MER: a Minimal Named-Entity Recognition Tagger and Annotation Server

Francisco M. Couto*, Luis F. Campos, and Andre Lamurias

LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. Named-Entity Recognition (NER) aims at identifying the fragments of a given text that mention a given entity of interest. This manuscript presents our Minimal named-Entity Recognizer (MER), designed with flexibility, autonomy and efficiency in mind. To annotate a given text, MER only requires a lexicon (text file) with the list of terms representing the entities of interest; and a GNU Bash shell grep and awk tools.
MER was deployed in a cloud infrastructure using multiple Virtual Machines to work as an annotation server and participate in the Technical Interoperability and Performance of annotation Servers (TIPS) task of BioCreative V.5. Preliminary results show that our solution processed each document (text retrieval and annotation) in less than 3 seconds on average without using any type of cache. MER is publicly available in a GitHub repository (`https://github.com/lasigeBioTM/MER`) and through a RESTful Web service (`http://labs.fc.ul.pt/mer/`).

Key words: Named-Entity Recognition, Annotation Server, Text Mining, Biomedical Ontologies, Lexicon

1 Introduction

Named-Entity Recognition (NER) aims at identifying mentions of entities in a given text. To define which type of entities to recognize, state-of-the-art tools usually require as input a training corpus. Their performance depends on the availability of a large corpus with an accurate and comprehensive set of annotations, which is usually arduous to create and maintain. Many other tools require only as input a lexicon consisting in a list of terms within some domain, for example a medical lexicon. A lexicon is normally much easier to find or to create. The input text is then matched against the terms in the lexicon.

We propose a Minimal named-Entity Recognizer (MER) designed with flexibility, autonomy and efficiency in mind. MER only requires a lexicon as input, in the form of a text file, in which each line contains a term representing a named-entity to recognize. MER makes really easy the addition of new lexicons. MER is not only minimal in terms of the input but also in its implementation, which was reduced to a minimal set of components and software dependencies. MER

* Corresponding author: fcouto@di.fc.ul.pt

Fig. 1. Example of the contents of an input file representing compounds CHEBI:18167, CHEBI:15940, CHEBI:15763 and CHEBI:76072.

```
α-maltose
nicotinic acid
nicotinic acid D-ribonucleotide
nicotinic acid-adenine dinucleotide phosphate
```

is then composed of just two components, one to process the lexicon (offline) and another to produce the annotations (on-the-fly). Both were implemented as a GNU Bash shell script[1], mainly for two reasons: i) efficiency, due to its direct access to high performance text and file processing tools, such as grep and awk; and ii) portability, since the scripts can be executed in any Unix-like operating systems (common in most servers) without requiring any additional software or modifications.

MER was deployed in a cloud infrastructure to work as an annotation server an participate in the Technical Interoperability and Performance of annotation Servers (TIPS) task of BioCreative V.5 [2]. This participation allowed us to assess the flexibility, autonomy and efficiency of MER in a realistic scenario. The preliminary results show that our annotation server achieved good reliability and performance indicators.

MER is publicly available in a GitHub repository[2]. The repository contains a small tutorial to help the user start using the program and test it. The remainder of this article will detail the MER tool, and how it was incorporated in our annotation server to participate in TIPS. We end by analyzing and discuss the preliminary results and present future directions.

2 MER

2.1 Input

Before being able to annotate any text, MER requires a lexicon with the list of terms to match. So, if for example we want to recognize terms that are present in ChEBI[3], the user just has to collect all ChEBI terms and store them in a text file, containing in each line one term representing a ChEBI entity. Figure 1 presents an example where four ChEBI compounds are represented by a list of terms based on their ChEBI's name. Currently MER is only performing NER, in the future we may allow the option to include an accession number to each line in the file in order to also perform entity linking.

