Package ‘validateIt’

May 16, 2023

Title Validating Topic Coherence and Topic Labels

Version 1.2.1

Description By creating crowd-sourcing tasks that can be easily posted and results retrieved using Amazon’s Mechanical Turk (MTurk) API, researchers can use this solution to validate the quality of topics obtained from unsupervised or semi-supervised learning methods, and the relevance of topic labels assigned. This helps ensure that the topic modeling results are accurate and useful for research purposes. See Ying and others (2022) <doi:10.1101/2023.05.02.538599>. For more information, please visit <https://github.com/Triads-Developer/Topic_Model_Validation>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports pyMTurkR, rlang (>= 0.4.11), tm (>= 0.7-11), here, SnowballC

Suggests roxygen2, testthat

Author Luwei Ying [aut, cre] (<https://orcid.org/0000-0001-7307-4834>), Jacob Montgomery [aut], Brandon Stewart [aut]

Maintainer Luwei Ying <triads.developers@wustl.edu>

Repository CRAN

Depends R (>= 3.5.0)

NeedsCompilation no

Date/Publication 2023-05-16 08:20:02 UTC

R topics documented:

allR4WSItasktest ................................................... 2
checkAgree .......................................................... 3
cmpMass ............................................................. 3
evalResults .......................................................... 4
getResults ............................................................ 5
Example R4WSI Tasks with Regular and Gold-Standard Tasks

Description
Data frame of 20 example R4WSI0 Tasks, with 5 of them being gold-standard and 15 of them not.

Usage
data(allR4WSItasktest)

Format
A data frame of 20 rows and 6 columns.

  topic  Index of topics
  id    Index of topics
  doc   Example documents associated with each topic
  opt1  Words set option 1
  opt2  Words set option 2
  opt3  Words set option 3
optcrt Words set option 4, also the correct choice
checkAgree

Check Agreement Rate between Identical Trails

Description

Check Agreement Rate between Identical Trails

Usage

checkAgree(results1, results2, key, type = NULL)

Arguments

results1 first batch of results; outputs from getResults()
results2 first batch of results; outputs from getResults()
key the local task record; outputs from recordTasks()
type Task structures to be specified. Must be one of "WI" (word intrusion), "T8WSI" (top 8 word set intrusion), "R4WSI" (random 4 word set intrusion), "LI" (Label Intrusion), and "OL" (Optimal Label)

Details

Evaluate workers’ performance by agreement rate between identical trails (Notice that this means the two input, results1 and results2, must be identical.); Return 1) the exact agreement rate when both workers agree on the exact same choice, and 2) the binary agreement rate when both workers get the task either right or wrong simultaneously

Value

A numeric value to be returned with output.

combMass

Combine the mass of words with the same root

Description

Combine the mass of words with the same root

Usage

combMass(mod = NULL, vocab = NULL, beta = NULL)
**Arguments**

- `mod`: Fitted structural topic models.
- `vocab`: A character vector specifying the words in the corpus. Usually, it can be found in topic model output.
- `beta`: A matrix of word probabilities for each topic. Each row represents a topic and each column represents a word. Note this should not be in the logged form.

**Details**

Use as a preparing step for validating unstemmed topic models.

**Value**

A list with two elements:

- `newvocab`: A matrix of new vocabulary. Each row represents a topic and each column represents a unique stemmed word.
- `newbeta`: A matrix of new beta. Each row represents a topic and each column represents the sum of the probabilities of the words with the same root.

---

**Description**

Evaluate results

**Usage**

```r
evalResults(results, key, type = NULL)
```

**Arguments**

- `results`: results of human choice; outputs from getResults()
- `key`: the local task record; outputs form recordTasks()
- `type`: Task structures to be specified. Must be one of "WI" (word intrusion), "T8WSI" (top 8 word set intrusion), "R4WSI" (random 4 word set intrusion), "LI" (Label Intrusion), and "OL" (Optimal Label)

**Details**

Evaluate worker performance by gold-standard HITs; Return the accuracy rate (proportion correct) for a specified batch

**Value**

A list containing the gold-standard HIT correct rate, gold-standard HIT correct rate by workers, and non-gold-standard HIT correct rate
**getResults**

