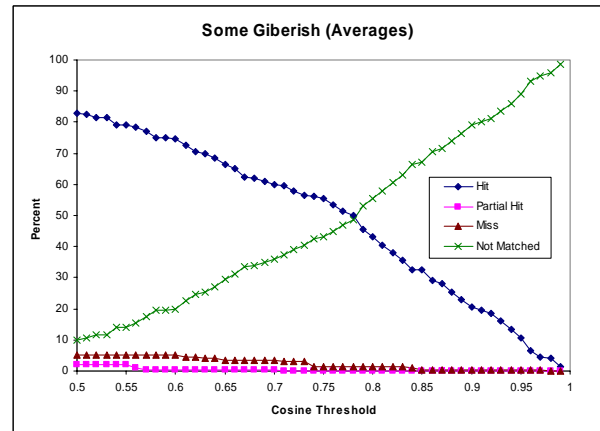


Figure 3: Results of experiments using instances with partial gibberish. Half of the instances had random statements added to a complete command while the other half replaced part of the command with a random statement.

will still be recognized as usual. The results of the system will, therefore, be providing a level of increased recognition that would not have been made by the original rule grammar alone. For this reason, it does not make sense to test exact matches. All of the test instances being used for these experiments *would not have been matched by the grammar alone*.

For any given instance of a phrase, the results of a matching attempt could be categorized in four ways: (1) a *hit* is when the phrase is exactly matched to the intended command, (2) a *miss* occurs when the phrase is matched to a command that is unrelated to the intended command, (3) a *partial hit* happens when a phrase is matched to a command that is not exactly the intended command but would result in the same action as the intended command along with necessary information, and (4) a phrase might *not be matched* to any command and would, therefore, not result in an action. All of the graphs presented here show the percentage of instances (y-axis) categorized into these four areas when a particular cosine threshold (x-axis) is used to make the distinction. As the cosine threshold increases, the requirements for a match (both correct and incorrect) become more stringent and the likelihood that the phrase will not be matched increases.

4.2.1 Gibberish

In our first experiments we needed to know how robust the system was at handling gibberish input. In other words, if the utterance was completely unrelated to the subjects addressed by the command and control grammar, how would the system categorize these? The gibberish used where random phrases extracted from a corpus of news articles [1]. For example, the following phrase was used as gibberish text: “to resign by the end of the year and urged Disney to split.” The graph in Figure 2 shows the results over 100 test instances at different cosine thresholds. For our complete gibberish instances there is no intended command that is supposed to match. Therefore, a perfect result would be one in which there were zero hits, zero misses, zero partial hits, and 100 percent unmatched. The system actually reaches this ideal at a cosine value of 0.87. Even at the

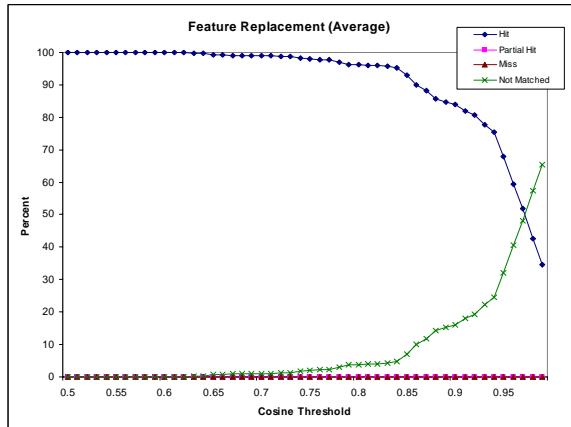


Figure 4: Results showing one or more features replaced by synonyms not present in the original grammar rules.

lowest cosine value of 0.5, the system has only 7% false positives and reaches 2% false positives at a cosine value of 0.64.

