Text to Matrix Generator[*]
User's Guide

Dimitrios Zeimpekis[†] Efstratios Gallopoulos[‡]

Department of Computer Engineering and Informatics,
University of Patras, Greece

April 2007

[*] Work conducted in the context of and supported in part by a University of Patras KARATHEODORI grant.

[†] e-mail: dsz@hpclab.ceid.upatras.gr, supported in part by a Bodossaki Foundation graduate fellowship.

[‡] e-mail: stratis@ceid.upatras.gr.

Contents

1 Introduction 1

2 Istallation Instuctions 1

3 Graphical User Interfaces 3

3.1 Indexing module (tmg_gui) 3

3.2 Dimensionality Reduction module (dr_gui) 6

3.3 Retrieval module (retrieval_gui) 9

3.4 Clustering module (clustering_gui) 11

3.5 Classification module (classification_gui) 13

A Appendix: Demonstration of Use 19

A.1 Indexing module (tmg_gui) 19

A.2 Dimensionality Reduction module (dr_gui) 26

A.3 Retrieval module (retrieval_gui) 30

A.4 Clustering module (clustering_gui) 34

A.5 Classification module (classification_gui) 39

B Appendix: Function Reference 43

about_tmg_gui 43

block_diagonalize 44

classification_gui 45

clsi 46

clustering_gui 47

cm 48

col_normalization 49

col_rearrange 50

compute_scat 51

create_kmeans_response 52

create_pddp_response 53

create_retrieval_response 54

diff_vector 55

dr_gui 56

ekmeans 57

entropy 58

get_node_scat 59

gui 60

init_tmg 61

knn_multi 62

knn_single 63

ks_selection 64

ks_selection1 65

llsf_multi 66

llsf_single 67

lsa 68
make_clusters_multi 69
make_clusters_single 70
make_labels 71
make_val_inds 72
merge_dictionary 73
merge_tdms 74
myperms 75
open_file 76
opt_2means 77
pca 78
pca_mat 79
pca_mat_afun 80
pca_propack 81
pca_propack_Atransfunc 82
pca_propack_afun 83
pddp 84
pddp_2means 85
pddp_optcut 86
pddp_optcut_2means 87
pddp_optcutpd 88
ps_pdf2ascii 89
retrieval_gui 90
rocchio_multi 91
rocchio_single 92
scut_knn 93
scut_llsf 94
scut_rocchio 95
sdd_tmg 96
skmeans 97
stemmer 98
strip_html 99
svd_tmg 100
tdm_downdate 101
tdm_update 102
tmg 104
tmg_gui 106
tmg_query 107
tmg_save_results 109
tmg_template 110
two_means_1d 111
unique_elements 112
unique_words 113
vsm 114

List of Figures

1 Structure and dependencies of GUI modules of TMG. 1
2 Structure of TMG root directory. 2
3 The `tmg_gui` GUI. 3
4 The `dr_gui` GUI. 6
5 The `retrieval_gui` GUI. 9
6 The `clustering_gui` GUI. 11
7 The `classification_gui` GUI. 13
8 Starting window of `tmg_gui`. 20
9 Next view of `tmg_gui` according to the user selection. 21
10 The `open_file` window. 22
11 The output '.mat' files of `tmg_gui`. 23
12 The MySQL view uppon `tmg` execution. 24
13 The GUIs' general help tab. 25
14 Starting window of `dr_gui`. 27
15 Next view of `dr_gui` according to the user selection. 28
16 The output '.mat' files of `dr_gui`. 29
17 Starting window of `retrieval_gui`. 31
18 Next view of `retrieval_gui` according to the user selection. . . . 32
19 The output of `retrieval_gui`. 33
20 Starting window of `clustering_gui`. 35
21 Next view of `clustering_gui` according to the user selection. . . 36
22 The output '.mat' files of `clustering_gui`. 37
23 The output of `clustering_gui` for PDDP. 38
24 Starting window of `classification_gui`. 40
25 Next view of `classification_gui` according to the user selection. 41

List of Tables

1 Description of use of tmg_gui components. 5
2 Description of use of dr_gui components. 8
3 Description of use of retrieval_gui components. 10
4 Description of use of clustering_gui components. 12
5 Description of use of classification_gui components. 15

1 Introduction

Text to Matrix Generator (TMG) is a MATLAB Toolbox that can be used for various Data Mining (DM) and Information Retrieval (IR) tasks. TMG uses the sparse matrix infrastracture of MATLAB that is especially suited for TM applications where data are extremely sparse. Initially built as a preprocessing tool, TMG offers now a wide range of DM tools. In particular, TMG is composed of five Graphical User Interface (GUI) modules, presented in Figure 1 (arrows show modules' dependencies).

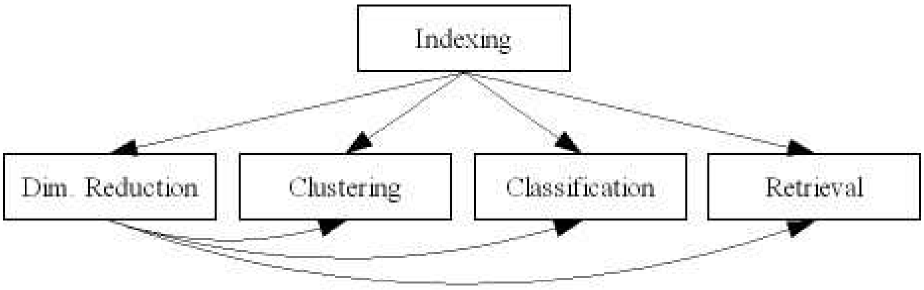


Figure 1: Structure and dependencies of GUI modules of TMG.

In the sequel, we first discuss the installation procedure of TMG and then describe in some detail the GUI's usage. In Appendix A we give a demonstation of use for all the TMG components, while Appendix B supplies a function reference.

2 Istallation Instuctions

Installation of TMG is straightforward by means of the `init_tmg` script. In particular, the user has to perform the following steps:

- For MySQL functionality, install MySQL and Java Connector.
- Download TMG by filling the form from: http://scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/tmg_request.php[1]
- Unzip TMG_X.XRX.zip and start MATLAB. Figure 2 depicts the directory structure of the TMG root directory.
- Change path to the TMG root directory.
- Run init_tmg. Give the MySQL login and password as well as the root directory of the MySQL Java Connector. The installation script creates all necessary information (including MySQL database TMG) and adds to the MATLAB path all necessary directories.
- Run gui. Alternatively, use the command line interface, type `help tmg`.

[1] We are currently on the development of a wiki page.

TMG requires the MySQL[2], PROPACK[3], SDDPACK[4] and SPQR[5] third party software packages. PROPACK, SDDPACK and SPQR packages are included into TMG, while the user has to download MySQL. However, we note that MySQL related software is necessary only if the user intends to use the database support implemented into TMG. Ordinary TMG will run without any problem on a Matlab 7.0 environment without any other special software.

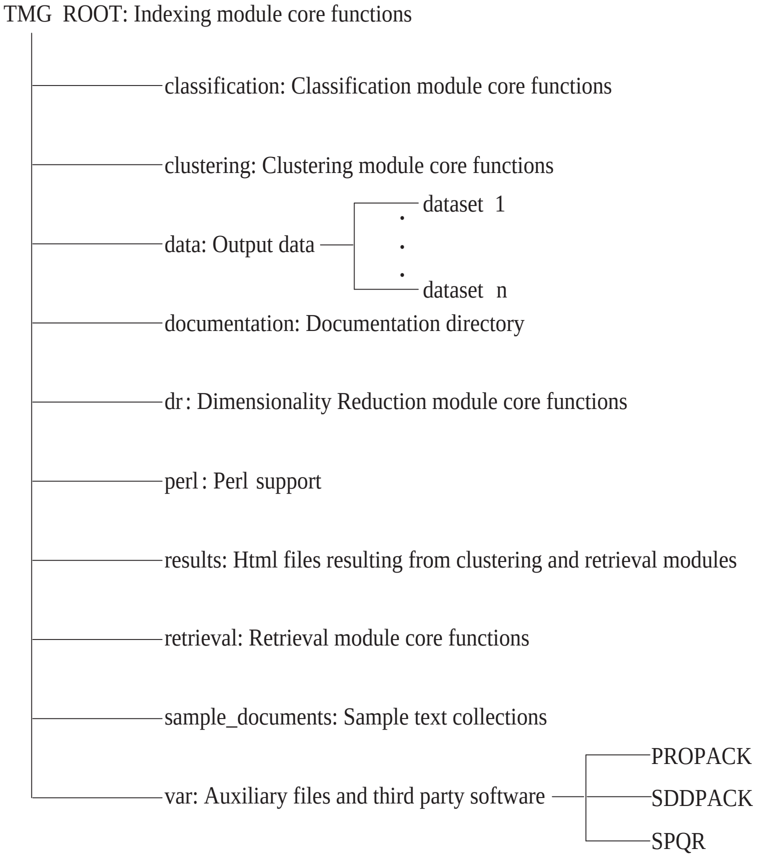
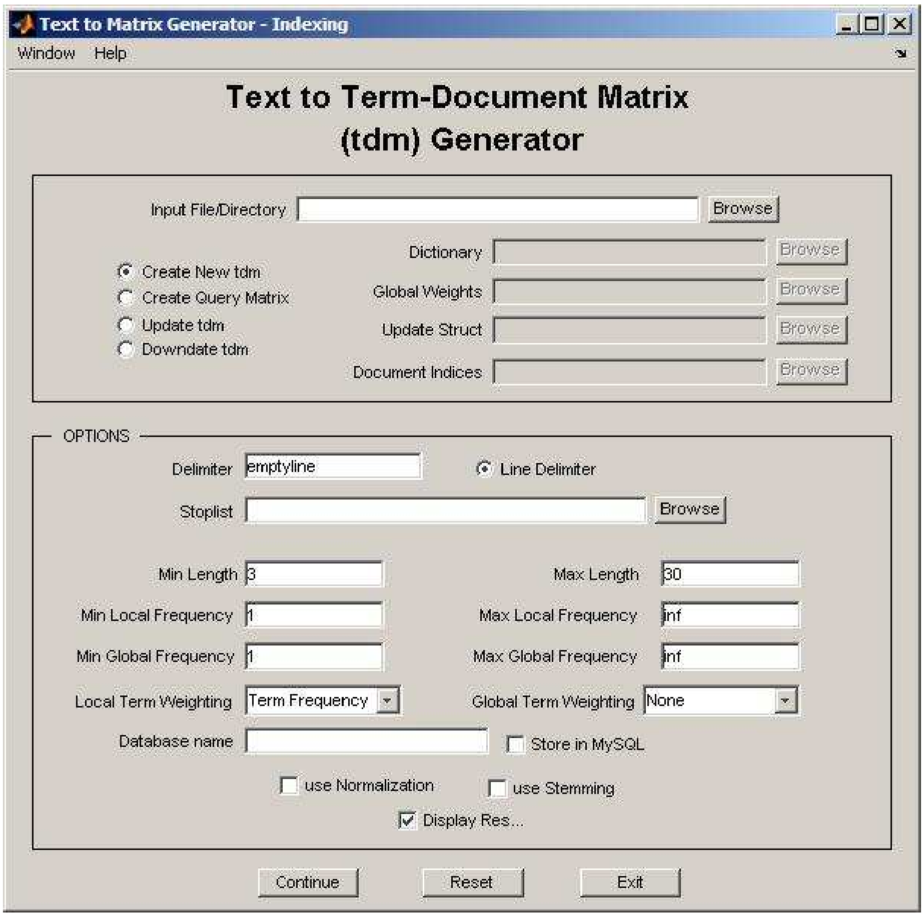

TMG ROOT: Indexing module core functions
classification: Classification module core functions
clustering: Clustering module core functions
data: Output data
dataset 1
dataset n
documentation: Documentation directory
dr: Dimensionality Reduction module core functions
perl: Perl support
results: Html files resulting from clustering and retrieval modules
retrieval: Retrieval module core functions
sample_documents: Sample text collections
var: Auxiliary files and third party software
PROPACK
SDDPACK
SPQR

Figure 2: Structure of TMG root directory.

[2] http://www.mysql.com/, http://dev.mysql.com/downloads/connector/j/5.0.html
[3] http://soi.stanford.edu/rmunk/PROPACK/index.html
[4] http://www.cs.umd.edu/oleary/SDDPACK/README.html
[5] http://portal.acm.org/citation.cfm?id=1067972

3 Graphical User Interfaces

3.1 Indexing module (`tmg_gui`)



Text to Term-Document Matrix (tdm) Generator

Input File/Directory

● Create New tdm
○ Create Query Matrix
○ Update tdm
○ Downdate tdm

Dictionary
Global Weights
Update Struct
Document Indices

OPTIONS

Delimiter: emptyline
● Line Delimiter
Stoplist

Min Length: 3
Max Length: 30
Min Local Frequency: 1
Max Local Frequency: inf
Min Global Frequency: 1
Max Global Frequency: inf
Local Term Weighting: Term Frequency
Global Term Weighting: None
Database name
□ Store in MySQL
□ use Normalization
□ use Stemming
☑ Display Res...

Figure 3: The `tmg_gui` GUI.

TMG can be used for the construction of new and the update of existing term-document matrices (tdms) from text collections, in the form of MATLAB sparse arrays. To this end, TMG implements various steps such as:

- Removal of stopwords.
- Apply stemming (currently Porter stemming algorithm [8]).
- Remove of short/long terms.
- Remove of frequent/infrequent terms (locally or globally).

- Term weighting and normalization.
- Html filtering, processing of Postscript and PDF.
- Store in MySQL (optionally).

The resulting tdms can be stored as “mat” files, while text can also be stored in MySQL for further procesing. TMG can also update existing tdms by efficient incremental updating or downdating operations. Finally, TMG can also construct query vectors using the existing dictionary that can be used from the retrieval and classification modules.

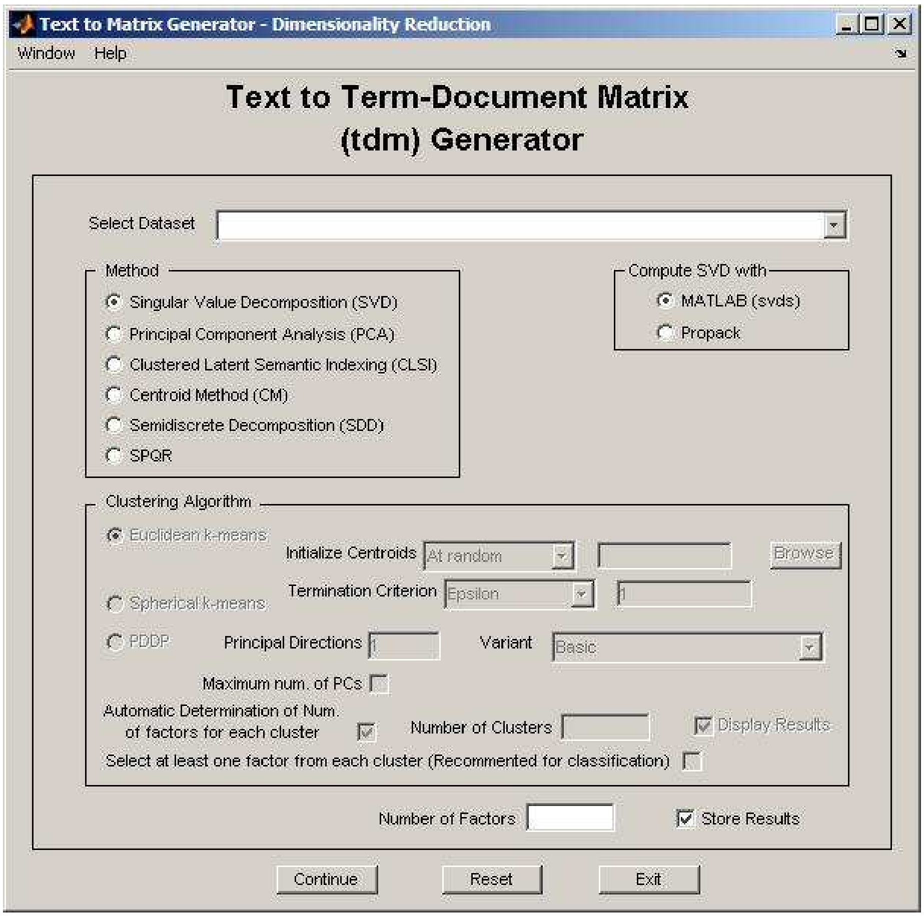
The indexing GUI module is depicted in Figure 3 while Table 1 describes in detail all the `tmg_gui` fields.

Field Name	Default	Description
Input File/Directory	-	Files to be parsed with resulting documents separated by “Delimiter”. Alternatively, each file in the input directory contains a single document.
Create New tdm	●	Checked if new tdm is to be created (default checked).
Create Query Matrix	-	Checked if new query matrix is to be created (default checked).
Update tdm	-	Checked if an existing tdm is to be updated with new documents. Alternatively, ckecked if an existing tdm is to be updated using different options (change update_struct).
Downdate tdm	-	Checked if an existing tdm is to be downdated according to the “Document Indices” field.
Dictionary	-	Name of .mat file or workspace variable containing the dictionary to be used by tmg_query function if the “Create Query Matrix” radio button is checked.
Global Weights	-	Name of .mat file or workspace variable containing the vector of global weights to be used by tmg_query function if the “Create Query Matrix” radio button is checked.
Update Struct	-	Name of .mat file or workspace variable containing the structure to be updated or downdated by tdm_update (or tdm_downdate) function if the “Udpate tdm” or “Downdate tdm” radio button is checked.
Document Indices	-	Name of .mat file or workspace variable containing the document indices marked for deletion when the “Downdate tdm” radio button is checked.
Field Name	Default	Description

Line Delimiter	●	Checked if the “Delimiter” takes a whole line of text.
Delimiter	emptyline	The delimiter between tmg’s view of documents. Possible values are ’emptyline’, ’none_delimiter’ (treats each file as single document) or any other string.
Stoplist	-	Name of file containing stopwords, i.e. common words not used in indexing.
Min Length	3	Minimum term length.
Max Length	30	Maximum term length.
Min Local Frequency	1	Minimum local term frequency.
Max Local Frequency	inf	Maximum local term frequency.
Min Global Frequency	1	Minimum global term frequency.
Max Global Frequency	inf	Maximum global term frequency.
Local Term Weighting	TF	Local term weighting function. Possible values: ’Term Frequency’ (TF), ’Binary’, ’Logarithmic’, ’Alternate Log’, ’Augmented Normalized Term Frequency’.
Global Term Weighting	None	Global term weighting function. Possible values: ’None’, ’Entropy’, ’Inverse Document Frequency (IDF)’, ’GfIdf’, ’Normal’, ’Probabilistic Inverse’.
Database Name	-	The name of the folder (under ’data’ directory) where data are to be saved (currently supported only for the “Create New tdm” module).
Store in MySQL	-	Checked if results are to be saved into MySQL (currently supported only for the “Create New tdm” module).
use Normalization	-	Indicates normalization method. Possible values: ’None’, ’Cosine’.
use Stemming	-	Indicates if stemming is to be applied. The algorithm currrently supported is due to Porter.
Display Results	●	Display results or not to the command windows.
Continue	-	Apply the selected operation.
Reset	-	Reset window to default values.
Exit	-	Exit window.

