A set of data bases to support intelligent agents in Internet Infrastructure routing domains

Juliao Braga\textsuperscript{1,2}, Joao Nuno Silva\textsuperscript{1}, Nizam Omar\textsuperscript{2}

\textsuperscript{1}INESC-ID, Lisbon University, Lisbon, PT
\textsuperscript{2}Mackenzie University, Sao Paulo, SP, Brazil

\{juliao.braga,joao.n.silva\}@tecnico.ulisboa.pt
nizam.omar@mackenzie.br

Abstract. This paper presents a set of three data bases that make up the Internet Infrastructure Data Base (IIDB). IIDB has three data bases – iidb.rfc, iidb.person, and iidb.acronym – that are key pieces to support the development of machine learning techniques by the intelligent elements of the Autonomous Architecture Over Restricted Domains (A2RD). The data contained in iidb.rfc and iidb.person were created after processing the contents available at the RFC Index web page. While the data contained in the iidb.acronym was created after processing the contents of the files available at the Request for Comments (RFC) repository, produced and maintained by the RFC Editor. The data format of IIDB data is JavaScript Object Notation (JSON), whose templates are available in the same site where the data bases are deposited, making them accessible through any programming language.

Resumo. Este artigo apresenta um conjunto de três bases de dados que compõem o Internet Infrastructure Data Base (IIDB). O IIDB é um conjunto formado pelas bases iidb.rfc, iidb.person e iidb.acronym, peças-chave para apoiar o desenvolvimento do aprendizado de máquina desejado aos elementos inteligentes do projeto Arquitetura Autônoma Sobre Domínios Restritos (A2RD). Os dados contidos em iidb.rfc e iidb.person foram criados após o processamento do conteúdo disponível na página web do RFC Index. Enquanto os dados contidos no iidb.acronym foram criados após o processamento do conteúdo dos arquivos disponíveis no repositório Request for Comments (RFC) produzido e mantido pelo RFC Editor. Todos os dados no IIDB são formatados em JavaScript Object Notation (JSON), cujos respectivos modelos estão disponíveis no mesmo site onde as bases de dados são depositadas, acessíveis através de qualquer linguagem de programação.

1. Introduction

There is a permanent concern to convey enough intelligence to Autonomous Architecture Over Restricted Domains (A2RD) agents to make them autonomous. This requires an organized integration of the resources shown in Figure 1, where the A2RD model that can be implemented in each Autonomous System AS (or routing domain) [Colel et al. 1994, Hares and Katz 1989], is represented as item (11) in the Figure 1, integrated to the Structure for Knowledge Acquisition, Use, Learning, and Collaboration (SKAU) model.
The SKAU model components and activities are:

- Each RFC is captured/updated and stored locally and transformed in a corpus, ready to be process by the Python Natural Language Tool Kit\(^1\) (NLTK), (1) [Bird et al. 2009];
- A set of tools is responsible for acting lexical and syntactically on RFCs (2), transforming them into intermediary data bases (3);
- Other tools (4), like Semantic Distillation, act on the intermediary data bases producing inputs for Domain Data Sets (DDS) construction (6) and supporting the data set production that forms the Training Data Sets (TDS) (8). Also, these tools will support part of the Knowledge Base (KB) (9) [Isotani and Bittencourt 2015]. A relevant part of DDS is the Internet Infrastructure Data Base (IIDB), whose construction description is the goal of this paper;
- Learning algorithms (7) support the construction and use of TDS to renew the knowledge base and meet the demand A2RD agents in the process of developing and applied intelligent actions. The efficient use of TDS will respond to the classic algorithms of Machine Learning (ML): (a) supervised learning, (b) unsupervised learning, (c) reinforcement learning and (d) semi-supervised learning, that combines (a) and (b). [Musumeci et al. 2018] in Section II has an appropriate ML overview, with a focus on optical networks;
- Each implemented A2RD model build a data base, named IIBlockchain (10) and stored it together, in the Git Hub (i.e. in the cloud). This then supports the process of collaboration and effective interaction, inter/intra agents of the models [Braga et al. 2018]. The IIBlockchain cloud interacts with the learning algorithm and KB allowing agents to exercise offline and online computation\(^2\) [Poole and Mackworth 2010].

