

# CNIO at BARR IberEval 2017: exploring three biomedical abbreviation identifiers for Spanish biomedical publications

Ander Intxaurreondo, Martin Krallinger

CNIO - Spanish National Cancer Research Center, 28029 Madrid, Spain  
{aintxaurreondo,mkrallinger}@cnio.es

**Abstract.** This paper describes the adaptation and assessment of three state-of-the-art publicly available, widely used, biomedical abbreviation recognition systems developed originally to process English scientific literature. The underlying assumption of using these tools was that abbreviations, and abbreviation-definition pairs do show similar properties shared by texts written in both languages. The three systems, ADRS, Ab3P and BADREX were evaluated at the Biomedical Abbreviation Recognition and Resolution (BARR) task of IberEval 2017. These three tools are based on heuristics that exploit aspects such as the presence of parentheses surrounding abbreviation mentions, which are commonly mentioned in the same sentence after the abbreviation description or long form. The obtained results showed that the heuristics used by these systems work well also for medical publications in other languages, such as Spanish and Portuguese.

## 1 Introduction

This paper describes the IberEval 2017 Biomedical Abbreviation Recognition and Resolution (BARR) task and the benchmarking CNIO participation in this track ([3]). The BARR track requires essentially finding abbreviations and their corresponding long forms (descriptions or definitions) in medical publications written in Spanish.

In the latest years, the interest of applying natural language processing tools to the biomedical domain has increased. A considerable number of publications describe methods related to biomedical named entity recognition approaches, for entity types such as diseases/symptoms, proteins, genes, drugs and chemicals. Moreover, a considerable number of domain-specific information retrieval and extraction systems specifically tailored to process biomedical and medical texts have been implemented during the last decade. We can find an extensive collection of research publications for the English language in this area; meanwhile, there is a lack of research for other languages.

An important challenge studied intensely by the biomedical text mining is the recognition and resolution of abbreviations and acronyms in biomedical documents. It is very common to find abbreviations of concepts and entities in clinical records without their long form or definition. Due to the lack of widely followed standardizations for abbreviations and their meanings, interpretation of abbreviations is a challenge both for humans as well as machines. Disambiguating abbreviations can help to construct medical abbreviation dictionaries, and so to improve the performance of different text processing

approaches applied to the biomedical domain. This may help health care professionals work in the interpretation of ambiguous abbreviations.

The aim of the BARR track was promoting the recognition and resolution of abbreviations found in Spanish medical publications. The task consisted of two tracks:

- Abbreviation mention (entity) evaluation.
- Abbreviation Short form - long form relation detection evaluation.

For this track, a corpus collection of medical article abstracts written in Spanish was released, the BARR document collection, while a manually annotated corpus of abbreviations and long forms served to train and test systems of participating teams, the BARR Gold Standard corpus.

This paper is structured as follows. In section 2 we briefly introduce the tracks of the BARR task. In section 3 we explain the three tools we used to extract abbreviations and their long forms. In section 4 we focus on the results of the submissions resulting from the use of these tools. And finally, in section 5, we draw some conclusions.

## 2 Evaluation tracks

In this section, we make a brief introduction of the two evaluations tracks present in the BARR task.

### 2.1 Entity evaluation track

In this track, participants had to detect mentions of abbreviations, i.e. short forms and their corresponding long forms and nested long forms in documents.

In the BARR corpus, among other annotations, the main types of entity corresponded to: *LONG*, *SHORT*, *MULTIPLE* and *NESTED* mention types. Abbreviations and acronyms were labelled as *SHORT*, while their descriptions (co-mentioned in the same sentence) were tagged as *LONG*. Note that short forms that were mentioned somewhere else in the record, were labelled as *MULTIPLE*.

Sometimes long forms did not correspond to a continuous string of text. In these special situations, long forms corresponded basically to several fragments of text and were labeled as *NESTED*.

### 2.2 Relation evaluation track

For the BARR track, participants had to detect mentions of short forms together with their long forms (SF-LF relation pairs) or nested long forms (*NESTED-SF*). The systems tested through the CNIO submissions were unable to detect nested cases, and thus did not return results for this relation type.

Figure 1 shows an example of manual annotation. The figure shows a long form and short form pair in the same context. We can find the short form mentioned again later in the abstract, which is labeled as *Multiple*.

Long Form                      Short Form  
 Las secuelas en la articulación temporomandibular (ATM) son (...)  
 La exploración y diagnóstico de los problemas de la ATM no son  
 sencillos (...) Multiple

**Fig. 1.** Manual annotation example.

### 3 Abbreviation detection and recognition

We used three different state-of-the-art tools to detect abbreviations and acronyms. These tools were initially developed to detect short forms and their long forms in biomedical documents. Although they were developed for the English language by default in the biomedical domain, we wanted to try their performance with Spanish and Portuguese documents.

