

Package ‘nametagger’

July 22, 2025

Type Package

Title Named Entity Recognition in Texts using 'NameTag'

Version 0.1.3

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Wraps the 'nametag' library <<https://github.com/ufal/nametag>>, allowing users to find and extract entities (names, persons, locations, addresses, ...) in raw text and build your own entity recognition models. Based on a maximum entropy Markov model which is described in Strakova J., Straka M. and Hajic J. (2013) <https://ufal.mff.cuni.cz/~straka/papers/2013-tds_ner.pdf>.

URL <https://github.com/bnosac/nametagger>

License MPL-2.0

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.5), utils

Suggests udpipe (>= 0.2)

LinkingTo Rcpp

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph],
BNOSAC [cph],
Institute of Formal and Applied Linguistics, Faculty of Mathematics and
Physics, Charles University in Prague, Czech Republic [cph],
Milan Straka [ctb, cph],
Jana Straková [ctb, cph]

Repository CRAN

Date/Publication 2023-09-13 11:00:02 UTC

Contents

europenanews	2
europena_read	3
nametagger	3
nametagger_download_model	6
nametagger_load_model	6
nametagger_options	7
predict.nametagger	10
write_nametagger	11
Index	12

europenanews	<i>Tagged news paper articles from Europeana</i>
--------------	--

Description

BIO-tagged news articles in different languages extracted from <https://github.com/EuropeanaNewspapers/ner-corpora> using [europena_read](#)

- Dutch from the Koninklijke Bibliotheek
- Austrian from the National Library of Austria
- German from the Dr. Friedrich Teßmann Library
- French from the National Library of France, in cooperation with LIP6-ACASA

References

Europeana Newspapers project, (2014), KB Europeana Newspapers NER Dataset. KB Lab: The Hague. <http://lab.kb.nl/dataset/europeana-newspapers-ner>

Examples

```
data(europenanews)
str(europenanews)
```

europeana_read	<i>Read Europeana Newspaper data</i>
----------------	--------------------------------------

Description

Read Europeana Newspaper data. Data is made available at <https://github.com/EuropeanaNewspapers/ner-corpora> under the CC0 license.

Usage

```
europeana_read(x)
```

Arguments

x	path to the file
---	------------------

Value

a data.frame with columns doc_id, sentence_id, token, entity

Examples

```
library(udpipe)
r <- "https://raw.githubusercontent.com/EuropeanaNewspapers/ner-corpora/master"

x <- file.path(r, "enp_NL.kb.bio", "enp_NL.kb.bio")
x <- europeana_read(x)
x <- file.path(r, "enp_FR.bnf.bio", "enp_FR.bnf.bio")
x <- europeana_read(x)
x <- file.path(r, "enp_DE.sbb.bio", "enp_DE.sbb.bio")
x <- europeana_read(x)
x <- file.path(r, "enp_DE.onb.bio", "enp_DE.onb.bio")
x <- europeana_read(x)
x <- file.path(r, "enp_DE.lft.bio", "enp_DE.lft.bio")
x <- europeana_read(x)
```

nametagger	<i>Train a Named Entity Recognition Model using NameTag</i>
------------	---

Description

Train a Named Entity Recognition Model using NameTag. Details at <https://ufal.mff.cuni.cz/nametagger/1>.

Usage

```

nametagger(
  x.train,
  x.test = NULL,
  iter = 30L,
  lr = c(0.1, 0.01),
  lambda = 0.5,
  stages = 1L,
  weight_missing = -0.2,
  control = nametagger_options(token = list(window = 2)),
  type = if (inherits(control, "nametagger_options")) control$type else "generic",
  tagger = if (inherits(control, "nametagger_options")) control$tagger else "trivial",
  file = if (inherits(control, "nametagger_options")) control$file else
    "nametagger.ner"
)