[1] `https://www.gnu.org/software/bash/`
[2] `https://github.com/lasigeBioTM/MER`
[3] `https://www.ebi.ac.uk/chebi/`

Fig. 2. Each block represents the content of each of the four files created after pre-processing the input file shown in Figure 1.

```
== one-word (...word1.txt) ========================
α.maltose
== two-word (...word2.txt) ========================
nicotinic acid
== more-words (...words.txt) ======================
nicotinic acid d.ribonucleotide
nicotinic acid.adenine dinucleotide phosphate
==  first-two-words (...words2.txt) ===============
nicotinic acid
nicotinic acid.adenine
```

Fig. 3. Example of a given sentence to be annotated (first line), and its one-word and two-word patterns created by MER.

```
α-maltose and nicotinic acid was found, but not
   nicotinic acid D-ribonucleotide

α.maltose|nicotinic|acid|d.ribonucleotide|found|nicotinic|acid

α.maltose nicotinic|acid d.ribonucleotide|found nicotinic
   |nicotinic acid|d.ribonucleotide found|nicotinic acid
```

Fig. 4. Output example of MER for the sentence in Figure 3 and the lexicon in Figure 1

```
0       9       α-maltose
14      28      nicotinic acid
65      79      nicotinic acid
14      45      nicotinic acid D-ribonucleotide
```

2.2 Pre-Processing

Each lexicon has to go through two pre-processing steps. The first step splits the lexicon in three files containing the terms composed by one (one-word), two (two-word) and three or more words (more-words). The second step creates a fourth file containing the first two words (first-two-words) of all the terms in the more-words file. During the above steps, MER makes the following minor modifications to the terms: convert all text to lowercase; contiguous white spaces are replaced by one white space; full stops are removed; leading and trailing white spaces are removed; and all special characters are replaced by a full stop. Since some special characters may cause matching problems, MER assumes that all the special characters (characters that are not alphanumeric or a whitespace, for example, hyphens) can be matched by any other character, so these characters are replaced by a full stop, like in regular expressions. Figure 2 presents the contents of each of the four files created using the terms shown in Figure 1. Note that the word *acid-adenine* was replaced by *acid.adenine*, and the last file presents the first two words of each entry in the third file.

2.3 Recognition

Given an input text, and the name of the lexicon already pre-processed, the goal is to identify which terms of the lexicon are mentioned in the text. The first step of MER is to apply the same minor modifications to the input text as described in the Pre-Processing section, but also remove stop words, and words with less than 3 characters. This will result in a processed input text derived from the original one. Note that MER only recognizes direct matches, if lexical variations of the terms are needed they have to be added in the lexicon, for example by using a stemming algorithm.

A common solution would be to apply grep directly to the input text. However, the execution time is proportional to the size of the lexicon, since each term of the lexicon will correspond to an independent pattern to match. To optimize the execution time we inverted this solution, i.e. we use the words in the processed input text as patterns to be matched against the lexicon file. Since the number of words in the input text is much smaller than the number of terms in the lexicon, grep has much less patterns to match. For example, finding the pattern *nicotinic acid* in the two-word chemical file created for TIPS is more than 100 times faster than using the common solution. This requires the creation of two alternation patterns: i) one-word pattern, with all the words in the input text; and ii) two-word pattern, with all the consecutive pairs of words in the input text. Figure 3 shows an example of these two patterns.

Next, MER creates three background jobs to match the terms composed of: i) one word, ii) two words, and iii) three or more words. The one-word job uses the one-word pattern to find matching terms in the one-word file. Similarly for the two-word job, that uses the two-word pattern and file. The last job uses the two-word pattern to find matches in the two-first-word file, and the resulting matches are then used as a pattern to find terms in the more-words file. The last job is less efficient since it executes grep twice, however the resulting list of matches with the two-first-word file is usually small, so the second execution is negligible. In the end, each job will create a list of matching terms that are mentioned in the input text.

Since the processed input text cannot be used to find the exact position of the term, MER uses the list of matching terms to find their exact position in the original input text. MER uses awk to find the multiple instances of each term in the original input text. The awk tool has the advantage of working well with UTF-8 characters that use more than one byte, in opposition to grep that just counts the bytes to find the position of a match. MER provides partial overlaps, i.e. a shorter term may occur at the same position as a longer one, but not full overlapping matches (same term in the same position). We also developed a test suite to refactor the algorithm with more confidence that nothing is being done incorrectly. The test suite is available in the GitHub repository branch dedicated to development[4].

Figure 4 shows the output of MER when using as input text the sentence in Figure 3, and the lexicon of Figure 1. Note that *nicotinic acid* appears twice at

[4] `https://github.com/lasigeBioTM/MER/tree/dev`

Fig. 5. Number of terms, words, and characters in the lexicons used in TIPS, obtained by using the following shell command: `wc -lmw *.txt`.

```
  #terms     #words      #char #filename
 116616     137702    1027369 CELL LINE AND CELL TYPE.txt
 332167     446423   10397574 CHEMICAL.txt
  26216      92688     808366 DISEASE.txt
  73954      73954     991012 MIRNA.txt
 597867    1372326   11863642 PROTEIN.txt
   8146      26117     228167 SUBCELLULAR STRUCTURE.txt
   5238      16283     126024 TISSUE AND ORGAN.txt
1160204    2165493   25442154 total
```

position 14 and 65, as expected, without affecting the match of *nicotinic acid D-ribonucleotide*.

