Application of Natural Language Processing Techniques to Marine V-22 Maintenance Data for Populating a CBM-Oriented Database

Huston Bokinsky[ac], Amber McKenzie[a], Abdel Bayoumi[bc], Rhea McCaslin[ac],
Andrew Patterson[c], Manton Matthews[a], Joseph Schmidley[d], Major General Lester Eisner[e]

[a]Department of Computer Science, [b]Department of Mechanical Engineering, [c]Condition-Based Maintenance Center
University of South Carolina, Columbia, South Carolina, USA
[d]United States Navy, [e]South Carolina Army National Guard (United States Army)

Abstract: This paper discusses modifications made to the Natural Language Toolkit, a well-known natural language processing software package, to achieve improved information extraction results when applied to helicopter maintenance records. In doing so, it will also attempt to elaborate the components of a tool under development to allow for machine analysis of the free-text fields of V-22 Osprey maintenance records. The authors have found that by adapting existing natural language processing software to suit peculiarities of the language found in maintenance records, substantive improvements can be made in the accuracy of information extraction. In particular, by modifying an existing text pre-processor to 1) take in multiple-sentence inputs, 2) treat all code tokens as the same, and 3) ignore distinctions in punctuation, part-of-speech tagging accuracy has improved from 92.49% to 96.59%; subsequently, entity chunking precision has improved from 91.5% to 92.3%.

Keywords: Natural language processing, Information extraction, Data preprocessing

Introduction

Condition-Based Maintenance (CBM), the practice of using vibration data, engine oil debris measurements, and other potential indicators of the health of complex machinery as guides in determining maintenance scheduling and procedures, distinguishes itself by its reliance on immense amounts of data both to study and understand the significance of different indicators as well as to collect accurate and meaningful information relevant to those indicators. In many cases, this data exists in the form of digital signals collected by sensors; in other cases, however, there is a large store of potentially valuable data in the form of hand-entered text. For the most part, the size of text archives, and the huge labor costs of going through them record by record, render such data sources entirely inaccessible to all but the most determined (and well-heeled) of institutions. Techniques of natural language processing (NLP), however, offer potential avenues into such data sources using machine processing to extract relevant information in a timely and cost-effective manner.

The Condition-Based Maintenance Research center at the University of South Carolina is currently developing a tool for the V-22 Osprey project management team that will allow for machine analysis of free-text fields from a large corpus of the aircraft's maintenance records. This work developed from a similar tool developed at USC to perform information extraction from maintenance records of a fleet of US Army Apache helicopters, and which achieved promising results by using "off the shelf" NLP software packages [1]. The focus of this discussion is a group of modifications made to that software in an effort to adapt it to the awkward language that is often used in maintenance records.

Figure 1 shows a graphical outline of the evolution of USC's efforts related to the V-22 project, including completed components (in green), discarded ideas that shaped the original design (in obfuscated grey), components currently under

development (in blue), and ideas for where the project might head in the future. The remainder of this paper will follow a similar outline: it looks first at the reasons for which textual analysis of maintenance records is a project worth pursuing for the V-22 project managers; it then gives an overview of the data and its characteristics that complicate normal NLP techniques; it then presents USC's current information extraction tool in its component stages – text pre-processor, part-of-speech tagger, and entity chunker – in the context of what modifications have been made to the tool and what results these have returned; finally, it concludes with a brief overview of where USC sees the project going in the immediate future.

Motivation for Examining Text Data

On a general level, maintenance records contain maintainers' observations of problems as they occur on an aircraft, symptoms associated with these problems, actions taken to correct the problems, and (sometimes) the degree of success of these actions; they also contain observations from when there are no problems reported, such as their notes made during a regular inspection. Relative to other sources and types of information able to be tapped for CBM research, this information is unique in nature, not only because it comes from a human source, but also because in many cases it is not channeled through a specific medium, measuring a specific indicator, drawn at a specific time. That is, a maintainer is able to record whatever draws his or her attention and seems, by his/her judgment, to be relevant. The information in maintenance records, then, has the potential to add a dimension to CBM data that digital sensor data cannot.