Table 1: Description of use of `tmg_gui` components.

3.2 Dimensionality Reduction module (`dr_gui`)



Text to Term-Document Matrix
(tdm) Generator

Select Dataset

Method
● Singular Value Decomposition (SVD)
○ Principal Component Analysis (PCA)
○ Clustered Latent Semantic Indexing (CLSI)
○ Centroid Method (CM)
○ Semidiscrete Decomposition (SDD)
○ SPQR

Compute SVD with
● MATLAB (svds)
○ Propack

Clustering Algorithm
● Euclidean k-means
Initialize Centroids: At random
○ Spherical k-means
Termination Criterion: Epsilon 1
○ PDDP
Principal Directions: 1
Variant: Basic
Maximum num. of PCs ☐
Automatic Determination of Num. of factors for each cluster ☑
Number of Clusters
☑ Display Results
Select at least one factor from each cluster (Recommented for classification) ☐

Number of Factors
☑ Store Results

Figure 4: The `dr_gui` GUI.

This module deploys a variety of powerful techniques designed to efficiently handle high dimensional data. Dimensionality Reduction (DR) is a common technique that is widely used. The target is dual: (a) more economical representation of data, and (b) better semantic representation. TMG implements six DR techniques.

- Singular Value Decomposition (SVD).
- Principal Component Analysis (PCA).
- Clustered Latent Semantic Indexing (CLSI) [12, 13].
- Centroids Method (CM) [7].
- Semidiscrete Decomposition (SDD) [6].

- SPQR Decomposition [2].

DR data can be stored as '.mat' files and used for further processing.

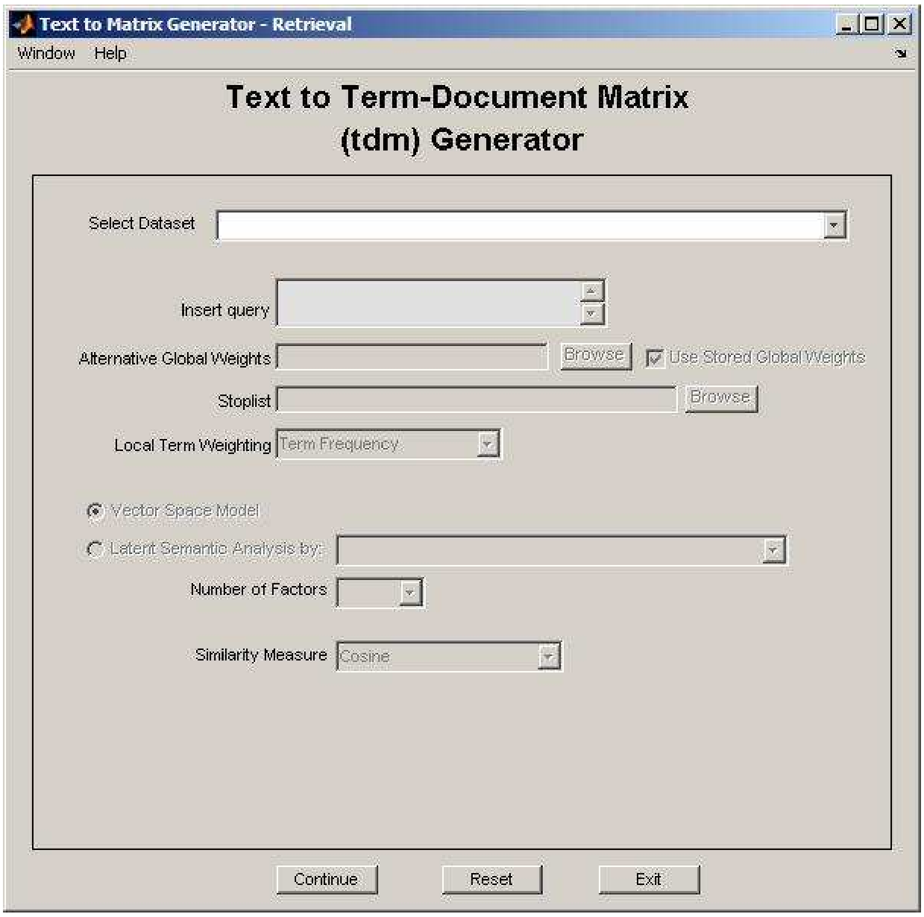
The dimensionality reduction GUI module is depicted in Figure 4 while Table 2 describes in detail all the `dr_gui` fields.

Field Name	Default	Description
Select Dataset	-	Select the dataset.
Singular Value Decomposition (SVD)	●	Apply the SVD method.
Principal Component Analysis (PCA)	-	Apply the PCA method.
Clustered Latent Semantic Indexing (CLSI)	-	Apply the CLSI method.
Centroid Method (CM)	-	Apply the CM method.
Semidiscrete Decomposition (SDD)	-	Apply the SDD method.
SPQR	-	Apply the SPQR method.
MATLAB (svds)	●	Check to use MATLAB function svds for the computation of the SVD or PCA.
Propack	-	Check to use PROPACK package for the computation of the SVD or PCA.
Euclidean k-means	●	Check to use the euclidean k-means clustering algorithm in the course of CLSI or CM.
Spherical k-means	-	Check to use the spherical k-means clustering algorithm in the course of CLSI or CM.
PDDP	-	Check to use the PDDP clustering algorithm in the course of CLSI or CM.
Initialize Centroids	At random	Defines the method used for the initialization of the centroid vector in the course of k-means. Possibilities are: initialize at random and supplly a variable of '.mat' file with the centroids matrix.
Termination Criterion	Epsilon (1)	Defines the termination criterion used in the course of k-means. Possibilities are: use an epsilon value (default 1) and stop iteration when the objective function improvement does not exceed epsilon or perform a specific number of iterations (default 10).
Principal Directions	1	Number of principal directions used in PDDP.

Maximum num. of PCs	-	Check if the PDDP(max-l) variant is to be applied.
Variant	Basic	A set of PDDP variants. Possibe values: ’Basic’, ’Split with k-means’, ’Optimat Split’, ’Optimal Split with k-means’, ’Optimal Split on Projection’.
Automatic Determination of Num. of factors for each cluster	●	Check to apply a heuristic for the determination of the number of factors computed from each cluster in the course of the CLSI algorithm.
Number of Clusters	-	Number of clusters computed in the course of the CLSI algorithm.
Display Results	●	Display results or not to the command windows.
Select at least one factor from each cluster	-	Use this option in case low-rank data are to be used in the course of classification.
Number of factors	-	Rank of approximation.
Store Results	●	Check to store results.
Continue	-	Apply the selected operation.
Reset	-	Reset window to default values.
Exit	-	Exit window.

Table 2: Description of use of `dr_gui` components.

3.3 Retrieval module (`retrieval_gui`)



Text to Matrix Generator - Retrieval

Text to Term-Document Matrix
(tdm) Generator

☑ Use Stored Global Weights

Local Term Weighting: Term Frequency

● Vector Space Model

○ Latent Semantic Analysis by:

Similarity Measure: Cosine

Figure 5: The `retrieval_gui` GUI.

TMG offers two alternatives for Text Mining.

- Vector Space Model (VSM) [9].
- Latent Semantic Analysis (LSA) [1, 4],

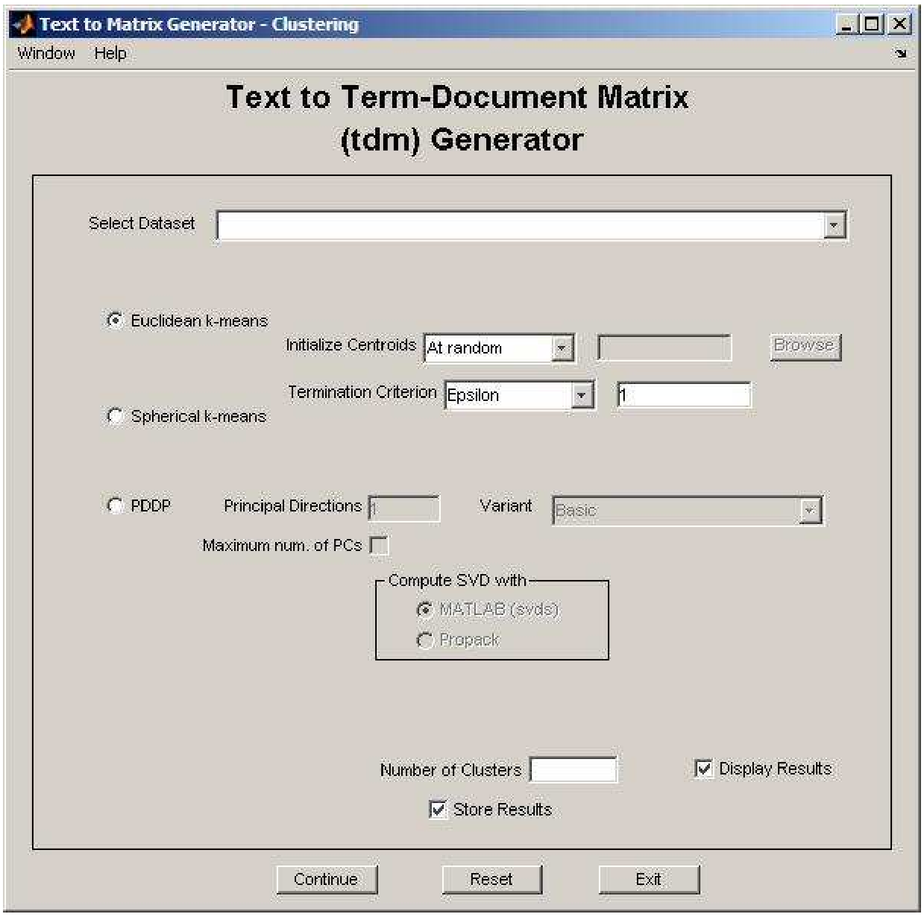
using a combination of any DR technique and Latent Semantic Indexing (LSI). Using the corresponding GUI, the user can apply a question to an existing dataset using any of the aforementioned techniques and get HTML response.

The retrieval GUI module is depicted in Figure 5 while Table 3 describes in detail all the `retrieval_gui` fields.

Field Name	Default	Description
Select Dataset	-	Select the dataset.
Insert Query	●	The query to be executed.
Alternative Global Weights	-	Global weights vector used for the construction of the query vector.
Use Stored Global Weights	●	Use the global weights vector found on the container directory of the dataset.
Stoplist	-	Use a stoplist.
Local Term Weighting	TF	The local term weighting to be used.
Vector Space Model	●	Apply the Vector space Model retrieval method.
Latent Semantic Analysis	-	The method used in the course of the Latent Semantic Analysis technique. Possible values: 'Singular Value Decomposition', 'Principal Component Analysis', 'Clustered Latent Semantic Analysis', 'Centroid Mathod', 'Semidiscrete Decomposition', 'SPQR'.
Number of Factors	-	Select the number of factors used during the retrieval process.
Similarity Measure	Cosine	Similarity measure used during the retrieval process.
Continue	-	Apply the selected operation.
Reset	-	Reset window to default values.
Exit	-	Exit window.

Table 3: Description of use of `retrieval_gui` components.

3.4 Clustering module (clustering_gui)



Text to Term-Document Matrix (tdm) Generator

Select Dataset

● Euclidean k-means

Initialize Centroids: At random

Termination Criterion: Epsilon, 1

○ Spherical k-means

○ PDDP

Principal Directions: 1

Variant: Basic

Maximum num. of PCs □

Compute SVD with: ● MATLAB (svds) ○ Propack

Number of Clusters

☑ Display Results

☑ Store Results

Figure 6: The clustering_gui GUI.

TMG implements three clustering algorithms.

- k-means.
- Spherical k-means [5].
- Principal Direction Divisive Partitioning (PDDP) [3, 11].

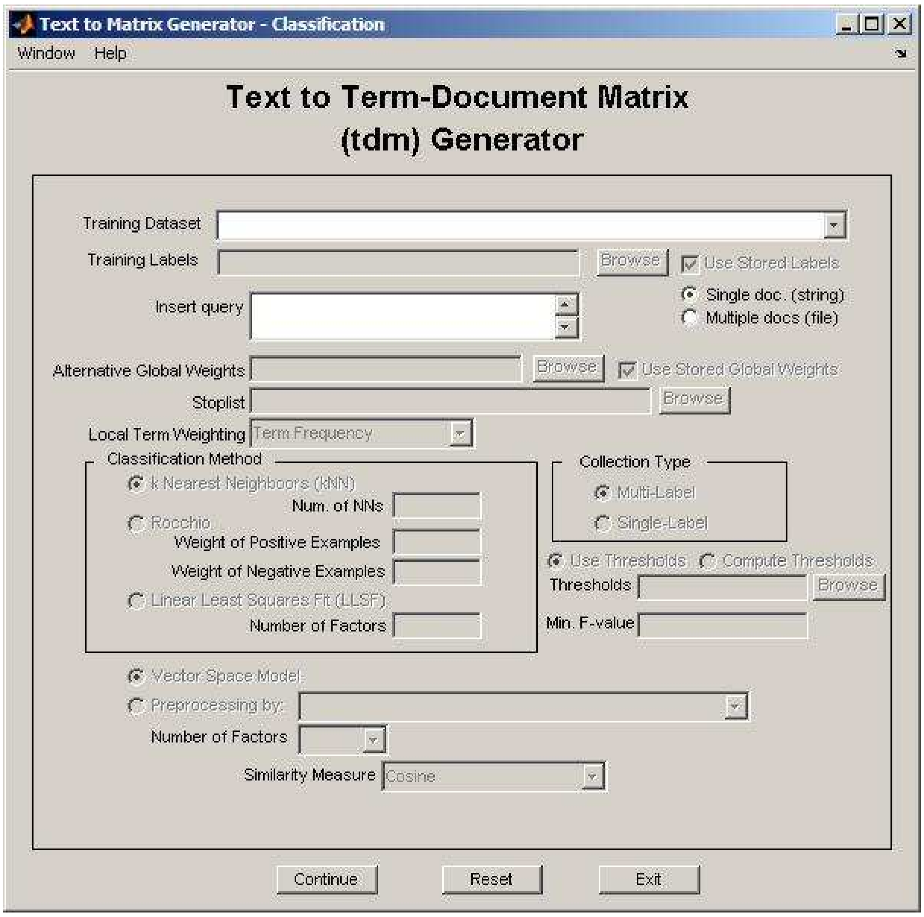
Regarding PDDP, TMG implements the basic algorithm as well as the PDDP(l) [11] along with some recent hybrid variants of PDDP and kmeans [15].

The clustering GUI module is depicted in Figure 6 while Table 4 describes in detail all the clustering_gui fields.

Field Name	Default	Description
Select Dataset	-	Select the dataset.
Euclidean k-means	●	Check to use the euclidean k-means clustering algorithm.
Spherical k-means	-	Check to use the spherical k-means clustering algorithm.
PDDP	-	Check to use the PDDP clustering algorithm.
Initialize Centroids	At random	Defines the method used for the initialization of the centroid vector in the course of k-means. Possibilities are: initialize at random and supplly a variable of '.mat' file with the centroids matrix.
Termination Criterion	Epsilon (1)	Defines the termination criterion used in the course of k-means. Possibilities are: use an epsilon value (default 1) and stop iteration when the objective function improvement does not exceed epsilon or perform a specific number of iterations (default 10).
Principal Directions	1	Number of principal directions used in PDDP.
Maximum num. of PCs	-	Check if the PDDP(max-l) variant is to be applied.
Variant	Basic	A set of PDDP variants. Possibe values: 'Basic', 'Split with k-means', 'Optimat Split', 'Optimal Split with k-means', 'Optimal Split on Projection'.
MATLAB (svds)	●	Check to use MATLAB function svds for the computation of the SVD in the course of PDDP.
Propack	-	Check to use PROPACK package for the computation of the SVD in the course of PDDP.
Number of Clusters	-	Number of clusters computed.
Display Results	●	Display results or not to the command windows.
Store Results	●	Check to store results.
Continue	-	Apply the selected operation.
Reset	-	Reset window to default values.
Exit	-	Exit window.

Table 4: Description of use of `clustering_gui` components.

3.5 Classification module (`classification_gui`)



☑ Use Stored Labels

● Single doc. (string)

☑ Use Stored Global Weights

Local Term Weighting Term Frequency

● k Nearest Neighboors (kNN)

● Multi-Label

● Use Thresholds

● Vector Space Model

Similarity Measure Cosine

Figure 7: The `classification_gui` GUI.

TMG implements three classification algorithms.

- k Nearest Neighboors (kNN).
- Rocchio.
- Linear Least Squares Fit (LLSF) [10].

All these algorithms can be combined with CLSI, CM and SVD DR techniques.

The classification GUI module is depicted in Figure 7 while Table 5 describes in detail all the `classification_gui` fields.

Field Name	Default	Description
Training Dataset	-	The training dataset.
Training Labels	-	The labels of the training dataset.
Use Stored Labels	●	Check to use the stored vector of labels of training documents in the container folder.
Insert query	-	The test document(s).
Single doc. (string)	●	Check if a single test document is to be inserted.
Multiple docs (file)	-	Check if multiple test document are to be inserted.
Filename	-	In 'Multiple docs (file)' is checked, insert the filename containing the test documents.
Delimiter	-	In 'Multiple docs (file)' is checked, insert the delimiter o be used for the test documents.
Line Delimiter	●	In 'Multiple docs (file)' is checked, check if delimiter of test documents' file takes a whole l of text.
Alternative Global Weights	-	Global weights vector used for the construction of the test documents' vectors.
Use Stored Global Weights	●	Use the global weights vector found on the container directory of the training dataset.
Stoplist	-	Use a stoplist.
Local Term Weighting	TF	The local term weighting to be used.
k Nearest Neighboors (kNN)	●	Check if the kNN classifier is to be applied.
Num. of NNs	-	Number of Nearest Neighboors in kNN classifier.
Rocchio	-	Check if Rocchio classifier is to be applied.
Weight of Positive Examples	-	The weight of the positive examples in the formation of the centroids vectors in Rocchio.
Weight of Negative Examples	-	The weight of the negative examples in the formation of the centroids vectors in Rocchio.
Linear Least Squares Fit (LLSF)	-	Check if LLSF classifier is to be applied.
Number of Factors	-	Number of factors used in the course of LLSF.
Multi-Label	●	Check if classifier is to be applied for a multi-label collection.
Single-Label	-	Check is classifier is to be applied for a single-label collection.
Use Thresholds	●	If 'Multi-Label' radio button is checked, use a stored vector of thresholds.
Compute Thresholds	-	If 'Multi-Label' radio button is checked, compute thresholds.
Thresholds	-	If 'Multi-Label' and 'Use Thresholds' radio buttons are checked, supply a stored vector of thresholds.

