Abstract: Natural Language Requirements are widely used in the aerospace industry, at least as the first level of description of a system. This paper compares some approaches in lexical analyses as a part of natural language processing applied to the analysis of requirements.

Keywords: natural language processing, requirements, NLU.

1. INTRODUCTION

Natural Language Requirements are widely used in the aerospace industry, at least as the first level of description of a system. Using formal language to specify requirements, despite a profit in the absence of ambiguity inherent in natural language, raises other problems.

The process leading to requirements of quality begins with the analysis of requirements written in natural language and continues with its formalization and verification. On the way to reduce such ambiguities are used standards for writing requirements.

NLP is a field of artificial intelligence that studies the problems of generation and automatic understanding of natural languages. More specifically, in this work, the statistical approach to natural language processing. The statistical NLP is multidisciplinary and deeply related to Linguistics, Statistics and Computer Science.

Evading discussions about what is the understanding of a language, NLP tries, in the context of this work, understanding natural language, at least enough for a particular purpose. The understanding of natural language (Natural Language Understanding - NLU) is a subarea of the natural language processing which studies the problems of automatic natural language understanding. Language understanding systems convert natural occurrences of human language into formal representations.

Classically, the first step of NLP program is the preprocessing of the text and the lexical analyze. This work compares different approaches in lexical analyses, component of natural language processing (NLP), applied to the analysis of requirements. The lexical analysis provides some innovative approaches, and produces good results compared with the specialized literature. For practical purposes, lexical analysis does not provide any expressive result. However, as a first step of a NLP, this step is fundamental due bad results at this step are transmitted to later steps, growing the errors exponentially. (1)

2. PURPOSE

One of the objectives of DO-178B provides that the requirements should be consistent and accurate. These characteristics are defined in Table A3 - objective 2 and table A4 - objective. This process requires the specification of requirements to increase the rigor in the process.

In the performance of a process aimed at the certification of an aircraft, there is a need for direct management of requirements. During the lifecycle of a requirement, huge efforts are made in order to follow this purpose. In the development of software, the management requirement has a fundamental role because it is crucial for certification’s purpose as to reduce the whole development effort, with the early errors detection.

Despite the standardization of writing requirements, much effort is required to ensure the quality of specifications. This happens primarily through meetings of requirements validation. However, in more complex systems the number of requirements rises to astronomical proportions. By the very nature of the problem, the greater the number of requirements more difficult is the task of verification of requirements. The experience shows that during long validation meetings, the results have a tendency to, over the time, to decreasing the accuracy, to validating requirements for a more detailed assessment would be rejected.

Similar problem occurs in case of changes in requirement. In the process, there is traceability between requirements of different levels. But there is no traceability between requirements correlated belonging to the same level. This causes another problem: when modifying a requirement may be conflicting with another of the same level. It is hoped that this problem is solved through manual verification. However, the same as during the validation, the growing number of requirements leads to greater effort and possibly errors.
Given this context, it is evident the need to optimize the writing and validation of requirements. Several approaches are possible for this problem, since improvements in the process of writing the training requirements of people.

One possibility would be to use some software that somehow assists in writing or in the validation of requirements.

In short, the process of requirements development in the airline industry demands great efforts. In order to explore the implications of creating a software to assist in this process, an application was developed as a proof of concept that anticipates problems and solutions in developing an enterprise application that helps write requirements, find related requirements (traceability between requirements of a same level) and be able to detect inconsistencies between those requirements.

3. METHODS

1.1. Pre-processing

1.1.1 Token

Token is a set of characters that form a linguistic unit with meaning. Normally one of the first tasks performed in the pre-processing of a text is tokenization, which identifies and separates expressions that appear in the text separated by spaces, commas, periods, etc. Specifically, it is the process of grouping characters, forming words or punctuation marks. Each group of characters obtained is called a token. The single character is dispensed into space. A very simple method for tokenization of a text is to divide the text with blanks. You can do this via the NLTK command re.split (r'\s+', text). Tokenize criterion for division of the string only in the blanks to produce results like 'flap actuator's' has not the same meaning that 'flap actuator' or 'herself,' A technique with better results and is easy to use tokenization with regular expressions.