Each AS can implement an A2RD, which is then controlled by the IE – IE Controller – and receives the identification x:0, where x is the AS Number (ASN).

\(^1\)https://www.nltk.org

\(^2\)Offline computation is the computation done by the agent before it has to act, and online computation is the computation done by the agent between observing the environment and acting in the environment
2. IIDB

IIDB evolved from the efforts to build WordNet [Fellbaum 1998]. It was later realized that its usefulness would be amplified if it represented not only words (from the domain of the Internet Infrastructure) and their lexical equivalents but also any representation associated with it meaning (proper names, numbers, dates, acronyms, etc.). Table 1 shows an example of the IIDB contents (excluding implementation details).

<table>
<thead>
<tr>
<th>Representation</th>
<th>Meaning</th>
<th>Ext</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
<td>-</td>
<td>ietf</td>
</tr>
<tr>
<td>3978</td>
<td>RFC</td>
<td>-</td>
<td>doc</td>
</tr>
<tr>
<td>3978</td>
<td>OpenTTD game (masterserver and content service)</td>
<td>TCP;UDP</td>
<td>tcp</td>
</tr>
<tr>
<td>Jon Postel</td>
<td>RFC00001</td>
<td>img:lnk;text:url</td>
<td>human</td>
</tr>
<tr>
<td>protocol</td>
<td>rules determining the format and transmission of data</td>
<td>-</td>
<td>wordnet</td>
</tr>
</tbody>
</table>

Approaching the formal notations and definitions of WordNet [Miller 1995], the IIDB is defined as $W_1 = (f, s, e, d)$ where $f$ is a form composed by a string over a finite alphabet, $s$ is sense got from a given set of meanings found in the unstructured bases (as RFC repository), $e$ is an extension which complements $s$ and $d$ is the sub-domain to which form $s$ belongs.

Hence, IIDB is a data set that covers the Internet Infrastructure domain and can be used for quick access not only by IEs but also by third parties and will help build the KB and support to update the KB and will be used as learning content for ML algorithms. The first three data bases which initially make up the IIDB – iidb.rfc (3,903 MBytes), iidb.person (4,961 MBytes) and iidb.acronym (9,590 MBytes) – are available in the repositories Open System Foundation (OSF) [Braga et al. 2019].

2.1. The value of IIDB

For several reasons it is necessary to find in which RFC an acronym has been defined. For example, a way to improve knowledge in a specific subject, which the acronym stands for. Having the number of the RFC in which it was defined and the RFCs that refer it, other bases can also be searched for the purpose of refining the knowledge. This set of research can contribute to become better the agents learning.

It is immediate to research by words in the acronym meaning. In this way, via the acronym it is possible to identify which RFCs treat the subject referenced by the word. For example when looking for 'NAT' we can get to RFC05720 and RFC06346, which should address the subject Network Address Translation, one of the meanings of the acronym (lines 5 and 6 in Table 4).

It is also possible to immediately identify which RFCs refer to an acronym or words of their meaning. Such facility are suitable for AS administrators, as well as technicians and researchers interested in interacting with RFCs.

The above privileges are appropriate, given the storage form of the IIDB bases, for testing, training and learning the ML algorithms.
Table 2 shows a set of meanings extracted from the corpus of the RFCs, showing that there are ambiguities in the meaning of the acronyms.