Min. F-value	-	If ’Multi-Label’ and ’Compute Thresholds’ radio buttons are checked, supply minimum F1 value used in the thresholding algorithm.
Vector Space Model	●	Use the basic Vector Space Model.
Preprocessing by	-	Use preprocessed training data with: ’Singular Value Decomposition’, ’Principal Component Analysis’, ’Clusteredd Latent Semantic Analysis’, ’Centroid Mathod’, ’Semidiscrete Decomposition’, ’SPQR’.
Number of Factors	-	Number of factors for preprocessed training data.
Similarity Measure	Cosine	The similarity measure to be used.
Continue	-	Apply the selected operation.
Reset	-	Reset window to default values.
Exit	-	Exit window.

Table 5: Description of use of `classification_gui` components.

Acknowledgments

TMG was conceived after a motivating discussion with Andrew Knyazev regarding a collection of MATLAB tools we had put together to aid in our clustering experiments. We thank our collegues Ioannis Antonellis, Anastasios Zouzias, Efi Kokiopoulou and Constantine Bekas for many helpful suggestions, Jacob Kogan and Charles Nicholas for inviting us to contribute to [14], Elias Houstis for his help in the initial phases of this research and Michael Berry, Tamara Kolda and Rasmus Munk Larsen for letting us use and distribute SPQR, SDDPACK and PROPACK software respectively. Special thanks are due to many of the users for their constructive comments regarding TMG. This research was supported in part by a University of Patras "Karatheodori" grant. The first author was also supported by a Bodossaki Foundation graduate fellowship.

References

[1] M. Berry, Z. Drmac, and E. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review **41** (1998), 335–362.

[2] M. W. Berry, S. A. Pulatova, and G. W. Stewart, *Computing sparse reduced-rank approximations to sparse matrices*, ACM TOMS **31** (2005), no. 2.

[3] D. Boley, *Principal direction divisive partitioning*, Data Mining and Knowledge Discovery **2** (1998), no. 4, 325–344.

[4] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and Harshman R., *Indexing by Latent Semantic Analysis*, Journal of the American Society for Information Science **41** (1990), no. 6, 391–407.

[5] I. S. Dhillon and D. S. Modha, *Concept decompositions for large sparse text data using clustering*, Machine Learning **42** (2001), no. 1, 143–175.

[6] T. Kolda and D. O'Leary, *Algorithm 805: computation and uses of the semidiscrete matrix decomposition*, ACM TOMS **26** (2000), no. 3.

[7] H. Park, M. Jeon, and J. Rosen, *Lower dimensional representation of text data based on centroids and least squares*, BIT **43** (2003).

[8] M.F. Porter, *An algorithm for suffix stripping*, Program (1980), no. 3, 130–137.

[9] G. Salton, C. Yang, and A. Wong, *A Vector-Space Model for Automatic Indexing*, Communications of the ACM **18** (1975), no. 11, 613–620.

[10] Y. Yang and C. Chute, *A linear least squares fit mapping method for information retrieval from natural language texts*, In 14th Conf. Comp. Linguistics, 1992.

[11] D. Zeimpekis and E. Gallopoulos, *PDDP(l): Towards a Flexing Principal Direction Divisive Partitioning Clustering Algorithms*, Proc. IEEE ICDM '03 Workshop on Clustering Large Data Sets (Melbourne, Florida) (D. Boley, I. Dhillon, J. Ghosh, and J. Kogan, eds.), 2003, pp. 26–35.

[12] D. Zeimpekis and E. Gallopoulos, *CLSI: A flexible approximation scheme from clustered term-document matrices*, In Proc. SIAM 2005 Data Mining Conf. (Newport Beach, California) (H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman, eds.), April 2005, pp. 631–635.

[13] D. Zeimpekis and E. Gallopoulos, *Linear and non-linear dimensional reduction via class representatives for text classification*, In Proc. of the 2006 IEEE International Conference on Data Mining (Hong Kong), December 2006, pp. 1172–1177.

[14] D. Zeimpekis and E. Gallopoulos, *TMG: A MATLAB toolbox for generating term-document matrices from text collections*, Grouping Multidimensional Data: Recent Advances in Clustering (J. Kogan, C. Nicholas, and M. Teboulle, eds.), Springer, Berlin, 2006, pp. 187–210.

[15] D. Zeimpekis and E. Gallopoulos, *k-means steering of spectral divisive clustering algorithms*, In Proc. of Text Mining Workshop (Minneapolis), 2007.

A Appendix: Demonstration of Use

A.1 Indexing module (`tmg_gui`)

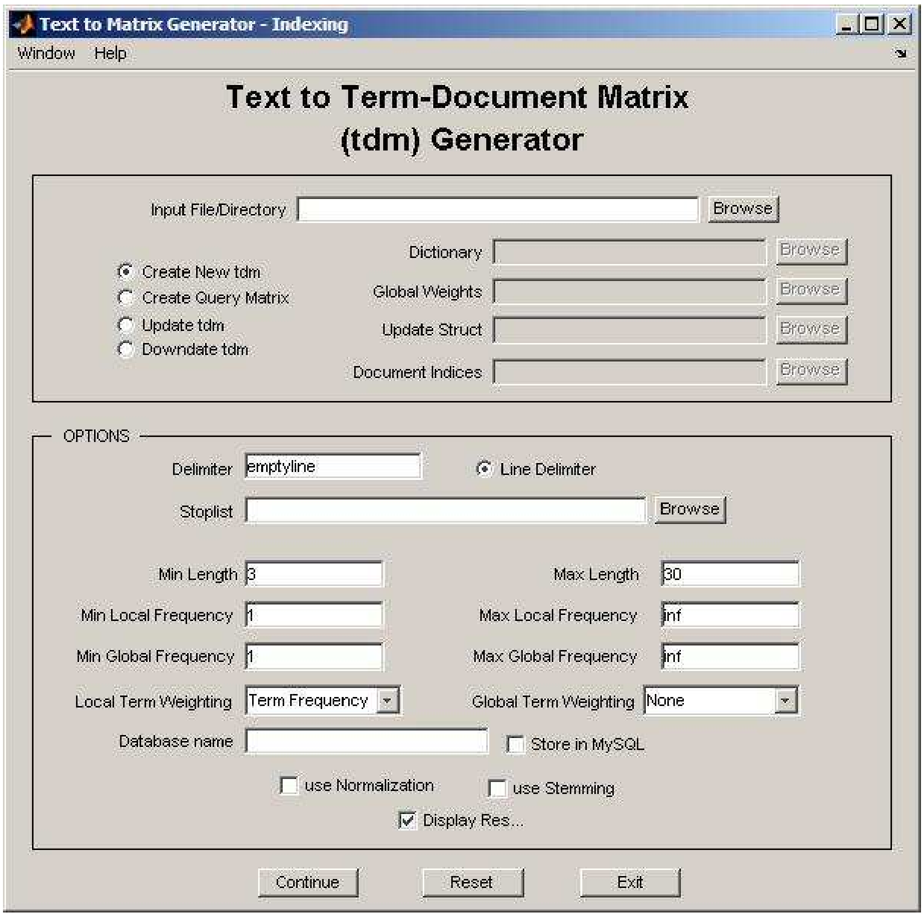
Assume we want to run `tmg.m` for the following input:

- filename: sample_documents/sample1
- delimiter: emptyline
- line_delimiter: yes
- stoplist: common_words
- minimum length: 3
- maximum length: 30
- minimum local frequency: 1
- maximum local frequency: inf
- minimum global frequency: 1
- maximum global frequency: inf
- local term weighting: logarithmic
- global term weighting: IDF
- normalization: cosine
- stemming: -

and you want to store results in sample1 directory and in MySQL.

1. Initially select the operation you want to perform, by pressing the corresponding radio button to the upper frame.

2. The selection of a radio button activates the required fields in the GUI, while deactivating the rest fields and changing their background color.



Text to Matrix Generator - Indexing

Text to Term-Document Matrix
(tdm) Generator

Input File/Directory

● Create New tdm
○ Create Query Matrix
○ Update tdm
○ Downdate tdm

Dictionary
Global Weights
Update Struct
Document Indices

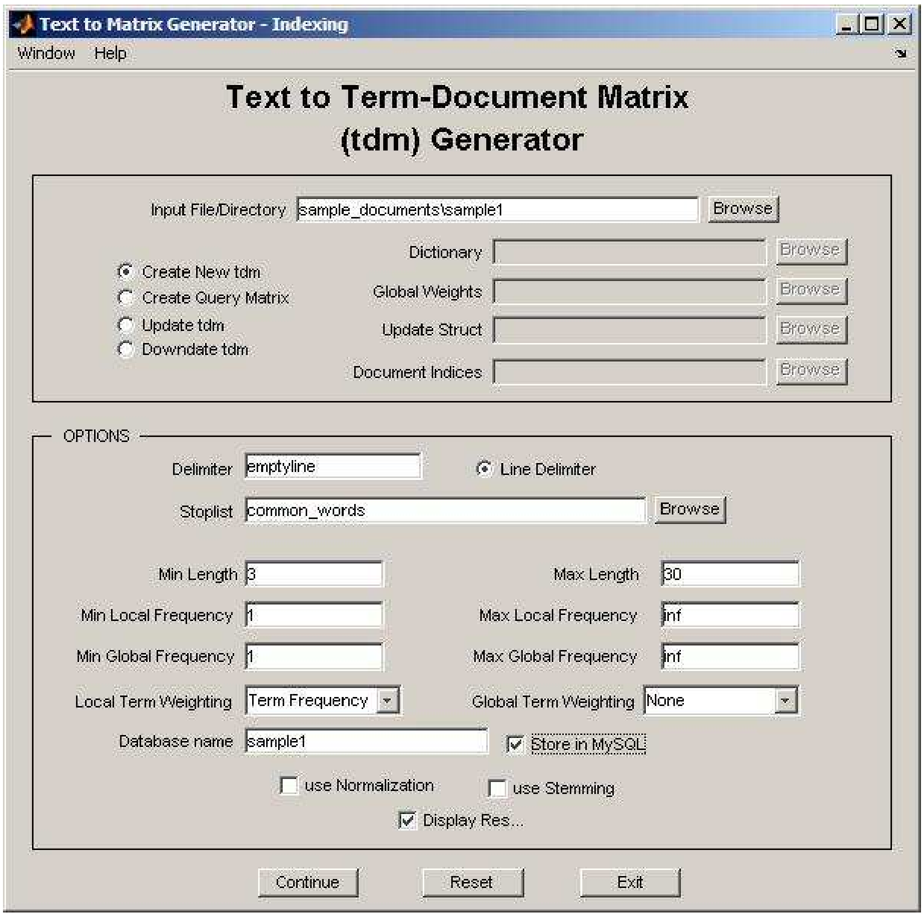
OPTIONS

Delimiter: emptyline
● Line Delimiter
Stoplist

Min Length: 3
Max Length: 30
Min Local Frequency: 1
Max Local Frequency: inf
Min Global Frequency: 1
Max Global Frequency: inf
Local Term Weighting: Term Frequency
Global Term Weighting: None
Database name
□ Store in MySQL
□ use Normalization
□ use Stemming
☑ Display Res...

Figure 8: Starting window of `tmg_gui`.

3. Fill in the required fields, by pressing the check buttons, editing the edit boxes or selecting the appropriate files/variables by pressing a “Browse” button.



Text to Matrix Generator - Indexing

Text to Term-Document Matrix (tdm) Generator

Input File/Directory: sample_documents\sample1

● Create New tdm
○ Create Query Matrix
○ Update tdm
○ Downdate tdm

Dictionary:
Global Weights:
Update Struct:
Document Indices:

OPTIONS

Delimiter: emptyline ● Line Delimiter

Stoplist: common_words

Min Length	3	Max Length	30
Min Local Frequency	1	Max Local Frequency	inf
Min Global Frequency	1	Max Global Frequency	inf
Local Term Weighting	Term Frequency	Global Term Weighting	None

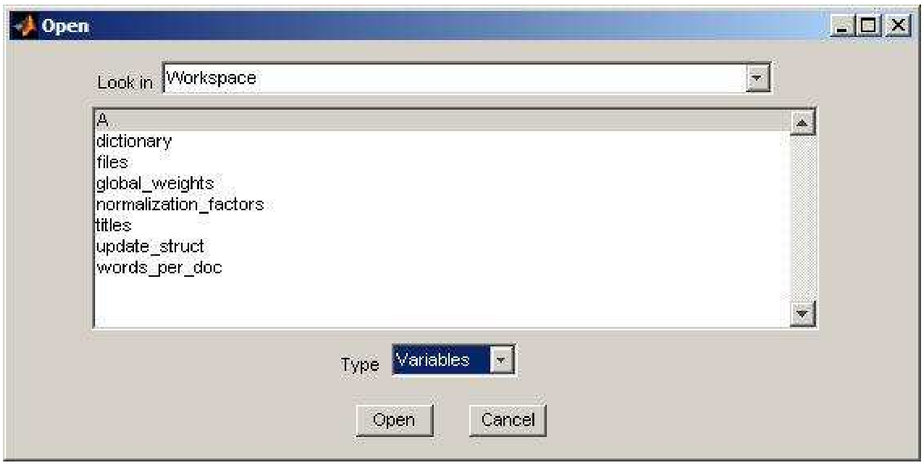
Database name: sample1 ☑ Store in MySQL

□ use Normalization □ use Stemming

☑ Display Res...

Figure 9: Next view of `tmg_gui` according to the user selection.

4. Pressing a “Browse” button the user has the ability to chose a file or a variable.



Look in Workspace

A
dictionary
files
global_weights
normalization_factors
titles
update_struct
words_per_doc

Type Variables

Figure 10: The `open_file` window.

5. Press the ”Continue” button in order to perform the selected operation.

6. Results have been saved to the workspace. Furthermore, directory ‘sample1’ has been created under ‘TMG_HOME/data’ with each output variable stored in a single ‘.mat’ file.

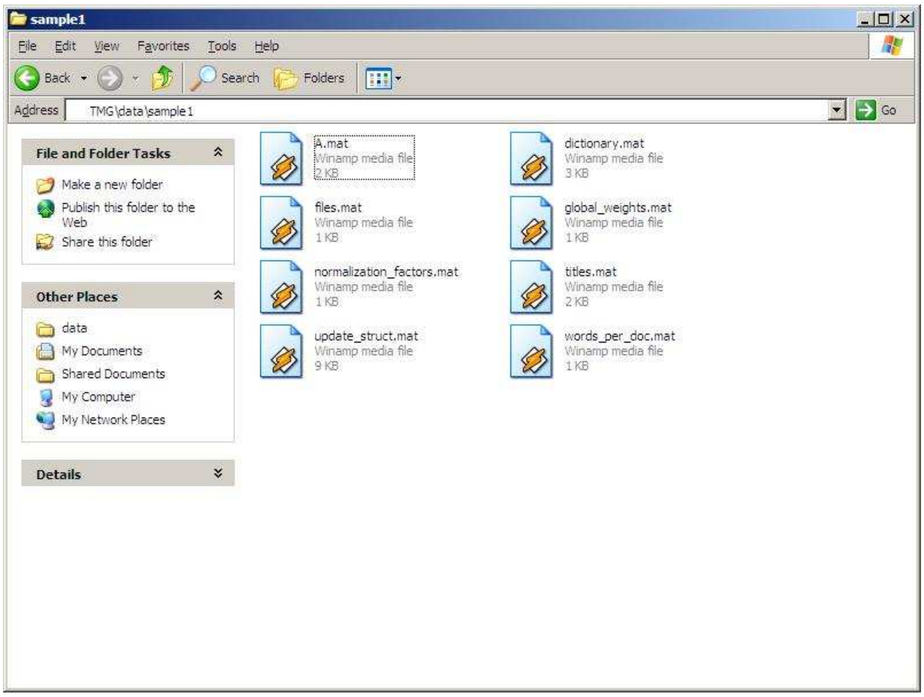
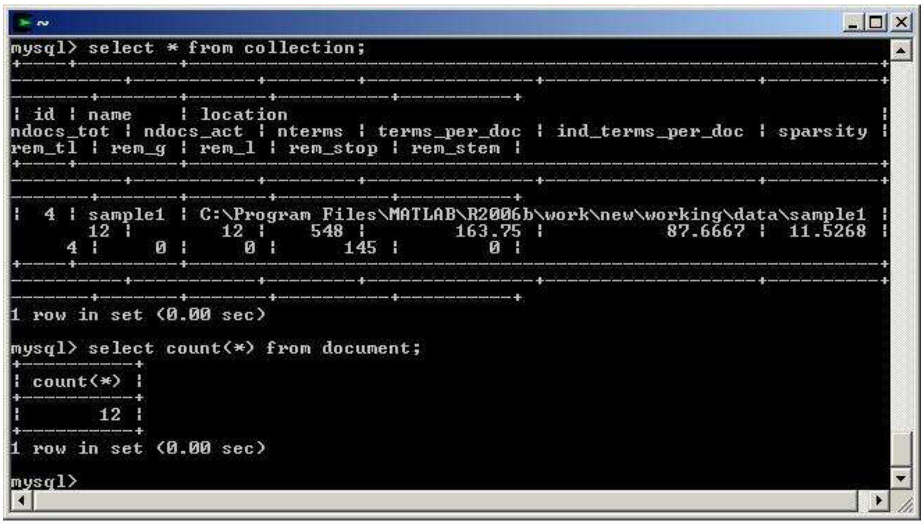


Figure 11: The output ‘.mat’ files of `tmg_gui`.

