Package 'readme'

September 21, 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
Creative Commons Attribution-Noncommercial-No Derivative Works 4.0, for academic use only.

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 (`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://gking.harvard.edu/words

Examples

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

#Generate synthetic 25-d word vector corpus.
my_wordVecs_corpus <- data.frame(matrix(rnorm(11*25), ncol = 25))
```

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

#Generate 100 ``documents'' of between 5-10 words each.
my_documentText <- replicate(100, paste(sample(my_wordVecs_corpus[[1]], sample(5:10, 1)), collapse = " ") )

#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

#perform estimation
readme_results <- readme(documentText = my_documentText,
       labeledIndicator= my_labeledIndicator,
       categoryVec = my_categoryVec,
       wordVecs_corpus = my_wordVecs_corpus,
       nboot = 1)
print(readme_results$point_readme)
```

---

readme *readme*

Usage

```
readme(documentText = NULL, labeledIndicator, categoryVec,
  wordVecs_corpus = NULL, dfm = NULL, nboot = 10, sgd_iters = 1000,
  verbose = F, diagnostics = F, justTransform = F)
```

Arguments

documentText	A vector in which each entry corresponds to a document. The function will automatically "clean" the text. For more control over the cleaning process, users should pre-process the text themselves, use the undergrad function, and leave the "documentText" parameter NULL.
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.
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. If wordVecs_corpus, wordVecs_corpusPointer, and dfm are all NULL, readme will download and use the GloVe 50-dimensional embeddings trained on Wikipedia.
dfm	'document-feature matrix'. A data frame where each row represents a document and each column a unique feature. Note that this parameter should be NULL if the user is supplying the raw document text into readme (i.e. documentText is not null). #'
nboot	A scalar indicating the number of times the estimation will be re-run (useful for reducing the variance of the final output).
sgd_iters	How many stochastic gradient descent iterations should be used?
verbose	Should diagnostic plots be displayed?
justTransform	A Boolean indicating whether the user wants to extract the quanficiation-optimized features only.

Value

A list consiting of

- estimated category proportions in the unlabeled set (point_readme);
- the transformed dfm optimized for quantification (transformed_dfm);

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_corpus <- data.frame(matrix(rnorm(11*25), ncol = 25))
my_wordVecs_corpus <- cbind(c("the","true", "thine", "stars", "are" , "fire", ".", "to", "own", "self", "be")
my_wordVecs_corpus <- data.table::as.data.table(my_wordVecs_corpus)

#Generate 100 ``documents'' of between 5-10 words each.
my_documentText <- replicate(100, paste(sample(my_wordVecs_corpus[[1]], sample(5:10, 1)), collapse = " ") )

#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

#perform estimation
```

```
readme_results <- readme(documentText = my_documentText,
       labeledIndicator= my_labeledIndicator,
       categoryVec = my_categoryVec,
       wordVecs_corpus = my_wordVecs_corpus,
       nboot = 1)
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_corpus = NULL)
```

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. If `wordVecs_corpus` and `wordVecs_corpusPointer` are `NULL`, `undergrade` will download and use the `GloVe` 50-dimensional embeddings trained on Wikipedia.
`wordVecs_corpusPointer`	A character string denoting where to find the `wordVecs_corpus` for loading into memory as a data.table. If `wordVecs_corpus` and `wordVecs_corpusPointer` are `NULL`, `undergrade` will download and use the `GloVe` 50-dimensional embeddings trained on Wikipedia.

Value

A data.frame consisting of the `10th`, `50th`, and `90th` 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_corpus <- data.frame(matrix(rnorm(11*50), ncol = 50))
my_wordVecs_corpus <- cbind(c("the","true", "thine", "stars", "are" ,
                              "fire", ".", "to", "own", "self", "be"),
                            my_wordVecs_corpus)
my_wordVecs_corpus <- data.table::as.data.table(my_wordVecs_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 = my_wordVecs_corpus)
print( my_dfm )
```

Index