LOGIC AND STATISTICAL MODELING FOR LANGUAGE UNDERSTANDING

Bradley Cope and Stephen Boemler
NAVAIR Training Systems Division
Orlando Florida

ABSTRACT

Team leader training applications often include critical decision making based on multiple streams of communications where an overload, or interference from secondary sources, can occur. Artificial intelligence based parsing of language could aid in the presentation of the information contained in communications. Semantic and pragmatic context of a group of messages can be restricted in terms of specific domain information relating to events. Further, rule based software written in programmable logic can model the rules of the domain phraseology.

Computational linguistics makes use of syntactic parsing whereby a stream of symbols is analyzed for its conformance to some specific set of rules. For instance, if you read the previous sentence, you "syntactically parsed" it according to some grammar of "English" and determined that it was grammatical. In the simplest terms, parsers analyze some target string of symbols according to some specified grammar.

In a probabilistic associative chain, the occurrence of each word is determined by the immediately preceding word or series of words, where each response serves as a stimulus for the next response. In bounded phraseologies such as those used in training devices, it is possible, using programmable logic, to model grammars. Similarly, the association of words in very large grammars can be modeled statistically. Two innovative computational linguistic approaches were investigated and presented in this paper, the cross entropy syntactic identifier and the Markov model syntactic parser. The cross entropy syntactic identifier identifies phonetic similarity and the Markov model syntactic parser adds variability to otherwise rigid parsing techniques.

AUTHOR S BIOGRAPHIES

Brad Cope is a Supervisory Engineer with the Simulation & Models Division of the Naval Air Warfare Center Training Systems Division. He is currently responsible for exploratory research in speech recognition and language understanding for Navy training. He holds a Bachelors of Science in Electrical Engineering from Temple University (5/83) and a Masters in Engineering Science from Penn State University (5/91). Brad was previously responsible for applied research in advanced flight control and electromagnetic effects at the Naval Air Development Center in Warminster, Pennsylvania.

Steve Boemler is an Electronics Engineer with the speech recognition research laboratory Simulation & Models Division, Research and Engineering Competency/Training Systems Department at NAWCTSD. He holds a Bachelors of Science in Electrical Engineering from the University of Florida (8/75) and a Master of Science in Electrical Engineering from the University of South Florida (5/95). Steve s previous experience was in the science of electrical/electronic measurement at the Naval Metrology Standards Laboratory in Pensacola, Florida.

LOGIC AND STATISTICAL MODELING FOR LANGUAGE UNDERSTANDING

Bradley Cope and Stephen Boemler
NAVAIR Training Systems Division
Orlando Florida

INTRODUCTION

Language is a system of rules that behavior follows, where the language is a set of symbols that represent words. Language is not continuous, rather it is discrete where in-between sounds are interpreted one way or another. The structure of language is divided into five levels: phonology (sounds), morphology (word formation), syntax (sentence structure), semantics (meaning), and pragmatics (context). Semantics or meaning, is where the assignment of meaning is relative to the situation and there may be multiple meanings for the same statement thus ambiguity. Pragmatics is the human ability to disambiguate. Pragmatics go beyond the literal meanings of the words such as in the string of words: Can you close the door? where it is probably implied that you should go close the door and not a question as to whether or not you are capable of closing the door [1].

The human mind is proficient at identifying grammatical correctness and relations in a sentence. Similarly, the mind is capable of word sense disambiguation. Underlying knowledge and language structures can be represented by overt speech behavior where a descriptive grammar can describe the knowledge people must have in order to speak and understand language. We have a built-in understanding of a coherent ordering of words, mostly represented by the subject-verb-object (SVO) relationship and can automatically identify meaning based on this coherence of order. This is true even where our syntactical rules allow many ways to say the same thing. Similarly, a transformational grammar allows us to go beneath the surface of structured sentences and transform the apparent structures into deeper structures, which reveal underlying meaning. Human structured and transformational grammars are a measure of competence as revealed in spoken utterances [1].

