Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2001), *15*, 295–305. Printed in the USA.
Copyright © 2001 Cambridge University Press 0890-0604/01 $12.50

Extracting information from free-text aircraft repair notes

BENOIT FARLEY
Institute for Information Technology, National Research Council of Canada, Ottawa, Ontario, K1A 0R6, Canada

(Received November 10, 2000; Accepted December 5, 2000)

Abstract

For every problem mentioned by crew members in an aircraft log book, an associated repair action note is entered in the same log book by a maintenance technician after the problem has been handled. These hand-written repair notes, subsequently transcribed into a database, give an account of the actions undertaken by the technicians to fix the problems. Written in a free-text format with peculiar linguistic characteristics, including many arbitrary abbreviations and missing auxiliaries, they contain valuable information that can be used for decision support methods such as case-based reasoning. We use natural language techniques in our information extraction system to analyze the structure and contents of these notes in order to determine the pieces of equipment involved in a repair and what was done to them. Lexical information and domain knowledge are extracted from an electronic version of the illustrated parts catalog for the particular airplane, and are used at different stages of the process, from the morpholexical analysis to the evaluation of the semantic expression generated by the syntactical analyzer.

Keywords: Aircraft Maintenance; Free-Text; Information Extraction; Natural Language; Semantic Interpretation

1. INTRODUCTION

When an aircraft is held on the ground for repair, time is of major importance. If a malfunction has been detected, it is crucial to rapidly identify the source of the problem and to make the repair. The defective parts may have to be replaced, and replacement parts must be readily available. With the objective of minimizing repair and maintenance time, the Integrated Diagnostic System (IDS) has been developed at the National Research Council of Canada (NRC) in collaboration with Air Canada (Wylie et al., 1997). This system monitors every message transmitted from the airplane's computers to the ground, sorts them into clusters—each representing a probable specific problem, and outputs for each problem-related cluster a diagnosis in terms of probable causes along with suggestions on repair actions. All this is done while the airplane is still in the air, which saves troubleshooting time on the ground to determine the source of the problem. It also saves on the repair time since the parts involved in the suggested repair may be identified, ordered, and received before the airplane arrives. Part of the IDS diagnosis reasoning, including causes and repair suggestions, is based on a huge set of rules, automatically extracted from an electronic version of the manufacturer's troubleshooting manual.

To improve diagnosis performance, the system will rely on a set of cases, where observed symptoms and repair actions along with the repair's results are paired together. That information can be found in the hand-written free-text notes of the airplane's log book where the crew write down their observations about any problem or malfunction they are aware of, and where the technicians write down what they did to fix the problem. Each problem-related entry of the log book is called a "snag." The snags are ultimately transcribed into a database along with supplementary information such as the airplane's identification. As a first step toward the automation of the case creation process, we have been applying natural language processing techniques, namely grammatical analysis and knowledge-based semantic interpretation evaluation, to the free-text repair action notes (FTRAN) of the snags in order to identify which parts were involved in the repair and what was done to them.

This paper describes the implementation of SNAGIE (Snag Information Extractor), our natural-language-based information extraction system for extracting useful information from free-text aircraft repair actions notes (see Farley, 1999, for an overview of the concepts). In the first part of this

Reprint requests to: Benoit Farley, Institute for Information Technology, National Research Council of Canada, Montreal Road-M50, Ottawa, Ontario, K1A 0R6, Canada. E-mail: Benoit.Farley@nrc.ca

https://doi.org/10.1017/S0890060401154041 Published online by Cambridge University Press

paper, we describe the linguistic characteristics and peculiarities of the FTRANs. In the second part, we describe the various modules of the implementation of SNAGIE.

2. THE LANGUAGE OF THE REPAIR NOTES

The FTRANs are written by maintenance technicians, with conciseness as a goal, and the assumption that whoever reads them will understand. This care for conciseness translates at two linguistic levels: *lexical* and *syntactical*. At the lexical level, numerous abbreviations are used. Abbreviations are words from which certain letters, assumingly unnecessary for their recognition, have been removed. They account for approximately 13% of all the words in a corpus of 884 notes that we randomly selected. Some words are more or less consistently abbreviated throughout the notes, whereas for many others, the abbreviated forms are many and seemingly arbitrary. Most of the abbreviated words are easily recognizable, but for some of them, one has to take into account the context of the phrase to decipher them: "VLV" for "valve," "CHKD" for "checked," "S/V" for "serviceable." Along with the abbreviations, one can also find approximately 14% acronyms. Acronyms are lexical items generally agreed upon in the community and formed by the initial letters of a few words, like "FADEC" for "Full Authority Digital Engine Control." The head item of those words is most often a noun and consequently, acronyms are effectively used as nouns: they are used where a noun would be used, and are often marked with the plural "s."

One must also consider the typographical errors and the misspellings, which can be found in close to 17% of the "TEST" notes. Wrong characters, for example, "EMG" instead of "ENG" for "ENGINE"; inverted characters, for example, "VLAVE" instead of "VALVE"; characters in excess, for example, "SSERV" and "SER5V" instead of "SERV" for "SERVICEABLE"; skipped characters, for example, "RESITANCE" instead of "RESISTANCE" are examples of typographical errors. As for misspellings, one can find "IGNITER" versus "IGNITOR," "EVEDENCE" versus "EVIDENCE," "FLAGED" versus "FLAGGED," and many others. We can also find examples of a few consecutive words glued together, instead of being separated by spaces, as in "CALLUPPAGE" for "CALL UP PAGE," and examples of words which have inserted signs, as in "RE-CALL," "FURTH-ER," and "CHG;D" for "CHANGED."