We named these tools as Ab3P, ADRS and BADREX. They all use the following heuristic to find abbreviations in texts: if there are opening and closing parentheses in the same sentences, we will likely find the long or short forms inside the parentheses, and their other form nearby. They check the characters inside the parentheses, and look for words that could match with those characters.

Before executing each tool, we split the sentences in the abstracts using IXA pipes [1], and looked for long and short form pairs in each sentence individually. Splitting sentences we prevented making short and long forms between entities detected at the beginning and the end of the abstract, making it possible to detect pairs only when they were in the same context. We considered titles as a single sentence.

None of these tools return the offsets of short or long forms. It is up to the users to get them.

The following subsections describe each tool, and how we adapted them for Spanish, in case it was necessary.

#### 3.1 ADRS

ADRS<sup>1</sup> is how we call the algorithm developed by [4], a state-of-the-art algorithm developed in Java. ADRS returns the abbreviations and their definitions found within sentences.

ADRS's main strategy consists of detecting parentheses, and considering the inner content, with a maximum of two words and ten characters, as a potential short form, following the pattern "*long-form (short-form)*". Long forms must be in the same sentence. Every character in the short form matches a character in the long form, following the order of the characters in the short form. The heuristic also handles the inverse form "*short-form (long-form)*".

To use ADRS, we integrated the original code with our system's code. Before executing the tool for each publication, we split all sentences of each abstract using Ixa

<sup>1</sup> <http://biotext.berkeley.edu/code/abbrev/ExtractAbbrev.java>

Pipes ([1]), and analysed each line with ADRS, in order to get all short form and long form pairs. We considered titles as a single sentence.

### 3.2 Ab3P

Ab3P<sup>2</sup> ([5]) is a state-of-the-art tool used to detect abbreviations precisely. It is developed in C++, and simple to use. Ab3P returns all abbreviations and their long forms detected in each line of the document, with their estimated precision. There is a Java fork available for download<sup>3</sup>, but it is still incomplete; it has not been improved for 3 years, and does not seem to be in the plans of the authors to finish it.

Ab3P's heuristics are based on the ADRS algorithm. The paper describes about 10 rules and 30 strategies used by their heuristic, and absent in ADRS. After applying all strategies, the tool estimates the accuracy of each given strategy; the strategy that returns the highest accuracy value is considered the most reliable, and so is selected as the long form of a short form.

We had many issues adapting the C++ code to our system in Java. In order to solve this, we executed Ab3P for each document individually, and later processed the outputs with our system. To execute this tool, all sentences need to be split by line in the input document, so we used Ixa Pipes once again to split the sentences. For each input file to be analysed by Ab3P, the first line of the input file belonged to the title.

### 3.3 BADREX

BADREX<sup>4</sup>, developed by [2], is a GATE<sup>5</sup> plug-in that detects abbreviations and their long forms in text using regular expressions.

This heuristic applies 5 steps to detect long and short forms. The first step is based on the ADRS algorithm. The second step uses subsets to discard conditions of short forms. Step 3 applies the regular expressions. Step 4 splits potential short and long form's non-alpha characters to match adjacent characters. And finally, Step 5 detects unpaired mentions of the long and short form in the same abstract (*MULTIPLE* mentions).

Regular expressions for long and short pairs are specified in separated files<sup>6</sup>. It is possible to adapt them to other languages or needs. We adapted them so it could work with acute (á), diaeresis (ü) and tildes (ñ). Giving them the possibility to work with Spanish special characters improved the tool's performance drastically. Table 1 shows BADREX's original regular expressions, together with the file names where these expressions are stored, and their variations to Spanish.

To make use of BADREX, we integrated the GATE API to our system, and executed the plug-in directly from the API. All sentences in the abstract were split once again.

<sup>2</sup> <https://github.com/ncbi-nlp/Ab3P>

<sup>3</sup> <https://github.com/aureooms/ab3p>

<sup>4</sup> <https://github.com/philgooch/BADREX-Biomedical-Abbreviation-Expander>