<table>
<thead>
<tr>
<th>#</th>
<th>Acronym</th>
<th>Description</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RFC</td>
<td>REQUEST FOR COMMENTS</td>
<td>RFC01175</td>
</tr>
<tr>
<td>2</td>
<td>RFC</td>
<td>Request For Comment</td>
<td>RFC00199</td>
</tr>
<tr>
<td>3</td>
<td>RFC</td>
<td>Request For Comments</td>
<td>RFC00724</td>
</tr>
<tr>
<td>4</td>
<td>RFC</td>
<td>Request For Connection</td>
<td>RFC00033</td>
</tr>
<tr>
<td>5</td>
<td>RFC</td>
<td>Request-For-Connection</td>
<td>RFC00663</td>
</tr>
<tr>
<td>6</td>
<td>RFC</td>
<td>Requests-For-Connection</td>
<td>RFC00634</td>
</tr>
<tr>
<td>7</td>
<td>RFCs</td>
<td>REQUESTS FOR COMMENTS</td>
<td>RFC01175</td>
</tr>
<tr>
<td>8</td>
<td>RFCs</td>
<td>Request For Connections</td>
<td>RFC00671</td>
</tr>
<tr>
<td>9</td>
<td>RFCs</td>
<td>Requests For Comment</td>
<td>RFC08280</td>
</tr>
<tr>
<td>10</td>
<td>RFCs</td>
<td>Requests For Comments</td>
<td>RFC00661</td>
</tr>
</tbody>
</table>

Accented, included for humans are the ambiguities represented by lines 2 and 4. But for machines, lines 1 through 6 are ambiguous and so lines 7 to 10. Ambiguities are common in documents where there is no rigidity in the patterns of their composition. The bases of the IIDD allow disambiguation of the acronyms by references to the RFCs and, eventually, to their authors.

In the repository the IIDD is a Jupyter notebook, with some Python scripts examples of using of the data bases.

3. Experimental design, material and methods

The techniques, features, and facilities used to construct the three bases that make up the IIDD are discussed below.

3.1. iibd.rfc and iibd.person

The iibd.rfc and iibd.person data bases were created based on data available in RFC Index web page. For example, the information regarding the RFC8039 [Shpiner et al. 2016], is:


A computer program processed this information to fit the following pattern, arbitrarily defined:

```
'number': 8039 'title': 'Multipath Time Synchronization'
'author': 'first': 'A' 'second': '' 'last': 'Shpiner'
'author': 'first': 'R' 'second': '' 'last': 'Tse' 'author':
'first': 'C' 'second': '' 'last': 'Schelp' 'author':
'first': 'T' 'second': '' 'last': 'Mizrahi' 'nauthor': 4
'date': 'year': 2016 'month': December 'day': '' 'status':
'EXPERIMENTAL' 'stream': 'IETF'
```

---

^3https://jupyter.org/index.html
^4https://www.rfc-editor.org/rfc-index2.html
From the above pattern, other appropriate computer program captured the data to fill the two data bases (iibd.rfc and iibd.person), according to the respective templates presented in Listings 1 and 2.

Listing 1. iidb.rfc template

```
{
  "representation": representation,
  "meaning": {
    "title": title,
    "year": year,
    "month": month,
    "day": day,
    "status": status,
    "stream": stream,
    "words": {
      "volume": "len(text)",
      "vocabulary": "len(set(text))"
    },
    "lexical_diversity": "vocabulary/volume",
    "extension": "",
    "subdomain": "rfc"
  }
}
```

3.2. iibd.acronym

3.2.1. The construction process

Figure 2 shows the four steps that allow to prepare the RFCs for the extraction of acronyms.

![Diagram of RFC processing steps]

Figure 2. Preparing to process RFCs

Following the figure, the process of add/update RFCs to a local directory based on a search using the Document Retrieval features of the RFC-Editor5.

Listing 2. iidb.person template

```
{
  "representation": {
    "firstname": ""
  }
}
```

5https://www.rfc-editor.org/retrieve/
An RFC is also published in text format such as RFC8039 (an arbitrary choice). By transforming the RFC8039 and others one into a single string as can be seen in Figure 3 and then using the tokenize techniques, available in NLTK, we construct the corpus of the RFCs, that is, a bunch of properly formatted documents, gathered in a directory [Perkins 2014].