At the syntactical level, things are worse. Whole categories of words are consistently missing. For example, one cannot find any of the articles "THE" and "A" (or "AN"). In a great number of sentences, the auxiliary verb between the subject and the past participle of the verb is not present. Similarly, the predicative verb "TO BE" is most often left out betwen the subject and the predicate. For example, "#2 EIU CHANGED" must be read as "(The) #2 EIU (has been) CHANGED"; also, "FADEC PWR BACK" reads as "(The) FADEC POWER (is) BACK." Normally those words would be there if the notes were written grammatically or spoken. In a fair proportion of notes, the subject of the sentence is omitted, likely because it is clear that the subject is the technician responsible for the repair. All this results in grammatical structures different from what is considered grammatical English. Table 1 summarizes the different grammatical sentence structures found in the test notes, after delimiting thoses sentences by hand. A noun phrase is a group of at least one word, a noun, with optional modifiers, like an article, one or more adjectives, noun premodifiers, and so forth. An adjectival phrase is a group of at least one word, an adjective, with optional modifiers, like an adverb, or a postpositioned prepositional phrase. A prepositional phrase is merely a noun phrase introduced by a preposition. A verb phrase is a group of words the head of which is a verb, accompanied by adverbs, auxiliaries, and complements.

Another characteristic is the deficient or absent punctuation between sentences. For example, in the following partial note: "FADEC PWR BACK FADEC 2B CLASS 1 MESS. ON UPPER ECAM," there are really 2 sentences, with no sign to tell the end of the first one from the beginning of the second, which should read as: "(The) FADEC POWER (is) BACK (.⟨end of sentence⟩) (A) FADEC 2B CLASS 1 MESSAGE (is) ON (the) UPPER ECAM." Actually, this problem in discriminating the sentences is made more challenging by an inconsistent and incoherent punctuation scheme. Many signs are used in many different ways that vary greatly from what we are accustomed to seeing and differ from one note to another. For example, one can

Table 1. *Proportional distribution of the propositional structures in a test set of FTRANs*

Sentence structures	% of total sentences
Noun phrase + Passive verb phrase ex.: "ignition lead b changed"	33.94%
Noun phrase + Adjectival phrase ex.: "anti ice valves operation normal"	5.74%
Noun phrase + Prepositional phrase ex.: "13 psi min allowable pressure within limits as per aom 01-70-08"	1.8%
Active verb + noun/prepositional/adjectival phrase ex.: "replaced tank drain valve"	21.96%
Adjectival phrase ex.: "normal as per amm"	12.11%
Simple noun phrase/Substantive ex.: "ground run"	9.85%
Negative noun phrase ex.: "no faults"	6.2%
Prepositional phrases ex.: "as per above comments"	0.5%
Noun phrase + Active verb phrase (complete) ex.: "motoring test and ground run check serviceable"	7.87%

find in the same note the sign "-" used as a sentence delimiter; a few words farther, the sign "." has been put at the end of the acronym "CFDS" as if it were an abbreviation by truncation; and a few words farther again, the sign "." is used as a sentence delimiter. This is shown in the following example: "#1 EIU RESET - ⟨end of sentence⟩ CFDS. FADEC TESTED - ⟨end of sentence⟩ NIL FAULT. ⟨end of sentence⟩ #1 THRUST REVERSER OPERATION NORMAL ON BOTH CHANNELS." In another note, the sign "-" is used like quotes: "#2 ENG -B- IGNITION U/S #110198."

3. SNAGIE

The goal of our system is to extract from free-text notes very specific information, namely the pieces of equipment involved in a repair job and the actions upon those parts. In that, it closely relates to the field of Information Extraction (IE). Effectively, the goal of an IE system is to identify references to the concepts of interest for a particular domain in an unrestricted text. Each domain needs a set of text extraction rules based on the vocabulary, semantic classes, and writing style peculiar to the domain and the target concepts (Soderland, 1996). There are basically two approaches to designing an information extraction system: the knowledge-engineering approach and the learning approach (Appelt, 1999). In the knowledge-engineering approach, a system developer, familiar with such systems, with natural language processing techniques, and with the domain, writes the text extraction rules based on his analysis of sample texts. On the other hand, in an opposite and complementary way, the learning approach requires that a domain expert annotate a large corpus of sample texts to mark the specific concepts of interest, and some machine-learning process automatically creates the text extraction rules from the annotated training text. This last approach was applied, for example, to a problem similar to ours in the medical world (Lehnert et al., 1994). Insofar as we could not have the help of a domain expert for annotating training text, our approach is clearly a knowledge-engineering approach. Moreover, the availability of electronic versions of domain-specific manuals and the many promising possibilities that this offered us in terms of sources of lexical and domain knowledge favored that approach. A thorough analysis of our corpus of close to 1000 repair notes showed recurrent grammatical patterns (cf. Table 1), which indicated the possibility of using natural language processing techniques to treat them and extract the desired information.

