Package 'readme'

December 12, 2018

Title An Algorithm for Text Quantification

Version 2.0

Authors 'Gary King <king@harvard.edu> [aut], Anton Strezhnev <astrezhn@law.upenn.edu> [aut, cre], Connor Jerzak <cjerzak@g.harvard.edu> [aut, cre]'

Description An R package for estimating category proportions in an unlabeled set of documents given a labeled set, by implementing the method described in Jerzak, King, and Strezhnev (2018, copy at \{ }url{http://GaryKing.org/words}). This method is meant to improve on the ideas in Hopkins and King (2010), which introduced a quantification algorithm that harnesses the Law of Total Expectation. We apply this law in a feature space we craft minimizes the error of the resulting estimate. Automatic differentiation, stochastic gradient descent, and batch re-normalization are used to carry out the optimization. Other pre-processing functions are available, as well as an interface to the earlier version of the algorithm for comparison. The package also provides users with the ability to extract the generated features for use in other tasks.

Depends R (>= 3.3.3)

License Creative Commons Attribution-Noncommercial-No Derivative Works 4.0, for academic use only.

Encoding UTF-8

LazyData true

Maintainer 'Connor Jerzak' <connor.jerzak@gmail.com>

Imports tensorflow, limSolve, tokenizers, data.table, optmatch

RoxygenNote 6.1.0

R topics documented:

readme-package *A algorithm for quantification that harnesses the Law of Total Expectations in an optimal feature space*

Description

An R package for estimating category proportions in an unlabeled set of documents given a labeled set, by implementing the method described in Jerzak, King, and Strezhnev (2018, copy at http://GaryKing.org/words). This method is meant to improve on the ideas in Hopkins and King (2010), which introduced a quantification algorithm that harnesses the Law of Total Expectation. We apply this law in a feature space we craft minimizes the error of the resulting estimate. Automatic differentiation, stochastic gradient descent, and batch re-normalization are used to carry out the optimization. Other pre-processing functions are available, as well as an interface to the earlier version of the algorithm for comparison. The package also provides users with the ability to extract the generated features for other tasks.

The package provides two main functions: `undergrad` and `readme`.

- `undergrad` takes as an input a word vector corpus (or pointer to such a corpus) and a vector housing cleaned text for cross-referencing with the vector corpus. It returns document-level summaries of each of the dimensions of the word vectors (`10th`, `50th`, and `90th` quantiles of each dimension within each document are calculated). Options also exist for generating a document-term matrix from the text. Useful for those wanting control over the linkup between documents and word vector corpus.
- `readme` takes as an input raw text (or optionally, the output from `undergrad`). It also takes as an input an indicator vector denoting which documents are labeled and a vector indicating category membership (`NA`s for unlabeled documents). The algorithm then generates an optimal projection for harnessing the Law of Total Expectation in calculating the estimated category proportions in the unlabeled set.

Usage

For advice on usage, see **Examples**. Many users will just interface with the `readme` function, as this approach takes care of much of the pre-processing in an automatic fashion. Some users may want more control over the linkup between the word vector corpus and the raw text; in that case, combining `undergrad` with `readme` is a good option.

For bug reports or support, please contact `<connor.jerzak@gmail.com>`.

Authors

- Connor Jerzak, Anton Strezhnev, and Gary King.
- Maintainer: Connor Jerzak <cjerzak@gmail.com>

References

- Hopkins, Daniel, and King, Gary (2010), *A Method of Automated Nonparametric Content Analysis for Social Science*, *American Journal of Political Science*, Vol. 54, No. 1, January 2010, p. 229-247. https://gking.harvard.edu/files/words.pdf
- Jerzak, Connor, King, Gary, and Strezhnev, Anton. Working Paper. *An Improved Method of Automated Nonparametric Content Analysis for Social Science*. https://GaryKing.org/words