Figure 3. Reduced RFC8039 text (partial view)

Thus, the corpus of RFCs facilitates the extraction of acronyms. This operation using an algorithm described below allows us to prepare some intermediate files that facilitate the availability of the acronyms database as desired.

6https://tools.ietf.org/rfc/rfc8039.txt
3.2.2. Acronym in the context of RFCs

Acronyms and their expansions (or meaning), when first introduced in the text, are usually adjacent [Osiek et al. 2010]. In RFCs the universally used standard has the format \textit{expansion (acronym)}, one of four formats identified by Pustejovsky et al. [Pustejovsky et al. 2001]. In this format, the (ACRONYM) will always be shown in capital letters. Usually, the expansion is composed of words with the first letters in upper case. Respecting this pattern and using some tools available in the Python language, and NLTK we extract the acronyms of the RFCs following the scheme shown in Figure 4.

![Figure 4. Preparing to process RFCs](image)

On the other hand, the purpose of the \textit{iibd.acronym} data base is to store acronyms and the respective RFC it was first occurred, in which RFCs it was referenced and how many times it appeared in each of them.

An acronym is a word created from the initial components of a phrase or name, called the expansion [Jacobs et al. 2018]. An acronym can be short-lived; if it was used in an RFC and is never referenced again. Also, an acronym can have more than one meaning. This is called \textit{polysemy}. Polysemy, or lexical ambiguity, is the property of some words to have multiple meanings or senses [Moldovan and Novischi 2004]. In linguistics, \textit{disambiguation}\footnote{https://en.wikipedia.org/wiki/Word-sense_disambiguation} refers to the process of explaining the message that has more than one meaning. To try to appease the disambiguation, the same acronym may appear several times in \textit{iidb.acronym}, with divergent meanings, but referencing the RFC where it was first quoted and at other times.

The acronyms for this work were taken from all RFCs up to RFC08540 [Stewart et al. 2019]. The algorithm used to extract the acronyms from RFCs is summarized in Figure 5.

![Figure 5. Algorithm used to extract acronyms from RFCs](image)
Three steps of the algorithm are relevant: *Apply some heuristics, delete text that precedes it* and *establish a rank to confirm it*. The last two steps are part of the kernel of the algorithm and considered to be primarily responsible for its success.

### 3.2.3. Step 1: "Apply some heuristics"

The requirements of this step can be summarized in the following topics:

- An acronym only exists if it is between ‘(’ and ‘)’.
- The possible acronym must have more than 1 character.
- If the possible acronym is numeric, ignore it.
- If the possible acronym is all lowercase, ignore it.
- The words are stacked until a ‘)’ is found. If an acronym is found, the stack is destroyed. So if there is not in the stack, at least the number of items corresponding to the number of letters found between ‘(’ and ‘)’, there is no acronym.
- It is acceptable that the acronym has the characters ‘&’, ‘/’ or ‘-’.
- It is acceptable ‘.’ if it appears by following each of all the letters of the possible acronym. In this case all the letters should be uppercase.
- We make the assumption that an acronym’s meaning (or expansion) lies before to the acronym. The reason for this assumption is associated with the pattern followed in the RFCs.
- Some authors consider that if the acronym contains numerical letters, its preceding letter (if there is any) or its following letter is repeated that many times to create a new acronym. This newly-created acronym is also used to find possible expansions by using other rules. For example, “W3C” can be changed to “WWW-C” in order to discover W3C (World Wide Web Consortium) [Ji et al. 2008]. But, this does not work, in S7 Application Part (S7AP), defined in RFC02719 [Ong et al. 1999]. The algorithm adopts the criterion of reproducing the letters on the left, when it finds 2 or 3 in the possible acronym.
- Some compound words such as ‘multiprotocol’ have been separated when are part of the acronym’s meaning these words were given an additional weight, to strengthen the acronym. For example, Multiprotocol Label Switching (MPLS).