7. Results have also been saved in MySQL (used for further processing, e.g. `retrieval_gui`).

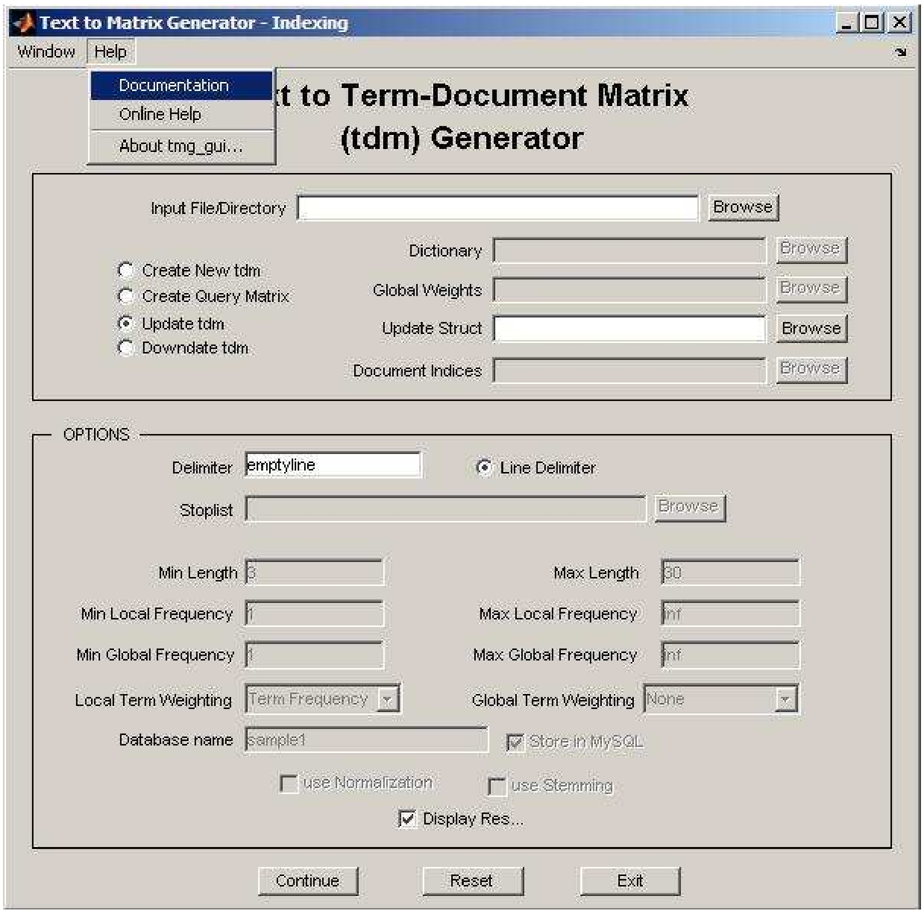


```
mysql> select * from collection;
+----+---------+--------------------------------------------------------------+-
----------+-----------+--------+---------------+-------------------+----------+-
-------+-------+-------+----------+----------+
| id | name    | location                                                     |
ndocs_tot | ndocs_act | nterms | terms_per_doc | ind_terms_per_doc | sparsity |
rem_tl | rem_g | rem_l | rem_stop | rem_stem |
+----+---------+--------------------------------------------------------------+-
----------+-----------+--------+---------------+-------------------+----------+-
-------+-------+-------+----------+----------+
|  4 | sample1 | C:\Program Files\MATLAB\R2006b\work\new\working\data\sample1 |
       12 |        12 |    548 |        163.75 |           87.6667 |  11.5268 |
     4 |     0 |     0 |      145 |        0 |
+----+---------+--------------------------------------------------------------+-
----------+-----------+--------+---------------+-------------------+----------+-
-------+-------+-------+----------+----------+
1 row in set (0.00 sec)

mysql> select count(*) from document;
+----------+
| count(*) |
+----------+
|       12 |
+----------+
1 row in set (0.00 sec)

mysql>
```

Figure 12: The MySQL view uppon `tmg` execution.

8. Press the “Reset” button in order to change the input.

9. For further documentation type ‘`help tmg_gui`’ at the MATLAB command window, or select the “Documentation” tab from the “Help” menu.



Input File/Directory

○ Create New tdm
○ Create Query Matrix
● Update tdm
○ Downdate tdm

Dictionary
Global Weights
Update Struct
Document Indices

Delimiter emptyline
● Line Delimiter
Stoplist

Min Length 3
Max Length 30
Min Local Frequency 1
Max Local Frequency Inf
Min Global Frequency 1
Max Global Frequency Inf
Local Term Weighting Term Frequency
Global Term Weighting None
Database name sample1
☑ Store in MySQL
□ use Normalization
□ use Stemming
☑ Display Res...

Figure 13: The GUIs’ general help tab.

10. In order to update a tdm, give the “input file/directory” and the update_struct corresponding to the initial collection. In case you just want to alter some options, give a blank “input file/direcory” and change the corresponding fields of update_struct.

11. In order to downdate a tdm, give the update_struct corresponding to the initial collection and the document indices vector you want to remove.

12. In order to construct a term-query matrix, give the dictionary char array of the initial collection and the corresponding vector of global weights (optional).

A.2 Dimensionality Reduction module (`dr_gui`)

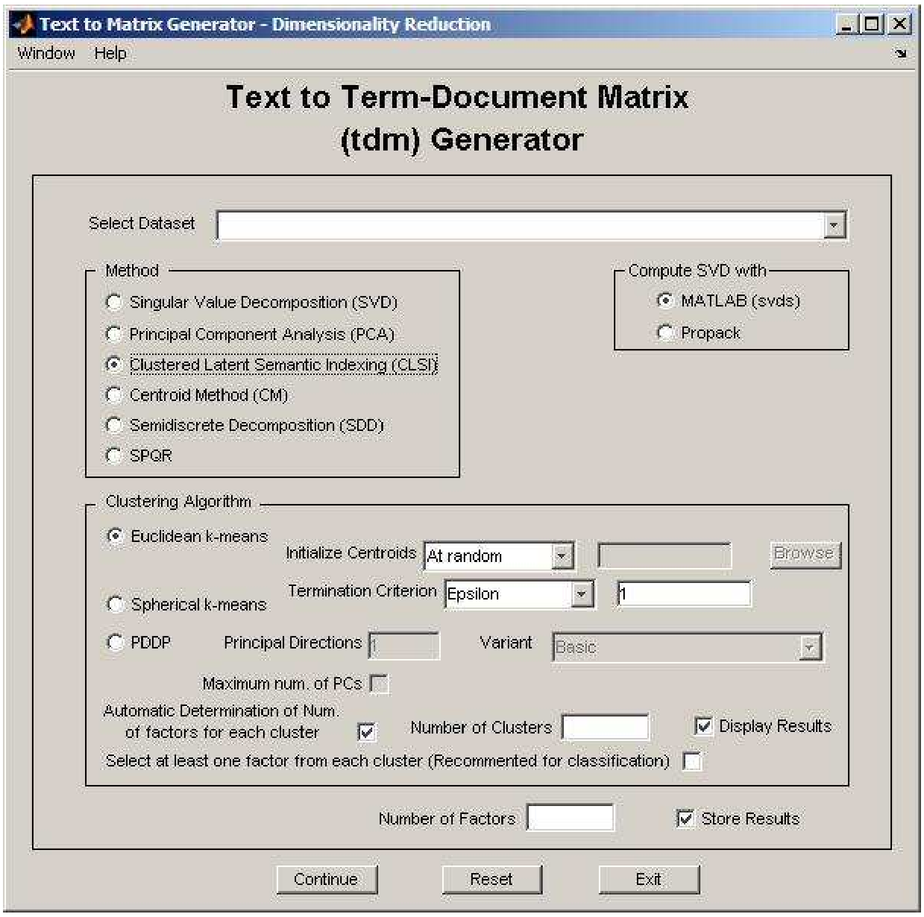
Suppose we have processed a collection with `tmg_gui`, construct a tdm with 1,033 documents and 12,184 terms (corresponding to the well-known MEDLINE collection) and store the results in 'TMG_HOME/data/medline'. Assume then, we want to construct a low-rank approximation of the tdm ,using the Clustered Latent Semantic Indexing (CLSI) technique for the following input:

- compute SVD with: Propack
- clustering algorithm: PDDP
- principal directions: 1
- maximum number of PCs: -
- variant: basic
- automatic determination of num. of factors from each cluster: yes
- number of clusters: 10
- number of factors: 100

and you want to store results in the medline directory.

1. Initially select the operation you want to perform, by pressing the corresponding radio button to the upper left frame.

2. The selection of a radio button activates the required fields in the GUI, while deactivating the rest fields and changing their background color.



Text to Matrix Generator - Dimensionality Reduction
Window
Help
Text to Term-Document Matrix (tdm) Generator

Select Dataset

Method
○ Singular Value Decomposition (SVD)
○ Principal Component Analysis (PCA)
● Clustered Latent Semantic Indexing (CLSI)
○ Centroid Method (CM)
○ Semidiscrete Decomposition (SDD)
○ SPQR

Compute SVD with
● MATLAB (svds)
○ Propack

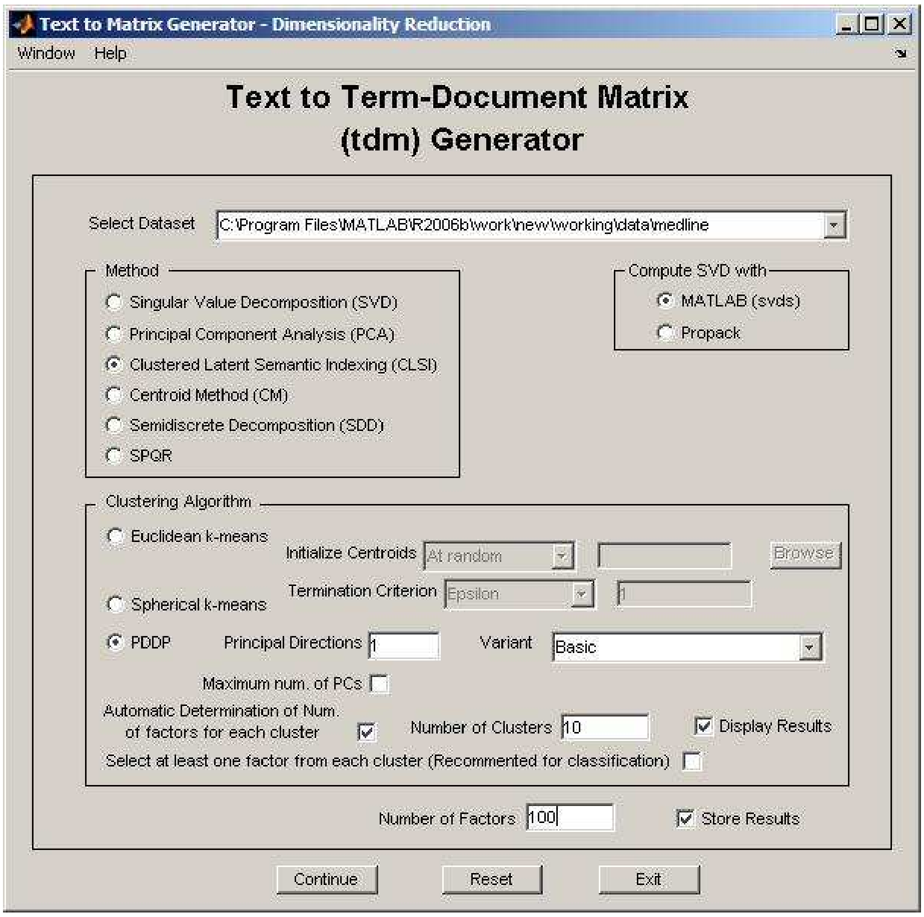
Clustering Algorithm
● Euclidean k-means
○ Spherical k-means
○ PDDP

Initialize Centroids: At random
Termination Criterion: Epsilon 1
Principal Directions: 1
Variant: Basic
Maximum num. of PCs □
Automatic Determination of Num. of factors for each cluster ☑
Number of Clusters
☑ Display Results
Select at least one factor from each cluster (Recommented for classification) □

Number of Factors
☑ Store Results

Figure 14: Starting window of `dr_gui`.

3. Fill in the required fields, by pressing the check buttons, editing the edit boxes or selecting the appropriate files/variables by pressing a “Browse” button.



Text to Term-Document Matrix (tdm) Generator

Select Dataset: C:\Program Files\MATLAB\R2006b\work\new\working\data\medline

Method
- ○ Singular Value Decomposition (SVD)
- ○ Principal Component Analysis (PCA)
- ● Clustered Latent Semantic Indexing (CLSI)
- ○ Centroid Method (CM)
- ○ Semidiscrete Decomposition (SDD)
- ○ SPQR

Compute SVD with
- ● MATLAB (svds)
- ○ Propack

Clustering Algorithm
- ○ Euclidean k-means
- ○ Spherical k-means
- ● PDDP

Initialize Centroids: At random

Termination Criterion: Epsilon 1

Principal Directions: 1

Variant: Basic

Maximum num. of PCs □

Automatic Determination of Num. of factors for each cluster ☑

Number of Clusters: 10

☑ Display Results

Select at least one factor from each cluster (Recommented for classification) □

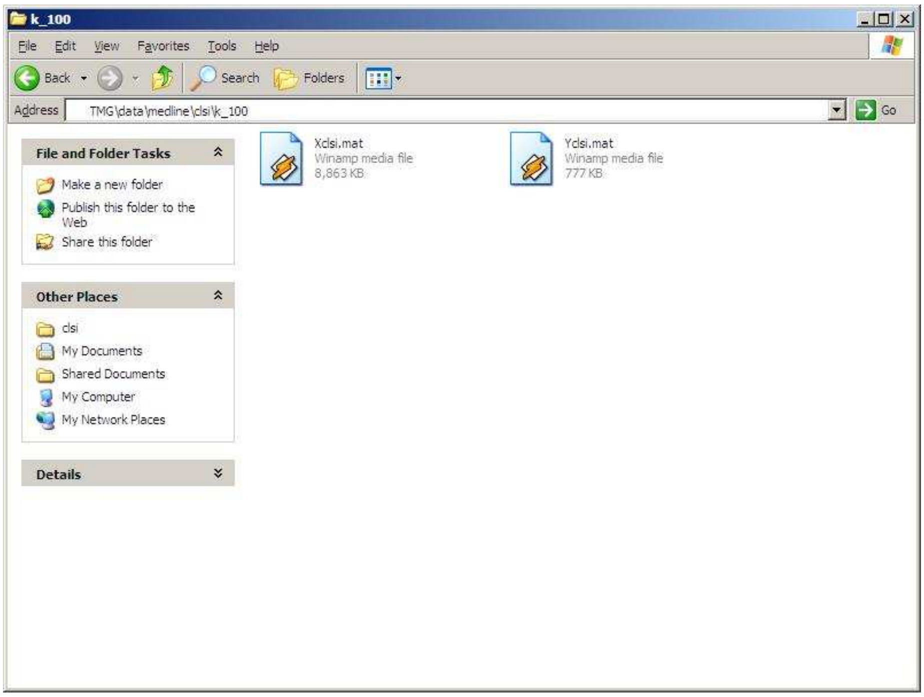
Number of Factors: 100

☑ Store Results

Figure 15: Next view of `dr_gui` according to the user selection.

4. Press the “Continue” button in order to perform selected operation.

5. Results have been saved to the workspace. Furthermore, directory ‘clsi/k_100’ has been created under ‘TMG_HOME/data/medline’ with each output variable stored in a single ‘.mat’ file.



Address TMG\data\medline\clsi\k_100

Xclsi.mat
Winamp media file
8,863 KB

Yclsi.mat
Winamp media file
777 KB

Figure 16: The output ‘.mat’ files of `dr_gui`.

6. Press the “Reset” button in order to change the input.

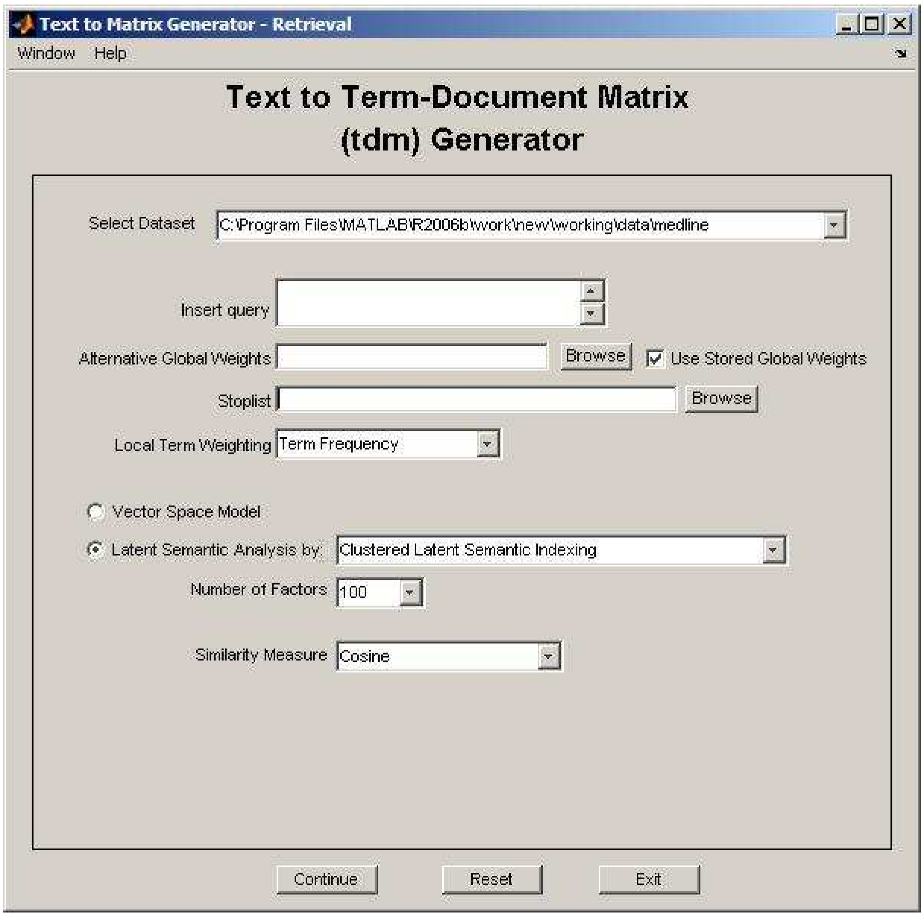
A.3 Retrieval module (`retrieval_gui`)

Suppose we have processed a collection `with tmg_gui`, construct a tdm with 1,033 documents and 12,184 terms (corresponding to the well-known MEDLINE collection) and store the results in 'TMG_HOME/data/medline'. Assume then, we want to retrieve the relevant documents to a specific query for the following input:

- insert query: 'the crystalline lens in vertebrates, including humans'
- use stored global weights: yes
- stoplist: common_words
- local term weighting: Term Frequency
- latent semantic analysis by: Clustered Latent Semantic Indexing
- number of factors: 100
- similarity measure: Cosine

1. Initially select the retrieval method you want to apply, by pressing the corresponding radio button.

2. The selection of a radio button activates the required fields in the GUI, while deactivating the rest fields and changing their background color.