<sup>5</sup> Open-source text analyser. <https://gate.ac.uk/>

<sup>6</sup> Directory: `BADREX_DIR/resources/regex`

RegEx file name	RegEx for English	RegEx for Spanish
inner.post.txt	) (,;:\s+w+)?(\)\]	) (,;:\s+p(L)+)?(\)\]
inner.post.2.txt	\2(,;:\s+w+)?(\)\]	\2(,;:\s+p(L)+)?(\)\]
inner.pre.txt	)\s+(\[\(\[\2[\w\-\w*\.\\/\+\s]\{1,	)\s+(\[ \(\[\2[\p(L)\-\w*\.\\/\+\s]\{1,
inner.pre.2.txt	)\b(\w)(\w+(\[^\[\+\s]\{1,2}))\s*(\[ \(\{. {1,})\b(\p(L))(\p(L)+[^\[\+\s]\{1,2}))\s*(\[ \(\{. {1,	)\b(\p(L))(\p(L)+[^\[\+\s]\{1,2}))\s*(\[ \(\{. {1,
outer.pre.txt	\b(\w)\W(0,2)(\w+[^\s*\[\+\s]\{1,2})\{1,	\b(\p(L))\W(0,2)(\p(L)+[^\s*\[\+\s]\{1,2})\{1,
outer.pre.txt	\b(. {1,	\b(. {1,

Table 1. BADREX regular expressions, for English and Spanish.

```

1 BiomedicalAbbreviationExpander badrex = new BiomedicalAbbreviationExpander ();
2 URL configUrl = new File ("BADREX_DIR/resources/config.txt").toURI().toURL ();
3 URL gazUrl = new File ("BADREX_DIR/resources/lookup/abbrevs.def").toURI().toURL ();
4 badrex.setConfigFileURL (configUrl);
5 badrex.setGazetteerListsURL (gazUrl);
6 badrex.setExpandAllShortFormInstances (Boolean.FALSE);
7 badrex.setLongType ("Long");
8 badrex.setLongTypeFeature ("longForm");
9 badrex.setMaxInner (10);
10 badrex.setMaxOuter (10);
11 badrex.setSentenceType ("Sentence");
12 badrex.setShortType ("Short");
13 badrex.setShortTypeFeature ("shortForm");
14 badrex.setSwapShortest (Boolean.TRUE);
15 badrex.setThreshold (0.9f);
16 badrex.setUseBidirectionMatch (Boolean.FALSE);
17 badrex.setUseLookups (Boolean.FALSE);
18
19 Gate.init ();
20 File pluginsDir = Gate.getPluginsHome ();
21 // load the Tools plugin
22 File aPluginDir = new File (pluginsDir, "ANNIE");
23 // load the plugin.
24 Gate.getCreoleRegister ().registerDirectories (aPluginDir.toURI().toURL ());
25
26 badrex.init ();

```

Listing 1.1. BADREX plug-in and GATE initialization.

Code listing 1.1 shows how we initialized the plug-in and GATE. Listing 1.2 shows how we executed the plug-in to analyse sentences and extract short forms and their long forms from each sentence.

### 3.4 Labelling ‘MULTIPLE’ entities:

After getting all short and long form pairs, we detected the offsets of all entities participating in each relation. If we detected a short form twice in the same sentence, we considered as pairs those long and short forms that were closest to each other, while the other short form would be labelled as *MULTIPLE*.

We also checked the appearances of each entity in the rest of the document. We labeled those appearances as *MULTIPLE* as well.

## 4 Results

This section shows the final results we obtained after evaluating the predictions of each track through Markyt. To prepare our system for the background set, we initially worked using the sample set and the available train sets.

```

1 Document d = Factory.newDocument(sentence);           // sentence to analyse
2 Corpus corpus = Factory.newCorpus("test_corpus");
3
4 LanguageAnalyser sentenceSplitter = (LanguageAnalyser)
5     Factory.createResource("gate_creole_splitter.RegexSentenceSplitter");
6 SerialAnalyserController serialController = (SerialAnalyserController)
7     Factory.createResource("gate_creole.SerialAnalyserController");
8 serialController.add(sentenceSplitter);
9
10 corpus.add(d);
11 serialController.setCorpus(corpus);
12 serialController.execute();
13 corpus.clear();
14
15 badrex.setDocument(d);
16
17 badrex.execute();
18
19 AnnotationSet abbrevAS = d.getAnnotations().get("Short");
20 AnnotationSet termAS = d.getAnnotations().get("Long");
21
22 // HashMap of strings to store short and long forms
23 Map<String, String> pairs = new HashMap<String, String>();
24 Iterator<Annotation> termIter = termAS.iterator();
25 while (termIter.hasNext())
26 {
27     Annotation term = termIter.next();
28     Annotation abbrev = abbrevAS.iterator().next();
29     FeatureMap termFeats = term.getFeatures();
30     FeatureMap abbrevFeats = abbrev.getFeatures();
31     String shortForm = (String)termFeats.get("shortForm");
32     String longForm = (String)abbrevFeats.get("longForm");
33
34     pairs.put(shortForm, longForm);
35 }
36
37 Factory.deleteResource(d);

```

Listing 1.2. BADREX plug-in execution through GATE.