Because of the many peculiarities found in our analysis, such as the multiple meanings of the punctuation signs and of the words and the multiple possibilities for abbreviations, we chose Prolog (Sterling & Shapiro, 1994) as the programming language for our implementation, due to its ability to deal with heuristics, its backtracking facility, and its definite-clause grammar paradigm. Several modules, such as the lexicon, the morpholexical analyzer, and the grammar, are derived from the Alvey Natural Language Toolkit (ANLT; Carroll et al., 1991; Grover et al., 1993). Some have been merely translated from Alvey's original language, Lisp, to Prolog, whereas others have been directly created in Prolog. Figure 1 shows the functional diagram of

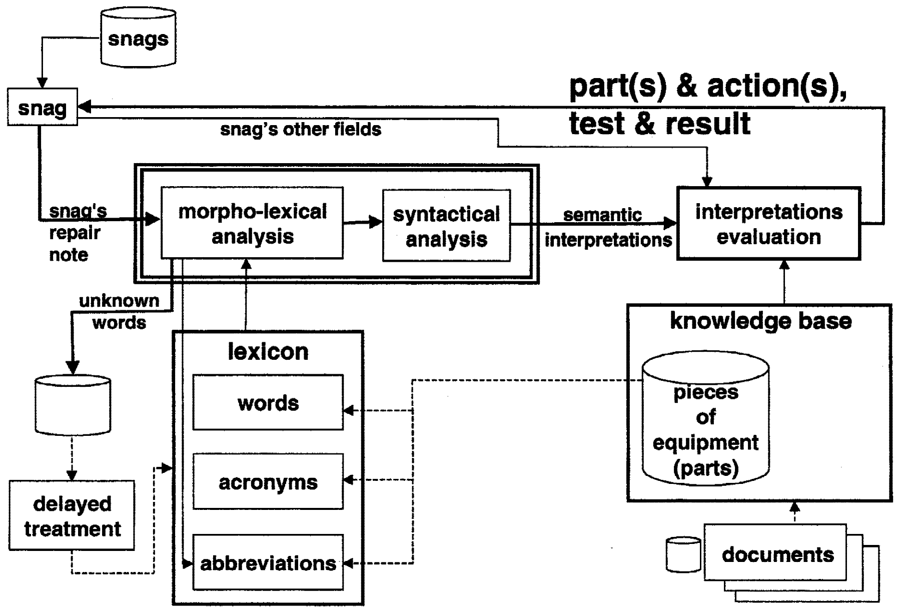

Fig. 1. Functional diagram of SNAGIE.

SNAGIE, with the four main modules: the lexicon, the knowledge base, the natural language processor, and the semantic interpretation evaluator.

The snags are stored as records in a database. Apart from the repair note, which is fed as input to the natural language processor, snag records have several fields which may be of help at the interpretation evaluation stage, namely: the snag's identifier, the FIN (Functional Item Number) aircraft number, the ATA (Air Transport Association) system and subsystem codes, the snag's description (the problem), the "continued from" and the "continued on" snag numbers, the repair note, the UNI (Unit Numerical Index) code of the part removed, and the UNI code of the part put back in.

3.1. The lexicon

The complete snag database contains 62,523 snags, and the snags' repair notes contain a total of 35,506 terms. After removing numbers, single letters, prepositions, acronyms, abbreviations, and misspellings to keep only nouns, adjectives, and verbs, we ended up with approximately 15,000 different words, ranging from common nouns and verbs like "GO," "DO," "FIND," "NOISE," and so on, to domain-specific words like "VALVE," "ACTUATOR," and "ENGINE." To parse the notes, make sense of them, and extract information about pieces of equipment and actions upon them, our system must know about the words and the terms they are composed of.

Our lexicon comprises three modules: the words, the acronyms, and the abbreviations. The word lexicon's main source is the ANLT dictionary, a substantial English dictionary derived semiautomatically with little manual intervention from the machine-readable version of a conventional learner's dictionary (Boguraev et al., 1987; Carroll & Grover 1989) that we translated from Lisp to Prolog and augmented with a few properties. It contains 28,240 nouns, 23,273 verbs, 9073 adjectives, 1069 adverbs, 1008 prepositions, modal and auxiliary verbs, pronouns, and various function words. A word may have several entries, one for each of its typical usages. Each entry is defined as a set of properties in the form of attribute-value pairs, like [plu,-] to indicate the singular and [per,3] to indicate the third person of a verb or a pronoun, supplemented with a term or an expression that is going to be used as its semantic expression. Actually, the lexicon consists of a set of source lexical files, where each entry is declared with a minimal set of properties. Each source file is then compiled, where each entry's set of properties is augmented with properties specific to the word's category and default values. Finally, the compiled lexical files are indexed for an easy and fast access to the words. Figure 2 shows an example of the source and compiled versions of the lexical entry of the word "VALVE" from the Alvey dictionary.