Used by itself or in conjunction with sensor data, maintenance record information can be exploited to draw strong connections between what is observable to the human eye and what is

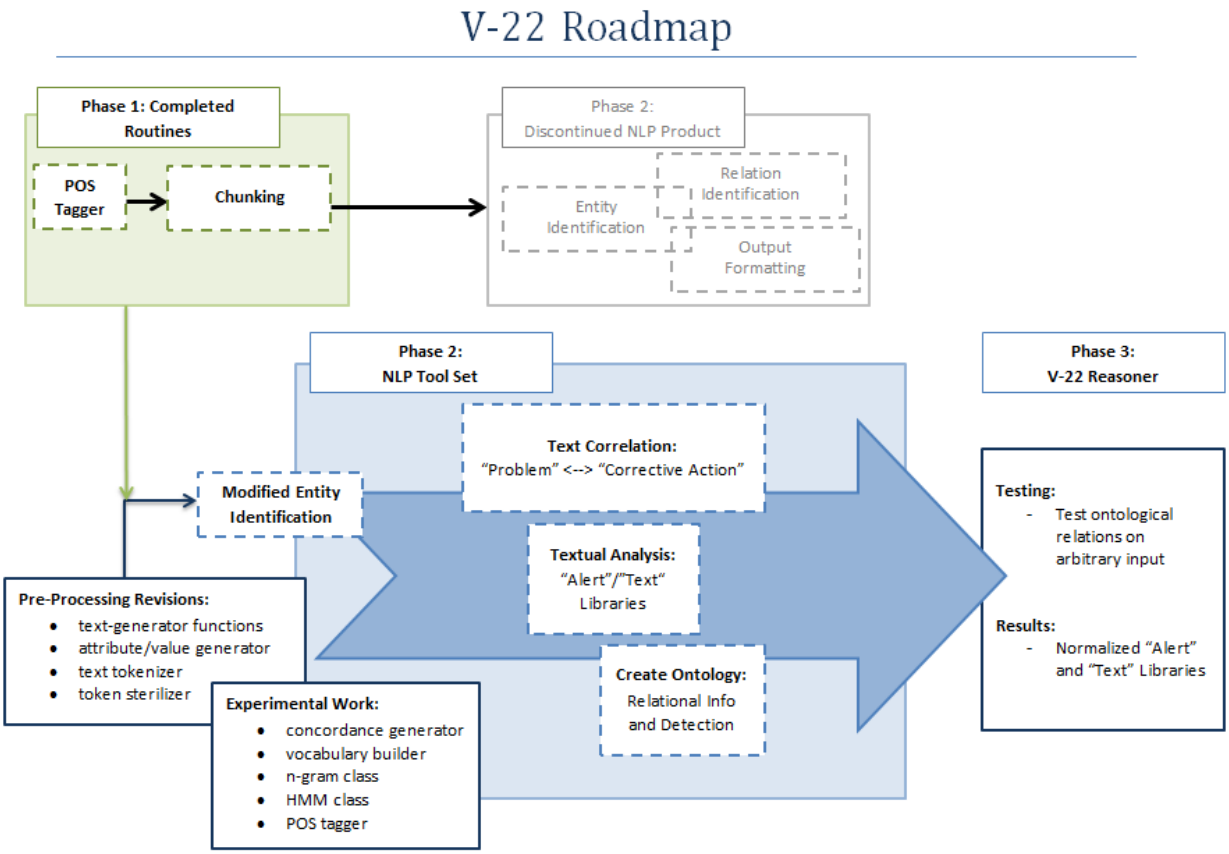

Figure 1: Proposed context of NLP Tools and results.

happening in the interior of an aircraft. From there, it can be used to identify patterns involving problems and maintenance actions, or help to augment troubleshooting techniques and guidelines available to the maintainers. Furthermore, the algorithms that process sensor data can be refined given the perspective of the maintainers. Nonetheless, all of these potential applications of this important source of information are contingent upon the ability to organize and store the information as data in electronic format, and to query it easily and reliably by machine.