In many NLP problems, it is common to find textual entries where there are several situations, for example: repeated scores (,,), abbreviations (Calif.) etc. The present study assumes that the requirements levied by the pre-word processing, are previously in a certain pattern. Therefore, the task of tokenization in this work is not very complex.

1.2. Lexical Analysis

The lexical analysis within the context of language processing is to connect each word with its corresponding label in a dictionary. However, many words have more than one meaning, which may make it impossible to choose the correct meaning of the word considering only the highlighted word in its context.

A morphological analyzer identifies words or phrases in one sentence alone, marking each with a token symbol. This process is aided by delimiters (punctuation and blanks), that are recognized in the stage of preprocessing. The tokens identified are classified according to their use (grammatical class).
You can cluster words in a language class who has similar behaviors syntactic and semantic types. These groups of words are called syntactic classes ("parts of speech"). The peculiarity of morphology is to study the words by looking at them individually and not in the context of the sentence or period. The most common test for whether two words belong to the same syntactic class is the replacement, as shown in Figure 2.

Figure 2 - Example of substitution of words in the same syntactic

In this context, an instance of a word in a grammatically valid sentence can be replaced by another of the same grammatical class, maintaining the validity of the sentence. Within the same class of words, there are groups of rules that characterize the behavior of a subset of words from one language (e.g. formation of the gerund in English finish with ing). Therefore, the morphology is the words on its structure, form and classification, with respect to each of the types of words.

The morphological analyzer is the key to understanding a sentence, as to form a coherent structure, it is necessary to understand the meaning of most of his words forming. For a precise meaning, it would be necessary to know all the words. However, due to the ambiguity and redundancy of natural languages it is possible for some natural language applications to accept unknown lexicons without significant decrease in its understandability.

The lexical analysis can have several applications. Among them, helps to predict the function of grammatical words initially unknown. For example, if a text in English has the word "scrobbling", is possible infer that it is a verb with a root scrobble. In other classes of problems, lexical marking is also useful in the synthesis and speech recognition; where without proper marking would be impossible to define the correct pronunciation.

1.2.1 Statistical Inference

The statistical NLP depends on the statistical inference for natural language. The statistical inference is the act to collect data from some phenomenon (with an unknown probability distribution) and inference of its probability distribution. For example, it can be from multiple instances of sentences to predict the probability that the next word is a given word.

1.2.2 Reliability X Discrimination

Typically, in an attempt to infer a characteristic of a model is used others characteristics that allow the inference. That is an a priori concept: the past behavior is a good guide of what will happen in the future. To be exact, it is assumed that the model is stationary. This assumes that the data set is grouped into classes of similar conduct, and by doing so we are grouping the data into classes of equivalence, using these classes to predict the value of a given data. This means that it is implicitly made an assumption of independence, the data of a class are so weakly associated to other classes that is possible overlook this interrelationship. However, this task of division into classes of equivalence, despite generating more specific classes should not be so detailed as to compromise the reliability of the sample. Therefore, the compromise between a sufficiently large number of samples and an adequate specificity that allows a satisfactory prediction. (2)

1.2.3 Markov model

The task of predicting the next word may be presented as an attempt to estimate the probability function P:

\[ P(w_n | w_1, \ldots, w_{n-1}) \]

This stochastic problem uses the classification of the previous word to predict the next word. After examining a large amount of text, it is possible to know which words tend to follow which.

For this task is impossible use a single word of a specific text. In general, the input is a text that was never read before, and there is not a prior identical text that can be taken as a basis for prediction. One possibility is to assume that a sequence of markings can be modeled as a Markov chain.

A Markov model of marking the search words in a string of markup text as a Markov chain. A Markov chain is a type of stochastic process in which the behavior of a random variable at a given moment is independent of its history. The transitions depend only on the current state, and these relationships remain stable over time.