```

Arguments

<code>x.train</code>	a file with training data or a data.frame which can be passed on to write_nametagger
<code>x.test</code>	optionally, a file with test data or a data.frame which can be passed on to write_nametagger
<code>iter</code>	the number of iterations performed when training each stage of the recognizer. With more iterations, training take longer (the recognition time is unaffected), but the model gets over-trained when too many iterations are used. Values from 10 to 30 or 50 are commonly used.
<code>lr</code>	learning rates used. Should be a vector of length 2 where • element 1: learning rate used in the first iteration of SGD training method of the log-linear model. Common value is 0.1. • element 2: learning rate used in the last iteration of SGD training method of the log-linear model. Common values are in range from 0.1 to 0.001, with 0.01 working reasonably well.
<code>lambda</code>	the value of Gaussian prior imposed on the weights. In other words, value of L2-norm regularizer. Common value is either 0 for no regularization, or small real number like 0.5.
<code>stages</code>	the number of stages performed during recognition. Common values are either 1 or 2. With more stages, the model is larger and recognition is slower, but more accurate.
<code>weight_missing</code>	default value of missing weights in the log-linear model. Common values are small negative real numbers like -0.2.
<code>control</code>	the result of a call to nametagger_options a file with predictive feature transformations serving as predictive elements in the model
<code>type</code>	either one of 'generic', 'english' or 'czech'
<code>tagger</code>	either one of 'trivial' (no lemma used in the training data), 'external' (you provided your own lemma in the training data)
<code>file</code>	path to the filename where the model will be saved

Value

an object of class `nametagger` containing an extra list element called `stats` containing information on the evolution of the log probability and the accuracy on the training and optionally the test set

Examples

```
data(europeannews)
x <- subset(europeannews, doc_id %in% "enp_NL.kb.bio")
traindata <- subset(x, sentence_id > 100)
testdata <- subset(x, sentence_id <= 100)
path <- "nametagger-nl.ner"

opts <- nametagger_options(file = path,
                           token = list(window = 2),
                           token_normalisedsuffix = list(window = 0, from = 1, to = 4),
                           ner_previous = list(window = 2),
                           time = list(use = TRUE),
                           url_email = list(url = "URL", email = "EMAIL"))

model <- nametagger(x.train = traindata,
                   x.test = testdata,
                   iter = 30, lambda = 0.5,
                   control = opts)

model
model$stats
plot(model$stats$iteration, model$stats$logprob, type = "b")
plot(model$stats$iteration, model$stats$accuracy_train, type = "b", ylim = c(95, 100))
lines(model$stats$iteration, model$stats$accuracy_test, type = "b", lty = 2, col = "red")

predict(model,
        "Ik heet Karel je kan me bereiken op paul@duchanel.be of www.duchanel.be",
        split = "[[:space:]]+")

features <- system.file(package = "nametagger",
                        "models", "features_default.txt")
cat(readLines(features), sep = "\n")
path_traindata <- "traindata.txt"

write_nametagger(x, file = path_traindata)

model <- nametagger(path_traindata, iter = 30, control = features, file = path)
model
```

`nametagger_download_model`*Download a Nametag model*

Description

Download a Nametag model. Note that models have licence CC-BY-SA-NC. More details at <https://ufal.mff.cuni.cz/nametag/1>.

Usage

```
nametagger_download_model(  
  language = c("english-conll-140408"),  
  model_dir = tempdir()  
)
```

Arguments

language	'english-conll-140408'
model_dir	a path where the model will be downloaded to.

Value

an object of class `nametagger`

References

<https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3118>

Examples

```
model <- nametagger_download_model("english-conll-140408", model_dir = tempdir())
```

`nametagger_load_model` *Load a Named Entity Recognition*

Description

Load a Named Entity Recognition from your hard disk

Usage

```
nametagger_load_model(file)
```

Arguments

file character string with the path to the file on disk

Value

an object of class `nametagger`

Examples

```
path <- system.file(package = "nametagger", "models", "exampltagger.ner")
model <- nametagger_load_model(path)
model
```

<code>nametagger_options</code>	<i>Define text transformations serving as predictive elements in the nametagger model</i>
---------------------------------	---

Description

Define text transformations which are relevant in predicting your entity. Typical text transformations are the token itself, the lemma, the parts of speech tag of the token or the token/lemma's and parts of speech tags in the neighbourhood of the word.