Information extraction (IE) is a subset of NLP techniques developed to isolate meaningful data pieces from unstructured text sources. By far the majority of information extraction research has been done on standard English texts, such as news articles or literary texts. What work has been done on shorthand-style language has focused on medical records [2-5]. There has been very little research done in the domain of maintenance records. Thus, for the most part, the task of rendering free-text maintenance record entries into machine-utilizable CBM data remains undone.

Raw Data and the Task at Hand

The data that serve as the basis of this discussion are XML exports of past Maintenance Action Form (MAF) records extracted from the Decision Knowledge Programming for Logistics Analysis and Technical Evaluation (DECKPLATE) system, the NAVAIR data warehouse for aircraft maintenance, flight, and usage data. Every time a maintainer does work on an aircraft, whether it be for repair or for inspection, this action must be documented with a complete MAF. There is considerable data in these MAFs that is in structured form (either selected from drop-down lists or, more likely, entered automatically by the information system). Structured data presents no complications for machine processing, since it is all regular in form and content. Yet a MAF also contains two fields for the maintainer to note down 1) the problem encountered and 2) the action taken to fix the problem, in unstructured text. The field for the problem description is labelled "descrepnarr;" the field for the corrective action description "corract." USC's data is 82 XML files, each representing anywhere from two weeks' to two months' worth of MAF records. The total number of records in these files is 414,893, of which 385,680, or %92.96 of the total records, contain corract and descrepnarr fields. Since this project is concerned with the corract fields specifically, this paper will pay more attention to these.

With respect to MAF records, the intent of the V-22 team is to be able to match a corrective action recorded in a record with a corrective action recommended in the electronic maintenance manual in order to determine to what extent the work actually being done on an aircraft aligns with what is called for in the maintenance documentation. The work of correlating one piece of text to another can be quite complex. It should come as no surprise that the technique of simply matching strings[1] will yield very unsatisfactory results – rare are the occasions when the same meaning is expressed in exactly the same combination of words! A more promising approach is IE, which identifies meaningful sequences of tokens, then determines a linguistic relationship between them. By comparing, for example, a noun phrase and verb phrase from one string to a noun phrase and verb phrase from another string, the working hypothesis is that there is a better likelihood of drawing a positive and accurate correlation between the two. From the beginning, USC's project with maintenance records has been geared toward IE.

The tools used have been primarily from the Natural Language Toolkit (NLTK) package, an open-source set of software freely available for download and study. The NLTK is written in Python, a programming language that lends itself well to text processing because of the built-in ease of handling words as entities that it provides; a Python

[1] A bit on terminology: a *character* is any letter, space, punctuation mark, etc. that can be individuated as a basic linguistic symbol. A *string* is a series of characters taken together as a grouping: strings can be anything from single letters, to words, to sentences, to multiple paragraphs. A *token* is the building block of a sentence; usually, 'token' is synonymous with 'word,' but there are many possible exceptions to this rule.

programmer can more naturally issue instructions to the computer to view textual data in much the same way that a human reader views it, circumventing many of the contortions needed for processing text data in a language such as C.

At the outset, USC's objective was to remain domain-independent; that is, to develop a tool that could be applied to any sort of input data – not specifically maintenance record data. Thus, every effort was made not to alter any of the original NLTK tools to suit maintenance texts specifically. However, there are substantial differences in the nature of the language used in maintenance records and the language for which the NLTK tools were written. An exact characterization of the vocabulary and style in which a corract string is composed is elusive; hence so is a comprehensive set of rules for parsing and interpreting any given string. In maintenance record data, 1) critical words are often abbreviated or misspelled, eg. "PLOACES" or "PRGB;" 2) non-critical words are often omitted, and words run together, eg. "CONSUMABLESRECIEVED;" 3) confusing lexica that thwart the use of standard English dictionaries for parsing, eg. "TROUBLESHOT L ABCV FADEC A TO BAD VALVE IAW IETMS 7325;" 4) frequently abbreviated words and use of acronyms, eg. "R L/H NLG TIRE ASSY. IAW SSS 3240 ATAF APAF AREA FOD FREE;" 5) a great deal of inconsistency in the use of abbreviations and acronyms, eg. "LEFT HAND," "LEFTHAND," "LEFT," "L/H," "LH," and "L" are all used interchangeably from one record to another. The NLTK (as well as all other readily-available language processing tools) is written to handle standard, edited, grammatically-correct English input. Consequently, the results of running some NLTK tools on the V-22 data are not as good as they would be when applied to standard English texts -- a number of the difficulties encountered during USC's initial work were related not to the difficulty of applying NLTK's toolset, but rather to the incompatibility of this toolset and the data at hand.