Formally, a Markov model can be defined as:

\[ P(X_{i+1} = t^j | X_1, \ldots, X_i) = P(X_{i+1} = t^j | X_i) \]

Onde,

\[ X_i = \text{Current State (present Tag)} \]
1.2.4 Collocation

regarded as the most successful. (2)

the most widely used techniques in this field and are broadly statistical method of modeling used in the moderns systems trigram for n = 2 and 3 respectively.

know: why we say a stiff breeze but not a stiff wind…” (2)

explainable patterns of word usage that native speakers all require intermediate understanding of some structures reason, much research in NLP has focused on tasks that

inherent in the language without requiring their complete markings to mark the next word is a model of order n -1, also called the 3 -gram. N-gram is usually called bigram and called an n-gram model. For example, a model of order 2 is also called the 3-gram. N-gram is usually called bigram and trigram for n = 2 and 3 respectively.

Hidden Markov models (HMM) have been a chief statistical method of modeling used in the moderns systems of NLP. Despite their limitations, variants of HMMs are still the most widely used techniques in this field and are broadly regarded as the most successful. (2)

1.2.4 Collocation

The term "collocation" is defined as an expression consisting of two or more words that correspond to some conventional way of saying things. Are sequences of words or terms that are used more frequently than other words at random.

“Collocations include noun phrases like strong tea and weapons of mass destruction, phrasal verbs like to make up, and other stock phrases like the rich and powerful. Particularly interesting are the subtle and not-easily-explainable patterns of word usage that native speakers all know: why we say a stiff breeze but not a stiff wind…” (2)

Certainly, the easiest way to find placements in a text is counting. If two words occur together rather often, is an indication that they have a special function that is not completely explained just by the function resulting from their combination.

Just chose the most frequent tuples of tokens can be not interesting so the most frequent usually are not a collocation. However, it is possible to improve results by using a simple heuristic: passing through a filter, the sequence of words, which is expected a particular pattern of lexical groups, make possible excellent results.

1.2.5 Pos Tagging

The ultimate goal of research in natural language processing is syntactic analysis and understanding of language. However, we are still far from this goal. For this reason, much research in NLP has focused on tasks that require intermediate understanding of some structures inherent in the language without requiring their complete understanding. One of these tasks is the Tagging.

The process of classifying the words of a sentence into its part of speech (morphological classes) and to label them

is known as part-of-speech tagging (POS tagging), or simply tagging. Decide for each word whether it is a verb, noun, adjective or something else. An example of a sentence with their tags:

[(The AT), (representative, JJ), (put, NN), (chairs, vbz) (on, IN) (the AT) (table, NN) (...)]

However, this tagging leads to a semantic understanding incoherent. This tagging is also syntactically unusual, because the use of "put" as a noun and "chairs" as intransitive verb is rare. This example shows a case with limited syntactic disambiguation. POS Tagging attempted to determine which of syntactic categories is most appropriate for the private use of the word in the sentence.

The tagging is more complicated than just from a list of words, to associate each word with its corresponding tag; as some words can represent more than one word class. This is common in natural languages (different to the formal languages), a large percentage of inflections of words are ambiguous.

The example above is intentionally created to show certain peculiarities of POS Tagging. However, the task is significantly simpler than parsing a complete and relatively little effort you can get a high hit rate. About 96% of the tokens correctly disambiguated in most common approaches. (3) On the other hand, it is important to consider that in texts with sentences with many words that can mean an average of almost one error per sentence.

Despite the limited information that she provides in this step, it is essential. For the tagger results is used to perform shallow parsing (Partial Parsing). The insight is to use the tagging as an intermediate layer of representation that is useful and more tractable than a full parsing.

1.2.6 Sources of information for a tagging

How is it possible to decide which part of speech belongs a word? There are essentially two sources of information. The first source of information is look for the word itself. The word itself contains morphological information. Through a corpus can get statistics of their lexical classification. Some words have only one grammatical class, which already solves the problem. The words that have multiple parts of speech can be disambiguated by the frequency of use of their possible classes. Using only information enclosed in the word itself, out of context provides a tagger with about 95% accuracy.

The second source of information is to look at the tags in other words in the context in which that word is used. Words can be ambiguous as to their grammatical class, but it is important to note that some sequences of parts of speech are common while others are unusual or non-grammatical. For example:

AT-JJ - NN (article - adjective - noun) is common.
AT-JJ - VBP (article - adjective - verb) is unusual.