3 Annotation Server

TIPS is a novel task in BioCreative aiming at the evaluating the performance of NER web servers, based on reliability and performance metrics. The entities to be recognized were not restricted to a particular domain.

The web servers had to respond to single document annotation requests. The servers had to be able to retrieve the text from documents in the patent server, the abstract server and PubMed, without using any kind of cache for the text or for the annotations. The annotations had to be provided in, at least, one of the following formats: BeCalm JSON, BeCalm TSV, BioC XML or BioC JSON.

3.1 Lexicons

The first step to participate in TIPS was to select the data sources from which we could collect terms related with the following accepted categories: Cell line and cell type: Cellosaurus[5]; Chemical: HMDB[6], ChEBI[7] and ChEMBL[8]; Disease: Human Disease Ontology[9]; miRNA: miRBase[10]; Protein: Protein Ontology[11]; Subcellular structure: cellular component aspect of Gene Ontology[12]; Tissue and organ: tissue and organ subsets of UBERON[13].

All these data files suffered a post-extraction processing which consisted in lowercasing all terms, deleting leading and trailing white spaces and removing

[5] http://web.expasy.org/cellosaurus/
[6] http://www.hmdb.ca/
[7] https://www.ebi.ac.uk/chebi/
[8] https://www.ebi.ac.uk/chembl/
[9] http://www.obofoundry.org/ontology/doid.html
[10] http://www.mirbase.org/
[11] http://www.obofoundry.org/ontology/pr.html
[12] http://www.geneontology.org/
[13] http://uberon.github.io/

Fig. 6. Output example of MER using BeCalm TSV format for the sentence in Figure 3 and the lexicon in Figure 1

```
1 A 0   9 0.54488  α-maltose                        lexicon 1
1 A 14 28 0.621077 nicotinic acid                   lexicon 1
1 A 48 62 0.621077 nicotinic acid                   lexicon 1
1 A 48 79 0.708793 nicotinic acid D-ribonucleotide lexicon 1
```

repeated terms. Since overlapping annotations were not allowed, we created another lexicon containing terms that appeared on more than one of the other lexicons. The terms matched to this lexicon were considered to be of the category Unknown, as suggested by the organization. The software to extract the list of terms from the above data sources can be found in the GitHub repository branch dedicated to TIPS[14].

Figure 5 shows the number of terms, the number of words, and the number of characters of each lexicon created. MER was therefore recognizing more than 1M terms composed of more than 2M words and more than 25M characters. All lexicons are available for reuse as a zip file [15].

3.2 Input and Output

We adapted MER to provide the annotations in the BeCalm TSV format. Thus, besides the input text and the lexicon, MER had to receive also the document identifier and the section as input. In Figure 6, the document identifier is 1 and section is A. The score column is calculated by $1 - 1/\ln(nc)$, where nc represents the number of characters of the recognized term. This is based on the assumption that longer terms are less ambiguous, and in that case the match should have a higher confidence score. Note that MER only recognizes terms with three or more characters, so the minimum score is 0.08976 and the score is always lower than 1. An instance of MER with this output format and using the lexicons described above is available through a RESTful Web service[16].

We used jq[17] a command-line JSON processor to parse the requests. The download of each document was implemented using the popular *cURL* tool, and we developed a specific parser for each data source to extract the text to be annotated. The parsers are also available at the TIPS branch[18].

[14] https://github.com/lasigeBioTM/MER/tree/biocreative2017/data_parsers
[15] https://github.com/lasigeBioTM/MER/raw/biocreative2017/data/TIPS_MER_lexicons_Jan2017.zip
[16] http://labs.fc.ul.pt/mer/
[17] https://stedolan.github.io/jq/
[18] https://github.com/lasigeBioTM/MER/tree/biocreative2017/external_services

	Patent Server	Abstract Server	PubMed	Total
Total requests	88,556	135,938	92,811	317,305
Total predictions	1,388k	4,035k	2,766k	8,189k
Total processing time	2d 01h	4d 18h	3d 19h	10d 14h
Mean predictions/document	15.6	29.7	29.8	22.4
Mean processing time/document (s)	2.02	3.03	3.57	2.90
MDTV	-	-	-	0.00238
MPDV	-	-	-	0.0184

Table 1. Annotation server performance values at April 20, 2017. MTDV (mean time in seconds per document volume) = Total processing time(s)/sum of document sizes (bytes). MPDV (total predictions per document volume) = Total predictions/sum of document sizes (bytes).