Text Pre-Processor

Presented with a text file, a computer only sees a single string of characters, including spaces and newline characters. A pre-requisite step of IE is to break this single string into segments that can be processed individually, in a fashion similar to the way in which a human reader would. Hence, pre-processing is the conversion of a collection of digital characters into a series of items that can be processed as language. Most of pre-processing is the individuation of words and punctuation called 'tokenizing,' though there are other features that can be added to a pre-processor.

The native NLTK tokenizer takes as input a single sentence, which it first splits around white space. Next, each resulting token is subjected to further processing. Any tokens are broken around punctuation marks, excepting periods which, since the tokenizer is only accepting single sentences, are all assumed to have significance as abbreviation points or decimal points. Also excepted is the '/' symbol to allow for occurrences such as 'w/in.' Significantly, contractions such as "won't," "pilot's," or "they'll" are broken around the apostrophe. The tokenizer then returns a list of tokens -- a structure in which each word, punctuation mark, symbol, or contraction piece which was separated from the others is an element by itself.

There are several features of the native NLTK tokenizer that were considered for modification to better fit the V-22 data:

1) The NLTK native tokenizer was made to break up tokens around hyphens; for most English texts this is sensible, since most often an analysis is more concerned with what is"blue" and "green" than with what is "blue-green." In the V-22 maintenance data, though, hyphens frequently occur in the middle of entities that should be processed as individual tokens. These include part codes such as as "901-060-857-114" and references to outside manuals such as "NA16-30PRQ7-2" or "4270NWA57-01." For example, the native NLTK tokenizer would break "NA13-1-6.1-1" into seven tokens: "NA13," "-," "1," "-," "6.1," "-," and "1." Consequently, the POS-tagger and named entity chunker (IE steps that follow pre-processing) would have to process seven tokens rather than just one. USC's hypothesis was that changing this standard behavior of the NLTK pre-processor could reduce the number of tokens to

process in later steps, reducing the statistical variation in tokens and hence improving IE accuracy.

2) The NLTK tokenizer does not tokenize any arbitrarily large sequence of characters; it only tokenizes one sentence at a time. Thus, before text can be tokenized it must first be broken into individual sentences ("sentence tokenizing" in NLTK's terminology), which NLTK does simply by finding terminating characters such as question marks and exclamation points. Periods present the only significant difficulty in this process, since the same character can also be used to abbreviate a word ("Mr., Mrs."), mark decimal places ("$10.91"), or construct ellipses ("..."). Each sentence is then fed to the tokenizer and is converted into a series of words and punctuation.

One reason NLTK does this is to preserve the meaning of punctuation. In the natural English for which the NLTK was written, periods, colons, semicolons, and commas all have specific uses and meanings; in maintenance data, however, punctuation is often used interchangeably -- these marks all generally serve to separate one complete idea from another. USC's second hypothesis was that by no longer giving sentence-ending periods a functional role different from that of semicolons, colons, or even commas, statistical variation of input tokens could be further reduced. As an added benefit, the step of sentence tokenizing required by the NLTK tokenizer could be obviated.

3) For standard language processing, every word has a particular meaning; the perceptible difference between two tokens such as "read" and "lead" is slight (one letter). However, it is a difference that has enormous consequences for the results of information extraction and must at all costs be preserved. On the other hand, in processing maintenance records, there is no semantic difference between two manual references such as "NA16-30PRQ7-2" and "4270NWA57-01." For the purposes of information extraction, until it comes time to display extracted data items to the user, these can be treated as one and the same without any loss of efficacity. USC's third hypothesis was that by treating any tokens that act like code references as the same token, statistical variation of input tokens could be reduced further.