Text to Matrix Generator - Retrieval
Window
Help
Text to Term-Document Matrix
(tdm) Generator
Browse
Browse
Continue
Reset
Exit

Select Dataset C:\Program Files\MATLAB\R2006b\work\new\working\data\medline

Insert query

Alternative Global Weights

☑ Use Stored Global Weights

Stoplist

Local Term Weighting Term Frequency

○ Vector Space Model

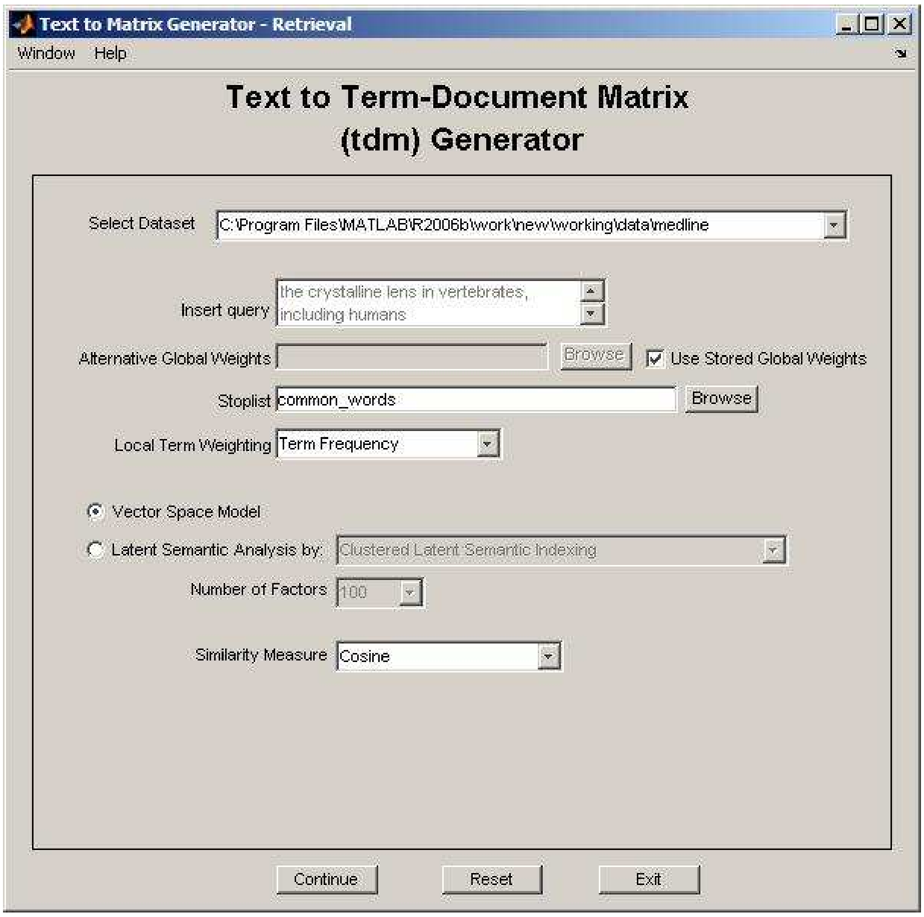
● Latent Semantic Analysis by: Clustered Latent Semantic Indexing

Number of Factors 100

Similarity Measure Cosine

Figure 17: Starting window of `retrieval_gui`.

3. Fill in the required fields, by pressing the check buttons, editing the edit boxes or selecting the appropriate files/variables by pressing a “Browse” button.



Text to Term-Document Matrix (tdm) Generator

Select Dataset: C:\Program Files\MATLAB\R2006b\work\new\working\data\medline

Insert query: the crystalline lens in vertebrates, including humans

Alternative Global Weights:

☑ Use Stored Global Weights

Stoplist: common_words

Local Term Weighting: Term Frequency

● Vector Space Model

○ Latent Semantic Analysis by: Clustered Latent Semantic Indexing

Number of Factors: 100

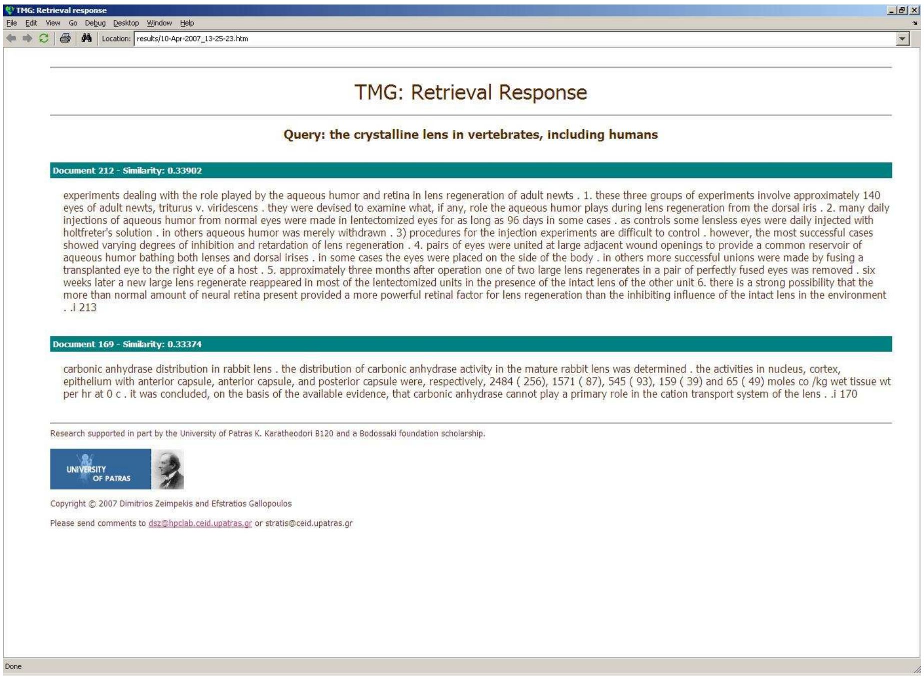
Similarity Measure: Cosine

Figure 18: Next view of `retrieval_gui` according to the user selection.

4. Press the “Continue” button in order to perform selected operation.

5. Results have been saved to the workspace.

6. Furthermore, in case data have been stored to MySQL, the user gets an html response.



Location: results/10-Apr-2007_13-25-23.htm

TMG: Retrieval Response

Query: the crystalline lens in vertebrates, including humans

Document 212 - Similarity: 0.33902

experiments dealing with the role played by the aqueous humor and retina in lens regeneration of adult newts . 1. these three groups of experiments involve approximately 140 eyes of adult newts, triturus v. viridescens . they were devised to examine what, if any, role the aqueous humor plays during lens regeneration from the dorsal iris . 2. many daily injections of aqueous humor from normal eyes were made in lentectomized eyes for as long as 96 days in some cases . as controls some lensless eyes were daily injected with holtfreter's solution . in others aqueous humor was merely withdrawn . 3) procedures for the injection experiments are difficult to control . however, the most successful cases showed varying degrees of inhibition and retardation of lens regeneration . 4. pairs of eyes were united at large adjacent wound openings to provide a common reservoir of aqueous humor bathing both lenses and dorsal irises . in some cases the eyes were placed on the side of the body . in others more successful unions were made by fusing a transplanted eye to the right eye of a host . 5. approximately three months after operation one of two large lens regenerates in a pair of perfectly fused eyes was removed . six weeks later a new large lens regenerate reappeared in most of the lentectomized units in the presence of the intact lens of the other unit 6. there is a strong possibility that the more than normal amount of neural retina present provided a more powerful retinal factor for lens regeneration than the inhibiting influence of the intact lens in the environment . .i 213

Document 169 - Similarity: 0.33374

carbonic anhydrase distribution in rabbit lens . the distribution of carbonic anhydrase activity in the mature rabbit lens was determined . the activities in nucleus, cortex, epithelium with anterior capsule, anterior capsule, and posterior capsule were, respectively, 2484 (256), 1571 (87), 545 (93), 159 (39) and 65 (49) moles co /kg wet tissue wt per hr at 0 c . it was concluded, on the basis of the available evidence, that carbonic anhydrase cannot play a primary role in the cation transport system of the lens . .i 170

Research supported in part by the University of Patras K. Karatheodori B120 and a Bodossaki foundation scholarship.

Copyright © 2007 Dimitrios Zeimpekis and Efstratios Gallopoulos

Please send comments to dsz@hpclab.ceid.upatras.gr or stratis@ceid.upatras.gr

Figure 19: The output of `retrieval_gui`.

7. Press the “Reset” button in order to change the input.

A.4 Clustering module (clustering_gui)

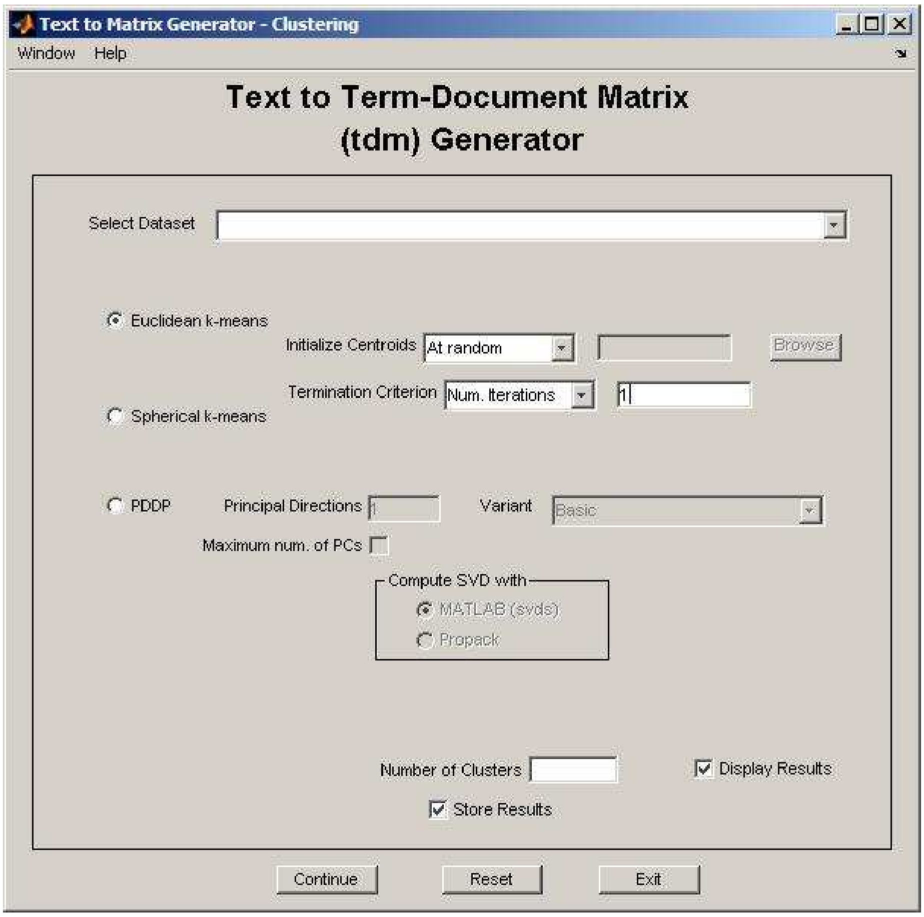
Suppose we have processed a collection with tmg_gui, construct a tdm with 1,033 documents and 12,184 terms (corresponding to the well-known MEDLINE collection) and store the results in 'TMG_HOME/data/medline'. Assume then, we want to cluster the tdm ,using the k-means clustering algorithm with the following input:

- initialize centroids: At random
- termination criterion: Num. iterations, value 10
- number of clusters: 10

and you want to store results in the medline directory.

1. Initially select the clustering algorithm you want to apply, by pressing the corresponding radio button.

2. The selection of a radio button activates the required fields in the GUI, while deactivating the rest fields and changing their background color.


Text to Matrix Generator - Clustering
Window
Help
Text to Term-Document Matrix
(tdm) Generator
Browse
Compute SVD with
Continue
Reset
Exit

Select Dataset

● Euclidean k-means

Initialize Centroids: At random

Termination Criterion: Num. Iterations 1

○ Spherical k-means

○ PDDP

Principal Directions: 1

Variant: Basic

Maximum num. of PCs □

● MATLAB (svds)

○ Propack

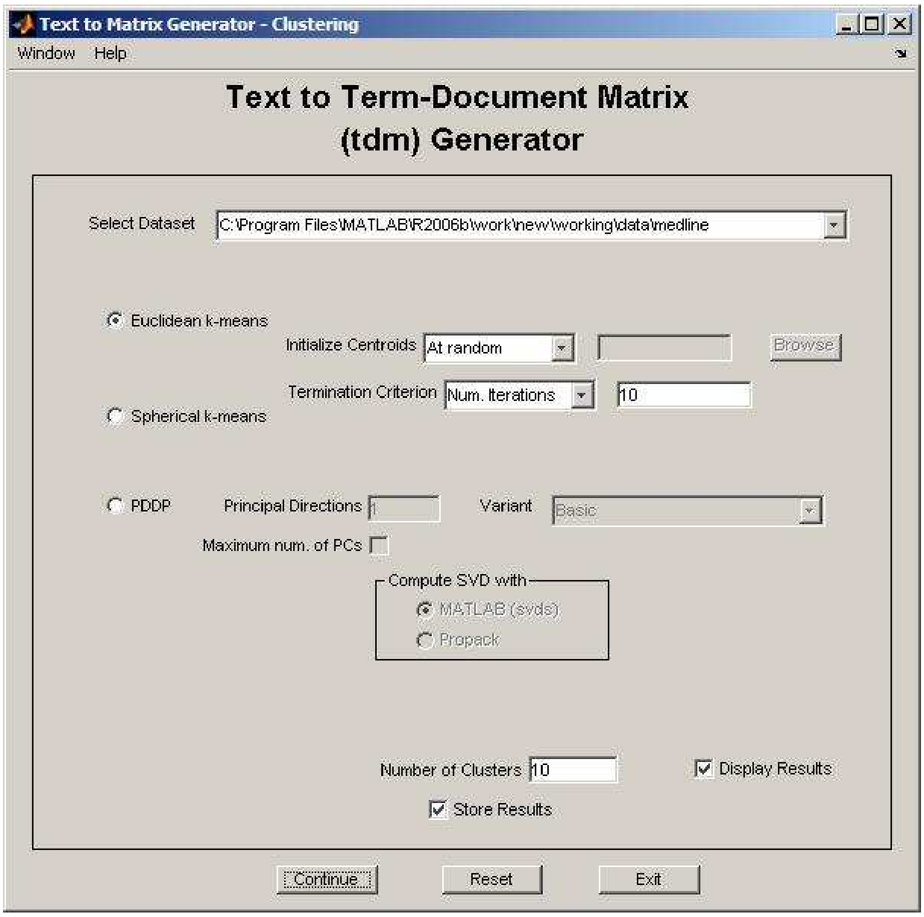
Number of Clusters

☑ Display Results

☑ Store Results

Figure 20: Starting window of `clustering_gui`.

3. Fill in the required fields, by pressing the check buttons, editing the edit boxes or selecting the appropriate files/variables by pressing a “Browse” button.



Text to Term-Document Matrix (tdm) Generator

Select Dataset: C:\Program Files\MATLAB\R2006b\work\new\working\data\medline

● Euclidean k-means

Initialize Centroids: At random

Termination Criterion: Num. Iterations 10

○ Spherical k-means

○ PDDP Principal Directions: 1 Variant: Basic

Maximum num. of PCs □

Compute SVD with

● MATLAB (svds)

○ Propack

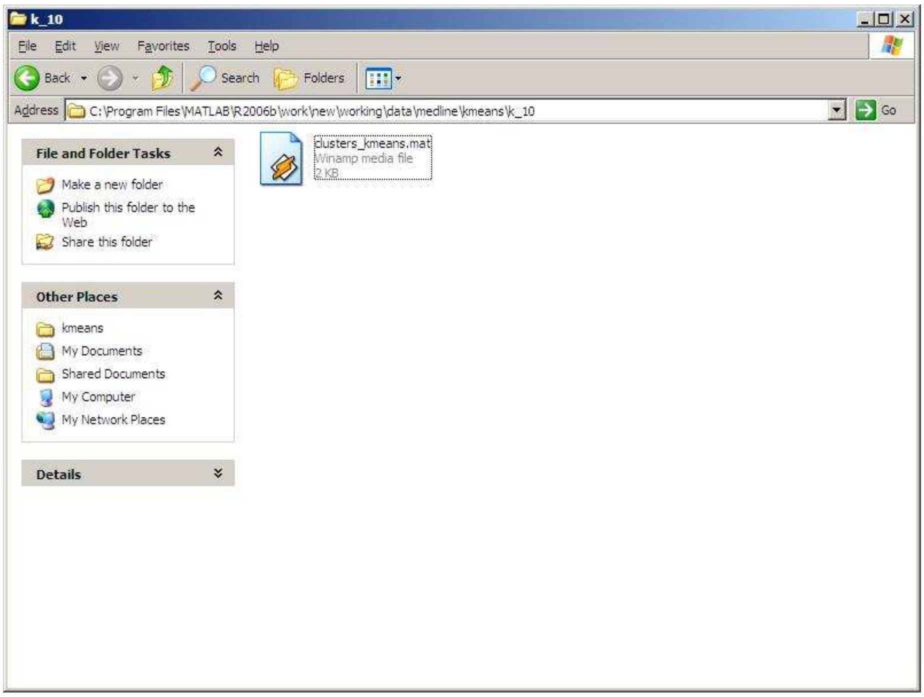
Number of Clusters: 10

☑ Display Results

☑ Store Results

Figure 21: Next view of `clustering_gui` according to the user selection.

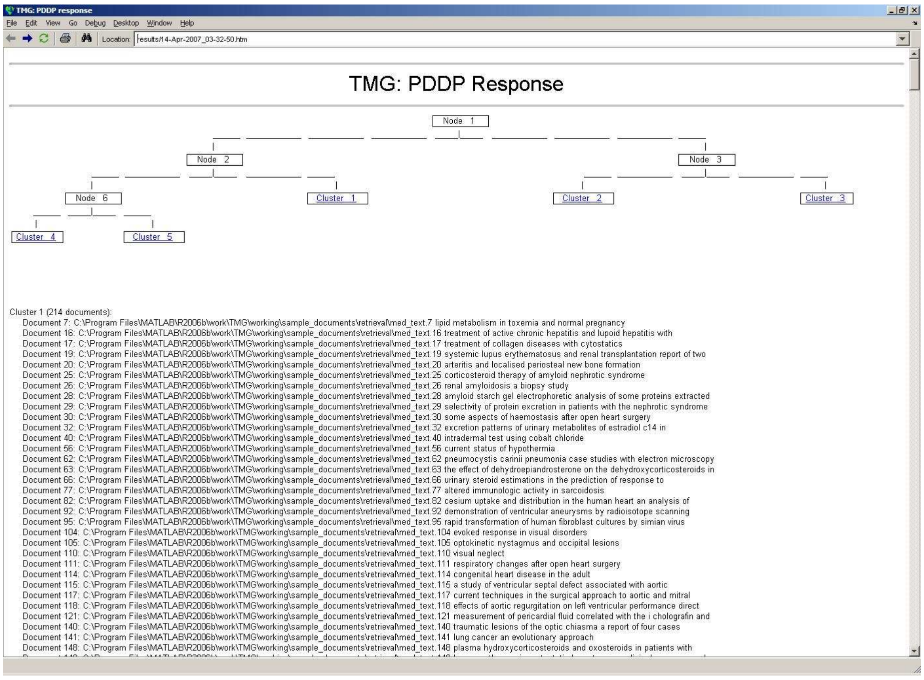
4. Press the “Continue” button in order to perform selected operation.
5. Results have been saved to the workspace. Furthermore, directory ‘kmeans/k_10’ has been created under ‘TMG_HOME/data/medline’ with each output variable stored in a single ‘.mat’ file.