### 3.2.4. Step 2: "delete text that precedes it"

When the algorithm finds an acronym such as *(PPP)*, it pops up three previously stacked words. Probably, the three words unstacked will be: ‘Protocol’, ‘Point-to-Point’ and a third word whatsoever, say ‘word3’. Arranged, the result would be: ‘word3 Point-to-Point Protocol’. Thus, the algorithm in this step eliminates the word ‘word3’ concluding that the result of the meaning of ‘PPP’ will be: ‘Point-to-Point Protocol. Once this is done, the next step should be to ensure that, indeed, this is the result.
3.2.5. Step 3: "establish a rank to confirm it"

Keeping the PPP acronym example, this step removes the hyphens ('-') and verifies that there is a stop word⁸ (the 'to') in the meaning of the acronym. Then the algorithm removes the stop word, adding to the acronym PPP, a negative weight, by having such a stop word. Then the algorithm continues analyzing the meaning and for each word that matches its first letter, with its position in the acronym, also receives a negative weight. When there is no coincidence, the weight is positive. In the example 'SS7 Application Part (S7AP)', the acronym and its meaning are immediately accepted because their weight will be sufficiently negative to maintain the valid acronym ranking. Finally, let’s look at the case of the acronym W3C. This acronym stands for a polysemy. The following meanings were found: (1) 'World Wide Web Consortium' (in the RFC05945), and (2) 'Worldwide Web Consortium' (RFC02768). This acronym is well known and ambiguity may disappear in the context. This acronym is well known and ambiguity may disappear in this context. But the different RFCs, which refer to them, will certainly elucidate the issue. This is the case of the polysemy of the acronym AIA, with three meanings: (3) 'Association America' (RFC04688), (4) 'Authority Info Access' (RFC04809) and (5) 'Authority Information Access’ (RFC05280).

4. Conclusion

4.1. Application of the algorithm

The final result of the execution of this algorithm, implemented in Python, is represented by the numbers in Table 3.

<table>
<thead>
<tr>
<th>#</th>
<th>Representation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RFCs processed (files)</td>
<td>8,340</td>
</tr>
<tr>
<td>2</td>
<td>Processing time (seconds)</td>
<td>14,163.1</td>
</tr>
<tr>
<td>3</td>
<td>Mean size of acronyms (chars)</td>
<td>3.36</td>
</tr>
<tr>
<td>4</td>
<td>Total number of acronyms extracted</td>
<td>69,198</td>
</tr>
<tr>
<td>5</td>
<td>Total number of acronyms extracted (no repetition)</td>
<td>12,273</td>
</tr>
<tr>
<td>6</td>
<td>Acronyms (no repetition) automatically confirmed</td>
<td>11,098</td>
</tr>
<tr>
<td>7</td>
<td>Precision</td>
<td>90.42%</td>
</tr>
</tbody>
</table>

The number of RFCs processed is less than the number of the last RFC processed, because there are numbers without RFCs, (1). The processing time is the result of the execution time given by the Sublime Text⁹, (2). The acronyms average size, in number of characters, (3). The total number of acronyms extracted, with repetition, (4). Number of non-repeating acronyms from a non-human point of view, (6), representing 90.42% of (5), (7).

The file obtained in row (4) of Table 3 is sorted in alphabetical order and used to populate the acronym.json file, based on the template displayed in Listing 3.