The basic acronym lexicon contains 458 acronyms from the Airbus General Information Manual, another 126 acronyms from the Airbus Minimum Equipment List Manual, and 70 acronyms from other Aerospatiale sources including Air Canada. The very great majority of the acronyms stand for a sequence of a few words, the main one being a noun generally in the final position of the sequence, like "HPV," which stands for "High Pressure Valve" and designates a valve. A few acronyms, though, do not follow that pattern, like "AOA," which stands for "Angle Of Attack": the head noun is not in the final position; or like "D/A" for "digital/analog," which is an adjective. So, in every acronym entry, we marked the head word, and when the acronym is looked up in the dictionary, it is returned with the properties of the head word's entry in the word lexicon. For example, the

	lexical data
word	valve
source properties	[[v,-], [n,+], [bar,0], [count,+]]
compiled properties	[[v,-], [n,+], [bar,0], [count,+], [compound,not], [lat,+], [at,+], [conj,null], [acr,-], [num,-], [demon,-], [timepn,-], [timetype,-], [part,-], [adv,-], [per,3], [nform,norm], [plu,-], [pn,-], [pro,-], [protype,none], [poss,-], [colour,-], [infl,+], [subcat,null], [fix,not], [coord,_], [refl,_], [prd,_], [case,_]]
semantic expression	valve

Fig. 2. Source and compiled lexical entry of the noun "VALVE."

acronym “HPV” is returned with the properties of the noun “VALVE,” with this difference, that the attribute “acr” now has the value “+.”

Abbreviations represent a complex problem. When an abbreviation is encountered in a repair note, the abbreviated word must be determined. The person who sees an abbreviation in a repair note recognizes the word immediately because he knows about the domain, and the language and the words of that domain. For example, “VLV” will immediately read “valve.” Similarly, when “CHGD” is encountered while reading “IGNITION LEAD B CHGD,” it will be read as “changed” even though the term might also be the abbreviation of “charged” or “challenged,” because he knows the word “change,” because he knows that an ignition lead is a part and that it can be “changed,” and because he has met the word “change” often in that context. The key here is knowledge and context: the recognition of the word is done through contextual knowledge.

The repair notes of the complete database of 62,523 snags do provide us with a contextual source of lexical knowledge. For example, in regard of “VLV,” if we look up every term starting with “V” and containing the letters “L” and “V” in that order, we find 18 terms, like “VLAVE,” “VALVA,” “VALVE,” “VLVE,” “VALVES,” and so on, with a total of 2304 appearances in the whole set of notes. Of those 18 terms, 1 is known in the word lexicon, “VALVE” (“VALVES” is recognized as the plural of “VALVE” by the morpholexical analyzer), which appears 2215 times, that is, it accounts for 96% of all appearances of the 18 terms. This makes “VALVE” a very good candidate for “VLV.” Similarly, 37 terms start with the letter “C” and contain the letters “H,” “G,” and “D” in that order. One of them, “CHANGED,” is recognized in our word lexicon, and this one accounts for 89% of all appearances of the 37 terms. This makes “CHANGED” a very good candidate for “CHGD.” We say “candidate” because we cannot be sure at that moment. Once the whole process has been completed, when the semantic evaluation has been done, a valid candidate will have resulted in the identification of some part and an action upon it. In that case, the abbreviations and the words they were assumed to abbreviate are added to the abbreviation lexicon.

The Alvey dictionary is general purpose and does not contain certain words specific to the domain, specially nouns designating pieces of equipment. A number of words in the repair notes are not contained in it, like “actuator” (586 appearances in all the repair notes), “connector” (272 appearances), “transceiver” (37 appearances), and “transducer” (193 appearances), all words more or less common in the repair notes, which means hundreds of notes that could not be treated.

The Airbus Illustrated Parts Catalog (IPC) is a catalog of all the parts of the particular type of aircrafts where the snags originated. It details every system and subsystem, installation and assembly. We will come back to the IPC in Section 3.2. The IPC also contains a list of all the part numbers and for each part number, a keyword designating the part. This is the PNR (Part Number) file. Logically, from this list, it is thus possible to draw a list of all the nouns that name parts. However, the keywords are seldom full words. Most of them are arbitrary abbreviations. To recover the full words, each part number was looked for in the IPC to pick the first item with that part number, and the textual description of that item was analyzed to find a word corresponding to the keyword associated with the part number. For example, the part number “30001-01-111,” associated with the keyword “ACCLRMTR,” can be found in the IPC with the description “ACCELEROMETER-THREE AXES.” The word to the left of the hyphen in the part descriptions is generally the main noun describing the part; the words to the right further describe the part. In this example, we have a three-axis accelerometer, that is, an accelerometer further determined with the characteristic of having “three axes.” We compare the keyword with the main noun, here “ACCLRMTR,” with “ACCELEROMETER,” looking for a clear correspondence between the two terms. Then, on the basis that the part descriptions generally contain full words and not abbreviations, the main word, here “ACCELEROMETER,” is looked up in the dictionary as a noun. If it cannot be found, it is added as a noun. It may also happen that the keyword is an acronym, found as the main word of the part description, with its development on the right side, like “ACP-AUDIO CONTROL PANEL.” In such cases, if the acronym cannot be found in the acronym lexicon, it is added; the head noun will be marked later, as was done for the other acronyms. This way, 350 new nouns, 38 new acronyms, and 341 abbreviations were found. Those abbreviations may be of little use, though, because of their arbitrary nature.