#### 4.1 Entity evaluation results

We submitted three runs for this track. The first run belongs to the Ab3P tool, explained above in section 3.2, the second one to the ADRS tool (section 3.1), and finally BADREX, in section 3.3.

We can find our results of the training set in Table 2. We obtain the best results with the tool ADRS, with Ab3P not being far. None of the systems was able to detect a single *NESTED* entity. After the predictions, we discovered that each tool worked well detecting abbreviations, but they often were not able to find the correct long form nearby.

BADREX is a good tool to detect abbreviations, but its heuristics to find the long form do not work that well. While this tool is very useful to detect long-short pairs in English, it still needs to be adapted for other languages.

Table 3 shows our final results of the entity evaluation track. We can find the same tendency of the training set here, with ADRS being the best system, Ab3P not far, and BADREX the last one.

Entity evaluation			
Tool	Precision	Recall	F-measure
Ab3P	<b>86.21</b>	56.04	67.92
ADRS	83.71	<b>59.84</b>	<b>69.79</b>
BADREX	81.27	45.39	58.25

**Table 2.** Entity evaluation results. Train set.

Entity evaluation			
Tool	Precision	Recall	F-measure
Ab3P	<b>87.95</b>	56.72	68.96
ADRS	84.47	<b>62.29</b>	<b>71.70</b>
BADREX	81.60	47.06	59.69

**Table 3.** Entity evaluation results. Test set.

## 4.2 Relation evaluation results

We also submitted three runs for this track. These three runs were based on the detected entities in the entity evaluation track, each run with the corresponding tool.

We can find our results of the training set in Table 4. Once more, we can see that ADRS is the best tool for abbreviation and long form detection. Ab3P is very close to ADRS again. Meanwhile, BADREX is far from getting the same performance of the other two systems. None of the systems was able to detect a single *NESTED* relation.

Table 5 shows our final results of the relation evaluation track. Just like in the entity evaluation track, results have the same tendency here.

An interesting project for the future would be extending these tools to work with *NESTED* entities, and be able to associate them with long and short forms.

## 5 Conclusions

In this paper, we presented our results of our participation in the entity and relation evaluation tracks for the Biomedical Abbreviation Recognition and Resolution (BARR) task at the IberEval 2017 workshop. We worked with 3 different state-of-the-art tools used to detect long forms and short forms for English biomedical texts, applying them to the Spanish language. We submitted 3 runs in total, being each submission for each tool. Two of the tools perform quite well for Spanish, giving good results when detecting biomedical entities, and relating abbreviations found in the text with their long forms in the same context; meanwhile, the third tool needs more polishing to perform better in Spanish. The tools used show that applying algorithms focused for abbreviation resolution in English, based on patterns and regular expressions, can also be used in other languages, such as Spanish and Portuguese.

For future work, we would like to investigate on the improvement of these tools for Spanish, in order to improve performance, detect nested entities, and make relations between nested and short and long forms possible.

Relation evaluation			
Tool	Precision	Recall	F-measure
Ab3P	<b>83.60</b>	51.55	63.48
ADRS	78.96	<b>54.17</b>	<b>64.26</b>
BADREX	64.57	36.75	46.84

**Table 4.** Relation evaluation results. Train set.

Relation evaluation			
Tool	Precision	Recall	F-measure
Ab3P	<b>84.23</b>	53.29	65.28
ADRS	79.05	<b>56.90</b>	<b>66.17</b>
BADREX	64.46	39.28	48.81

**Table 5.** Relation evaluation results. Test set.

## 6 Acknowledgments

We acknowledge the the encomienda MINETAD-CNIO/OTG Sanidad Plan TL and OpenMinted (654021) H2020 project for funding.

## References

1. Agerri, R., Bermudez, J., Rigau, G.: Ixa pipeline: Efficient and ready to use multilingual nlp tools. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14) (2014)
2. Gooch, P.: Badrex: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. CoRR (2012)
3. Intxaurreondo, A., Pérez-Pérez, M., Pérez-Rodríguez, G., López-Martín, J., Santamaría, J., de la Peña, S., Villegas, M., Akhondi, S., Valencia, A., Lourenço, A., Krallinger, M.: The biomedical abbreviation recognition and resolution (barr) track: benchmarking, evaluation and importance of abbreviation recognition systems applied to spanish biomedical abstracts. SEPLN (2017)
4. Schwartz, A., Hearst, M.: A simple algorithm for identifying abbreviation definitions in biomedical text. In: In Proceedings of Pacic Symposium on Biocomputing (2003)
5. Sohn, S., Comeau, D.C., Kim, W., Wilbur, W.J.: Abbreviation definition identification based on automatic precision estimates. BMC Bioinformatics (2008)