When choosing the tag for the word "play" between NN and VBP in the phrase "the new play" should be chosen for training more common.
1.2.7 Unknown words - Default Tagger

Above it was discussed how to estimate the probability occurrence of a word in a text, based on a corpus already known. Nevertheless, several words in texts that are analyzed are not present in a corpus already studied. Some might never be found in a dictionary. According to (4), the biggest problem of tagging, in practice they are unknown words. Dissimilar rates of success, at lexical analyze, for different texts from many corpus comes mainly from the proportion of unknown words. The final results are strongly influenced by the ability of the algorithm of guessing the correct part of speech of a word.

The first approach is to check mark all the words with the same tags. This step down despite a trivial milestone. In an attempt to obtain better results, choose the tag that represents the most words in case the tag noun (NN). The marking pattern as a noun contains a key to their choice, the language is mutable emerging new words all the time, yet its structure remains, ie, prepositions appear and disappear at a much slower speed than the nouns for example. Therefore, when we face with unknown words is likely to be a noun.

It is possible to enhance this approach. Often is used morphological information and other clues to infer that the best possible part of speech it belongs to the unknown word. For example, in English, words ending in -ed is possibly a conjugated verb in the past or past participle (past tense and past participles). Verbs ending with -ing indicates that the unknown word probably expresses the idea of continuation, incomplete action, or gerund.

1.2.8 Unigram tagger

Unigram tagger is based on a simple statistical algorithm: for each token defines the tag that most frequently occurs at a referenced corpus. For each token there is a base frequency of use of tags associated with the token. Simply, is adopted as the correct token, the most common. Recalling the discussion on information sources for a tagging; the unigram tagger is exactly this approach.

1.2.9 N-Gram tagger

To accomplish the task of tagging based on a unigram, is taken into account only the word out of context. From this model, the best that can be done is to mark each word with the most frequent tag for that word.

An n-gram tagger is a generalization of a unigram tagger in which the context of analysis is the most current word tags of preceding words, as shown in Figure 3.

Where:
- \( t_n \) = tag to be chosen
- \( w_n \) = current token
- \( t_{(n-1)} \) = token on the previous word

The n-gram takes into account n-1 tags of preceding words. For example, a 3-gram considers the tag of the two preceding words and the current word. Building a model where all the database is formed by n-1 words in the same equivalence class, means a Markov model of order n-1 or a n-gram model of order n. The n-gram tagger uses the parts of speech more frequent in a corpus, in each context, to define the most appropriate tag. Or 1-gram is another term for unigram. The terms "2-gram" and "3-gram" are commonly called bigram and trigram tagger.

1.2.10 Combining taggers

With background, a n-gram tagger and get in back to the discussion of "Reliability X Discrimination" there are two desirable characteristics, however contradictory: the reliability of the tagger and the coverage ratio. For reliability tagger, understand how the average probability of a word to be labeled (tagged) correctly. The coverage ratio must be understood as the probability of a word receives a mark by any tagger, this means the probability that the same sequence in the text is found in the reference corpus.

At first, looks possible think that would be desirable that the n-gram models were of high order. For higher order model, the higher the correlation between the estimated and stretch the body. But by dividing a larger number of equivalence classes, have many parameters to estimate. With increasing order of the tagger (value of "n" in n-gram) the less likely that a set of words is found in a given corpus. This effect can be seen in Table 1– number of parameters for a model n-gram with 20,000 tokens, where it is explained the exponential relationship between the model order and number of parameters.

<table>
<thead>
<tr>
<th>Modelo</th>
<th>Número de Parâmetros</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ª ordem (bigram)</td>
<td>20,000 * 19,999 = 400 milhões</td>
</tr>
<tr>
<td>2ª ordem (trigram)</td>
<td>20,000² * 19,999 = 8 trilhões</td>
</tr>
<tr>
<td>3ª ordem (four-gram)</td>
<td>20,000³ * 19,999 = 1.6 * 10¹⁷</td>
</tr>
</tbody>
</table>

Table 1– number of parameters for a model n-gram with 20,000 tokens
"At half a million words, this book does not contain enough data to produce a good bigram model, let alone a trigram model. There are about 15,000 different words in the lexicon of this book, so the bigram model include 15,000^2 = 225 million word pairs. Clearly, at least 99.8% of these pairs will have a count of zero..." (5)

For n-gram models, the combination of several models with different orders is usually the key to success. There are several possible approaches to deal with this problem of sparse data.