Examples

```
#set seed
set.seed(1)

#Generate synthetic 25-d word vector corpus.
my_wordVecs <- matrix(rnorm(11*25), ncol = 25)
row.names(my_wordVecs) <- c("the","true", "thine", "stars", "are" , "fire", ".", "to", "own", "self", "be")

#Generate 100 ``documents'' of between 5-10 words each.
my_documentText <- replicate(100, paste(sample(row.names(my_wordVecs), sample(5:10, 1), replace = T), collap

#Assign labeled/unlabeled sets. The first 50 will be labeled; the rest unlabeled.
my_labeledIndicator <- rep(1, times = 100)
my_labeledIndicator[51:100] <- 0

#Assign category membership randomly
my_categoryVec <- sample(c("C1", "C2", "C3", "C4"), 100, replace = T)
true_unlabeled_pd <- prop.table(table(my_categoryVec[my_labeledIndicator==0]))
my_categoryVec[my_labeledIndicator == 0] <- NA

#Get word vector summaries
my_dfm <- undergrad(documentText = my_documentText, wordVecs = my_wordVecs)

#perform estimation
readme_results <- readme(dfm = my_dfm,
                         labeledIndicator = my_labeledIndicator,
                         categoryVec = my_categoryVec,
                         nboot = 2)
print(readme_results$point_readme)
```

`cleanme` *cleanme*

Description

Standard preprocessing code for ASCII texts. Removes HTML tags, URLs, linebreaks. Converts standard emoticons to tokens. Removes non-informative punctuation.

Usage

```
cleanme(my_text)
```

Arguments

`my_text`	Vector of character strings containing the raw document texts.

Value

A vector of character strings with the processed texts, each token is separated by a space.

readme *readme*

Description

Implements the quantification algorithm described in Jerzak, King, and Strezhnev (2018) which is meant to improve on the ideas in Hopkins and King (2010). Employs the Law of Total Expectation in a feature space that is tailoed to minimize the error of the resulting estimate. Automatic differentiation, stochastic gradient descent, and batch re-normalization are used to carry out the optimization. Takes an inputs (a.) a vector holding the raw documents (1 entry = 1 document), (b.) a vector indicating category membership (with NAs for the unlabeled documents), and (c.) a vector indicating whether the labeled or unlabeled status of each document. Other options exist for users wanting more control over the pre-processing protocol (see undergrad and the dfm parameter).

Usage

```
readme(dfm, labeledIndicator, categoryVec, nboot = 4, sgd_iters = 2000,
  sgd_momentum = 0.9, numProjections = 20, mLearn = 0.01,
  dropout_rate = 0.5, kMatch = 3, nBoot_matching = 100,
  batchSizePerCat = 10, bootSizePerCat = 20, minMatch = 10,
  verbose = F, diagnostics = F, justTransform = F, winsorize = T)
```

Arguments

dfm	'document-feature matrix'. A data frame where each row represents a document and each column a unique feature.
labeledIndicator	An indicator vector where each entry corresponds to a row in dfm. 1 represents document membership in the labeled class. 0 represents document membership in the unlabeled class.
categoryVec	An factor vector where each entry corresponds to the document category. The entires of this vector should correspond with the rows of dtm. If wordVecs_corpus, wordVecs_corpusPointer, and dfm are all NULL, readme will download and use the GloVe 50-dimensional embeddings trained on Wikipedia.
nboot	A scalar indicating the number of times the estimation procedure will be re-run (useful for reducing the variance of the final output).
sgd_iters	How many stochastic gradient descent iterations should be used? Input should be a positive number.
sgd_momentum	Momentum parameter for stochastic gradient descent (default = 0.90)
numProjections	How many projections should be calculated? Input should be a positive number. Minimum number of projections = number of categories + 2.
mLearn	Learning parameter for moments in batch normalization (numeric value from 0-1). Default to 0.01
dropout_rate	What should the dropout rate be in the sgd optimization? Input should be a positive number.
kMatch	What should k be in the k-nearest neighbor matching? Input should be a positive number.

batchSizePerCat
: What should the batch size per category be in the sgd optimization and knn matching?

verbose
: Should progress updates be given? Input should be a Boolean.

diagnostics
: Should diagnostics be returned? Input should be a Boolean.

justTransform
: A Boolean indicating whether the user wants to extract the quanficiation-optimized features only.

winsorize
: Should columns of the raw dfm be Windorized?