Get results from Mturk

**Usage**

```r
getResults(
  batch_id = "unspecified",
  hit_ids,
  retry = TRUE,
  retry_in_seconds = 60,
  AWS_id = Sys.getenv("AWS_ACCESS_KEY_ID"),
  AWS_secret = Sys.getenv("AWS_SECRET_ACCESS_KEY"),
  sandbox = getOption("pyMTurkR.sandbox", TRUE)
)
```

**Arguments**

- `batch_id`: any number or string to annotate the batch
- `hit_ids`: hit ids returned from the MTurk API, i.e., output of `sendTasks()`
- `retry`: if TRUE, retry retrieving results from Mturk API five times; default to TRUE
- `retry_in_seconds`: default to 60 seconds
- `AWS_id`: `AWS_ACCESS_KEY_ID`
- `AWS_secret`: `AWS_SECRET_ACCESS_KEY`
- `sandbox`: sandbox setting

**Details**

this function works for complete or incomplete batches

**Value**

a data frame with columns:

- `batch_id`: an annotation for the batch
- `local_task_id`: an identifier for the task in the batch
- `mturk_hit_id`: the ID of the HIT in MTurk
- `assignment_id`: the ID of the assignment in MTurk
- `worker_id`: the ID of the worker who completed the assignment
- `result`: the worker’s response to the task
- `completed_at`: the time when the worker submitted the assignment
Example Gold-Standard R4WS10 Tasks

Description
Data frame of 5 example gold-standard R4WS10 Tasks.

Usage
data(goldR4WS1test)

Format
A data frame of 5 rows and 6 columns.
- topic  Index of topics
- doc   Example documents associated with each topic
- opt1  Words set option 1
- opt2  Words set option 2
- opt3  Words set option 3
- optcrt Words set option 4, also the correct choice

An Example Heldout Test Set

Description
An output from the make.heldout function of the stm package.

Usage
data(heldouttest)

Format
A list of the heldout documents, vocab, and missing.

Source
See https://CRAN.R-project.org/package=stm for more details.

References
**Example Answer Keys**

**Description**
Example Answer Keys

**Usage**
data(keypostedtest)

**Format**
A list of two data frames. Similar to recordtest.

data.frame1 A data frame of tasks with the optcrt indicating the machine predicted choice.
data.frame2 A data frame of tasks with randomized choices. Exactly the same with what would be sent online.

**masstest**
***An Example of the Combined Mass for Words with the Same Roots***

**Description**
A list of two with the words (the most frequent form in each topic) and the corresponding word probabilities.

**Usage**
data(masstest)

**Format**
A list of two.

**Details**
vocab A matrix of words for each topic. Each row represents a topic and each column represents the words. Words with the same roots are only represented by the most common form in that topic.

beta A matrix of combined word probabilities for each topic. Each row represents a topic and each column represents a combined word.
Mix the gold-standard tasks with the tasks need to be validated

**Description**

Mix the gold-standard tasks with the tasks need to be validated

**Usage**

mixGold(tasks, golds)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tasks</td>
<td>All tasks need to be validated</td>
</tr>
<tr>
<td>golds</td>
<td>Gold standard tasks with the same structure</td>
</tr>
</tbody>
</table>

**Value**

A data frame with the same structure as the input, where gold-standard tasks are randomly inserted

---

**modtest**

An Example Topic Model

**Description**

A structural topic model (STM) object generated from the stm package using a random sample of US senators’ Facebook posts.

**Usage**

data(modtest)

**Format**

A STM object.

**Source**

See https://CRAN.R-project.org/package=stm for more details.

**References**

pickLabel

Pick the optimal label from candidate labels

**Description**

Pick the optimal label from candidate labels

**Usage**

```r
pickLabel(
  n,
  text.predict = NULL,
  text.name = "text",
  top1.name = "top1",
  labels.index = NULL,
  candidate.labels = NULL
)
```

**Arguments**

- **n**: The number of desired tasks
- **text.predict**: A data frame or matrix containing both the text and the indicator(s) of the model predicted topic(s).
- **text.name**: variable name in `text.predict` that indicates the text
- **top1.name**: variable name in `text.predict` that indicates the top1 model predicted topic
- **labels.index**: The topic index in correspondence with the labels, e.g., c(10, 12, 15).
- **candidate.labels**: A list of vectors containing the user-defined labels assigned to the topics, Must be in the same length and order with `labels.index`.