---

⁸A word that is part of the meaning of the acronym, but usually is not represented in the acronym.
⁹https://www.sublimetext.com/
4.2. Acronym examples

Table 4 presents acronyms on which we will make some comments to reinforce the heuristic algorithm used.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6LoWPANs</td>
<td>IPv6 over Low-Power Wireless Personal Area Networks</td>
<td>RFC06550</td>
</tr>
<tr>
<td>2</td>
<td>AAA</td>
<td>Authentication Authorization and Accounting</td>
<td>RFC05887</td>
</tr>
<tr>
<td>3</td>
<td>AAA</td>
<td>AUTHENTICATION AUTHORIZATION AND ACCOUNTING</td>
<td>RFC02881</td>
</tr>
<tr>
<td>4</td>
<td>AAA</td>
<td>Authentication Authorization and Accountability</td>
<td>RFC02888</td>
</tr>
<tr>
<td>5</td>
<td>CGN</td>
<td>Carrier-Grade NAT</td>
<td>RFC05720</td>
</tr>
<tr>
<td>6</td>
<td>CGNs</td>
<td>Carrier-Grade NATs</td>
<td>RFC06346</td>
</tr>
<tr>
<td>7</td>
<td>EBCDIC</td>
<td>Extended Binary-Coded Decimal Interchange Code</td>
<td>RFC00109</td>
</tr>
<tr>
<td>8</td>
<td>(FDV, also known as Jitter)</td>
<td>Frame Delay Variation</td>
<td>RFC07023</td>
</tr>
<tr>
<td>9</td>
<td>NETRJS</td>
<td>Remote Job Service</td>
<td>RFC00252</td>
</tr>
<tr>
<td>10</td>
<td>RJE</td>
<td>Remote Job Entry Protocol</td>
<td>RFC00105</td>
</tr>
<tr>
<td>11</td>
<td>RJE</td>
<td>Remote Job Entry Protocol</td>
<td>RFC00707</td>
</tr>
<tr>
<td>12</td>
<td>USASCII</td>
<td>USA Standard Code for Information Interchange</td>
<td>RFC00109</td>
</tr>
<tr>
<td>13</td>
<td>WRU</td>
<td>Who Are You</td>
<td>RFC00109</td>
</tr>
</tbody>
</table>

In line 1, ’6LoWPANs’. If the number 6 appears in an acronym and does not follow a small ’v’ letter, then it represents ’IPv6’. The same happens with the number 4, which turns into ’IPv4’. The acronyms in lines 2–3 represent the same meaning (from the human point of view). The algorithm, however, considers them different and implements the two in the iidb.acronym file. It remains for the future work, the normalization of this difference. The acronym on line 4, has the seemingly strange meaning with the word ’Accountability’. This is a good example of the absence of standardization, whose proposal was put into future work. The RFC Editor should define between ’Accounting’ and ’Accountability’, for an acronym such as ’AAA’. Lines 5–6 serve to illustrate the fact that a lowercase ’s’ at the end of the acronym does not influence the choice. This is the orientation that intelligent agents will receive when using IIDB. There is similarity to
this case with lines 2–3, relating to the acronym ‘AAA’. The acronym of line 7 has size 6. If you remove the hyphen from the second word of meaning, you have 6 words in the meaning. This is how the algorithm behaves. In line 8, very rarely is there an oversight of an author, escaping aggressively from the pattern of acronyms. The algorithm of this work fails in cases like this even though it is not complicated, in the context of a ‘(, end with a ‘)’, instead of a ‘)’. But, this ran away from the more common pattern of acronyms in RFCs. In line 9, the algorithm will fail (but acknowledge and identify the failure), to Remote Job Service (NETRJS). In lines 10–11, same acronym with the word ‘Protocol’ in the end of the meaning. This word does not create ambiguity, because the algorithm admits the existence of it, or not at the end of the meaning. The acronym of line 12, ‘USASCII’ has size 7, but has one of the words of its meaning, integrally in the acronym: ‘USA’. Then the size of the acronym becomes 5. However, it has a stop word, ‘for’ that reduces its size to 4. Acronym Who Are You (WRU), line 13, is only possible to be identified if we use a phonetic dictionary, since the characters R and U correspond to ARE and YOU. This was done in the algorithm, which gives each phoneme the appropriate punctuation.