In a probabilistic left-to-right associative chain, the occurrence of each word is determined by the immediately preceding word or series of words, where each response serves as a stimulus for the next response. Depending on what was said, a given word can be followed by a variety of words. Usually, sentences are not remembered as a string of words. Syntactic structure and meaning are key to remembering what has been said. This is manifested in the fact that it is easy to paraphrase, but not easy to remember entire strings of words and the more complicated the syntax, the more difficult it is to remember. In bounded phraseologies such as those used in training devices, it is possible, using programmable logic, to model the structured and transformational grammars. Similarly, the left-to-right association of words (lexicon) can be modeled statistically [2]. A goal of this research is to separate dissimilar phraseologies for the purpose of focusing attention on a single stream of military communication, which is analogous to focusing on one person speaking at a cocktail party.

BACKGROUND

Computational linguistics makes use of syntactic parsing whereby a stream of symbols is analyzed for its conformance to some specific set of rules. For instance, if you read the previous sentence, you "syntactically parsed" it according to some grammar of "English" and determined that it was grammatical. Symbols can be analyzed from left to right or right to left to make individual phrases. Mathematically based computational linguistic methods are feasible due to the available computational speed of the personal computer. In the simplest terms, parsers analyze some target string of symbols according to some specified grammar. Insertions and substitutions are due to the statistical property of the current speech recognition technology. The following syntactic parsers were investigated for their appropriateness and utility for resolving insertions and substitutions in statistical pattern recognition systems. The cross entropy syntactic identifier was evaluated for its ability to match strings of phonetic symbols

Bottom-Up Syntactic Parser

The parse begins at the word level and uses the grammar rules to build higher-level structures (’bottom-up’),which are combined until a goal state is reached or until all the applicable grammar rules have been exhausted [3][4][5]. Bottom-up parsers are the typical approach. Bottom-up parsers for typical military phraseologies were prepared and used as the baseline or control model and were subsequently evaluated for accuracy.

Markov Model Syntactic Parser

The Markov model is typically used for pattern recognition [6], but a Markov syntactic parser appears to be innovative and unique approach to get the gist of spoken language or any stream of organized symbols. The bottom-up parser typically expects some precise sequence of symbols. When the sequence is within the scope of the parser, a result is presented. The concept of the Markov parser introduces variability in the allowable response and can tolerate random insertions and occasional substitutions. The trade-off is the possible presentation of incorrect results. Here is a diagram showing a Markov model based parser for an unclassified military phraseology:

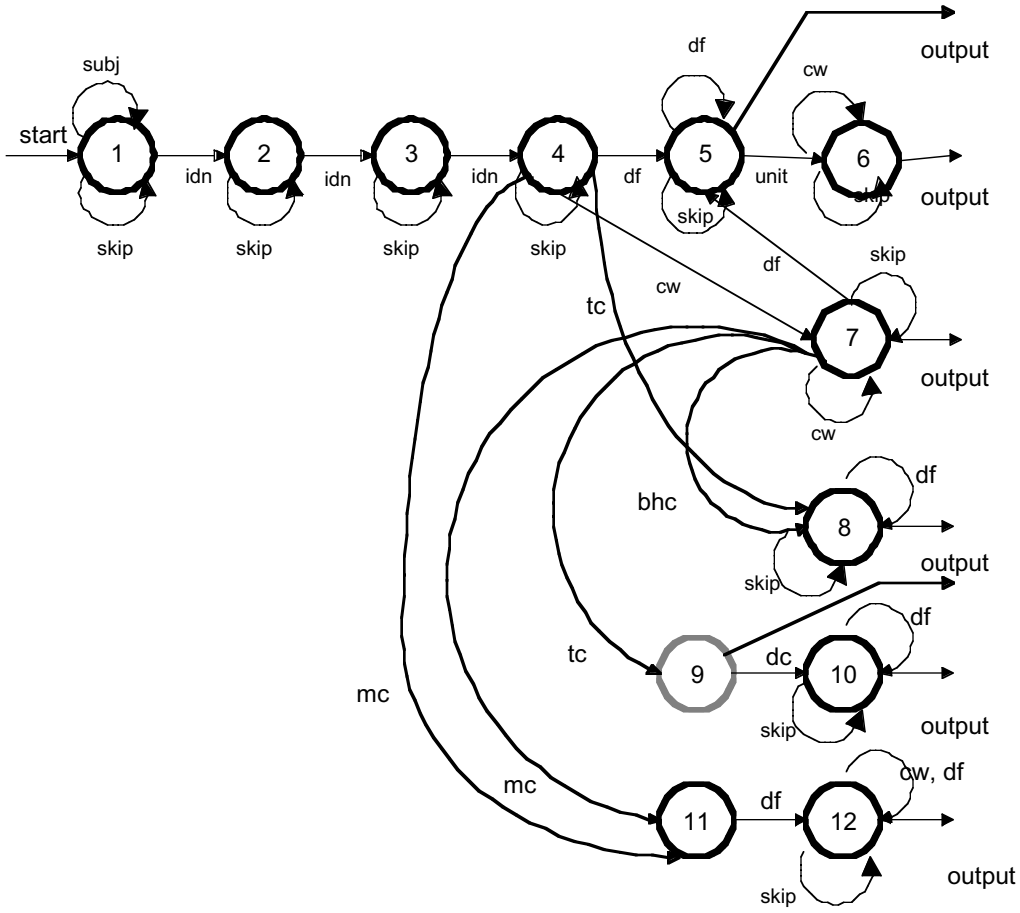

Figure 1. An example Markov Parser Topology

where the following symbols were used:
cw- command word
bhc — bearing and heading command
dc- direction command
df- digits
idn- call sign numbers
mc- mode command
tc — turn command

Clearly, by virtue of the relationships shown in Figure 1, the Markov syntactic parser allows variability in the possible symbols emanated from a source thus mitigating the rigidity seen in the typical bottom-up parser. The phraseology shown in the figure is for air traffic control. An example command that demonstrates the paths taken in the Markov parser is something like LOOSEFOOT SIX ZERO

TWO TAKE ANGELS ONE POINT TWO where LOOSEFOOT is the **subj**, SIX ZERO TWO are the three **idn s**, TAKE ANGELS is the **cw**, and ONE POINT TWO are the **df s**. This particular example of the Markov based topology handles a 117 word vocabulary with many thousands of possible word sequences. It is simplistic, and easy to modify.

Markov Syntactic Parser Tests

The Markov syntactic parser was tested with 100,000 randomly generated strings of symbols of which 999,312 were syntactically correct and 688 deliberately contained errors. The errors were such that a human being could interpret the correct meaning, but a typical computer based parser would not. For example, the command may be missing the word TURN, but the word RIGHT was present. In the case of this particular phraseology, there is no other interpretation for the word RIGHT other than to turn to the right, therefore just having the word RIGHT is sufficient to get the intended meaning. Unless the grammar rules in the typical bottom-up parser allow the omission of the word TURN, the phrase would not be parsed and it would count as an error. The bottom up parser found all of the errors as expected. The Markov syntactic parser found and produced the correct result for 453 of the 688. This is a 66 % increase in producing a correct response from the known errors when compared to the bottom-up parser. The trade-off is the increase in incorrectness with the Markov syntactic parser. The bottom-up parser made no mistakes and did not produce any incorrect parses whereas the Markov syntactic parser produced 98 incorrect parses. This means that there is an incorrectness of 14% in the Markov parser when compared to the bottom-up parser. In other words, the Markov parser allowed 14% of the errors to be presented as something that may not be accurate. Since most of us don t hear every uttered word, the error rate may be tolerable in a system that considers other domain related information to discern truth. Because current parsing techniques do not include higher level cognitive processes, the precise words that are missed by the parser have no deeper meaning; therefore the parser is incapable of asking for a clarification.

The Cross Entropy Syntactic Identifier

The concept of the cross entropy syntactic identifier originates from a property of statistically based pattern recognition systems whereby statistically equivalent observations occur, but are not reduced to parts for further analysis due to computational restraints. The cross entropy syntactic identifier requires several minutes to process each string of symbols on a personal computer with no other processes running. As the personal computer becomes faster, this approach may become feasible for real time application. Typically, statistically based speech recognition systems have language models that are only accurate to the word level because of the enormous possible phonetic combinations. The speech recognition system will produce phonetically correct but nonsensical results. Entropy is a measure of how much information emanates from a model. Entropy depends upon the variability of the values as well as the number of values. Well-ordered data indicates entropy is closer to 1 and randomness indicates entropy is closer to 0. The theorem defined to measure the entropy has the form:

$$\mathbf{H} = -K \sum p_i \log p_i$$

Figure 2. General Entropy Equation

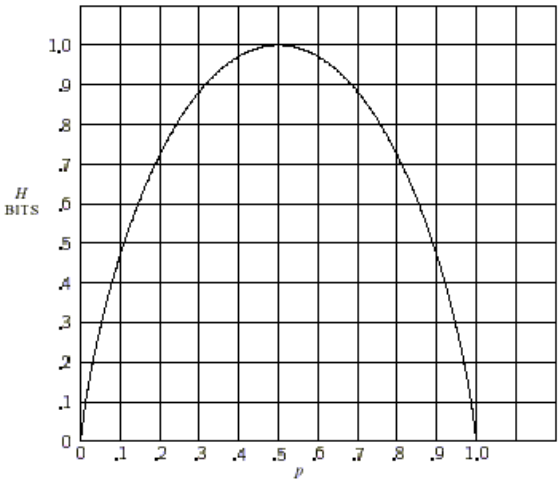

Figure 3. Entropy solution for the case of two probabilities p and q=1-p [7]

H is defined as the entropy, K is a constant, and pi is the probability of a system being in cell i of its phase space. The entropy in the case of two possibilities with probabilities p and q = 1 — p is shown infigure 3.

Randomness indicates entropy is closer to 0. The entropy *H* formula [7] with probabilities *p* and *q=1-p* (*p* and *q* are the only two possible probabilities) is:

$$\mathbf{H} = -(p \log p + q \log q)$$

Figure 4 . Entropy Equation for Two Variables

where q and p can be two probabilities from two dissimilar corpuses and the logs are in base 2.

Cross entropy determines how much of p is in q where the number of bits H indicates how much information. More information indicates less match

and fewer bits indicates more match between p and q where *p* can be the observation and *q* can be used as a variable from any one of a set of multiple phraseologies. The cross entropy is a measure of how much of one type of data is in another. Separating two dissimilar communications may be achievable computationally by knowing two independent phraseologies and measuring the cross entropy. We want to know how much of an observation appears in a particular phraseology. This approach appears to be unique for separation of two mixed communications effectively focusing attention on one rather than the other. Further, there is potential application of the cross entropy identifier to separate phraseologies where the phraseologies are significantly dissimilar.

The formula for the cross entropy simply swaps the *logp* for the *logq* in equation shown in figure 5. An observed input can be measured for it s cross entropy to determine how much of the observation appears in each of two dissimilar phraseologies.

$$\mathbf{H} = -(q \log p + p \log q)$$

Figure 5. Cross Entropy Equation for Two Variables

The cross entropy process was implemented computationally to investigate the potential for resolving membership in dissimilar domains. American English speech consists of 44 defined speech phonemes where each word can be described as a sequence of these phonemes. The cross entropy is used to compare two words by its components, where each word is represented by its sequence of phonemes. The phoneme sequences are represented as transcriptions where the transcriptions can be used to measure the cross entropy. Each phoneme, is assigned a probability, which was predicted using a statistical language modeling (SLM) algorithm. The SLM algorithm uses a training corpus representing individual domain phraseologies. A corpus is a set of example symbol strings that represent the types of expected behavior intended to be represented by the model. In applying the cross entropy, the a priori SLM containing phoneme probabilities in conjunction with a dictionary module containing all the possible words with their transcriptions is cross calculated using the observed words with their associated transcriptions. Each domain has its own SLM and each SLM contains the single phoneme probabilities as well as the dual phoneme or biphone probabilities for calculating digram probabilities.

Using Shannon s Mathematical Theory of Communication [7], probabilities depend on preceding symbols and in the simplest case only the immediately preceding symbol. The structure is described as a set of transition probabilities $p_i(j)$, the probability that symbol i is followed by symbol j. The indices i and j range over all the possible symbols. The frequencies $p(i)$, or unigram probabilities, and the transition probabilities $p_i(j)$, and the digram probabilities $p(i,j)$are related by the following formulas:

$$p(i) = \sum_j p(i,j) = \sum_j p(j,i) = \sum_j p(j)p_j(i)$$

$$p(i,j) = p(i)p_i(j)$$

$$\sum_j p_i(j) = \sum_i p(i) = \sum_{i,j} p(i,j) = 1.$$

Figure 6. Unigram and Digram Probability Equations

The digram probabilities are obtained by multiplying the unigram probability $p(i)$ by the transition probability matrix $p_i(j)$, and indicate the probability of symbol pair. As a specific example, given the $p(i)$ and $p_i(j)$ for symbols A B & C, the $p(i,j)$ would appear as follows:

$p_i(j)$		j		
		A	B	C
i	A	0	$\frac{4}{5}$	$\frac{1}{5}$
	B	$\frac{1}{2}$	$\frac{1}{2}$	0
	C	$\frac{1}{2}$	$\frac{2}{5}$	$\frac{1}{10}$

i	$p(i)$
A	$\frac{9}{27}$
B	$\frac{16}{27}$
C	$\frac{2}{27}$

$p(i,j)$		j		
		A	B	C
i	A	0	$\frac{4}{15}$	$\frac{1}{15}$
	B	$\frac{8}{27}$	$\frac{8}{27}$	0
	C	$\frac{1}{27}$	$\frac{4}{135}$	$\frac{1}{135}$

Figure 7. Unigram and Digram Example

Fundamentally, $p_i(j)$ is the transition matrix, p(i) s are the unigram probabilities and p(i,j) s are the digram probabilities. For example, p(B,A) = 8/27 where the p(i,j) in this case is the probability that a A will be followed by a B. Using digram probabilities is more computationally intensive than using just unigrams, but the accuracy of the pattern matching is significantly improved because there is more contextual information to determine the relevance of the current observation with the previous observation.

Cross Entropy Tests

Shown in Figure 8 is the square law relationship between entropy and perplexity. Perplexity can be roughly described as the number of possible symbols following a given symbol. The chart shows that for the smaller language models, such as LM_1.0 through LM_1.4, that the perplexity does not increase significantly for small entropy values.

These smaller language models contained on the order of one hundred possible words. In contrast, the larger language models such as the 10.50_no_oops, 10.50_oops, 2.10 and 2.50 which contain many thousands of possible words, result in a significant increase in perplexity. The no_oops simply means that the twelve test words were not included in the

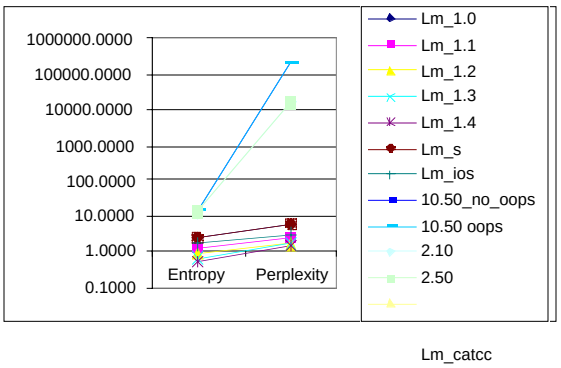

Figure 8. Entropy and Perplexity for Test Cases

training corpus used to develop the language model. The 10.50 represents 50,000 possible strings of symbols each containing 10 words each. The 2.10 is 10,000 strings of symbols with only two possible words in each. The test words were removed in one of the test cases to see the effect of not having the test words trained in the language model and it became evident that having the test words included in the training set did have a positive effect. The 10.50 and 2.10 language models used a dictionary of the English language to draw their possible strings.

In Figure 9, the leftmost three bars show the very first N-Best results for each of 12 test words and the rightmost three bars show the second N-Best results for the same 12 test words. Just looking at the total errors, that is 52 for the very first N-Best and 34 for the second N-Best, there is a 35% increase in accuracy when using the second N-Best results over using the very first N-Best results for each of the twelve test cases.