**Details**

Users need to specify four plausible labels for each topic

**Value**

A matrix with n rows and 6 columns (topic, doc, opt1, opt2, opt3, optcrt) where optcrt is the correct label that was picked.
plotResults  

Description
Plot results

Usage
plotResults(path, x, n, taskname, ...)

Arguments
- path: path to store the plot
- x: a vector of counts of successes; could be obtained from getResults()
- n: a vector of counts of trials
- taskname: the name of the task for labeling, e.g., Word Intrusion, Optimal Label.
- ...: additional arguments to be passed to plot function

Details
Visualize the accuracy rate (proportion correct) for a specified batch

Value
Nothing is returned; a plot is created and saved as a pdf file.

R4WSItasktest  

Description
Data of 15 example R4WSI0 Tasks structured as a matrix.

Usage
data(R4WSItasktest)
**Format**

A matrix with 15 rows and 6 columns.

- **topic**: Index of topics
- **doc**: Example documents associated with each topic
- **opt1**: Words set option 1
- **opt2**: Words set option 2
- **opt3**: Words set option 3
- **optcrt**: Words set option 4, also the correct choice

**Details**

Please note that the difference between the R4WSI0 examples used here and the R4WSI tasks is that the R4WSI tasks do not present any documents.

---

**Description**

Reform tasks to facilitate sending to Mturk

**Usage**

```r
record(type, tasks, path)
```

**Arguments**

- **type**: (character) one of WI, T8WSI, R4WSI
- **tasks**: (data.frame) outputs from validateTopic(), validateLabel(), or mixGold() if users mix in gold-standard HITs
- **path**: (character) path to record the tasks (with meta-information)

**Details**

Randomize the order of options and record the tasks in a specified local directory

**Value**

A list of two data frames, containing the original tasks and the randomized options respectively.
recordtest  Example Local Record of the R4WSI Tasks

Description

Local record generated by the recordTasks function.

Usage

data(recordtest)

Format

A list of two data frames.

data.frame1  A data frame of tasks with the optcrt indicating the machine predicted choice.
data.frame2  A data frame of tasks with randomized choices. Exactly the same with what would be sent online.

Details

To be compared with the answers from the online workers to evaluate the topic model performance.

resultstest  Example Results Retrieved from Mturk

Description

Example Results Retrieved from Mturk

Usage

data(resultstest)

Format

A data frame of ten example tasks retrieved from the Mturk with or without online workers’ answers.

assignment_id  Assignment id. Mturk assigned. If 0, then the task hasn’t been completed.
batch_id  User specified batch id.
completed_at  Timestamp when the task was completed. If 0, then the task hasn’t been completed.
local_task_id  Local task id.
murk_hit_id  Mturk HIT id. Mturk assigned.
result  Choice made by the worker. 1-4. If 0, then the task hasn’t been completed.
worker_id  Mturk worker id. If 0, then the task hasn’t been completed.
Send prepared task to Mturk and record the API-returned HIT ids.

Description

Send prepared task to Mturk and record the API-returned HIT ids.

Usage

```r
sendTasks(
  hit_type = NULL,
  hit_layout = NULL,
  type = NULL,
  tasksrecord = NULL,
  tasksids = NULL,
  HITidspath = NULL,
  n_assignments = 1,
  expire_in_seconds = as.character(60 * 60 * 8),
  batch_annotation = NULL
)
```

Arguments

- **hit_type**: find from the Mturk requester’s dashboard
- **hit_layout**: find from the Mturk requester’s dashboard
- **type**: one of WI, T8WSI, R4WSI
- **tasksrecord**: output of recordTasks()
- **tasksids**: ids of tasks to send in numeric form. If left unspecified, the whole batch will be posted
- **HITidspath**: path to record the returned HITids
- **n_assignments**: number of of assignments per task. For the validation tasks, people almost always want 1
- **expire_in_seconds**: default 8 hours
- **batch_annotation**: add if needed

Details

Pairs the local ids with Mturk ids and save them to specified paths
### Value

A list containing two elements:

- **current_HIT_ids**: A vector of the HIT IDs returned by the API.
- **map_ids**: A data frame that maps the tasksids to their corresponding HIT ids.