3.2. The knowledge base

Since we are looking for the piece of equipment involved in a repair and what was done to it, we need to know how those parts are referred to, that is, we need to know the nouns used to designate the parts. In our processing of the PNR file for collecting new words unknown to our general lexicon (discussed in the preceding section), every main noun and acronym associated with a part number and its keyword is compiled into a list of part names. This way, when a repair note says “REPLACED VALVE,” its interpretation evaluation may establish immediately that the topic “valve” is a piece of equipment. But we need more than that. In the following repair note, “REPLACED TANK DRAIN VALVE,” we have a cluster of three nouns, of which the grammar of English says that the last one is the head noun (we are talking here of a “valve”) and the others determine it. But how do they determine it? Does “TANK” modify and determine “DRAIN,” or does it modify and determine “VALVE”? The answer to this problem lies in the knowledge of the domain.

As mentioned in Section 3.1, the IPC describes the component makeup of an airplane. Each item is described with its part number, the vendor number, a textual description, the number of such items in the assembly, the FINs (Functional Item Number, from Airbus) of each copy of the item in the assembly, its hierarchical rank as a component of some other item or assembly, the range of A320 models where the part can be found, and when applicable, information about the interchangeability of the part with other part numbers. From the SGML version of the IPC, we have created a DB2 relational database of about 1.4 Gb, 392 tables for the 32,768 items of the IPC, the 51,319 part numbers of the PNR file, and their data.

The IPC's items have textual descriptions, and the subsystems and the systems are described within sections and chapters, which have titles. So, effectively, we have here a textual hierarchical representation of the aircrafts. The words of a compound noun can be looked for in the items' descriptions and, going up the hierarchy, the sections' and chapters' titles; the locations where the words are found can tell us how the words relate to one another. As an example of this, we will follow the evaluation of the interpretation of "TANK DRAIN VALVE" in the repair note "REPLACED TANK DRAIN VALVE," where "TANK" determines "VALVE." In agreement with the English grammar, in compound-noun constructions, the second to the last noun always modifies and determines the last noun; the first thing then is to look for items the description's main word of which is "VALVE" and that relate somehow to "DRAIN," with this term either in the description itself of the items as a secondary word, or in the description of higher-rank items, or even in the title of the section or the title of the chapter where the items are described. We do not need to span the whole IPC since the value of the "ATA System" field of the snag, 38, tells us in which chapter of the IPC to search the items. So, in chapter 38 (WATER/WASTE-GENERAL), there are 28 valves mentioned with a "drain" relationship. Of these 28 items, 2 are found to be components of the immediate higher-rank item the textual description of which is "WASTE DISPOSAL LINE-WASTE WATER TANK,Z170" found on Airbus A320 aircrafts 201 to 234, and two others as components of the immediate higher-rank item with the same description but for Airbus A319 aircrafts 251 to 285. According to the general structure of IPC item descriptions, the higher-rank item can be told as "the waste disposal LINE of the waste water TANK," and so, we have a connection between "TANK" and "VALVE" because in the end, the valve is a component of the tank. Since the "Aircraft Number" field of the snag has the value "206," only the two first mentioned are relevant, and since the two are described as mutually interchangeable, we end up with one item only.

3.3. The natural language processor

Prior to the syntactical analysis, the repair note is passed through the morpholexical analyzer, where the terms of the note, that is, its words and signs, are determined. Words are first examined one at a time by a morphological analyzer for specific morphological markers, for example the noun plural "s" or the verb past "ed." They are then looked up in the lexicon to find out, if they exist, what kind of words they are: nouns, adjectives, verbs, adverbs, prepositions, articles, and so forth or acronyms or abbreviations. It is possible for a word to be more than one lexical item and so, all possibilities are returned. The morphological analyzer has been built according to the description of the spelling rules of the ANLT morphological analyzer (Ritchie et al. 1987). For example, the word "TESTS" is analyzed as "TEST" and "+S." When looked up in the lexicon, "TEST" returns as a noun and as a verb, "+S" returns as a noun suffix and as a verb suffix, and their combinations result in the two following possibilities: a plural noun and a third person present tense verb.

Once all the words have been determined, they are fed to the syntactical analyzer, a bottom-up chart parser based on that proposed by Gazdar and Mellish (1989) and adapted to run with our grammar formalism and implementation. The parser looks for specific associations betwen the lexical items according to the grammar. The grammar describes the syntactical sentence structures of the repair notes as shown in Table 1, and substructures like noun phrases, verb phrases, and so forth. It is based on the ANLT grammar formalism, which is similar to that of the Generalized Phrase Structure Grammar (GPSG; Gazdar et al., 1985), details of which are discussed in Boguraev et al. (1988), Briscoe et al. (1987*a*, 1987*b*), and Carroll et al. (1991). Each constituent of a rule, the mother as well as the sisters, consists of a list of attribute-value properties, following the same paradigm as the property lists of the lexical entries. Figure 3 shows an example of a rule in our system as it is written in the grammar file. Like the lexical entries, only those properties which are necessary to express the rule are specified. Other properties will be assigned by default during the grammar compilation.