4.3. Use of IIDB databases

In the IIDB repository, as already mentioned, there is a file with some scripts in Python, using the data bases available there. By way of illustration, we show a simple script, but it uses two bases at the same time, to determine the first 30 most participant authors of RFCs. Figure 6 shows the program and on the right side, the result.

5. Related Works

The importance of extracting acronyms is acclaimed by Sánchez and Isern [Sánchez and Isern 2011]: The discovery of the definitions associated to an acronym is an important matter in order to support language processing and knowledge-related tasks as information retrieval, ontology mapping or question answering. Xiaonan Ji and colleagues [Ji et al. 2008], consider that techniques for being able to automatically identify acronym patterns are very important for enhancing a multitude of applications that rely upon search and present a new approach to discover acronyms patterns.
Other techniques such as logical-algebraic equations that combine grammatical and semantic characteristics of words of substantive, attributive and verbal collocations types exploit WordNet resources [Khairova et al. 2018].

Unsupervised learning techniques have been and are used in the medical field, where ambiguity of acronyms occur mainly when the context is not delimited. If we see the acronym RA in a cardiology report, then it can be normalized to “right atrial”; otherwise, if it occurs in the context of a rheumatology note, it is likely to mean “rheumatoid arthritis” or “rheumatic arthritis” [Pakhomov 2002]. So the method of using the global context to solve the ambiguity of an acronym as it is done in the medical field is not an adequate solution to the ambiguity of the acronyms found in the RFCs. It is noted that ambiguity is often produced, by lexical error or typos of the authors, both very common, as can be seen. Also, other techniques as machine-learning-based approach to automatically build an acronym dictionary from texts are proposed [Jacobs et al. 2018].

A simple formula, where a score is determined by Equation 1, and if it is below some threshold, then the pair is accepted. This was proposed by [Pustejovsky et al. 2001].

\[
\text{score} = \frac{\# \text{ of words in the match}}{\# \text{ of characters in the acronym}}
\]  

(1)

Reinforcement learning techniques like Markov chains [Paulino et al. 2018] and unsupervised learning techniques like Hidden Markov models (HMMs) are a powerful probabilistic tools for modeling time series data, and have been applied with success to many language-related tasks such as part of speech tagging, speech recognition, text segmentation and topic detection [Freitag and McCallum 1999]. Several authors have used acronyms extraction in medical unstructured texts [Osiek et al. 2010] [Conroy and O’leary 2001]. Others one, in speech recognition [Rabiner 1989].

The issue associated specifically with the extraction of acronyms is intensely addressed in Manuel Zahariev’s doctoral thesis [Zahariev 2004].

6. Future Works

RFCs are not ready to be evaluated by machines or smart codes. For example, Monkey in the Middle (MITM) should come closer to the machine understanding perspective being represented as (MitM), ensuring that stop words are in lower case, in the acronym. The absence of standards when writing an RFC creates ambiguity between acronyms, more often than was expected (CGN e CGNs, both with the same meaning). The IETF, as a careful standards-setting institution, should take better care of the standardization of its main document, since it is from RFCs that we can gain knowledge to provide intelligent agents. The authors believe and in the near future should lead to a draft proposing writing standards that could be required from the authors to facilitate the use of machine learning and other techniques of artificial intelligence. This kind of care should be extended to IETF peripheral institutions, such as Regional Internet Registers (RIRs), which produce a lot of important information for ASes administrators.

The algorithm used to extract the acronyms from RFCs can be improved to use machine learning techniques on rejected acronyms. Additionally, refinement of the files generated by the current algorithm could increase the accuracy of the result. Both al-
ternatives allow to produce appropriate results for measurements and evaluations of the acronyms capture algorithm [Batista et al. 2004]. This was not done in this project, because it was outside the goal of the global experiment.

Acknowledgment

This work is supported by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Brazil’s Ministry of Education, by national funds through FCT with reference UID/CEC/50021/2019 and by MackPesquisa.

References