The ideas behind these three hypotheses were implemented with changes made directly to the NLTK tokenizer code and through the added step of a token "sterilizer." This latter takes in a series of tokens and outputs the same series with any punctuation mark replaced by '<punc>' and any token containing numerals replaced by '<code>'. Sentence-ending periods are the only punctuation mark that were left intact. Periods used inside numeric tokens ("1.23 psi") and as abbreviating marks ("a.c.") were not distinguished from their context. As an example, the string "ED/W IAFC-0044; INCORP OF NACELLE COOLER BLOWER PAN REINFORCEMENT MOD.," after tokenization and sterilization, is shown in Table 1.

Consequent to the evident changes made to

Original Token	Sterilized Token
ED/W	ED/W
IAFC-0044	<code>
;	<punc>
INCORP	INCORP
OF	OF
NACELLE	NACELLE
COOLER	COOLER
BLOWER	BLOWER
PAN	PAN
REINFORCEMENT	REINFORCEMENT
MOD	MOD
.	.

Table 1: example text tokenized and sterilized

the tokens themselves, token sterilization also changes the demographics of the tokens being processed. The *vocabulary* of a collection of text data is the number of distinct tokens encountered; this number can be divided by the total number of tokens to give a measure of the variety of tokens used throughout a collection. The total number of tokens counted in the data's corract strings is 6,327,420. The vocabulary of the corract strings before sterilization is 110,141; after sterilization, this figure becomes 26,240. Thus the variation in tokens as measured by vocabulary / total tokens encountered goes from 0.017407 to 0.004147 -- a reduction by a factor of over 4.

Part-of-Speech Tagger

After the tokens in a record have been individuated comes the task of selecting those of interest in analysis. One possible method would be, for example, to pick out all nouns. However, there are several complexities of English that make computer processing of natural language difficult, and this task in particular inefficient. One such complexity is the way in which a same word can have multiple meanings depending on context. Thus, to identify the word 'pilot' has limited use, since it isn't clear if that instance of 'pilot' occurs as a noun or as a verb, or even an adjective. Part-of-Speech (POS) taggers can reduce this area of doubt.

While there are a number of tagging algorithms that have been developed, the ones best suited to this context are N-gram tagging algorithms. An N-gram tagger builds a statistical model from hand-tagged training data by examining the tokens that appear before and after a given token and judging the most likely tag based on what token precedes and follows it, as well as what were the parts of speech of the preceding and succeeding tokens. This method is advantageous over some other, more sophisticated methods because very often tokens appear in the test data that were not encountered in the training data; rather than grind to a halt, an N-gram tagger can easily pass responsibility for tagging an unknown token to a more generalizable tagger. Such a series of POS taggers, aligned from more to less refined, can be chained together in a "back-off" system. A higher *n* N-gram tagger has the advantage of bringing more information to its tag decision; however, the scope of what it can recognize and tag with confidence also becomes more limited. A lower *n* N-gram tagger makes less-informed tagging decisions, but it will tag a greater range of vocabulary. At the bottom is a default tagger which tags everything it sees with the same tag. This latter is a last-resort tool to prevent tokens unseen in the training data from going