This rule recognizes noun phrases like "#1 EIU." It says that a term (cat 1) with the property [sign,nbr] (this is how the sign "#" is defined in the lexicon), followed by a term (cat 2), which is syntactically an intermediate-level adjective structure with the property [num,+], which is what a mere number like "1" is, followed by (cat 3), an intermediate-level noun structure, which an acronym like "EIU" can be, is to be considered a numerically premodified intermediate-level noun structure (cat 0).

In addition to checking for valid grammatical associations, the syntactical analyzer generates an analysis tree representing the hierarchical links between the syntactical structures. In fact, because words may have different meanings with several lexical entries, and because a set of words may associate in different ways for different interpretations, there may be more than one analysis tree. In the case of the example "#1 EIU," our grammar generates two different analyses: in the first one, "EIU" is a noun premodified by a numeric identifier, as parsed by the rule shown in

rule	N1/IDNUM2_PRE/+
source declaration	regle_gram('n1/idnum2_pre/+', [cat(0,n1([num,+],[mod,pre],[modkind,ident])), cat(1,[[sign,nbr]]), cat(2,a1([num,+],[agr,_])), cat(3,n1([num,_],[modkind,none]))], [[[], [lambda,[x],[and,[3,x],[ident,2,x]]]]]

Fig. 3. Example of a grammar rule as declared in the grammar file.

Figure 3; in the second one, "EIU" is a noun premodified by a special type of definite noun phrase which has the form of a numeric identifier, for a pattern like "#1" in our context may stand by itself for some undefined but definite object. Figure 4 shows the syntactical analysis tree of the first interpretation. Each node is the name of a grammar rule and defines a grammatical structure level, from the top level to the words.

Each lexical item in the lexicon is defined with an expression, most often a symbol, that stands as its semantic interpretation. Likewise, each grammar contains a semantic expression, in the form of a lambda-calculus (λc; Dowty et al., 1981) logical expression which is the combination of the semantic λc expressions of its syntactical constituents. That λc expression conveys how the structure defined by the grammar rule must be interpreted semantically. As the syntactical analyzer parses a repair note, the λc expressions of each rule's constituents are combined by means of rule-to-rule λc translations. Eventually, an unscoped event-based (Davidson, 1967; Parsons, 1990; Alshawi, 1992) logical form is generated which is the representation of the compositional semantics of the input sentence. For example, [lambda,[x],[and,[3,x],[ident,2,x]]] is the λc expression of the rule N1/IDNUM2_PRE/+ (cf. Fig. 3) that the rule will contribute to the overall semantic expression. The "2" is replaced by the semantic expression of the second element (the numerical adjective, cat 2) and the "3" by the semantic expression of the third element (the noun, cat 3). Figure 5 shows the semantic expression of the first interpretation of the noun phrase "#1 EIU." This expression says that there is a definite object X12 which has the properties of being a "EIU" and of being identified by the value "1."

3.4. The semantic interpretation evaluator

The semantic expressions generated by the syntactical analyzer for a repair note are passed to the semantic interpretation evaluator. This is where the information for which

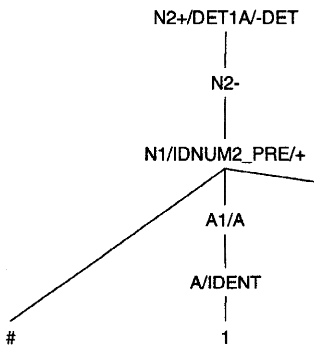

Fig. 4. Syntactical analysis tree of one interpretation of the noun phrase "#1 EIU."

phrase	#1 EIU
semantic expression	[dd, [the, [x12], [and, [sg,x12], [and, [eiu,x12], [ident,1,x12]]]]]

Fig. 5. Semantic expression of one interpretation of the noun phrase "#1 EIU."

phrase	#1 EIU RESET
semantic expression	[decl, [some, [x11], [entity,x11], [the, [x12], [and, [sg,x12], [eiu,x12], [ident,1,x12]], [some, [e3], [pres,e3], [reset,e3,x11,x12]]]]]

Fig. 6. Semantic expression of the sentence "#1 EIU RESET."

we are looking is to be determined. The meaning of the λc expressions is assessed by evaluating the validity, or correctness, of their assumptions. Figure 6 shows the semantic expression of the sentence "#1 EIU RESET" with the first interpretation of the noun phrase "#1 EIU" as described in Section 3.3.

This expression can be paraphrased as the following declarative statement: "There is some entity X11, such that there is the definite single object X12 which is a EIU and which is identified by the value 1, such that there is some event E3, such that the event is a RESETTING by the entity X11 of the object X12," or in other words: "Someone has reset the eiu no. 1." The semantic evaluator goes down this expression and tries to assess its meaning by determining the objects. When an expression like [the, [x12], [and, [sg,x12], [eiu,x12], [ident,1,x12]], ⟨...⟩] is encountered, each part is evaluated. For the structure [the,⟨variable⟩,⟨properties⟩,⟨facts⟩], one looks to find one or more definite objects with the given properties for which the facts have been stated. In our example, the evaluation of [eiu,x12] finds out from the list of part names that "eiu" names a piece of equipment, and retrieves from the IPC database, the airplane's equipment list, all the items which are EIUs in the appropriate system/chapter. Two items are returned. But their part numbers are defined as interchangeable, so one can keep only one of them. The evaluation of [ident,1,x12] checks the items' properties and keeps only those parts which can be identified by the number 1, that is, only those parts for which there are more than one in the assembly or installation. The item that was retained is described as there being two copies of it, and so it is still retained. Finally, the interpretation of [sg,x12] in the context of the formula [the, ...] says that there should be only one such object left, which is the case. Thus we have identified an item from the evaluation of the ⟨properties⟩ part of the [the, ...] expression, which gives it a meaning. Now, the ⟨facts⟩ part must be evaluated. Since "reset" is a transitive verb with the part "eiu" as its direct object, we can infer that this is an action done upon its object. And so, we have a full meaning for the semantic expression of the repair note. We have found a piece of equipment and an action upon it. The data related to that piece of equipment in the IPC database gives us all the information we want about that piece. If anything could not be assessed, the interpretation is put aside as meaningless and the next one, if any, is examined. Tables 2 and 3 summarize the processing steps and results from the input of the repair note "#1 EIU RESET" to the output of the sought information.