Value

A list consiting of

- estimated category proportions in the unlabeled set (point_readme);
- the transformed dfm optimized for quantification (transformed_dfm);
- (optional) a list of diagnostics (diagnostics);

References

- Hopkins, Daniel, and King, Gary (2010), *A Method of Automated Nonparametric Content Analysis for Social Science*, *American Journal of Political Science*, Vol. 54, No. 1, January 2010, p. 229-247. https://gking.harvard.edu/files/words.pdf
- Jerzak, Connor, King, Gary, and Strezhnev, Anton. Working Paper. *An Improved Method of Automated Nonparametric Content Analysis for Social Science*. https://gking.harvard.edu/words

Examples

```
#set seed
set.seed(1)

#Generate synthetic 25-d word vector corpus.
my_wordVecs <- matrix(rnorm(11*25), ncol = 25)
row.names(my_wordVecs) <- c("the","true", "thine", "stars", "are" , "fire", ".", "to", "own", "self", "be")

#Generate 100 ``documents'' of between 5-10 words each.
my_documentText <- replicate(100, paste(sample(row.names(my_wordVecs), sample(5:10, 1), replace = T), collap

#Assign labeled/unlabeled sets. The first 50 will be labeled; the rest unlabeled.
my_labeledIndicator <- rep(1, times = 100)
my_labeledIndicator[51:100] <- 0

#Assign category membership randomly
my_categoryVec <- sample(c("C1", "C2", "C3", "C4"), 100, replace = T)
true_unlabeled_pd <- prop.table(table(my_categoryVec[my_labeledIndicator==0]))
my_categoryVec[my_labeledIndicator == 0] <- NA

#Get word vector summaries
my_dfm <- undergrad(documentText = my_documentText, wordVecs = my_wordVecs)

#perform estimation
readme_results <- readme(dfm = my_dfm,
                         labeledIndicator = my_labeledIndicator,
                         categoryVec = my_categoryVec,
```

```
                          nboot = 2)
print(readme_results$point_readme)
```

---

undergrad *undergrad*

Description

Preprocessing for `readme` function - creates a document-feature matrix (saved as a data frame in output) to be passed to `readme`. Users can either input word-specific vectors using the `wordVecs_corpus` or `wordVecs_corpusPointer` parameters. Primarily intended for users wanting control over the pre-processing protocol.

Usage

```
undergrad(documentText, wordVecs = NULL, word_quantiles = c(0.1, 0.5,
  0.9), reattempt = T, reattempt_regex = list(c("\#", ""),
  c("#\\S+", "<hashtag>"), c("[[:punct:]]+", ""), c("ing\\b", ""),
  c("s\\b", ""), c("ed\\b", ""), c("ies\\b", "y")),
  unique_terms = T, verbose = T)
```

Arguments

`documentText`	A vector in which each entry corresponds to a "clean" document. Note that the function will take as a "word" all whitespace-separated elements in each vector entry. For example, `"star."` would have to have an exact analogue in the vector corpus, otherwise it will be dropped in the calculations. It will be more common to space separate punctuation marks (i.e. `"star."` would become `"star ."`), since punctuation marks often have their own entries in the vector database.
`wordVecs`	A matrix where each row denotes a word and each column a word vector. Words should be stored as the rownames of the matrix.
`word_quantiles`	A numeric vector denoting the quantiles (0-1) used to summarize each word vector dimension. Defaults to 0.10th, 0.50th and 0.90th quantiles.
`reattempt`	If TRUE, attempts to match terms missing from the wordVec corpus with alternate representations.
`reattempt_regex`	A list of character vectors containing regular expression pairs to be used for generating alternate representations of words to attempt to match with the wordVec corpus when terms initially cannot be matched. Order matters.
`unique_terms`	If TRUE, removes duplicate terms from each document - each document is represented only by the presence or absence of a term.
`verbose`	If TRUE, prints updates as function runs

Value

A data.frame consisting of the `word_quantiles` quantiles of the word vectors by document. Each row corresonds to a document, and the columns to a particular summary of a particular word vector dimension.

Examples

```
#set seed
set.seed(1)

#Generate synthetic word vector corpus.
my_wordVecs <- matrix(rnorm(11*50), ncol = 50)
row.names(my_wordVecs) <- c("the","true", "thine", "stars", "are" ,
                            "fire", ".", "to", "own", "self", "be")

#Setup ``documents''
my_documentText <- c(
"the stars are fire .", #document 1
"to thine own self be true ", #document 2
"true stars be true ." #document 3
)

#Get document-level word vector summaries.
my_dfm <- undergrad(documentText = my_documentText, wordVecs = my_wordVecs)
print( my_dfm )
```

Index