Introduction to RIAMsystem

2019-03-18

Introduction

Matching text phrases to medical concepts automatically is important to facilitate tasks such as search, classification or organization of biomedical textual contents. The task is to identify mentions of concepts in textual content using terms from a controlled vocabulary. Many concept recognition systems already exist like MetaMap, MGrep or Noble.[1]

IAMsystem is a fast and lightweight algorithm for concepts recognition. It normalizes, tokenizes and stores a controlled vocabulary in a tree data structure for fast dictionary-lookup. Description of the algorithm is available in this article : https://arxiv.org/abs/1807.03674

The algorithm was implemented in Java and is open source, available here : https://github.com/scossin/IAMsystem.

RIAMsystem is an R package that uses the rJava package to create objects and invoke methods of the IAMsystem Java implementation, in a jar file (Java ARchive) attached to the package. It permits to use the IAMsystem algorithm from R.

This tutorial explains how it works and helps to get started.

Quick example

First we need to create a termDetector instance of the TermDetector class in Java, then we add terms from a controlled vocabulary (abbreviations and stopwords are optionals) then the algorithm will detect these terms in textual content :

```
termDetector <- RIAMsystem::newTermDetector()
RIAMsystem::addTerms(termDetector = termDetector,
                     terms = "ulcères gastriques",
                     codes = "K25")
RIAMsystem::addStopwords(termDetector = termDetector,stopwords = c("de","l"))
RIAMsystem::addAbbreviations(termDetector = termDetector,
                             token = "ulceres",
                             abbreviation = "ul")
RIAMsystem::addAbbreviations(termDetector = termDetector,
                             token = "gastriques",
                             abbreviation = "estomac")
RIAMsystem::detect(termDetector = termDetector,
                   text="le patient a un ul. de l'estomac")
```

```
##   candidateTermString   candidateTerm startPosition endPosition code
## 1    ul. de l'estomac ul de l estomac            16          31  K25
##      normalizedLabel
## 1 ulceres gastriques
```

In this example, “ul” is matched to “ulceres” then “de” and “l”’ are removed because they are stopwords, then “estomac” is a synonym of “gastriques” so the term “ulcères gastriques” is detected starting at position 16 and ending at position 31.

[1] Tseytlin E, Mitchell K, Legowski E, Corrigan J, Chavan G, Jacobson RS. NOBLE – Flexible concept recognition for large-scale biomedical natural language processing. BMC Bioinformatics. 14 janv 2016;17(1):32.

All the terms are normalized through accents (diacritical marks) and punctuation removal, lowercasing and stopwords removal. The same normalization steps are performed for the text to analyze.

Stopwords

We need to add terms (with associated codes) to the algorithm. Adding stopwords is optional. The order (adding stopwords BEFORE terms) is very important since the algorithm will try to remove stopwords in the terms of the controlled vocabulary. See the difference between :

1) Adding stopwords before the terms :

```
termDetector <- RIAMsystem::newTermDetector()
RIAMsystem::addStopwords(termDetector = termDetector,
                         stopwords = c("de","l"))
RIAMsystem::addTerms(termDetector = termDetector,
                     terms = "ulcère de l'estomac",
                     codes = "K25")
RIAMsystem::detect(termDetector = termDetector,
                   text="le patient a un ulcère estomac")
```

```
##   candidateTermString  candidateTerm startPosition endPosition code
## 1      ulcère estomac ulcere estomac            16          29  K25
##   normalizedLabel
## 1  ulcere estomac
```

The stopwords “de” and “l” were removed from the term “ulcère de l’estomac”, so “ulcère estomac” is detected.

2) Adding stopwords after the terms :

```
termDetector <- RIAMsystem::newTermDetector()
RIAMsystem::addTerms(termDetector = termDetector,
                     terms = "ulcère de l'estomac",
                     codes = "K25")
RIAMsystem::addStopwords(termDetector = termDetector,
                         stopwords = c("de","l"))
RIAMsystem::detect(termDetector = termDetector,
                   text="le patient a un ulcère estomac")
```

```
## [1] candidateTermString candidateTerm       startPosition
## [4] endPosition         code                normalizedLabel
## <0 rows> (or 0-length row.names)
```

The stopwords “de” and “l” were NOT removed in the term “ulcère de l’estomac”, so “ulcère estomac” is not detected.