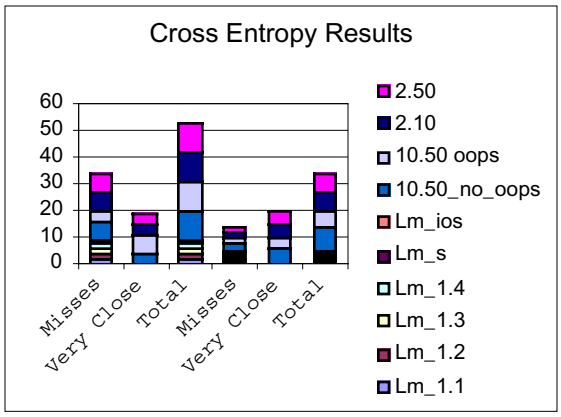

Figure 9. Cross Entropy Matches from Various Language Models

The N-Best is simply the top N possible results. An N-Best solution was used to determine if phonetically similar results were produced that would produce a more accurate solution. Shown in Tables 1 and 2 are the top two N-Best sets of results for all of the language models tested with the cross entropy algorithm. The bolded words are correct answers. The italicized words are the answers that are within one phoneme of being correct. For example, the very first N-Best results for the word HEAVEN was consistently HEAVEN S which is phonetically very close. The second N-Best results was precisely presented as the word HEAVEN. If there were some domain frame of reference, the exact interpretation could be determined between the top first and second N-Best results. It the word FEINER is phonetically identical to the word FINER therefore this is an exact match. The time to calculate the cross entropy for the larger language models was on the order of several minutes with a 600 MHz Pentium IV processor. Currently, the cross entropy is not computationally feasible, but it is evident that the potential exists for a significant increase in recognition accuracy.

The results of twelve test words used to see if the cross entropy could produces a correct phonetic match are shown in the example in Tables 1 and 2. The twelve test words were presented to each of the various language models. The cross entropy results indicate that the correct answer can probably be found in the top few N-Best iterations. Table 1 shows the first N-Best results. Only a few words were identified correctly. Table 2 shows much better results and at some point within the remaining N-Best there is a high probability that the exact pattern will be matched. Semantics are not considered in the cross entropy pattern match, but each of the possible N-Best results could be iterated through a semantic parser to determine if there is meaning to a string of symbols. With fast enough computation, this would

be feasible. The parses for five to ten word sentences required on the order of 100 ms for a single pass. Add in the several minutes to get the N-Best, then iterate each N-Best through the parser, and the result is excessive time for a real-time application using today s current standalone personal computers.

	10.50_no_oops	10.50 oops	2.10	2.50
HERO	*HERO'S*	*HERO'S*	*HERO'S*	*HERO'S*
DONE	UNUSED	*DONE'S*	UNUSED	UNUSED
WHO	HOOVES	*HEWES*	HOOVES	HOOVES
TREE	TIERS	TIERS	TIERS	TIERS
POUR	*POORS*	*POORS*	*POORS*	*POORS*
HIVE	*HIVES*	*HIVES*	*HIVES*	*HIVES*
PICKS	KIPPES	KIPPES	CUSIP	CUSIP
HEAVEN	*HEAVEN'S*	*HEAVEN'S*	*HEAVEN'S*	*HEAVEN'S*
LATE	TAILS	TAILS	TAILS	TAILS
DINE	WINDES	*DINES*	WINDES	WINDES
FINER	**FEINER**	**FEINER**	**FEINER**	**FEINER**
BANANA	BANYAS	BANYAS	BANYAS	BANYAS
Misses	7	4	7	7
Very Close	*4*	*7*	*4*	*4*
Total Errors	11	11	11	11

Table1. Examples of the First N-Best Results for Twelve Unique Test Words

	10.50_no_oops	10.50 oops	2.10	2.50
HERO	**HERO**	**HERO**	**HERO**	**HERO**
DONE	*DONES*	*DONE'S*	*DONES*	*DONES*
WHO	*HEW*	*HEW*	*HEW*	*HEW*
TREE	RETRIEVE	RETREAT	RETRIEVE	RETRIEVE
POUR	*POORS*	*POORS*	*POORS*	*POORS*
HIVE	*HIVES*	HIVE	*HIVES*	*HIVES*
PICKS	CHICKS	CHICKS	CHICKS	CHICKS
HEAVEN	**HEAVEN**	**HEAVEN**	**HEAVEN**	**HEAVEN**
LATE	TAILS	**LATE**	**LATE**	**LATE**
DINE	*DINES*	**DHEIN**	**DHEIN**	**DHEIN**
FINER	**FEINER**	**FEINER**	**FEINER**	**FEINER**
BANANA	*BANANAS*	*BANANAS*	*BANANAS*	*BANANAS*
Misses	3	2	2	2
Very Close	*6*	*4*	*5*	*5*
Total Errors	9	6	7	7

Table 2. Examples of the Second N-Best Results for Twelve Unique Test Words

CONCLUSIONS

In conclusion, the investigation into the Markov syntactic parser and the Cross Entropy syntactic identifier proved these to be effective methods for processing streams of symbols.