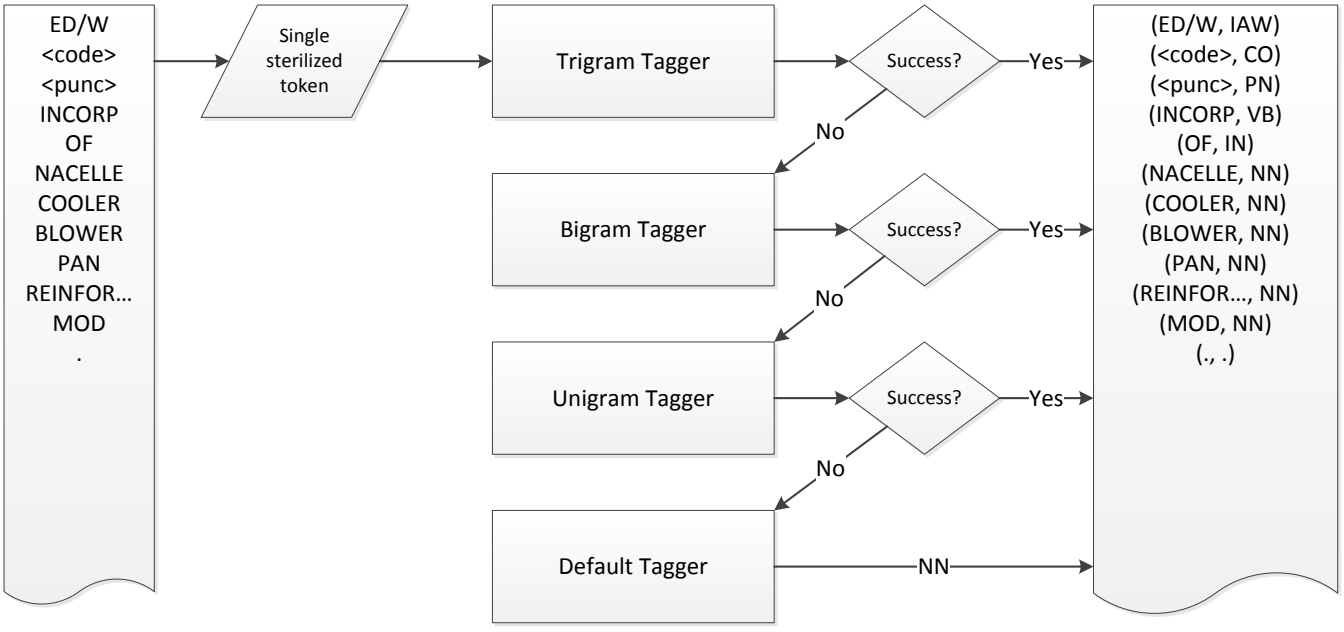

Figure 2: Flowchart for POS tagger.

Original Token	Sterilized Token	POS
ED/W	ED/W	IAW
IAFC-0044	<code>	CO
;	<punc>	PN
INCORP	INCORP	VB
OF	OF	IN
NACELLE	NACELLE	NN
COOLER	COOLER	NN
BLOWER	BLOWER	NN
PAN	PAN	NN
REINFORCEMENT	REINFORCEMENT	NN
MOD	MOD	NN
.	.	.

Table 2: example text tokenized, sterilized, and POS-tagged

untagged or breaking the system.

USC's implementation chains together four taggers. All tokens to be tagged are first sent to the trigram (3-gram) tagger, which has built a statistical model based on series of three tokens and parts of speech found in the training data. If this tagger is able to issue a tag for the token in question, it does so; otherwise, it backs off to the bigram tagger. Since the bigram tagger's statistical model is less refined than that of the trigram tagger, it is more likely to be able to issue a tag. If not, it can back off to a unigram tagger. If none of the taggers that have built models from the training data are able to tag a token, the token is tagged by a default tagger which simply gives all tokens the same tag. Since most of the words encountered in this maintenance data are nouns, this tagger has been set to issue 'NN' tags.

The set of POS tags used is a modified version of the commonly-used Penn-Treebank tagset. Since entries in maintenance records are typically straightforward grammatically, approximately twenty POS tags from this tagset suffice to tag the data. The following are five example tags from the tagset, what they mean, and what sort of words they tag:

CD	cardinal number ('1,' 'one')
JJ	adjective ('broken')
NN	noun ('nacelle')
RB	adverb ('inside')
VB	verb ('install')

The above example, after being part-of-speech tagged, is shown in Table 2.

The NLTK has a native POS tagger, which uses a similar system of N-gram taggers placed in a back-off cascade, and which has been trained on a hand-tagged corpus of texts known as the Brown corpus. This corpus is a collection of texts from classic literature, newspapers, culture magazines, scholarly journals, and stories, so represents the English language at its most carefully constructed level. Considering the vast differences between the language on which the NLTK tagger was trained and the language found in maintenance data, the need to construct a new training set from hand-tagged was immediately apparent.

