

Galileo: Cluster Analysis Algorithms

Justin Grimmer

Gary King

Version 0.02
March 25, 2008

affinity : Compute Similarity Between Documents

affinity provides a matrix of affinities between documents. To improve the topics detected in the data, **affinity** allows users to discard unimportant words, remove stop words, and compute a set of weights to identify important words. The output from **affinity** is provided to **galileo** to uncover topics.

Syntax

```
aff.mat<- affinity(undergrad, tfidf=T, type='cosine', stop=T,  
extra=F, extra.dat=NULL)
```

Inputs

- **undergrad** An object from **undergrad**, counting the occurrences of stems in each document.
- **tf-idf=T** if true, tf-idf weights are used to calculate the affinity between documents.
- **type='cosine'** Selects the measure of affinity between documents. Currently, only the cosine is implemented, but future methods include euclidean (normal distribution), correlation, and absolute deviation.
- **stop=T** if true, removes stop words from each document's representation when calculating the affinity between documents.
- **extra=T** if true, there is an additional set of words to be removed from each document's representation before calculating the affinity between documents
- **extra.dat** if **extra=T** then the additional words (in a data frame) must be supplied to **extra.dat**.

Details

`affinity` creates a matrix that measures the similarities between a set of documents, which is supplied to `galileo` to group documents with similar topics. Using the `stop`, `tf-idf`, `extra.dat` options allows the user to supply information to help increase the likelihood `galileo` will uncover meaningful topics from the data. See Frey and Dueck (2007).

Examples

```
##gt is an object from undergrad
```

```
library(Galileo)
```

```
data(SenPress)
```

```
data(CommWords)
```

```
output<- affinity(gt, stop=T, tfidf=T, extra=T,  
extra.dat=comm.words)
```

Output Values

if there are k documents to be analyzed, `affinity` returns a $k \times k$ matrix of similarities between documents.

Contributors

CONTRIBUTORS HERE.

galileo : Methods for Topic Detection in Texts

The `galileo` command provides a variety of methods for topic-detection in texts. `galileo` takes as an input a matrix of affinities between documents and returns a vector of cluster assignments between documents, which groups documents together based upon the topic of the text. This classification can be used by `mutinf` to label the clusters.

Syntax

```
clust <- galileo(affinity, method=c('Affinity', 'Spectral'), p=NULL,
, lambda=0.6, maxits=500, convits=50, num=NULL)
```

Inputs

- `affinity` output from `affinity` a matrix of affinities (similarities) between a set of documents.
- `method= c('Affinity', 'Spectral')` selects the method of text-clustering for topic detection. `Affinity` results in text-clustering by approximate mixture of von-Mises distribution, with estimation by Dueck and Frey's (2007) affinity propagation. `Spectral` computes the clusters using the two step clustering procedure developed in Ng, Jordan, and Weiss (2002).
- `p=NULL` If `Affinity` is selected, `p` sets the prior that each document forms a cluster center and therefore is a prior on the number of topics. Specifically, $p = \log \Pr(\text{doc}_i = \text{center}) + 1$, or the log probability an arbitrarily chosen document is a cluster center.
- `lambda=0.6` Dampening coefficient for method `affinity`, must be between (0.5, 1).
- `maxits=500` is the maximum number of iterations for affinity propagation to carry out before ceasing
- `convits=50` is the number of iterations necessary to establish convergence.
- `num` sets the number of topics if `Spectral` method is selected

Details

`galileo` implements two different topic detection methods. The first, `Affinity` estimates an approximate mixture of von Mises-Fisher distributions using Dueck and Frey's (2007) Affinity Propagation algorithm. DISCUSSION OF THE APPROXIMATE MIXTURE OF VMF DISTRIBUTIONS. Affinity propagation is a special case of 'loopy belief-propagation'—an approximate maximization approach for NP-Hard problems. The second method `Spectral` implements the spectral clustering method of Ng, Jordan, and Weiss (2002). Spectral clustering methods have recently been developed in Computer Science and provide a quick approach

to document clustering that have been shown to be competitive with mixture modeling approaches.

Examples

`##output is an object from the affinity function`

`##Example code for Affinity propagation`

```
cluster<- galileo(log(output), p = median(log(output))- 10,  
lambda=0.6, method='Affinity')
```

`##Example code for Spectral Clustering`

```
cluster<- galileo(output, num=30, method='Spectral')
```

Output Values

`galileo` returns a vector of cluster labels for each document.

Contributors

CONTRIBUTORS HERE.

mutinf : Compute Cluster Labels using Mutual Information

`mutinf` determines a set of cluster labels using output from `galileo`. Specifically, `mutinf` computes the information each stem w provides for correct classification of documents in topics. The output from `mutinf` and `galileo` can be passed to `clustTable` to produce a \LaTeX ready table.

Syntax

```
mutinf(cluster, undergrad, stop=T, extra=F, extra.dat=NULL)
```

Inputs

- `cluster` is an object from `galileo` containing the cluster assignments for a set of documents.
- `undergrad` is an object from `undergrad` containing the representation of each document.
- `stop=T` if true, removes stop words from each document's representation when calculating the affinity between documents.
- `extra=F` if true, there is an additional set of words to be removed from each document's representation before calculating the affinity between documents
- `extra.dat` if `extra=T` then the additional words (in a data frame) must be supplied to `extra.dat`.

Details

`mutinf` produces a set of labels for clusters of documents, regardless of the method used to compute the cluster assignments. The mutual information measures how well a stem w predicts whether a document is assigned to a topic k . The mutual information will reach its maximum if a stem is only found in documents that assigned to topic k and reaches its minimum if the stem does not help discriminate between the documents assigned to a category and the documents not assigned to the category.

Examples

```
##gt is an object from undergrad
```

```
library(Galileo)
```

```
data(SenPress)
```

```
data(CommWords)

output<- affinity(gt, stop=T, tfidf=T, extra=T,
extra.dat=comm.words)

cluster<- galileo(log(output), p = median(log(output))- 10,
lambda=0.6, method='Affinity')

mutts<- mutinf(cluster, gt, extra=T, extra.dat=comm.words)
```

Output Values

If there are k topics and W stems, produces a $k \times W$ matrix, describing the mutual information of each stem, for each topic.

Contributors

CONTRIBUTORS HERE.

clustTable : Table for Displaying Cluster Output

`clustTable` takes the output from `galileo` and `mutinf` and produces a \LaTeX ready table.

Syntax

```
clustTable(cluster, mut.inf, num=5)
```

Inputs

- `cluster` is an object from `galileo` containing the cluster assignments for a set of documents.
- `mut.inf` is an object from `mutinf` describing the mutual information for each word and for each cluster.
- `num` describes the number of stems used to describe each topic.

Details

`clustTable` orders the clusters from largest to smallest. Then, each cluster is labeled according to the `num` stems with the largest mutual information for a given cluster.

Examples

```
##gt is an object from undergrad
```

```
library(Galileo)
```

```
data(SenPress)
```

```
data(CommWords)
```

```
output<- affinity(gt, stop=T, tfidf=T, extra=T,  
extra.dat=comm.words)
```

```
cluster<- galileo(log(output), p = median(log(output))- 10,  
lambda=0.6, method='Affinity')
```

```
muts<- mutinf(cluster, gt, extra=T, extra.dat=comm.words)
```

```
clustTable(cluster, muts, num=5)
```

Output Values

A L^AT_EXready table

Contributors

Justin Grimmer and Gary King implemented `clustTable`.

References

Frey, B.J. and D. Dueck. 2007. “Clustering by Passing Messages Between Data Points.” *Science* 315(5814):972.