Our second experiment also involves some gibberish, but in this case only some of the incoming test phrase is gibberish, the rest is a valid command. For example, the following phrase was used as a test case, “begin a new meeting with the man of at on serious responsibility country was proud.” Current speech recognition systems will still occasionally misrecognize an individual’s utterance. Unlike when a person uses a speech recognition system to dictate text, the Intelligent Environment does not have a way for the user to correct the speech recognizer’s output. This partial gibberish test is intended to simulate when the speech recognition engine incorrectly translates a portion of the user’s utterance. Often the erroneous section of the output is completely unrelated to the current context of the statement even though the words themselves are valid. Figure 3 displays the averaged results of two different types of these tests. Half of the instances had random statements added to a complete command while the other half replaced part of the command with a random statement. The interesting line on this graph is the one representing false positives (misses). Even at the most liberal cosine value of 0.5, only 5% of the test instances were incorrectly matched. This means that even when the speech recognizer incorrectly translates a portion of the user’s utterance, this methodology is still likely to correctly ascertain the user’s intended purpose. This does not mean that all of the information needed to successfully complete the user’s request will be available. Under such circumstances, the system will specifically request that the user supply the missing data.

4.2.2 Feature Replacement

The next three experiments attempted to measure the system’s ability to correctly match an incoming utterance with a grammar rule that has a very similar meaning. As described above, these experiments replaced one or more features of the original grammar rule with synonym words not included in the original rule. All of these feature replacement tests produced very similar results with false positives accounting for no more than 1% of the test cases. Figure 4 is a graph showing the average for all three types of feature replacement experiments. Again, these tests

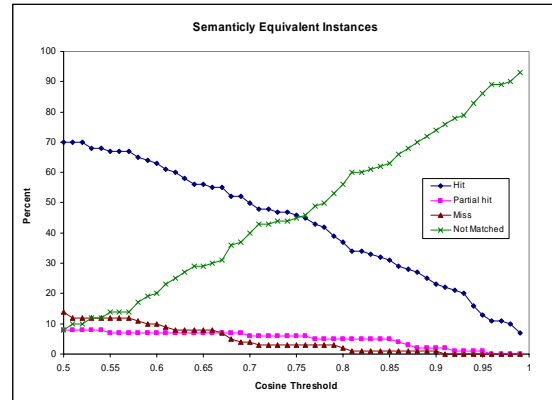


Figure 5: Experimental results in which rules were matched against semantically equivalent instances.

show only a 1% error rate with synonym replacements and a very high hit rate. These experiments give us confidence that small deviations from the provided grammar rule will nearly always be correctly identified.

4.2.3 Semantic Instances

Finally, the results for the experiment testing what we have called semantically equivalent instances were significantly different. These test cases were made up of phrases that mean the same thing (based on human interpretation) as a provided grammar rule, but do not necessarily share any structural similarity. For example, “What time is my date with that person with Dell computers?” had a 0.74 cosine with the rule instance, “when is my talk with that person with <organization>?” Figure 5 shows the results for the semantic equivalence tests. Just as we have seen previously, as the cosine threshold becomes more stringent, the unmatched instances increase while the positive instances (those correctly matched to a grammar rule) decrease. While, the number of false positives increases as expected, they remain below 12% for all cosine values above 0.5. The number of partial hits also goes up, but only reaches 8% even at the lowest threshold.

In initial tests (not shown here) we got a much higher error rate at the lower cosine levels (~40% at 0.5). Two things brought the error rate down to the level show in Figure 5. First, the process (Figure 1) was modified to remove information elements that were hypothesized to be semantically neutral or negative from the perspective of LSA. Second, during those early tests, partial hits were being incorrectly labeled as incorrect matches. For instance, some of the grammar rules being tested here involve follow-ups to previous commands. In other words, the user should have issued a command such as, “create a new meeting,” prior to providing information that would be matched to a follow-up rule. In these circumstances, the follow-up rule would not have been in context and, therefore, not considered for a match. For example, the test instance, “The representative of Taiwan will be meeting us on March the 16th at 10:00am,” is meant to instigate the creation of a new meeting. It was incorrectly matched to a follow-up rule of the form, “the meeting will be with the <position> of <organization> at <time> on <date>.” Semantically the incoming utterance about the representative of Taiwan is very similar to the follow-up rule. However, if the follow-up rule had not been

available as a possible match, then it might have been correctly matched to an appropriate creation command.