For evaluation purposes, a file of hand-tagged and hand-chunked corract data was prepared. This file contains approximately 30,000 lines, representing 1794 entries from the raw XML data. Compared with the Brown corpus, this data set is very small, especially considering that only one half of it can be used for training a tagger model while the other half is used for evaluating the performance of the model. Still, even being trained on slightly less than 17000 lines, the tagger achieved an accuracy rate of 96.59%.[2] This performance is considerably

[2] The accuracy of a part-of-speech tagger is a measure of what percentage of the tags the system issues align with the hand-assigned tags in the evaluation data. Accuracy = # correct tags / total # tags in data set.

Original Token	Sterilized Token	POS	Chunk
ED/W	ED/W	IAW	B-IAW
IAFC-0044	<code>	CO	B-RF
;	<punc>	PN	O
INCORP	INCORP	VB	B-VP
OF	OF	IN	O
NACELLE	NACELLE	NN	B-NP
COOLER	COOLER	NN	I-NP
BLOWER	BLOWER	NN	I-NP
PAN	PAN	NN	I-NP
REINFORCEMENT	REINFORCEMENT	NN	I-NP
MOD	MOD	NN	I-NP
.	.	.	O

Table 3: example text tokenized, sterilized, POS-tagged, and chunked

better than it was on unsterilized data. To make a comparison, a new tagger model was built from the same training data, though this time the data was not passed through the sterilization process. The same evaluation tests were run on the same test data, which also was not passed through the sterilization process. The accuracy of this model fell to 92.49%, indicating that the sterilization step of the preprocessor adds significant value to the tool.

Chunker

Another complexity of English is that often-times one word does not by itself correspond to one idea. More often than not, combinations of words must be read, or processed, all together in order to have meaning. For example, V-22 records could refer to a 'tail rotor swashplate.' If a computer is unable to put all three of these words together as representing a single entity, it will fail to properly analyze the sentence in which it finds them. Chunking is the task of grouping together words that go together meaningfully. Chunkers rely upon POS tags to accurately identify where a word chunk begins and ends and, like POS tagging, use tags to indicate when such groupings begin and end. Also like POS taggers, chunking algorithms work by building statistical models from a hand-tagged data set, then make chunking decisions based on the token encountered, tokens preceding and succeeding it in order, the POS tags of all environing tokens, and the chunk tags of those preceding it. For this work, USC's tool has been using the NLTK NEChunkParser class which assigns chunk tags based on an algorithm know as maximum entropy. The intuition behind the algorithm is to use features of a token (here, the token, its part of speech, and its immediate context), to narrow the possible tags which might be assigned to it, then making a most-likely decision from there.[3]

Chunk tags are much simpler to understand than POS tags. There are certain kinds of chunks that interest us: noun phrases ('NP'), verb phrases ('VP'), and references ('RF'). Pieces that are not of direct consequence may be tagged 'O' for 'other.' The USC tool adds a chunk label to take advantage of a certain feature of the maintenance data to help the chunker identify specific references to the electronic maintenance manual or to other documents. Very often, such references are introduced either by the acronym 'IAW' (short for "in accordance with") or some variation on this. Since the variation could use multiple words (sometimes "IN ACCORDANCE WITH" is spelled out in the data), USC introduced a chunk tag just for these items -- 'IAW.' After this, any token encountered either begins a chunk, in which case 'B-' is prepended to the chunk tag, or it is inside a chunk, in which case 'I-' is prepended to the chunk tag.

[3] For a description of Maximum Entropy modeling in general, see Chapter 6 of Jurafsky and Martin [6]. For a description of the algorithm used by the NLTK tool specifically, see Daumé's paper on his Megam code [7].

The above example, when run through the chunker, is shown in Table 3. The output shown in Table 3 means that the machine has taken the original string and extracted from it two items of interest: 1) a reference "IAFC-0044" and 2) "NACELLE COOLER BLOWER PAN REINFORCEMENT MOD," an entity to correlate to the manual. As with the POS tagger, sterilization of the input does contribute to an improvement in chunker performance, though the difference is much smaller here. On unsterilized data, the chunker had 91.5% precision and 87.8% recall.[4] On the same data set sterilized, the chunker had 92.3% precision and 88.3% recall.