---

### Description

An output from the `prepDocuments` function of the `stm` package.

### Usage

```r
data(stmPreptest)
```

### Format

A list containing a documents and vocab object.

### Source

See [https://CRAN.R-project.org/package=stm](https://CRAN.R-project.org/package=stm) for more details.

### References


---

### Description

The `Topic_Model_Validation` repository is a collection of scripts and functions for performing topic modeling and evaluating topic models. This document provides an overview of the different scripts and functions in the repository and their purpose.
validateLabel

Create validation tasks for labels assigned to the topics in the topic model of choice.

Usage

validateLabel(
    type,
    n,
    text.predict = NULL,
    text.name = "text",
    top1.name = "top1",
    top2.name = "top2",
    top3.name = "top3",
    labels = NULL,
    labels.index = NULL,
    labels.add = NULL
)
validateLabel

Arguments

type  Task structures to be specified. Must be one of "LI" (Label Intrusion) and "OL" (Optimal Label).

n  The number of desired tasks

text.predict  A data frame or matrix containing both the text and the indicator(s) of the model predicted topic(s).

text.name  variable name in 'text.predict' that indicates the text

top1.name  variable name in 'text.predict' that indicates the top1 model predicted topic

top2.name  variable name in 'text.predict' that indicates the top2 model predicted topic

top3.name  variable name in 'text.predict' that indicates the top3 model predicted topic

labels  The user-defined labels assigned to the topics

labels.index  The topic index in correspondence with the labels, e.g., c(10, 12, 15). Must be in the same length and order with 'label'.

labels.add  Labels from other broad categories. Default to NULL. Users could specify them to evaluate how well different broad categories are distinguished from one another.

# value A matrix containing the validation tasks as described in the return section.

Details

Users need to pick a topic model that they deem to be good and label the topics they later would like to use as measures.

Value

A matrix containing the validation tasks. The matrix has six value columns:

topic  The topic index associated with the document.
doc  The text of the document.

opt1  The first option label presented to the user.

opt2  The second option label presented to the user.

opt3  The third option label presented to the user.

optcrt  The correct label for the document.
validateTopic

Create validation tasks for topic model selection

Description
Create validation tasks for topic model selection

Usage
validateTopic(type, n, text = NULL, vocab, beta, theta = NULL, thres = 20)

Arguments

- **type**: Task structures to be specified. Must be one of "WI" (word intrusion), "T8WSI" (top 8 word set intrusion), and "R4WSI" (random 4 word set intrusion).
- **n**: The number of desired tasks
- **text**: The pool of documents to be shown to the Mturk workers
- **vocab**: A character vector specifying the words in the corpus. Usually, it can be found in topic model output.
- **beta**: A matrix of word probabilities for each topic. Each row represents a topic and each column represents a word. Note this should not be in the logged form.
- **theta**: A matrix of topic proportions. Each row represents a document and each column represents a topic. Must be specified if task = "T8WSI" or "R4WSI".
- **thres**: The threshold to draw words from, default to top 50 words.

Details
Users need to fit their own topic models.

Value
A matrix of validation tasks. Each row represents a task and each column represents an aspect of a task, including the topic label, the document text (for "T8WSI" and "R4WSI"), and five words, including four non-intrusive words and one intrusive word.
Index

* datasets
  allR4WSItasktest, 2
  goldR4WSItest, 6
  R4WSItasktest, 10
  resultstest, 12

allR4WSItasktest, 2
checkAgree, 3
combMass, 3
evalResults, 4
getResults, 5
goldR4WSItest, 6
heldouttest, 6
keypostedtest, 7
masstest, 7
mixGold, 8
modtest, 8
pickLabel, 9
plotResults, 10
R4WSItasktest, 10
record, 11
recordtest, 12
resultstest, 12
sendTasks, 13
stmPreptest, 14

Topic_Model_Validation_Overview, 14

validateLabel, 15
validateTopic, 17