Address: C:\Program Files\MATLAB\R2006b\work\new\working\data\medline\kmeans\k_10

Figure 22: The output ‘.mat’ files of `clustering_gui`.

6. The user gets an html response that summarizes the clustering result.



TMG: PDDP Response

```
Cluster 1 (214 documents):
   Document 7: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.7 lipid metabolism in toxemia and normal pregnancy
   Document 16: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.16 treatment of active chronic hepatitis and lupoid hepatitis with
   Document 17: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.17 treatment of collagen diseases with cytostatics
   Document 19: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.19 systemic lupus erythematosus and renal transplantation report of two
   Document 20: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.20 arteritis and localised periosteal new bone formation
   Document 25: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.25 corticosteroid therapy of amyloid nephrotic syndrome
   Document 26: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.26 renal amyloidosis a biopsy study
   Document 28: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.28 amyloid starch gel electrophoretic analysis of some proteins extracted
   Document 29: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.29 selectivity of protein excretion in patients with the nephrotic syndrome
   Document 30: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.30 some aspects of haemostasis after open heart surgery
   Document 32: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.32 excretion patterns of urinary metabolites of estradiol c14 in
   Document 40: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.40 intradermal test using cobalt chloride
   Document 56: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.56 current status of hypothermia
   Document 62: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.62 pneumocystis carinii pneumonia case studies with electron microscopy
   Document 63: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.63 the effect of dehydroepiandrosterone on the dehydroxycorticosteroids in
   Document 66: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.66 urinary steroid estimations in the prediction of response to
   Document 77: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.77 altered immunologic activity in sarcoidosis
   Document 82: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.82 cesium uptake and distribution in the human heart an analysis of
   Document 92: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.92 demonstration of ventricular aneurysms by radioisotope scanning
   Document 95: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.95 rapid transformation of human fibroblast cultures by simian virus
   Document 104: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.104 evoked response in visual disorders
   Document 105: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.105 optokinetic nystagmus and occipital lesions
   Document 110: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.110 visual neglect
   Document 111: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.111 respiratory changes after open heart surgery
   Document 114: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.114 congenital heart disease in the adult
   Document 115: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.115 a study of ventricular septal defect associated with aortic
   Document 117: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.117 current techniques in the surgical approach to aortic and mitral
   Document 118: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.118 effects of aortic regurgitation on left ventricular performance direct
   Document 121: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.121 measurement of pericardial fluid correlated with the i cholografin and
   Document 140: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.140 traumatic lesions of the optic chiasma a report of four cases
   Document 141: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.141 lung cancer an evolutionary approach
   Document 148: C:\Program Files\MATLAB\R2006b\work\TMG\working\sample_documents\retrieval\med_text.148 plasma hydroxycorticosteroids and oxosteroids in patients with
```

Figure 23: The output of `clustering_gui` for PDDP.

7. Press the “Reset” button in order to change the input.

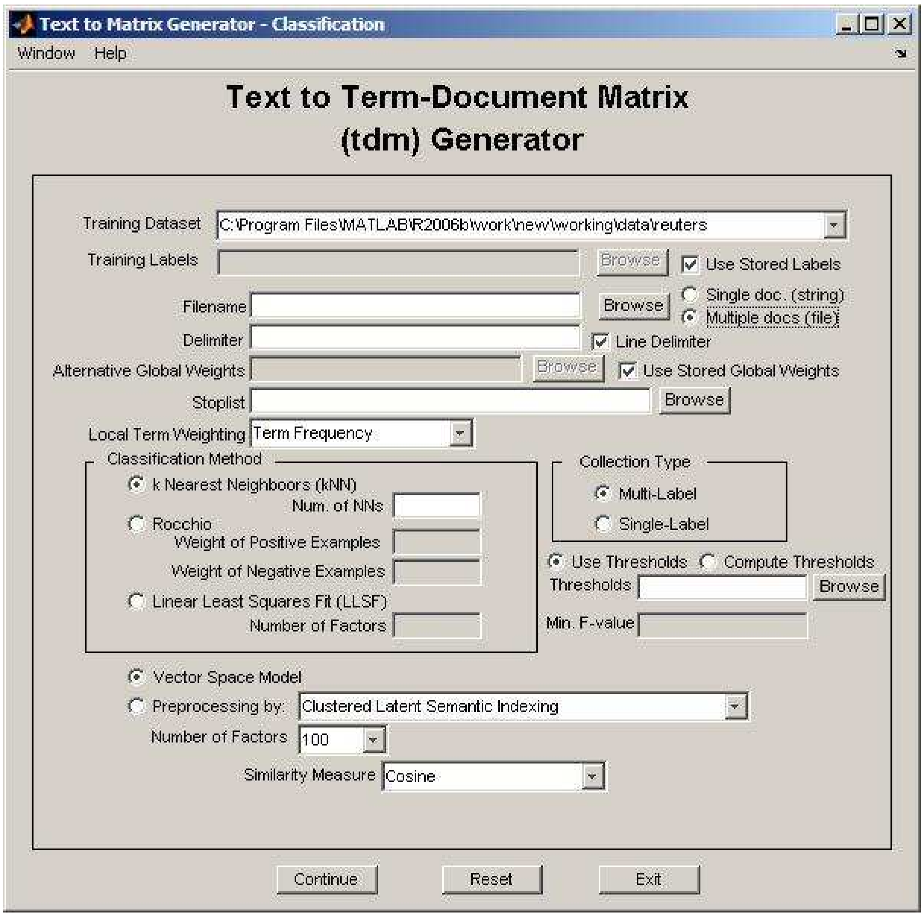
A.5 Classification module (`classification_gui`)

Suppose we have processed a collection with `tmg_gui`, construct a tdm with 6,495 documents and 21,764 terms (a single label dataset corresponding to the well-known modapte split of the Reuters-21578 collection) and store the results in 'TMG_HOME/data/reuters'. Assume then, we want to classify the test part of the modapte split,using the k-Nearest Neighboors classifier for the following input:

- Multiple docs (file): yes
- filename: sample_document/reuters.test
- delimiter: </reuters>
- line delimiter: yes
- use stored global weights: yes
- stoplist: common_words
- local term weighting: Term Frequency
- classification method: k Nearest Neighboors (kNN)
- num. of NNs: 10
- collection type: Single-Label
- preprocessed by: Clustered Latent Semantic Indexing
- number of factors: 100
- similarity measure: Cosine

1. Initially select the classification algorithm you want to apply, by pressing the corresponding radio button to left frame.

2. The selection of a radio button activates the required fields in the GUI, while deactivating the rest fields and changing their background color.



Training Dataset C:\Program Files\MATLAB\R2006b\work\new\working\data\reuters

☑ Use Stored Labels

● Multiple docs (file)

☑ Line Delimiter

☑ Use Stored Global Weights

Local Term Weighting Term Frequency

● k Nearest Neighboors (kNN)

● Multi-Label

● Use Thresholds

● Vector Space Model

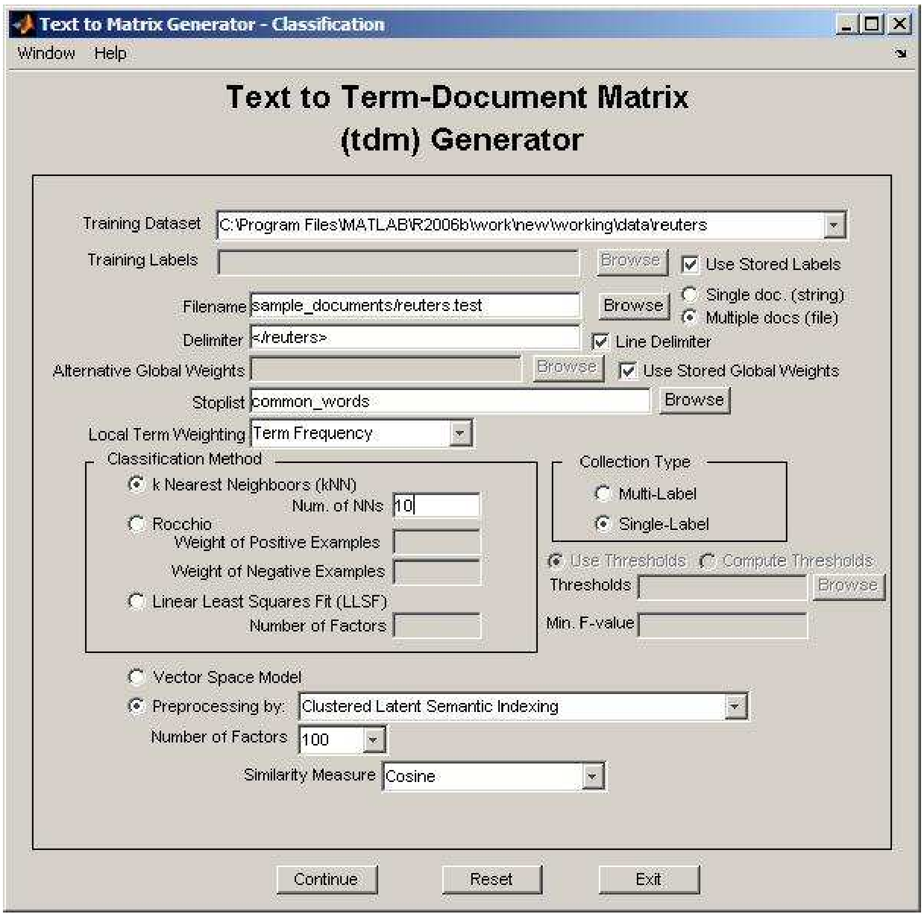
○ Preprocessing by: Clustered Latent Semantic Indexing

Number of Factors 100

Similarity Measure Cosine

Figure 24: Starting window of `classification_gui`.

3. Fill in the required fields, by pressing the check buttons, editing the edit boxes or selecting the appropriate files/variables by pressing a “Browse” button.



Text to Term-Document Matrix (tdm) Generator

Training Dataset C:\Program Files\MATLAB\R2006b\work\new\working\data\reuters

Training Labels ☑ Use Stored Labels

Filename sample_documents/reuters.test ○ Single doc. (string) ● Multiple docs (file)

Delimiter </reuters> ☑ Line Delimiter

Alternative Global Weights ☑ Use Stored Global Weights

Stoplist common_words

Local Term Weighting Term Frequency

Classification Method

● k Nearest Neighboors (kNN)
Num. of NNs 10

○ Rocchio
Weight of Positive Examples
Weight of Negative Examples

○ Linear Least Squares Fit (LLSF)
Number of Factors

Collection Type
○ Multi-Label
● Single-Label

● Use Thresholds ○ Compute Thresholds

Thresholds

Min. F-value

○ Vector Space Model

● Preprocessing by: Clustered Latent Semantic Indexing

Number of Factors 100

Similarity Measure Cosine

Figure 25: Next view of `classification_gui` according to the user selection.

4. Press the “Continue” button in order to perform selected operation.
5. Results have been saved to the workspace.
6. Press the “Reset” button in order to change the input.

B Appendix: Function Reference

about_tmg_gui

ABOUT_TMG_GUI
    ABOUT_TMG_GUI displays information for TMG.

`block_diagonalize`

BLOCK_DIAGONALIZE - reorders a matrix heuristically using a clustering result

[A, N_ROWS, N_COLS, ROW_INDS, COL_INDS]=BLOCK_DIAGONALIZE(A, CLUSTERS) reorders matrix A using the clustering result represented by the structure CLUSTERS. N_ROWS and N_COLS store the last row and column index for each row and column block resprectively, while ROW_INDS and COL_INDS contain the permuted row and column indices.

classification_gui

CLASSIFICATION_GUI

CLASSIFICATION_GUI is a graphical user interface for all classification functions of the Text to Matrix Generator (TMG) Toolbox.

clsi

CLSI - computes a rank-L approximation of the input matrix
using the Clustered Latent Semantic Indexing Method [1]

[X, Y]=CLSI(A, CLUSTERS, L, FUNC, ALPHA_VAL, SVD_METHOD)
computes the rank-L approximation X*Y of the input matrix
A with the Clustered Latent Semantic Indexing Method [1],
using the cluster structure information from CLUSTERS.
FUNC denotes the method used for the selection of the
number of factors from each cluster. Possible values for
FUNC:

- ’f’: Selection using a heuristic method from [1]
(see KS_SELECTION).
- ’f1’: Same as ’f’ but use at least one factor
from each cluster.
- ’equal’: Use the same number of factors from
each cluster.

ALPHA_VAL is a value in [0, 1] used in the number of
factors selection heuristic [1]. Finally, SVD_METHOD
defines the method used for the computation of the SVD
(svds or propack).

REFERENCES:
[1] D. Zeimpekis and E. Gallopoulos. CLSI: A Flexible
Approximation Scheme from Clustered Term-Document Matrices.
In Proc. 5th SIAM International Conference on Data Mining,
pages 631635, Newport Beach, California, 2005.

clustering_gui

CLUSTERING_GUI
CLUSTERING_GUI is a graphical user interface for all
clustering functions of the Text to Matrix Generator
(TMG) Toolbox.

cm

CM - computes a rank-L approximation of the input matrix
using the Centroids Method [1]

[X, Y]=CM(A, CLUSTERS) computes the rank-K approximation
X*Y of the input matrix A with the Centroids Method [1],
using the cluster structure information from CLUSTERS.

REFERENCES:
[1] H. Park, M. Jeon, and J. Rosen. Lower Dimensional Representation
of Text Data Based on Centroids and Least Squares.
BIT Numerical Mathematics, 43(2):427448, 2003.

`col_normalization`

COL_NORMALIZATION - normalizes the columns of the input matrix.

`col_rearrange`

```
COL_REARRANGE - reorders a matrix using a clustering result
    [A, N_COLS, COL_INDS]=COL_REARRANGE(A, CLUSTERS) reorders
    the columns of matrix A using the clustering result represented
    by the structure CLUSTERS. N_COLS stores the last column index
    for each column block, while COL_INDS containes the permuted
    column indices.
```

```
comppute_scat
```

COMPUTE_SCAT - computes the cluster selection criterion value
of PDDP

    SCAT=COMPUTE_SCAT(A, C) returns the square of the frobenius
    norm of A-C*ones(1, size(A, 2)).

create_kmeans_response

```
CREATE_KMEANS_RESPONSE returns an html response for k-means
    CREATE_KMEANS_RESPONSE(CLUSTERS, TITLES) creates a summary
    html file containing information for the result of the
    k-means algorithm, defined by CLUSTERS, when applied to
    the dataset with document titles defined in the TITLES
    cell array.
    CREATE_KMEANS_RESPONSE(CLUSTERS, TITLES, VARIANT) defines
    additionaly the k-means variant (possible values 'k-means'
    and 'skmeans'). The result is stored in the "results"
    directory and displayed using the default web browser.
```

`create_pddp_response`

CREATE_PDDP_RESPONSE returns an html response for PDDP

CREATE_PDDP_RESPONSE(TREE_STRUCT, CLUSTERS, L, TITLES)
creates a summary html file containing information for
the result of the PDDP algorithm, defined by TREE_STRUCT
and CLUSTERS, when applied to the dataset with document
titles defined in the TITLES cell array. L defines the
maximum number of principal directions used by PDDP.
The result is stored in the ”results” directory and
displayed using the default web browser.

`create_retrieval_response`

```
CREATE_RETRIEVAL_RESPONSE returns an html response for a query
    CREATE_RETRIEVAL_RESPONSE(DATASET, IDS, SIMILARITY, QUERY)
    creates an html file containing information for the text of
    documents of DATASET stored in MySQL defined by IDS and
    having SIMILARITY similarity coefficients against QUERY.
    The result is stored in the ”results” directory and displayed
    using the default web browser.
```

`diff_vector`

DIFF_VECTOR

DIFF_VECTOR returns the vector of differences between consecutive elements of the input vector.

dr_gui
DR_GUI DR_GUI is a graphical user interface for all dimensionality reduction functions of the Text to Matrix Generator (TMG) Toolbox.

`ekmeans`

```
EKMEANS - Euclidean k-Means Clustering Algorithm
    EKMEANS clusters a term-document matrix using the standard
    k-means clustering algorithm. CLUSTERS=EKMEANS(A, C, K,
    TERMINATION) returns a cluster structure with K clusters
    for the term-document matrix A using as initial centroids
    the columns of C (initialized randomly when it is empty).
    TERMINATION defines the termination method used in k-means
    (’epsilon’ stops iteration when objective function decrease
    falls down a user defined threshold - see OPTIONS input
    argument - while ’n_iter’ stops iteration when a user
    defined number of iterations has been reached).
    [CLUSTERS, Q]=EKMEANS(A, C, K, TERMINATION) returns also
    the vector of objective function values for each iteration
    and [CLUSTERS, Q, C]=EKMEANS(A, C, K, TERMINATION) returns
    the final centroid vectors.
    EKMEANS(A, C, K, TERMINATION, OPTIONS) defines optional
    parameters:
        - OPTIONS.iter: Number of iterations (default 10).
        - OPTIONS.epsilon: Value for epsilon convergence
          criterion (default 1).
        - OPTIONS.dsp: Displays results (default 1) or
          not (0) to the command window.
```

`entropy`

ENTROPY - computes the entropy of a clustering result
[VENTROPY, CONFUSION_MATRIX, MISTAKES]=ENTROPY(CLUSTERS,
LABELS) computes the entropy value of a clustering result
represented by the CLUSTERS structure. LABELS is a vector
of integers containing the true labeling of the objects.
The entropy value is stored in VENTOPY, while
CONFUSION_MATRIX is a k x r matrix, where k is the number
of clusters and r the number of true classes, and
CONFUSION_MATRIX(i, j) records the number of objects
of class j assigned to cluster i. Finally, MISTAKES contains
the number of misassigned objects, measured by m1+...+mk,
where mi=sum(CONFUSION_MATRIX(i, j)), j∼=i.