A simple and effective way of dealing with the conflict between reliability and coverage is the simple linear interpolation of probabilities. One possibility is to use more accurate models (wheat, for example) possible, but maintaining a cover in case of failure models of order smaller. Actually it is an infinite linear interpolation with weights for models of higher order with respect to a lower order. Although simpler, thus less accurate, this approach facilitates the training stage of the tagger and enables the use of "back-offs, which greatly simplifies the implementation of this algorithm.

For example, combining the results of a bigram, a unigram tagger and a default as follows:
1. Try to mark the token with the bigram tagger.
2. If the bigram tagger fails to match the token with the corpus, use the unigram tagger.
3. If the unigram tagger fails use the default tagger.

1.2.11 Application of Lexical Analysis

There are several classification schemes tags. The application developed as a proof of concept adopted the system of appointment of Brown University. The Brown corpus is probably one of the most widely used corpora. This is a set of texts labeled with approximately one million words compiled by Brown University. The texts are samples of American English into several categories: fiction, newspapers, scientific texts etc. After several interactions in use the entire corpus of Brown as "back off" and press reports as a standard tagger, was the most efficient configuration. The central portion of the code of POS tagger is illustrated in Figure 1 - Tagger.

```python
from nltk.corpus import brown
from nltk import pos_tag

# Example text
text = 'The AFCS shall provide to the FCS a roll rate demand signal whenever the autopilot engagement is enabled.

# POS tagging
pos_tags = pos_tag(text.split())

print(pos_tags)
```

Using the function nltk.tag.accuracy () is possible to calculate the accuracy of a tagger. This function compares the results of the tagger to try to score known texts. For Tagger implemented produces an index of 94.6% accuracy by reference to the corpus of press publications.

However, upon lexical marking in a sample of actual requirements resulted in an average of one error at six requirements. This represents a hit rate greater than the established building on the reference corpus. Probably, this effect is because requirements are written in a controlled natural language.

1.2.12 Collocation used as a tagging parameter

However, one error every 6 requirements is an unacceptable error rate. Although numerically lower, about 2% of tokens are bad tagged; the syntactic analysis depends of these data as input. This causes a 12% rate of failure in parsing stage.

When checking the origin of the problem, becomes explicit that virtually all the tags that receive a bad tag for the reason that the word is commonly used as a word class different from that found in the requirement.

For example, the follow requirement:

"The AFCS shall provide to the FCS a roll rate demand signal whenever the autopilot engagement is enabled.

Generates the tagged:

```plaintext
[('The', 'AT'), ('AFCS', 'NN'), ('shall', 'MD'), ('provide', 'VB'), ('to', 'TO'), ('FCS', 'NN'), ('a', 'AT'), ('roll', 'VB'), ('rate', 'NN'), ('demand', 'VB'), ('signal', 'NN'), ('whenever', 'WRB'), ('the', 'AT'), ('autopilot', 'NN'), ('engagement', 'NN'), ('is', 'BEZ'), ('enabled', 'VBD'), (',', 'LS'), (',', 'LS')]
```

The token "to" is bad tagged for the context in which it is used. The tagger algorithm marks the preposition "to" like it was a particle of the infinitive. However, "to" in this context has the function of a preposition. How could be recognized these and other similar cases?

From the same reference corpus used to construct the bigram can get up important data, from a simple script like the one in Erro! Fonte de referência não encontrada. Figure 2.

```python
for a in range(len(brown_s)): 
    t1 = []
    for (w1,t1) in brown_s[a]:
        if w1 == 'to' and t1 == 'IN':
            counter_prep = counter_prep + 1
        if w1 == 'to' and t1 == 'TO':
            counter_prep = counter_prep + 1
            t_ent = t1
```

Figure 2 - script to count frequency of preposition "to"

In the entire corpus token, "to" preceded by a verb occurs exactly 2 times as the infinitive particle than as preposition, 40 times as a particle of the infinitive and 20 times as a preposition. Considering that the words were marked using an n-gram model is easy to see that while the universe expands search or use other weights in the relations of n-grams the error is persistent. So the n-gram model does not consider such cases.