4. PORTABILITY

Like any knowledge-engineered system, SNAGIE is somewhat customized to the target problem. Our approach is based on knowledge engineering, and in order to apply our method to another domain, some knowledge-engineering work would have to be done to adapt some of the modules to that domain. The first criteria of success for portability would be that the new target problem be of the same nature as ours. Our approach will work more easily if applied to repair notes dealing with parts and maintenance actions on those parts. For example, we are looking at the possibility of applying our approach to the extraction of information from maintenance notes for a fleet of trucks and shovels in a huge open-pit mine in the north of Canada. Linguistically speaking, the same grammar rules could be used, probably with some adjustments. From the lexical point of view, domain-specific vocabulary and tokens with peculiar structures have to be identified and defined. For example, every domain has its own types of alphanumerical identifiers and idiosyncracies that have to be inventoried, and recognition rules created for them. Another important criteria is the availability of electronically formatted documents, troubleshooting manuals, parts catalogues, and so forth, where knowledge of the domain can be found and used. In our approach, those constitute an essential source of domain-specific vocabulary, part names, part numbers, semantic classification of the domain concepts, and the various relationships between them. From a semantic point, the semantic expressions generated by the parser are domain independent and the semantic evaluation process over those expressions is also domain independent, to a certain extent because at the ground level, the domain concepts and their relationships are necessarily expressed differently from one domain to the other and from one document to the other. What is important here is the methodology, that is, to identify clearly the knowledge-engineering tasks and steps, the sources of knowledge and their processing.

Table 2. *Summary of the syntactic analysis of the sentence "#1 EIU RESET"*

Input sentence	"#1 EIU RESET"
Syntactical analysis tree	1. t1
	2. s1a
	3. n2+/det1a/-det
	4. n2-
	5. n1/idnum2_pre/+
	6. #
	7. a1/a
	8. a/ident
	9. 1
	10. n1/n
	11. eiu
	12. vp/be_prd/-be
	13. prd4
	14. vp/np(passive/-)
	15. reset

5. CONCLUSION

The repair notes from aircraft maintenance technicians are written in a fairly telegraphic format in an English-like language where several types of words are missing: auxiliary verbs, articles, and so on, where the lack of punctuation makes it very difficult to tell sentences apart, where the abbreviations and the acronyms, not to mention the typographical errors, make for more than 30% of all the words on average, and where idiosyncracies are frequent. Yet a deep analysis of several hundred notes shows recurrent patterns which can be described in a grammatical form, allowing the notes to be parsed. The parsing rewrites a note into a set of unambiguous semantic expressions, the meaning of which can be assessed by finding relationships between the objects they suggest the existence of and the airplane parts described in a relational database, and by inferring actions upon them from the lexical and syntactical properties of the verbs.

We have succinctly described the type of language of the repair notes, and presented the four main modules of SNAGIE, our natural language information extraction system for processing those notes: the lexicon, the knowledge base, the syntactical analyzer, and finally the semantic interpretation evaluator. Through the description of each module, we have shown with examples how the free-text notes are processed, from the morpholexical analysis to the semantic expression's evaluation, with an aim at finding what pieces of equipment were involved in a repair and what was done to them. We have developed a basic implementation of the four modules which demonstrates the feasibility of our approach.

Still, much work remains to be done. The lexicon is fairly complete. Yet we intend to process a number of other documents, namely the SGML version of the Trouble Shooting Manual of Airbus and the Unit Numerical Index file from Air Canada, with an aim at discovering other nouns for part names, and links between UNI numbers and Airbus part numbers. We also intend to add a facility that will pick the unknown words during the morpholexical analysis and save the words and all the information pertaining to the repair notes where they were found, such that after the off-line definition of those words by the system manager, the notes will be analyzed again. The morphological analyzer and the parser are fully operational. The grammar remains to be completed. For the moment, it contains the set of rules that are needed for treating the kinds of sentences given as examples throughout this article. As for the knowledge base, we have to complete an index table that links the main nouns of the items' descriptions in the IPC to secondary words and to the chapters' and sections' titles. The implementation of the semantic interpretation evaluator is a con-

Table 3. *Summary of the semantics and results for the sentence "#1 EIU RESET"*

Semantic expression			[decl, [some, [x11], [entity,x11], [the, [x12], [and, [sg,x12], [eiu,x12], [ident,1,x12]], [some, [e3], [pres,e3], [reset,e3,x11,x12]]]]
Results	**Part**	**Name**	EIU
		Number	3957900612
		FIN	1KS1
		Chapter	73
		Section	25
		Unit	8
		Figure	2
		Item	40D
		Sheet	N273250802
	Action		reset

tinuing process, along with that of the grammar, and its capabilities will grow as we add new grammar rules with new semantic expressions.