Each argument should be a list with elements `use` and `window`.

- `use` is a logical indicating if the transformation should be used in the model.
- `window` specifies how many adjacent words can observe the feature template value of a given word. The default value of 0 denotes only the word in question, no surrounding words.

If you specify the argument without specifying `use`, it will by default use it. For arguments `brown`, `gazetteers` and `gazetteers_enhanced`, see the examples and the documentation at <https://ufal.mff.cuni.cz/nametagger/1>.

Usage

```
nametagger_options(
  file = "nametagger.ner",
  type = c("generic", "english", "czech"),
  tagger = c("trivial", "external"),
  token = list(use = FALSE, window = 1),
  token_capitalised = list(use = FALSE, window = 0),
  token_normalised = list(use = FALSE, window = 0),
  token_normalisedsuffix = list(use = FALSE, window = 0, from = 1, to = 4),
  lemma = list(use = FALSE, window = 0),
  lemma_capitalised = list(use = FALSE, window = 0),
  lemma_normalised = list(use = FALSE, window = 0),
```

```

lemma_normalisedsuffix = list(use = FALSE, window = 0, from = 1, to = 4),
pos = list(use = tagger == "external", window = 0),
time = list(use = FALSE, window = 0),
url_email = list(use = FALSE, url = "URL", email = "EMAIL"),
ner_previous = list(use = FALSE, window = 0),
brown = list(use = FALSE, window = 0),
gazetteers = list(use = FALSE, window = 0),
gazetteers_enhanced = list(use = FALSE)
)

```

Arguments

file	path to the filename where the model will be saved
type	either one of 'generic', 'english' or 'czech'. See the documentation at the documentation at https://ufal.mff.cuni.cz/nametag/1 .
tagger	either one of 'trivial' (no lemma used in the training data), 'external' (you provided your own lemma in the training data)
token	use forms as features
token_capitalised	use capitalization of form as features
token_normalised	use case normalized (first character as-is, others lowercased) forms as features
token_normalisedsuffix	shortest longest – use suffixes of case normalized (first character as-is, others lowercased) forms of lengths between shortest and longest
lemma	use raw lemmas as features
lemma_capitalised	use capitalization of raw lemma as features
lemma_normalised	use case normalized (first character as-is, others lowercased) raw lemmas as features
lemma_normalisedsuffix	shortest longest – use suffixes of case normalized (first character as-is, others lowercased) raw lemmas of lengths between shortest and longest
pos	use parts-of-speech tags as features
time	recognize numbers which could represent hours, minutes, hour:minute time, days, months or years
url_email	If an URL or an email is detected, it is immediately marked with specified named entity type and not used in further processing. The specified entity label to use can be specified with url and email (in that sequence)
ner_previous	use named entities predicted by previous stage as features
brown	file [prefix_lengths] – use Brown clusters found in the specified file. An optional list of lengths of cluster prefixes to be used in addition to the full Brown cluster can be specified.

- gazetteers** [files] – use given files as gazetteers. Each file is one gazetteers list independent of the others and must contain a set of lemma sequences, each on a line, represented as raw lemmas separated by spaces.
- gazetteers_enhanced** (formlrawlemmalrawlemmas) (embed_in_model|out_of_model) file_base entity [file_base entity ...] – use gazetteers from given files. Each gazetteer contains (possibly multiword) named entities per line. Matching of the named entities can be performed either using form, disambiguated rawlemma of any of rawlemmas proposed by the morphological analyzer. The gazetteers might be embedded in the model file or not; in either case, additional gazetteers are loaded during each startup. For each file_base specified in GazetteersEnhanced templates, three files are tried:
- file_base.txt: gazetteers used as features, representing each file_base with a unique feature
 - file_base.hard_pre.txt: matched named entities (finding non-overlapping entities, preferring the ones starting earlier and longer ones in case of ties) are forced to the specified entity type even before the NER model is executed
 - file_base.hard_post.txt: after running the NER model, tokens not recognized as entities are matched against the gazetteers (again finding non-overlapping entities, preferring the ones starting earlier and longer ones in case of ties) and marked as entity type if found