The Markov syntactic parser, while some incorrectness is introduced, produces a significant increase in correctly parsed streams of symbols and allows for variability that is not available in typical bottom-up parsing techniques. Both the bottom-up parser and the Markov syntactic parsers can be crafted to tolerate out-of-phraseology words that are inserted in between correct phraseology. Neither the bottom-up nor the Markov parser can tolerate substitutions since the semantic meaning is lost. When compared to a typical bottom-up parser, the Markov parser provided a 66% increase in producing a correct response when parsing information that contained syntactic errors, but could be correctly interpreted by a human being. Generally, this is because the bottom-up parser is not tolerant of deletions, whereas the Markov parser can tolerate deletions of non-value

added words. The tradeoff is that the Markov parser allowed 14% of the errors to be presented as something that may not be accurate. Additionally, the time required to develop the Markov Parser is significantly less than the time required to craft a similar bottom-up parser. Further, a Markov parser is significantly simpler to produce and can, with little effort, be adjusted to tolerate a broad range of insertions and deletions.

The cross entropy syntactic identifier is clearly an effective method for finding phonetically similar words. An example that has been used to represent the challenge is RECOGNIZE SPEECH or WRECK A NICE BEACH. These phrases are almost phonetically identical, but their meanings are very different. It was proven in this research, that the cross entropy syntactic identifier could be used to present phonetically similar results, but with computational limitation.

A goal of this research was to determine if the cross entropy syntactic identifier could separate streams of military phraseologies. Separations of dissimilar phraseologies by identifying non-domain words using the cross entropy method for double phonemes (bigrams) was at best 50% accurate and on average only 35% accurate. By itself, the cross entropy syntactic identifier was not effective in separating similar phraseologies. However, the cross entropy syntactic identifier was effective in identifying a correct stream of symbols in a language model using the N-Best approach where the test words were used in the training corpus. This is significant because a number of possible words, that are not necessarily close in meaning, but are indeed phonetically close, can be used to provide an accurate parse. In the experiments described in this paper, simply searching the top few alternative n-best results produced the correct phonetic match. Thus, the cross entropy provides the ability to parse word substitutions when used with a traditional parser or the Markov parser.

RECOMMENDATIONS

Adding order to the language models will intuitively increase the accuracy of the cross entropy pattern matching. This order can be added using the WordNet from Princeton University, some basic algorithms for applying grammar rules and randomly generating grammatically correct corpuses. Further, based on this research, experiments with a combined Markov syntactic parser and a cross entropy syntactic identifier is warranted. The experiments would evaluate how well a cross entropy identifier produces phonetically similar words which are iteratively processed through a Markov parser. The variability of the Markov parser could be adjusted to determine the optimal engineering tradeoff between correctness and tolerance of insertions and deletions. This research was built upon previous research in language processing and pattern recognition [8][9] at the Naval Air Warfare Center. Subsequent research in this area would follow the work described herein.

REFERENCES

1) Slobin, Dan,I. *Psycholinguistics*, Copyright 1971, Scott, Foresman and Company, Glenview Illinois, Library of Congress catalog number: 70-131220

2) Rosenfeld, Ronnie. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Pittsburgh, PA, Carnegie Mellon University, 1994.

3) Covington, Michael A. *Natural Language Processing For Prolog Programmers* Prentice Hall, Englewood Cliffs, New Jersey, Copyright 1994, ISBN 0-13-629213-5

4) Dougherty, R. C. *Natural Language Processing-A Generative Grammar in Prolog*, Linguistics Department -New York University, Lawrence Erlbaum Associates, Inc. 1994.

5) Dowding, John; Moore, Robert; Andry Francis; and Moran, Douglas. *Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser*. SRI Inernational, Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, June 1994.

6) Cope, Bradley and Kotick, David. *Developing Speech Recognition Models for use in Training Devices*. Proc. of the 19th I/ITSEC, 1997

7) Shannon, C.E. *A Mathematical Theory of Communication*. Bell Systems Technical Journal, Volume 27, 379-423 (Part I), pages 623-656 (Part II), 1948.

8) Cope, Bradley and Kotick, David. *Integration of Highly Accurate Speech Recognition With Natural Language Processing*. Proc. of the 18th I/ITSEC, 1996.

9) Cope, Bradley and Boemler, Stephen G. *Improved Speech Recognition Using Quantized Frequency Domain Filters*. Patent Pending, 1998.