Abbreviations

Abbreviations work at the token level, so if “ac” is the abbreviation of “acide” then all the terms containing the word “ac”, at any position, will be concerned.

```
termDetector <- RIAMsystem::newTermDetector()
RIAMsystem::addTerms(termDetector = termDetector,
                     terms = "ACIDE ALENDRONIQUE",
                     codes = "BNg4")
RIAMsystem::addAbbreviations(termDetector = termDetector,
```

```
                              token = "acide",
                              abbreviation = "ac")
RIAMsystem::addAbbreviations(termDetector = termDetector,
                              token = "ALENDRONIQUE",
                              abbreviation = "alendronic")
RIAMsystem::detect(termDetector = termDetector,text="le patient prend de l'ac alendronic")
```

```
##   candidateTermString candidateTerm startPosition endPosition code
## 1       ac alendronic ac alendronic             22          34 BNg4
##      normalizedLabel
## 1 acide alendronique
```

Abbreviation work for multiwords like “avc” for “accident vasculaire cérébral”.

```
termDetector <- RIAMsystem::newTermDetector()
RIAMsystem::addTerms(termDetector = termDetector,
                     terms = "accident vasculaire cérébral sylvien gauche",
                     codes = "I63")
RIAMsystem::addAbbreviations(termDetector = termDetector,
                              token = "accident vasculaire cérébral",
                              abbreviation = "avc")
RIAMsystem::detect(termDetector = termDetector,
                   text="le patient a eu un avc sylvien gauche")
```

```
##   candidateTermString      candidateTerm startPosition endPosition code
## 1  avc sylvien gauche avc sylvien gauche             19          36  I63
##                               normalizedLabel
## 1 accident vasculaire cerebral sylvien gauche
```

Load a terminology

Romedi is a French terminology of brand names and ingredients. It contains more than 10 000 terms. It takes approximately 10 seconds to load this terminology with a loop using ‘RIAMsystem::addTerms’ function.

```
data("romedi", package="RIAMsystem")
colnames(romedi)
```

```
## [1] "instance"  "type"      "label"     "typeLabel"
```

The first column, ‘instance,’ is the code.
The third column, ‘label’, is the term.
Java uses zero-based indexing, so the first column (colCode) has index 0 and the third column (colLabel) has index 2.

To load quickly a terminology, write it to a CSV file and use the ‘RIAMsystem::loadTerminoCSV’ function :

```
write.table(romedi, "romedi.csv",sep="\t",col.names = T, row.names = F)
RIAMsystem::loadTerminoCSV(termDetector = termDetector,
                           file = "./romedi.csv",
                           sep="\t",
                           colLabel = 2,
                           colCode = 0)
RIAMsystem::detect(termDetector, "le patient prend de l'escitalopram et de la dompéridone")
```

```
##   candidateTermString candidateTerm startPosition endPosition
## 1        escitalopram  escitalopram             22          33
## 2         dompéridone   domperidone             44          54
```

```
##                                                                  code
## 1 "http://www.romedi.fr/romedi/INrth34o4uohkjffrahl2pepb44lkgp5ui"
## 2 "http://www.romedi.fr/romedi/BNrm21178t8uhk4brb5n0h09umjp3257n4"
##   normalizedLabel
## 1    escitalopram
## 2     domperidone
```

Fuzzy matching

Fuzzy matching is not available yet in this R package.

Reference

Cossin et al. IAM at CLEF eHealth 2018 : Concept Annotation and Coding in French Death Certificates. https://arxiv.org/abs/1807.03674

Acknowledgement

This annotation tool is part of the Drugs Systematized Assessment in real-liFe Environment (DRUGS-SAFE) research platform that is funded by the French Medicines Agency (Agence Nationale de Sécurité du Médicament et des Produits de Santé, ANSM). This platform aims at providing an integrated system allowing the concomitant monitoring of drug use and safety in France.