Value

an object of class `nametagger_options` with transformation information to be used by `nametagger`

Examples

```
opts <- nametagger_options(token = list(window = 2))
opts
opts <- nametagger_options(time = list(use = TRUE, window = 3),
                           token_capitalised = list(use = TRUE, window = 1),
                           ner_previous = list(use = TRUE, window = 5))
opts
opts <- nametagger_options(
  lemma_capitalised = list(window = 3),
  brown = list(window = 1, file = "path/to/brown/clusters/file.txt"),
  gazetteers = list(window = 1,
                    file_loc = "path/to/txt/file1.txt",
                    file_time = "path/to/txt/file2.txt"))
opts
opts <- nametagger_options(
  lemma_capitalised = list(window = 3),
  brown = list(window = 2,
               file = "path/to/brown/clusters/file.txt"),
  gazetteers_enhanced = list(
    loc = "LOC", type_loc = "form", save_loc = "embed_in_model", file_loc = "path/to/loc.txt",
    time = "TIME", type_time = "form", save_time = "embed_in_model", file_time = "path/to/time.txt")
)
```

opts

predict.nametagger *Perform Named Entity Recognition on tokenised text*

Description

Perform Named Entity Recognition on tokenised text using a nametagger model

Usage

```
## S3 method for class 'nametagger'
predict(object, newdata, split = "[[:space:]]+", ...)
```

Arguments

object	an object of class nametagger as returned by nametagger_load_model
newdata	a data.frame with tokenised sentences. This data.frame should contain the columns doc_id, sentence_id and text where text contains tokens in vertical format, meaning each token is put on a new line. Column doc_id should be of type character, column sentence_id of type integer.
split	a regular expression used to split newdata. Only used if newdata is a character vector containing text which is not tokenised
...	not used

Value

a data.frame with columns doc_id, sentence_id, token and entity

Examples

```
path <- system.file(package = "nametagger", "models", "exampletagger.ner")
model <- nametagger_load_model(path)
model

x <- c("I ga naar Brussel op reis.", "Goed zo dat zal je deugd doen Karel")
entities <- predict(model, x, split = "[[:space:]][:punct:]]+")
entities

model <- nametagger_download_model("english-conll-140408", model_dir = tempdir())

x <- data.frame(doc_id = c(1, 1, 2),
                sentence_id = c(1, 2, 1),
                text = c("I\nlive\nin\nNew\nYork\nand\nI\nwork\nfor\nApple\nInc.",
                        "Why\ndon't\nyou\ncome\nvisit\nme",
                        "Good\nnews\nfrom\nAmazon\nas\nJohn\nworks\nthere\n."))
entities <- predict(model, x)
entities
```

write_nametagger	<i>Save a tokenised dataset as nametagger train data</i>
------------------	--

Description

Save a tokenised dataset as nametagger train data

Usage

```
write_nametagger(x, file = tempfile(fileext = ".txt", pattern = "nametagger_"))
```

Arguments

x	a tokenised data.frame with columns doc_id, sentence_id, token containing 1 row per token. In addition it can have columns lemma and pos representing the lemma and the parts-of-speech tag of the token
file	the path to the file where the training data will be saved

Value

invisibly an object of class `nametagger_traindata` which is a list with elements

- `data`: a character vector of text in the nametagger format
- `file`: the path to the file where data is saved to

Examples

```
data(europeannews)
x <- subset(europeannews, doc_id %in% "enp_NL.kb.bio")
x <- head(x, n = 250)

path <- "traindata.txt"

bio <- write_nametagger(x, file = path)
str(bio)
```

Index

européana_read, [2](#), [3](#)

europeannews, [2](#)

nametagger, [3](#), [9](#)

nametagger_download_model, [6](#)

nametagger_load_model, [6](#), [10](#)

nametagger_options, [4](#), [7](#)

predict.nametagger, [10](#)

write_nametagger, [4](#), [11](#)