`get_node_scat`

GET_NODE_SCAT - returns the PDDP node with the maximum scatter
value (see PDDP)

    [MAX_SCAT_IND, M_SCAT]=GET_NODE_SCAT(TREE_STRUCT, SPLITTED)
    returns the node index and the scatter value of the PDDP
    tree defined by TREE_STRUCT. SPLITTED is a vector that
    determines the active nodes.

gui

GUI

GUI is a simple, top graphical user interface of the Text to Matrix Generator (TMG) Toolbox. Using GUI, the user can select any of the four GUI modules (indexing, dimensionality reduction, clustering, classification) of TMG.

init_tmg

INIT_TMG - Installation script of TMG

INIT_TMG is the installation script of the Text to Matrix Generator (TMG) Toolbox. INIT_TMG creates the MySQL database and adds all TMG directories to the path.

`knn_multi`

KNN_MULTI - k-Nearest Neighboors classifier for multi-label collections

LABELS_AS=KNN_MULTI(A, Q, K, LABELS, NORMALIZED_DOCS, THRESHOLDS) classifies the columns of Q with the K-Nearest Neighboors classifier using the pre-classified columns of matrix A with labels LABELS (cell array of vectors of integers). THRESHOLDS is a vector of class threshold values. NORMALIZED_DOCS defines if cosine (1) or euclidean distance (0) similarity measure is to be used. LABELS_AS contains the assigned labels for the columns of Q.

knn_single

KNN_SINGLE - k-Nearest Neighboors classifier for single-label collections

LABELS_AS=KNN_SINGLE(A, Q, K, LABELS, NORMALIZED_DOCS) classifies the columns of Q with the K-Nearest Neighboors classifier using the pre-classified columns of matrix A with labels LABELS (vector of integers). NORMALIZED_DOCS defines if cosine (1) or euclidean distance (0) similarity measure is to be used. LABELS_AS contains the assigned labels for the columns of Q.

ks_selection

KS_SELECTION - implements the heuristic method from [2] for
the selection of the number of factors from each cluster
used in the Clustered Latent Semantic Indexing method [1].

    N_ST=KS_SELECTION(A, N_COLS, ALPHA_VAL, L) returns in N_ST
    a vector of integers denoting the number of factors (sum
    equals L) selected from each cluster of the tdm A. N_COLS
    is a vector containing the last column index for each column
    block, while ALPHA_VAL is a value in [0, 1].

ks_selection1

KS_SELECTION1 - implements the heuristic method from [2] for
the selection of the number of factors from each cluster used
in the Clustered Latent Semantic Indexing method [1]. The
number of factors from each cluster is at least 1.

N_ST=KS_SELECTION1(A, N_COLS, ALPHA_VAL, L) returns in N_ST
a vector of integers denoting the number of factors
(sum equals L) selected from each cluster of the tdm A.
N_COLS is a vector containing the last column index for
each column block, while ALPHA_VAL is a value in [0, 1].

llsf_multi

LLSF_MULTI - Linear Least Squares Fit for multi-label collections [2]

LABELS_AS=LLSF_MULTI(A, Q, CLUSTERS, LABELS, L, METHOD, THRESHOLDS, SVD_METHOD, CLSI_METHOD) classifies the columns of Q with the Linear Least Squares Fit classifier [2] using the pre-classified columns of matrix A with labels LABELS (cell array of vectors of integers). THRESHOLDS is a vector of class threshold values, while CLUSTERS is a structure defining the classes. METHOD is the method used for the approximation of the rank-l truncated SVD, with possible values:

- 'clsi': Clustered Latent Semantic Indexing [3].
- 'cm': Centroids Method [1].
- 'svd': Singular Value Decomosition.

SVD_METHOD defines the method used for the computation of the SVD, while CLSI_METHOD defines the method used for the determination of the number of factors from each class used in Clustered Latent Semantic Indexing in case METHOD equals 'clsi'.

llsf_single

```
LLSF_SINGLE - Linear Least Squares Fit for single-label
collections [2]
    LABELS_AS=LLSF_SINGLE(A, Q, CLUSTERS, LABELS, L, METHOD,
    SVD_METHOD, CLSI_METHOD) classifies the columns of Q
    with the Linear Least Squares Fit classifier [2] using
    the pre-classified columns of matrix A with labels LABELS
    (cell array of vectors of integers). CLUSTERS is a
    structure defining the classes. METHOD is the method
    used for the approximation of the rank-l truncated SVD,
    with possible values:
        - 'clsi': Clustered Latent Semantic Indexing [3].
        - 'cm': Centroids Method [1].
        - 'svd': Singular Value Decomosition.
    SVD_METHOD defines the method used for the computation
    of the SVD, while CLSI_METHOD defines the method used for
    the determination of the number of factors from each class
    used in Clustered Latent Semantic Indexing in case METHOD
    equals 'clsi'.
```

`lsa`

LSA - Applies the Latent Semantic Analysis Model to a document collection

[SC, DOCS_INDS] = LSA(D, P, Q, NORMALIZE_DOCS) applies LSA to the text collection represented by the latent semantic factors D, P of the collection's term - document matrix, for the query defined by the vector Q [1]. NORMALIZE_DOCS defines if the document vectors are to be normalized (1) or not (0). SC contains the sorted similarity coefficients, while DOC_INDS contains the corresponding document indices.

make_clusters_multi

MAKE_CLUSTERS_MULTI - auxiliary function for the classification algorithms

> CLUSTERS=MAKE_CLUSTERS_MULTI(LABELS) forms the cluster structure of a multi-label collection with document classes defined by LABELS (cell array of vectors of integers).

make_clusters_single

MAKE_CLUSTERS_SINGLE - auxiliary function for the classification algorithms

CLUSTERS=MAKE_CLUSTERS_SINGLE(LABELS) forms the cluster structure of a single-label collection with document classes defined by LABELS (vector of integers).

make_labels

MAKE_LABELS - creates a label vector of integers for the input
cell array of string

[LABELS, UNIQUE_LABELS]=MAKE_LABELS(INPUT_LABELS) creates a
vector of integer labels (LABELS) for the input cell array
of strings INPUT_LABELS. UNIQUE_LABELS contains the strings
of unique labels of the input cell array.

make_val_inds

MAKE_VAL_INDS - auxiliary function for the classification algorithms

INDS=MAKE_VAL_INDS(LABELS) constructs an index vector used during the thresholding phase of any classifier for the multi-label collection with document classes defined by LABELS (cell array of vectors of integers).

`merge_dictionary`

MERGE_DICTIONARY - merges two cell arrays of chars and returns only the distinct elements of their union (used by tmg.m, tmg_query.m, tdm_update.m)

[ALL_WORDS, ALL_DOC_IDS]=MERGE_DICTIONARY(ALL_WORDS, NEW_WORDS, ALL_DOC_IDS, NEW_DOC_IDS) returns in ALL_WORDS all distinct elements of the union of the cell arrays of chars ALL_WORDS, NEW_WORDS corresponding to two document collections. ALL_DOC_IDS and NEW_DOC_IDS contain the inverted indices of the two collections. Output argument ALL_DOC_IDS contains the inverted index of the whole collection.

merge_tdms

```
MERGE_TDMS - Merges two document collections
    [A, DICTIONARY]=MERGE_TDMS(A1, DICTIONARY1, A2, DICTIONARY2]
    merges the tdms A1 and A2 with corresponding dictionaries
    DICTIONARY1 and DICTIONARY2.
    MERGE_TDS(A1, DICTIONARY1, A2, DICTIONARY2, OPTIONS) defines
    optional parameters:
        - OPTIONS.min_local_freq: The minimum local frequency for
          a term (default 1)
        - OPTIONS.max_local_freq: The maximum local frequency for
          a term (default inf)
        - OPTIONS.min_global_freq: The minimum global frequency
          for a term (default 1)
        - OPTIONS.max_global_freq: The maximum global frequency
          for a term (default inf)
        - OPTIONS.local_weight: The local term weighting function
          (default 't'). Possible values (see [1, 2]):
              't': Term Frequency
              'b': Binary
              'l': Logarithmic
              'a': Alternate Log
              'n': Augmented Normalized Term Frequency
        - OPTIONS.global_weight: The global term weighting function
          (default 'x'). Possible values (see [1, 2]):
              'x': None
              'e': Entropy
              'f': Inverse Document Frequency (IDF)
              'g': GfIdf
              'n': Normal
              'p': Probabilistic Inverse
        - OPTIONS.normalization: Indicates if we normalize the
          document vectors (default 'x'). Possible values:
              'x': None
              'c': Cosine
```

`myperms`

MYPERMS - computes all possible combinations of the input
V=MYPERMS[P, L] returns all possible combinations of the
input vector of integers with L numbers.

open_file

```
OPEN_FILE
      OPEN_FILE is a graphical user interface for selecting a file,
      directory or variable from the workspace. The function returns
      the name of the selected file, directory or variable.
```

`opt_2means`

OPT_2MEANS - a special case of k-means for k=2

OPT_2MEANS(A, X) returns the clustering that optimizes the objective function of the k-means algorithm based on the ordering of vector X.

[CLUSTERS, S]=OPT_2MEANS(A, X) returns the cluster structure as well as the value of the objective function.

pca

PCA - Principal Component Analysis

[U, S, V]=PCA(A, C, K, METHOD) computes the K-factor
Principal Component Analysis of A, i.e. SVD of
A-C*ones(size(A, 2), 1), using either the svds function
of MATLAB or the PROPACK package [1].

REFERENCES:
[1] R.M.Larsen, PROPACK: A Software Package for the Symmetric Eigenvalue
Problem and Singular Value Problems on Lanczos and Lanczos Bidiagonalization
with Partial Reorthogonalization, Stanford University,
http://sun.stanford.edu/∼rmunk/PROPACK.

pca_mat

PCA_MAT - Principal Component Analysis with MATLAB (svds)

[U, S, V]=PCA_MAT(A, C, K) computes the K-factor Principal Component Analysis of A, i.e. SVD of A-C*ones(size(A, 2), 1), using the svds function of MATLAB.

`pca_mat_afun`

PCA_MAT_AFUN - Auxiliary function used in PCA_MAT.

pca_propack

PCA_PROPACK - Principal Component Analysis with PROPACK
[U, S, V]=PCA_PROPACK(A, C, K) computes the K-factor
Principal Component Analysis of A, i.e. SVD of
A-C*ones(size(A, 2), 1), using the PROPACK package [1].

REFERENCES:
[1] R.M.Larsen, PROPACK: A Software Package for the Symmetric Eigenvalue
Problem and Singular Value Problems on Lanczos and Lanczos Bidiagonalization
with Partial Reorthogonalization, Stanford University,
http://sun.stanford.edu/ rmunk/PROPACK.

`pca_propack_Atransfunc`

PCA_PROPACK_ATRANSFUNC - Auxiliary function used in PCA_PROPACK.

pca_propack_afun

PCA_PROPACK_AFUN - Auxiliary function used in TMG_PCA_PROPACK.

pddp

PDDP - Principal Direction Divisive Partitioning Clustering Algorithm

PDDP clusters a term-document matrix (tdm) using the Principal Direction Divisive Partitioning clustering algorithm [1, 2].

CLUSTERS=PDDP(A, K, L) returns a cluster structure with K clusters for the tdm A formed using information from the first L principal components of the tdm.

[CLUSTERS, TREE_STRUCT]=PDDP(A, K, L) returns also the full PDDP tree, while [CLUSTERS, TREE_STRUCT, S]=PDDP(A, K, L) returns the objective function of PDDP.

PDDP(A, K, L, SVD_METHOD) defines the method used for the computation of the PCA (svds - default - or propack), while PDDP(A, K, L, SVD_METHOD, DSP) defines if results are to be displayed to the command window (default 1) or not (0).

REFERENCES:

[1] D.Boley, Principal Direction Divisive Partitioning, Data Mining and Knowledge Discovery 2 (1998), no. 4, 325-344.

[2] D.Zeimpekis, E.Gallopoulos, PDDP(l): Towards a Flexible Principal Direction Divisive Partitioning Clustering Algorithmm, Proc. IEEE ICDM'03 Workshop on Clustering Large Data Sets (Melbourne, Florida), 2003.

pddp_2means

PDDP_2MEANS - Hybrid Principal Direction Divisive Partitioning Clustering Algorithm and k-means

PDDP_2MEANS clusters a term-document matrix (tdm) using a combination of the Principal Direction Divisive Partitioning clustering algorithm [1] and k-means [2].

CLUSTERS=PDDP_2MEANS(A, K) returns a cluster structure with K clusters for the tdm A.

[CLUSTERS, TREE_STRUCT]=PDDP_2MEANS(A, K) returns also the full PDDP tree, while [CLUSTERS, TREE_STRUCT, S]=PDDP_2MEANS(A, K) returns the objective function of PDDP.

PDDP_2MEANS(A, K, SVD_METHOD) defines the method used for the computation of the PCA (svds - default - or propack).

PDDP_2MEANS(A, K, SVD_METHOD, DSP) defines if results are to be displayed to the command window (default 1) or not (0).

Finally, PDDP_2MEANS(A, K, SVD_METHOD, DSP, EPSILON)defines the termination criterion value for the k-means algorithm.

REFERENCES:

[1] D.Boley, Principal Direction Divisive Partitioning, Data Mining and Knowledge Discovery 2 (1998), no. 4, 325-344.

[2] D.Zeimpekis, E.Gallopoulos, k-means Steering of Spectral Divisive Clustering Algorithms, Proc. of Text Mining Workshop, Minneapolis, 2007.

`pddp_optcut`

PDDP_OPTCUT - Hybrid Principal Direction Divisive Partitioning Clustering Algorithm and k-means

PDDP_OPTCUT clusters a term-document matrix (tdm) using a combination of the Principal Direction Divisive Partitioning clustering algorithm [1] and k-means [2].

CLUSTERS=PDDP_OPTCUT(A, K) returns a cluster structure with K clusters for the tdm A.

[CLUSTERS, TREE_STRUCT]=PDDP_OPTCUT(A, K) returns also the full PDDP tree, while [CLUSTERS, TREE_STRUCT, S]=PDDP_OPTCUT(A, K) returns the objective function of PDDP.

PDDP_OPTCUT(A, K, SVD_METHOD) defines the method used for the computation of the PCA (svds - default - or propack).

PDDP_OPTCUT(A, K, SVD_METHOD, DSP) defines if results are to be displayed to the command window (default 1) or not (0). Finally, PDDP_OPTCUT(A, K, SVD_METHOD, DSP, EPSILON) defines the termination criterion value for the k-means algorithm.

REFERENCES:

[1] D.Boley, Principal Direction Divisive Partitioning, Data Mining and Knowledge Discovery 2 (1998), no. 4, 325-344.

[2] D.Zeimpekis, E.Gallopoulos, k-means Steering of Spectral Divisive Clustering Algorithms, Proc. of Text Mining Workshop, Minneapolis, 2007.

pddp_optcut_2means

PDDP_OPTCUT_2MEANS - Hybrid Principal Direction Divisive
Partitioning Clustering Algorithm and k-means

PDDP_OPTCUT_2MEANS clusters a term-document matrix (tdm)
using a combination of the Principal Direction Divisive
Partitioning clustering algorithm [1] and k-means [2].
CLUSTERS=PDDP_OPTCUT_OPTCUT_2MEANS(A, K) returns a cluster
structure with K clusters for the tdm A.
[CLUSTERS, TREE_STRUCT]=PDDP_OPTCUT_2MEANS(A, K) returns also
the full PDDP tree, while [CLUSTERS, TREE_STRUCT, S]=
PDDP_OPTCUT_2MEANS(A, K) returns the objective function of
PDDP.
PDDP_OPTCUT_2MEANS(A, K, SVD_METHOD) defines the method used
for the computation of the PCA (svds - default - or propack).
PDDP_OPTCUT_2MEANS(A, K, SVD_METHOD, DSP) defines if results
are to be displayed to the command window (default 1) or not
(0). Finally, PDDP_OPTCUT_2MEANS(A, K, SVD_METHOD, DSP, EPSILON)
defines the termination criterion value for the k-means
algorithm.

REFERENCES:
[1] D.Boley, Principal Direction Divisive Partitioning, Data
Mining and Knowledge Discovery 2 (1998), no. 4, 325-344.
[2] D.Zeimpekis, E.Gallopoulos, k-means Steering of Spectral
Divisive Clustering Algorithms, Proc. of Text Mining Workshop,
Minneapolis, 2007.

`pddp_optcutpd`

PDDP_OPTCUTPD - Hybrid Principal Direction Divisive
Partitioning Clustering Algorithm and k-means

PDDP_OPTCUTPD clusters a term-document matrix (tdm) using
a combination of the Principal Direction Divisive
Partitioning clustering algorithm [1, 2] and k-means [3].
CLUSTERS=PDDP_OPTCUT_OPTCUTPD(A, K, L) returns a cluster
structure with K clusters for the tdm A formed using
information from the first L principal components of the
tdm.
[CLUSTERS, TREE_STRUCT]=PDDP_OPTCUTPD(A, K, L) returns
also the full PDDP tree, while [CLUSTERS, TREE_STRUCT, S]=
PDDP_OPTCUTPD(A, K, L) returns the objective function of
PDDP. PDDP_OPTCUTPD(A, K, L, SVD_METHOD) defines the method used
for the computation of the PCA (svds - default - or
propack). Finally, PDDP_OPTCUTPD(A, K, L, SVD_METHOD, DSP)
defines if results are to be displayed to the command window
(default 1) or not (0).

REFERENCES:
[1] D.Boley, Principal Direction Divisive Partitioning, Data
Mining and Knowledge Discovery 2 (1998), no. 4, 325-344.
[2] D.Zeimpekis, E.Gallopoulos, PDDP(l): Towards a Flexible
Principal Direction Divisive Partitioning Clustering
Algorithmm, Proc. IEEE ICDM’03 Workshop on Clustering Large
Data Sets (Melbourne, Florida), 2003.
[3] D.Zeimpekis, E.Gallopoulos, k-means Steering of Spectral
Divisive Clustering Algorithms, Proc. of Text Mining Workshop,
Minneapolis, 2007.

ps_pdf2ascii

PS_PDF2ASCII - converts the input ps or pdf file to ASCII

RESULT = PS_PDF2ASCII(FILENAME) converts the input ps or
pdf files to ASCII, using ghostscript's utility 'ps2ascii'.
RESULT returns a success indicator, e.g. -2 if the input
file does not exist or has a wrong format, -1 if gs is not
installed or the path isn't set, 0 if 'ps2ascii' didn't
work properly, and 1 if the conversion was successful.

`retrieval_gui`

RETRIEVAL_GUI

RETRIEVAL_GUI is a graphical user interface for all retrieval
functions of the Text to Matrix Generator (TMG) Toolbox.