To make the solution more widely as possible, must first be analyzed, as a human can recognize when "to" should be marked with 'TO' (infinitive) and when it should be marked 'IN' (preposition). Simple, if you are before you to interpret...
it as a verb infinitive particle and is followed by a paper as a preposition.

Nevertheless, this solution is very specific. A more general way of treating the problem is to use the term collocation. The concept of collocation diverge from author to author, while some think that only words that occur together and when given a special meaning can be treated as collocation. Others argue that any set of words, even though sparse in the text, and they occur with a high frequency, is a collocation. For this study, understand collocation as a sequence of words that occurs together more frequently that their frequency grammatical function. To test this idea, a script was developed. Although of the collocation theory, the development was widely empirical, refining at each interaction the weights of frequency distributions, classes of words to ignore and which set of optimal amount of tokens that must be used to improve POS tagging. The script to raise the number of collocation is implemented with the algorithm described below:

1. Consider the frequency distribution, defining the most relevant settings using the following equation to calculate the correlation index between the two words:

$$CorrelationIndex(w1,w2) = \frac{FreqTag(w1|w2)}{FreqTag(w1) * FreqTag(w2) + FreqTot(w1)}$$

Where:

- $FreqTag(w1|w2)$ = Distribution of Frequency; where $w1$ and $w2$ appears in sequence with a specific part of speech. Remark that $(x,y)$ is different of $(y,x)$
- $FreqTag(w)$ = Frequency distribution of a word with a specific part of speech.
- $FreqTot(w1)$ = Frequency distribution of a word, looking over all corpuses, even non tagged texts.

The number of words classified, test sets of words which meets the following criteria:

a) The tag must be placed in $w1$ than the most frequent tag for the word $w1$.
b) $w1$ and $w2$ should not be its personal name or words of foreign origin.
c) Set as collocation the first 1000 sets of $(w1, w2)$ with the highest correlation.

As a result it generated a list of the first 1000 sets of words that should be tagged in a special way when they occur in sequence and its corresponding tag. Interestingly, this process in some respects is similar to that conducted by the bigram with two differences: the bigram uses the previous symbol, while this treatment takes into account the word later. Table 6 contains the 10 placements with highest correlation.

<table>
<thead>
<tr>
<th>Collocation</th>
<th>Tag of the first word</th>
</tr>
</thead>
<tbody>
<tr>
<td>to the</td>
<td>IN</td>
</tr>
<tr>
<td>as well</td>
<td>QL</td>
</tr>
<tr>
<td>more than</td>
<td>AP</td>
</tr>
<tr>
<td>no longer</td>
<td>QL</td>
</tr>
<tr>
<td>rather than</td>
<td>IN</td>
</tr>
<tr>
<td>out of</td>
<td>IN</td>
</tr>
<tr>
<td>White-House</td>
<td>JJ-TL</td>
</tr>
<tr>
<td>so-that</td>
<td>CS</td>
</tr>
<tr>
<td>over-the</td>
<td>IN</td>
</tr>
<tr>
<td>more-than</td>
<td>AP</td>
</tr>
<tr>
<td>all-right</td>
<td>IN</td>
</tr>
</tbody>
</table>

Table 2- example of collocations that should be retagged.

4. RESULTS

Using the previous example:

The AFCS shall provide to the FCS a roll rate demand signal whenever the autopilot engagement is enabled.

Performing the procedure taking in count the collocation as a tagging parameter, at a sample of 64 requirements produces 8 substitutions, all corrects. With these corrections the algorithm of POS tagging perform rate of success approximately 99%. That is surprisingly good. This rate could be improved, with a corpus more matched to aeronautical vocabulary.

Observe that 64 sentences is a sample numerically weak and may not be representative. Besides the fact they were deliberately excluded from baseline, some requirements potentially problematic to NLP, like requirements with topics, equations or tables.

For the scope of this work, the approach adopted showed excellent results, both from the perspective of the outcome itself, as referring to the computer resources that the software demands.
REFERENCES


