Package ‘readme’

May 18, 2018

Title An Algorithm for Text Quantification

Version 2.0

Description An R package for estimating category proportions in an unlabeled set of documents by implementing the method described in Jerzak, King, and Strezhnev (2018). 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 that is now crafted to minimize the error of the resulting estimate. Automatic differentiation, stochastic gradient descent, and batch renormalization 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.

Depends R (>= 3.3.3)

License

Encoding UTF-8

LazyData true

Maintainer 'Connor Jerzak' <cjerzak@g.harvard.edu>

Suggests FNN,tensorflow, tm, data.table, optmatch,roxygen2

RoxygenNote 6.0.1

R topics documented:

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

Details

An R package for estimating category proportions in an unlabeled set of documents by implementing the method described in Jerzak, King, and Strezhnev (2018). 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 that is now crafted to minimize 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. 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 (`min`, `median`, and `max` of each dimension within each document are calculated). Options also exist for generating a document-term matrix from the text.
- `readme` takes as an input a document feature matrix (preferably, 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 guidance on usage, see **Examples**.

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://gking.harvard.edu/words

Examples

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

#Generate synthetic word vector corpus using a random vocabulary of size 100.
my_vocab <- replicate(100, paste(sample(letters, 4, replace = T), collapse = "") )
synthetic_vector_corpus <- data.frame(matrix(rnorm(100*50), ncol = 50))
synthetic_corpus <- cbind(my_vocab, synthetic_vector_corpus)
synthetic_corpus <- data.table::as.data.table(synthetic_corpus)

#Setup 50 ``documents'' of 10 words apiece.
my_documents <- replicate(50, paste(sample(my_vocab, 10), collapse = " ") )
```

```
#Get document-level word vector summaries.
my_dfm <- undergrad(documentText = my_documents, wordVecs_corpus = synthetic_corpus)

#randomly assign labeled/unlabeled assignment for the 50 documents
my_labeledIndicator <- sample(c(0,1), size = 50, replace = T)

#randomly assign category membership for the 50 documents. There are 3 categories in this example.
my_categoryVec <- sample(c("C1", "C2", "C3"), 50, replace = T)
true_unlabeled_pd <- prop.table(table(my_categoryVec[my_labeledIndicator==0]))
my_categoryVec[my_labeledIndicator == 0] <- NA

#perform estimation
readme_results <- readme(dfm = my_dfm,
                         labeledIndicator=my_labeledIndicator,
                         nboot = 3, categoryVec = my_categoryVec)
print(readme_results$point_readme)
print(true_unlabeled_pd)
```

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 crafted 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 document-feature matrix, (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.

Usage

```
readme(dfm, labeledIndicator, categoryVec, nboot = 10, resample = T,
  resampleTrim = 3, verbose = F, diagnostics = F, justTransform = F,
  readmeVersion = "2")
```

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`.

`nboot` A scalar indicating the number of times the estimation will be re-run, either with or without re-sampling of documents (see `resample`).

resample A Boolean indicating whether documents should be resampled using propensity weights in the estimation.

resampleTrim A scalar (usually between 1 and 10) which controls truncation of the resampling weights. A value of k means that all weights greater than k times the 3rd quartile will be truncated.

verbose A Boolean indicating whether exploratory plots should be shown.

diagnostics A Boolean indicating whether diagnostics should be saved in the output list.

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

readmeVersion A character string taking on values either "1" or "2" indicating the version of the readme algorithm to employ. "2" is highly recommended. If using "1", the dfm should be a document-term matrix. If using "2", document-level word vector summaries should be supplied.

Value

A list consiting of

- estimated category proportions in the unlabeled set (point_readme);
- (if readmeVersion = "2") transformed dfm optimized for quantification (transformed_dfm);
- (optionally and if readmeVersion = "2") a list containing various diagnostics from the estimation.

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)

#setup synthetic dfm
my_dfm <- matrix(rnorm(100*20), ncol = 20)

#randomly assign labeled/unlabeled assignment
my_labeledIndicator <- sample(c(0,1), size = 100, replace = T)

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

#perform estimation
readme_resutls <- readme(dfm = my_dfm,
       labeledIndicator=my_labeledIndicator,
       nboot = 3, categoryVec = my_categoryVec)
print(readme_resutls$point_readme)
print(true_unlabeled_pd)
```

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.

Usage

```
undergrad(documentText, wordVecs_corpus = NULL,
  wordVecs_corpusPointer = NULL, undergradVersion = "2",
  undergradVersion1_control = list())
```

Arguments

documentText A vector in which each entry corresponds to a “clean” document. Note that the function will take as a “word” all space-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_corpus A data.table object in which the first column holds the text of each word, and in which the remaining columns contain the numerical representation. Either wordVecs_corpus or wordVecs_corpusPointer should be null.

wordVecs_corpusPointer A character string denoting where to find the wordVecs_corpus for loading into memory as a data.table.

undergradVersion A character string taking on values either "1" or "2" indicating the version of the undergrad algorithm to employ. Use "2" for extracting document-level summaries of word vectors. Use "1" for extracting the unigrams (to create a document-term matrix).

undergradVersion1_control A list containing additional parameters passed to version "1".

Value

A data.frame consisting of the min, median, and maximum 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.
synthetic_vector_corpus <- data.frame(matrix(rnorm(11*50), ncol = 50))
```

```
synthetic_corpus <- cbind(c("the","true", "thine", "stars", "are" , "fire", ".", "to", "own", "self", "be"),
                          synthetic_vector_corpus)
synthetic_corpus <- data.table::as.data.table(synthetic_corpus)

#Setup ``documents''
my_documents <- 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_documents, wordVecs_corpus = synthetic_corpus)
print( my_dfm )
```

Index