A validation process has also to be implemented. This process will be twofold. When the system modules have been developed at a more operational level, a test group of several hundred notes will be fed to our system and the results evaluated by the knowledge engineer and system developer manually. This is part of the knowledge-engineering process. That will allow us to pinpoint where the system fails and misses and where to bring corrections. When the system is decided operational, with an acceptable percentage of failures and misses, the resulting extracted information will have to be validated by domain experts before it is used in cases for diagnosis and repair suggestions.

REFERENCES

Alshawi, H., Ed. (1992). *The core language engine*. Cambridge, MA: MIT Press.

Appelt, D.E. (1999). Introduction to information extraction. *AI-Communications 12(3)*, 161–172.

Boguraev, B., Briscoe, E., Carroll, J., Carter, D., & Grover, C. (1987). The derivation of a grammatically indexed lexicon from the Longman Dictionary of Contemporary English. *Proc. 25th Annual Meeting of the Association for Computational Linguistics*, 193–200.

Boguraev, B., Carroll, J., Briscoe, E., & Grover, C. (1988). Software support for practical grammar development. *Proc. 12th International Conference of Computational Linguistics*, 54–58.

Briscoe, E., Grover, C., Boguraev, B., & Carroll, J. (1987*a*). Feature defaults, propagation and reentrancy. In *Categories, Polymorphism and Unification* (Klein, E., & van Benthem, J., Eds.), pp. 19–34. Centre for Cognitive Science, University of Edinburgh.

Briscoe, E., Grover, C., Boguraev, B., & Carroll, J. (1987*b*). A formalism and environment for the development of a large grammar of English. *Proc. 10th International Joint Conference on Artificial Intelligence*, 703–708.

Carroll, J., Briscoe, E., & Grover, C. (1991). *A Development Environment for Large Natural Language Grammars*. Technical Report No. 233, Computer Laboratory, University of Cambridge, UK.

Carroll, J., & Grover, C. (1989). The derivation of a large computational lexicon for English from LDOCE. In *Computational Lexicography for Natural Language Processing* (Boguraev, B., & Briscoe, E., Eds.), pp. 117–134. Harlow, UK: Longman.

Davidson, D. (1967). The logical form of action sentences. In *The Logic of Decision and Action* (Rescher, N., Ed.), pp. 81–95. Pittsburgh, PA: University of Pittsburgh Press.

Dowty, D., Wall, R., & Peters, S. (1981). *Introduction to Montague Semantics*. Dordrecht, Germany: Reidel.

Farley, B. (1999). From free-text repair action message to automated case generation. *Proc. 1999 AAAI Spring Symposium: AI in Equipment Maintenance Service and Support*. Technical Report SS-99-04, Menlo Park, CA: AAAI Press.

Gazdar, G., Klein, E., Pullum, G., & Sag, I. (1985). *Generalized Phrase Structure Grammar*. Oxford, UK: Blackwell.

Gazdar, G., & Mellish, C. (1989). *Natural Language Processing in Prolog. An Introduction to Computational Linguistics*. Addison-Wesley, Reading, Massachusetts.

Grover, C., Carroll, J., & Briscoe, E. (1993). *The Alvey Natural Language Tools Grammar* (4th release). Technical Report 284, Computer Laboratory, Cambridge University, UK.

Lehnert, W., Soderland, S., Aronow, D., Feng F., & Shmueli, A. (1994). Inductive text classification for medical applications. *Journal for Experimental and Theoretical Artificial Intelligence (JETAI) 7(1)*, 49–80.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. Cambridge, MA: MIT Press.

Ritchie, G., Black, A., Pulman, S., & Russell, G. (1987). *The Edinburgh/Cambridge Morphological Analyser and Dictionary System*. Software Paper No. 10, Department of Artificial Intelligence, University of Edinburgh.

Soderland, S.G. (1996). *CRYSTAL: Learning Domain-specific Text Analysis Rules*. Technical Report TC-43, Center for Intelligent Information Retrieval, University of Massachusetts.

Sterling, L., & Shapiro, E. (1994). *The Art of Prolog*. Cambridge, MA: MIT Press.

Wylie, R., Orchard, R., Halasz, M., & Dubé, F. (1997). IDS: Improving aircraft fleet maintenance. *Proc. 14th National Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence (IAAI-97)*, 1078–1085.

Benoit Farley has a Master’s degree in Applied Sciences from Université de Sherbrooke, Province of Québec, Canada. As a research officer at the National Research Council of Canada, he has worked successively in Computer Assisted Learning, on Natural Language Front Ends to Expert-Tutoring Systems, on a Natural Language Interface to a 3-D visualization system, and in Natural Language Information Extraction. One of his main interests lies in the determination of information from diverse electronically formatted documents, and its utilization in practical natural-language-based systems.