Anticipated Application of Results

The two objectives of the V-22 project to which these techniques are to be applied are both focused on analysis of textual data. The first is to correlate the corrective actions entered in MAF records to a recommended action in the aircraft's technical manual. A tool that can accurately match, or give a leveraged conjecture of, which maintenance actions correspond to which recommended actions in the manual would provide the basis for an analysis of the program's technical manuals and, more tangibly, a basis for improving first-use rates. Yet answerable questions include: once a maintainer in the field has diagnosed a problem using methods enumerated in the manual, is the action s/he takes to correct that problem the same as what is recommended in the manual? How often does this "first-use" occur? By improving this rate, and by hence reducing the number of times an aircraft goes in for maintenance due to a single cause, costs represented by maintenance man hours (MMH), parts expended in replacements, and time an aircraft is unavailable for service can all be reduced. This is especially true in the area of unscheduled maintenance, resulting in maintenance savings for the program.

The second program objective also has potential for savings in man hours both in the field and in the support office. This is to analyze the separate Alert and Text libraries from the aircraft technical manual with the object of eliminating redundant entries and information. Because the manual is the result of many people writing entries according to their expertise over a long period of time, it contains many areas of duplication. Often this duplication is not exact, word for word copies, but only semantically similar entries, yet nonetheless represents added cost in inefficiency of authorship and of use time for a maintainer, and which it is desirable to eliminate.

As of this writing, a concrete implementation to use the results of the above discussion for the tasks of correlation of maintenance records to manual entries or of manual entries to each other has not been made (see Figure 1 above). The implementation envisioned involves three more loosely-defined steps: 1) taking the chunked results of the techniques outlined above and fixing, inasmuch as possible, misspellings as well as expanding abbreviations to known equivalents; 2) similarly deconstructing the technical manual entries to which corrective action maintenance entries are to be correlated and using these to build a database of actions, noun phrases, and references that can be queried by the statistical probability or proximity of MAF chunks to database records; and 3) evaluation of correlation results based on a limited-size, hand-correlated "gold" set of data. Though these steps remain loosely-defined, the first two are expected to provide a blueprint for meeting the concrete objectives of the V-22 project coordinators.

Acknowledgements

This project has been funded by grants from the United States Navy and the Boeing Corporation.

References

[1] A. McKenzie, M. Matthews, N. Goodman, and A. Bayoumi, "Information extraction from helicopter maintenance records as a springboard for the future

[4] Precision and recall are two statistics to measure chunker performance. Precision = # correct chunks given by system / total # chunks given by system. Recall = # correct chunks given by system / total # actual chunks in data. Notice that these formulae are measuring chunks, not chunk tags. Thus to be correct, the sequence of tokens that is labelled a chunk must share all of the same tags as the same sequence in the hand-tagged 'gold' data.

of maintenance text analysis," *The Twenty Third International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems* (IEA-AIE 2010), June 1-4, 2010

[2] Kraus, S., Blake, C., & West, S. L. (in press). Information extraction from medical notes. *Proceedings of the 12th World Congress on Health (Medical) Informatics -- Building Sustainable Health Systems (MedInfo)*, Brisbane, Australia, 1661-1664.

[3] Gaizauskas, R., Harkema, H., Hepple, M., & Setzer, A. (2006). Task-oriented extractions of temporal information: The case of clinical narratives. *Thirteenth International Symposium on Temporal Representation and Reasoning* (TIME'06), 188-195.

[4] Irvine, A. K. (2008). Natural language processing and temporal information extraction in emergency department triage notes. *A Master's Paper*.

[5] Han, B., Gates, D. & Levin, L. (2006). Understanding temporal expressions in emails. *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, 136-143.

[6] Jurafsky, D. & Martin, J.H. (2009). *Speech and Language Processing* (Second Edition), Pearson/Prentice Hall, Upper Saddle River, NJ.

[7] Daumé, H. III (2004). Notes on {CG} and {LM-BFGS} Optimization of Logistic Regression. Paper available at http://www.cs.utah.edu/~hal/docs/daume04cg-bfgs.pdf.