5. CONCLUSION

In an effort to free users from the constraints of the desktop, intelligent environments are being created that allow for device-free interaction. A key piece to this endeavor will be the accurate understanding of natural human language. Described above is a method for the enhancement of speech recognition and natural language understanding through the coupling of highly accurate rule-based grammars with a powerful classification technique. The results presented here, although not performed on “live” speech, demonstrate a clear benefit over rule-based grammars. Since none of the test instances could have been recognized by a rule-based approach, any command correctly categorized by this new approach can only serve to increase the accuracy of the overall system. The trade-off of possible miscategorizations seems reasonable although testing under real-life conditions needs to be conducted. The technique presented here in conjunction with standard rule-based methods should allow for accurate recognition and understanding without the constraints of strict phraseology. Future experiments will test the system under “real-world” conditions in order to gauge its full usefulness. In particular, we want to know what percentage of utterances is likely to match the original grammar rules as opposed to being semantic equivalents. We also would like to find out what percentage are going to be simple synonym replacements and how many will be complex restatements. Experiments with human subjects are currently underway.

Finally, the system was not tested as a whole unit. Instead, the NLU module was tested without the inclusion of variable bindings or text actually coming from the speech recognition engine. Future experiments will include these elements. The results presented here show that this method of supplementing the speech recognition system to aid in natural language understanding is worth additional testing and can increase the usability of verbal command and control software significantly.

6. ACKNOWLEDGEMENTS

We would like to acknowledge the following contributors to this work: Phanni Penumatsu and Andrew Olney provided essential assistance in collecting data and configuring LSA. Scott Dodson also volunteered his time in coding needed modules.

7. REFERENCES

- [1] CNN online news, <http://www.cnn.com>, 2004.
- [2] Resource.com, <http://thesaurus.reference.com>, 2004.
- [3] Baker, L.D. and McCallum, A.K., Distributional clustering of words for text classification. in *SIGIR'98*, (Melbourne, Australia, 1998), 96-103.
- [4] Bellegarda, J.R., Butzberger, J.W., Chow, Y.L., Coccaro, N. and Naik, D., Automatic Discovery of Word Classes Through Latent Semantic Analysis. in *EUSIPCO-96 Signal Processing VIII, Theories and Applications*, (1996), Edizioni Lint Trieste.
- [5] Bellegarda, J.R., Butzberger, J.W., Chow, Y.L., Coccaro, N. and Naik, D., A Novel Word Clustering Algorithm Based on Latent Semantic Analysis. in *ICASSP-96*, (1996), 172-175.
- [6] Burgess, C., Livesay, K. and Lund, K. Explorations in Context Space: Words, Sentences, Discourse. *Discourse Processes*, 25. 211-257.
- [7] Deerwester, S., Dumais, S.T., Fumas, G.W., Landaur, T.K. and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41. 391-407.
- [8] Dumais, S.T. Latent semantic indexing (LSI) and TREC-2. in Harman, D. ed. *National Institute of Standards and Technology Text Retrieval Conference*, NIST, 1994.
- [9] Gotoh, Y. and Renals, S. Topic-based mixture language modeling. in *Natural Language Engineering* 6, 2000.
- [10] Khudanpur, S. and Wu, J., A maximum entropy language model integrating n-grams and topic dependencies for conversational speech recognition. in *ICASSP-99*, (Phoenix, AZ, 1999), 553-556.
- [11] Kuhn, R. and De Mori, R. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (6). 570-583.
- [12] Landaur, T.K. and Dumais, S.T. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104. 211-240.
- [13] Rea, L. *Designing and Conducting Survey Research*. Jossey-Bass, 1997.
- [14] Siivola, V., Language modeling based on neural clustering of words. in *IDIAP-Com 02*, (Martigny, Switzerland, 2000).