3.3 Infrastructure

Our annotation server was deployed in a cloud infrastructure composed of three Virtual Machines (VM). Each VM had 8GB of RAM and 4 CPUs @ 1.7 GHz, using CentOS Linux release 7.3.1611 (Core) as the operating system. We selected one VM (primary) to process the requests, distribute the jobs, and execute MER. The other two VMs (secondary) just execute MER. We installed the NGINX HTTP server running CGI scripts given its high performance when compared with other web servers [3]. We also used the Task Spooler[19] tool to manage and distribute within the VMs the jobs to be processed.

The server is configured to receive the REST API requests defined in the BeCalm API documentation. Each request is distributed to one of the three VMs according to the least-connected method of NGINX. When a *getAnnotations* request is received, the server first downloads the documents from the respective sources, and then processes the title and abstract of each document in the same VM. Two jobs are spawned in background, corresponding to the title and abstract. Each annotation server handles all the entity types mentioned in Figure 5, spawning a separate job for each entity type. The name of the entity type is added as another column to the output of Figure 4. These jobs run in parallel since they are independent from each other and the output of each job can be easily merged into a final TSV output file. When a job finishes processing, a script checks if the other jobs associated with the same requests have also finished processing. If that is the case, then the results of every job are concatenated and sent back to BeCalm using the *saveAnnotations* method.

3.4 Results

Table 3.3 presents the performance values for our annotation server available at the BeCalm web interface on April 20, 2017. Our minimal annotation server was able to efficiently process the documents by taking less than 3 seconds on average

[19] `http://vicerveza.homeunix.net/~viric/soft/ts/`

without using any type of cache. We note that all documents, irrespectively of the source, were annotated using all the entity types presented in Section 3.1.

We compared the time necessary to process the same sentence on the same hardware using MER and a more complex machine learning system, IBEnt [1], using the sentence of Figure 3. While IBEnt took 8.25 seconds to process the sentence, MER took only 0.098 seconds. Although IBEnt is optimized for batch processing, therefore reducing the time per document as the number of documents increases, MER is still 84 times faster than IBEnt in this experiment. Thus, besides being easy to install and configure, MER is also a highly efficient and scalable NER tagger.

4 Conclusions

We presented MER a minimal NER tagger that was developed with the concepts of flexibility, autonomy and efficiency in mind. MER is flexible since it can be extended with any lexicon composed of a simple list of terms. MER is autonomous since it only requires a GNU Bash shell with awk and grep tools, which are omnipresent in almost any Unix-like operating systems. MER is efficient since it takes advantage of the high-performance capacity of grep as a file pattern matcher.

MER was integrated in an annotation server deployed in a cloud infrastructure for participating in the TIPS task of BioCreative V.5. Our server was fully developed in-house with minimal software dependencies and is open-source. Without using any kind of cache, our server was able to process each document in less than 3 seconds on average. In the future, we intend to implement the entity linking functionality in MER, without undermining its flexibility, autonomy and efficiency.

Acknowledgments. This work was produced with the support of the Portuguese National Distributed Computing Infrastructure (`http://www.incd.pt`). This work was supported by FCT through funding of the LaSIGE Research Unit, ref. UID/CEC/00408/2013

References

1. Lamurias, A., Filipe, L., Couto, F.M.: IBEnt: Chemical Entity Mentions in Patents using CheBI. Proceedings of the BioCreative V.5 Challenge Evaluation Workshop (2017)
2. Perez, M.P., Rodriguez, G.P., Mguez, A.B., Riverola, F.F., Valencia, A., Krallinger, M., Lourenco, A.: Benchmarking biomedical text mining web servers at BioCreative V.5: the technical Interoperability and Performance of annotation Servers - TIPS track. In: Proceedings of the BioCreative V.5 Challenge Evaluation Workshop. pp. 12–21 (2017)
3. Reese, W.: Nginx: the high-performance web server and reverse proxy. Linux Journal 2008(173), 2 (2008)