`rocchio_multi`

```
ROCCHIO_MULTI - Rocchio classifier for multi-label collections
    LABELS_AS=KNN_MULTI(A, CLUSTERS, BETA, GAMMA, Q, LABELS,
    NORMALIZED_DOCS, THRESHOLDS) classifies the columns of Q
    with the Rocchio classifier using the pre-classified columns
    of matrix A with labels LABELS (vector of integers).
    THRESHOLDS is a vector of class threshold values. BETA and
    GAMMA define the weight of positive and negative examples in
    the formation of each class centroid. NORMALIZED_DOCS defines
    if cosine (1) or euclidean distance (0) similarity measure is
    to be used. LABELS_AS contains the assigned labels for the
    columns of Q.
```

rocchio_single
ROCCHIO_SINGLE - Rocchio classifier for single-label collections LABELS_AS=KNN_SINGLE(A, CLUSTERS, BETA, GAMMA, Q, LABELS, NORMALIZED_DOCS) classifies the columns of Q with the Rocchio classifier using the pre-classified columns of matrix A with labels LABELS (vector of integers). BETA and GAMMA define the weight of positive and negative examples in the formation of each class centroid. NORMALIZED_DOCS defines if cosine (1) or euclidean distance (0) similarity measure is to be used. LABELS_AS contains the assigned labels for the columns of Q.

scut_knn

SCUT_KNN - implements the Scut thresholding technique from [1]
for the k-Nearest Neighboors classifier

THRESHOLD=SCUT_KNN(A, Q, K, LABELS_TR, LABELS_TE, MINF1, NORMALIZE, STEPS) returns the vector of thresholds for the k-Nearest Neighboors classifier for the collection [A Q]. A and Q define the training and test parts of the validation set with labels LABELS_TR and LABELS_TE respectively. MINF1 defines the minimum F1 value and NORMALIZE defines if cosine (1) or euclidean distance (0) measure of similarity is to be used. Finally, STEPS defines the number of steps used during thresholding.

[THRESHOLD, F, THRESHOLDS]=SCUT_KNN(A, Q, K, LABELS_TR, LABELS_TE, MINF1, NORMALIZE, STEPS) returns also the best F1 value as well as the matrix of thresholds for each step (row i corresponds to step i).

REFERENCES:

[1] Y. Yang. A Study of Thresholding Strategies for Text Categorization. In Proc. 24th ACM SIGIR, pages 137145, New York, NY, USA, 2001. ACM Press.

scut_llsf

```
SCUT_LLSF - implements the Scut thresholding technique from [2]
for the Linear Least Squares Fit classifier [3]
    THRESHOLD=SCUT_LLSF(A, Q, CLUSTERS, K, LABELS_TR, LABELS_TE,
    MINF1, L, METHOD, STEPS, SVD_METHOD, CLSI_METHOD) returns
    the vector of thresholds for the Linear Least Squares Fit
    classifier for the collection [A Q]. A and Q define the
    training and test parts of the validation set with labels
    LABELS_TR and LABELS_TE respectively. CLUSTERS is a
    structure defining the classes, while MINF1 defines the
    minimum F1 value and STEPS defines the number of steps
    used during thresholding.
    METHOD is the method used for the approximation of the
    rank-l truncated SVD, with possible values:
        - 'clsi': Clustered Latent Semantic Indexing [4].
        - 'cm': Centroids Method [1].
        - 'svd': Singular Value Decomosition.
    SVD_METHOD defines the method used for the computation of
    the SVD, while CLSI_METHOD defines the method used for the
    determination of the number of factors from each class used
    in Clustered Latent Semantic Indexing in case METHOD equals
    'clsi'.
    [THRESHOLD, F, THRESHOLDS]=SCUT_LLSF(A, Q, CLUSTERS, K,
    LABELS_TR, LABELS_TE, MINF1, L, METHOD, STEPS, SVD_METHOD,
    CLSI_METHOD) returns also the best F1 value as well as the
    matrix of thresholds for each step (row i corresponds to
    step i).

    REFERENCES:
    [1] H. Park, M. Jeon, and J. Rosen. Lower Dimensional
    Representation of Text Data Based on Centroids and Least
    Squares. BIT Numerical Mathematics, 43(2):427448, 2003.
    [2] Y. Yang. A Study of Thresholding Strategies for Text
    Categorization. In Proc. 24th ACM SIGIR, pages 137145,
    New York, NY, USA, 2001. ACM Press.
    [3] Y. Yang and C. Chute. A Linear Least Squares Fit
    Mapping Method for Information Retrieval from Natural
    Language Texts. In Proc. 14th Conference on Computational
    Linguistics, pages 447453, Morristown, NJ, USA, 1992.
    [4] D. Zeimpekis and E. Gallopoulos, "Non-Linear Dimensional
    Reduction via Class Representatives for Text Classification".
    In Proc. 2006 IEEE International Conference on Data Mining
    (ICDM'06), Hong Kong, Dec. 2006.
```

scut_rocchio

SCUT_ROCCHIO - implements the Scut thresholding technique
from [1] for the Rocchio classifier

THRESHOLD=SCUT_ROCCHIO(A, CLUSTERS, BETA, GAMMA, Q,
LABELS_TR, LABELS_TE, MINF1, NORMALIZE, STEPS) returns
the vector of thresholds for the Rocchio classifier
for the collection [A Q]. A and Q define the training
and test parts of the validation set with labels
LABELS_TR and LABELS_TE respectively. MINF1 defines
the minimum F1 value, while NORMALIZE defines if cosine
(1) or euclidean distance (0) measure of similarity is
to be used, CLUSTERS is a structure defining the classes
and STEPS defines the number of steps used during
thresholding. BETA and GAMMA define the weight of positive
and negative examples in the formation of each class
centroid.
[THRESHOLD, F, THRESHOLDS]=SCUT_ROCCHIO(A, CLUSTERS, BETA,
GAMMA, Q, LABELS_TR, LABELS_TE, MINF1, NORMALIZE, STEPS)
returns also the best F1 value as well as the matrix of
thresholds for each step (row i corresponds to step i).

REFERENCES:
[1] Y. Yang. A Study of Thresholding Strategies for Text
Categorization. In Proc. 24th ACM SIGIR, pages 137145,
New York, NY, USA, 2001. ACM Press.

sdd_tmg

SDD_TMG - interface for SDDPACK

[X, D, Y]=SDD_TMG(A, K) computes a rank-K Semidiscrete
Decomposition of A using the SDDPACK [1].

REFERENCES:
Tamara G. Kolda and Dianne P. O’Leary, Computation and Uses of the
Semidiscrete Matrix Decomposition, Computer Science Department Report
CS-TR-4012 Institute for Advanced Computer Studies Report UMIACS-TR-99-22,
University of Maryland, April 1999.

skmeans

SKMEANS - Spherical k-Means Clustering Algorithm

SKMEANS clusters a term-document matrix using the Spherical k-means clustering algorithm [1]. CLUSTERS=SKMEANS(A, C, K, TERMINATION) returns a cluster structure with K clusters for the term-document matrix A using as initial centroids the columns of C (initialized randomly when it is empty). TERMINATION defines the termination method used in spherical k-means ('epsilon' stops iteration when objective function increase falls down a user defined threshold - see OPTIONS input argument - while 'n_iter' stops iteration when a user defined number of iterations has been reached). [CLUSTERS, Q]=SKMEANS(A, C, K, TERMINATION) returns also the vector of objective function values for each iteration and [CLUSTERS, Q, C]=SKMEANS(A, C, K, TERMINATION) returns the final centroid vectors. SKMEANS(A, C, K, TERMINATION, OPTIONS) defines optional parameters:

- OPTIONS.iter: Number of iterations (default 10).
- OPTIONS.epsilon: Value for epsilon convergence criterion (default 1).
- OPTIONS.dsp: Displays results (default 1) or not (0) to the command window.

REFERENCES:

[1] I. S. Dhillon and D. M. Modha, "Concept Decompositions for Large Sparse Text Data using Clustering", Machine Learning, 42:1, pages 143-175, Jan, 2001.

`stemmer`

```
STEMMER - applies the Porter’s Stemming algorithm [1]
    S = STEMMER(TOKEN, DSP) returns in S the stemmed word of
    TOKEN. DSP indicates if the function displays the result
    of each stem (1).

    REFERENCES:
    [1] M.F.Porter, An algorithm for suffix stripping, Program, 14(3): 130-137,
    1980.
```

`strip_html`

STRIP_HTML - removes html entities from an html file
S = STRIP_HTML(FILENAME) parses file FILENAME and removes
the html entities, while the result is stored in S as a
cell array and written in file ”FILENAME.TXT”.

svd_tmg

SVD_TMG - Singular Value Decomposition

[U, S, V]=SVD_TMG(A, K, METHOD) computes the K-factor
truncated Singular Value Decomposition of A using either
the svds function of MATLAB or the PROPACK package [1].

REFERENCES:
[1] R.M.Larsen, PROPACK: A Software Package for the Symmetric Eigenvalue
Problem and Singular Value Problems on Lanczos and Lanczos Bidiagonalization
with Partial Reorthogonalization, Stanford University,
http://sun.stanford.edu/ rmunk/PROPACK.

tdm_downdate

TDM_DOWNDATE - renews a text collection by downdating the correspoding term-document matrix

A = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns the new term - document matrix of the downdated collection. UPDATE_STRUCT defines the update structure returned by TMG, while REMOVED_DOCS defines the indices of the documents that is to be be removed.

[A, DICTIONARY] = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns also the dictionary for the updated collection, while [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZED_FACTORS] = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns the vectors of global weights for the dictionary and the normalization factor for each document in case such a factor is used. If normalization is not used TDM_DOWNDATE returns a vector of all ones. [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC] = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns statistics for each document, i.e. the number of terms for each document.

[A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC, TITLES, FILES] = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns in FILES the filenames containing the collection's documents and a cell array (TITLES) that contains a declaratory title for each document, as well as the document's first line.

Finally [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC, TITLES, FILES, UPDATE_STRUCT] = TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS) returns the update structure that keeps the essential information for the collection' s update (or downdate).

TDM_DOWNDATE(UPDATE_STRUCT, REMOVED_DOCS, OPTIONS) defines optional parameters:

- OPTIONS.dsp: Displays results (default 1) or not (0) to the command window.

`tdm_update`

TDM_UPDATE renews a text collection by updating the correspoding term-document matrix.

A = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns the new term - document matrix of the updated collection. FILENAME defines the file (or files in case a directory is supplied) containing the new documents, while UPDATE_STRUCT defines the update structure returned by TMG. In case FILENAME variable is empty, the collection is simply updated using the options defined by UPDATE_STRUCT (for example, use another term-weighting scheme).

[A, DICTIONARY] = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns also the dictionary for the updated collection, while [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZED_FACTORS] = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns the vectors of global weights for the dictionary and the normalization factor for each document in case such a factor is used. If normalization is not used TDM_UPDATE returns a vector of all ones.

[A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC] = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns statistics for each document, i.e. the number of terms for each document.

[A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC, TITLES, FILES] = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns in FILES the filenames contained in directory (or file) FILENAME and a cell array (TITLES) that containes a declaratory title for each document, as well as the document's first line.

Finally [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC, TITLES, FILES, UPDATE_STRUCT] = TDM_UPDATE(FILENAME, UPDATE_STRUCT) returns the update structure that keeps the essential information for the collection' s update (or downdate).

TDM_UPDATE(FILENAME, UPDATE_STRUCT, OPTIONS) defines optional parameters:

- OPTIONS.delimiter: The delimiter between documents within the same file. Possible values are 'emptyline' (default), 'none_delimiter' (treats each file as a single document) or any other string.
- OPTIONS.line_delimiter: Defines if the delimiter takes a whole line of text (default, 1) or not.

- OPTIONS.update_step: The step used for the incremental
  built of the inverted index (default 10,000).
- OPTIONS.dsp: Displays results (default 1) or not (0) to
  the command window.

tmg

```
TMG - Text to Matrix Generator
    TMG parses a text collection and generates the term -
    document matrix.
    A = TMG(FILENAME) returns the term - document matrix,
    that corresponds to the text collection contained in
    files of directory (or file) FILENAME.
    Each document must be separeted by a blank line (or
    another delimiter that is defined by OPTIONS argument)
    in each file.
    [A, DICTIONARY] = TMG(FILENAME) returns also the
    dictionary for the collection, while [A, DICTIONARY,
    GLOBAL_WEIGHTS, NORMALIZED_FACTORS] = TMG(FILENAME)
    returns the vectors of global weights for the dictionary
    and the normalization factor for each document in case
    such a factor is used. If normalization is not used TMG
    returns a vector of all ones.
    [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS,
    WORDS_PER_DOC] = TMG(FILENAME) returns statistics for
    each document, i.e. the number of terms for each document.
    [A, DICTIONARY, GLOBAL_WEIGHTS, NORMALIZATION_FACTORS,
    WORDS_PER_DOC, TITLES, FILES] = TMG(FILENAME) returns in
    FILES the filenames contained in directory (or file)
    FILENAME and a cell array (TITLES) that containes a
    declaratory title for each document, as well as the
    document’s first line. Finally [A, DICTIONARY,
    GLOBAL_WEIGHTS, NORMALIZATION_FACTORS, WORDS_PER_DOC,
    TITLES, FILES, UPDATE_STRUCT] = TMG(FILENAME) returns a
    structure that keeps the essential information for the
    collection’ s update (or downdate).

    TMG(FILENAME, OPTIONS) defines optional parameters:
        - OPTIONS.use_mysql: Indicates if results are to be
          stored in MySQL.
        - OPTIONS.db_name: The name of the directory where
          the results are to be saved.
        - OPTIONS.delimiter: The delimiter between documents
          within the same file. Possible values are ’emptyline’
          (default), ’none_delimiter’ (treats each file as a
          single document) or any other string.
        - OPTIONS.line_delimiter: Defines if the delimiter
          takes a whole line of text (default, 1) or not.
        - OPTIONS.stoplist: The filename for the stoplist,
          i.e. a list of common words that we don't use for
          the indexing (default no stoplist used).
        - OPTIONS.stemming: Indicates if the stemming algorithm
          is used (1) or not (0 - default).
        - OPTIONS.update_step: The step used for the incremental
          built of the inverted index (default 10,000).
        - OPTIONS.min_length: The minimum length for a term
          (default 3).
        - OPTIONS.max_length: The maximum length for a term
          (default 30).
        - OPTIONS.min_local_freq: The minimum local frequency for
          a term (default 1).
        - OPTIONS.max_local_freq: The maximum local frequency for
          a term (default inf).
        - OPTIONS.min_global_freq: The minimum global frequency
          for a term (default 1).
        - OPTIONS.max_global_freq: The maximum global frequency
          for a term (default inf).
        - OPTIONS.local_weight: The local term weighting function
          (default 't'). Possible values (see [1, 2]):
             't': Term Frequency
             'b': Binary
             'l': Logarithmic
             'a': Alternate Log
             'n': Augmented Normalized Term Frequency
        - OPTIONS.global_weight: The global term weighting function
          (default 'x'). Possible values (see [1, 2]):
             'x': None
             'e': Entropy
             'f': Inverse Document Frequency (IDF)
             'g': GfIdf
             'n': Normal
             'p': Probabilistic Inverse
        - OPTIONS.normalization: Indicates if we normalize the
          document vectors (default 'x'). Possible values:
             'x': None
             'c': Cosine
        - OPTIONS.dsp: Displays results (default 1) or not (0) to
          the command window.

   REFERENCES:
   [1] M.Berry and M.Browne, Understanding Search Engines, Mathematical
   Modeling and Text Retrieval, Philadelphia, PA: Society for Industrial
   and Applied Mathematics, 1999.
   [2] T.Kolda, Limited-Memory Matrix Methods with Applications,
   Tech.Report CS-TR-3806, 1997.
```

tmg_gui

TMG_GUI

TMG_GUI is a graphical user interface for all indexing
routines of the Text to Matrix Generator (TMG) Toolbox.
For a full documentation type 'help tmg', 'help tmg_query',
'help tdm_update' or 'help tdm_downdate'.
For a full documentation of the GUI's usage, select the
help tab to the GUI.

tmg_query

TMG_QUERY - Text to Matrix Generator, query vector constructor
TMG_QUERY parses a query text collection and generates the
query vectors corresponding to the supplied dictionary.
Q = TMG_QUERY(FILENAME, DICTIONARY) returns the query
vectors, that corresponds to the text collection contained
in files of directory FILENAME. DICTIONARY is the array of
terms corresponding to a text collection.
Each query must be separeted by a blank line (or another
delimiter that is defined by OPTIONS argument) in each file.
[Q, WORDS_PER_QUERY] = TMG_QUERY(FILENAME, DICTIONARY)
returns statistics for each query, i.e. the number of terms
for each query.
Finally, [Q, WORDS_PER_QUERY, TITLES, FILES] =
TMG_QUERY(FILENAME) returns in FILES the filenames contained
in directory (or file) FILENAME and a cell array (TITLES)
that containes a declaratory title for each query, as well
as the query's first line.

TMG_QUERY(FILENAME, DICTIONARY, OPTIONS) defines optional
parameters:

- OPTIONS.delimiter: The delimiter between queries within
  the same file. Possible values are 'emptyline' (default),
  'none_delimiter' (treats each file as a single query)
  or any other string.
- OPTIONS.line_delimiter: Defines if the delimiter takes a
  whole line of text (default, 1) or not.
- OPTIONS.stoplist: The filename for the stoplist, i.e. a
  list of common words that we don't use for the indexing
  (default no stoplist used).
- OPTIONS.stemming: Indicates if the stemming algorithm is
  used (1) or not (0 - default).
- OPTIONS.update_step: The step used for the incremental
  built of the inverted index (default 10,000).
- OPTIONS.local_weight: The local term weighting function
  (default 't'). Possible values (see [1, 2]):
  't': Term Frequency
  'b': Binary
  'l': Logarithmic
  'a': Alternate Log
  'n': Augmented Normalized Term Frequenct
- OPTIONS.global_weights: The vector of term global
  weights (returned by tmg).
- OPTIONS.dsp: Displays results (default 1) or not (0).

REFERENCES:

[1] M.Berry and M.Browne, Understanding Search Engines, Mathematical Modeling and Text Retrieval, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[2] T.Kolda, Limited-Memory Matrix Methods with Applications, Tech.Report CS-TR-3806, 1997.

tmg_save_results

TMG_SAVE_RESULTS

TMG_SAVE_RESULTS is a graphical user interface used from TMG_GUI, for saving the results to a (or multiple) .mat file(s).

tmg_template

TDM_TEMPLATE - demo script

This is a template script demonstrating the use of TMG,
as well as the application of the resulting TDM'S in
two IR tasks, quering and clustering. The quering models
used is the Vector Space Model (see vsm.m) and LSI
(see lsi.m), while two versions of the k-means algorithm
(euclidean and spherical, see ekmeans.m and skmeans.m)
cluster the resulting matrix (see pddp.m). The user can
edit this code in order to change the default OPTIONS of
TMG, as well as to apply other IR tasks or use his own
implementations regarding these tasks.

`two_means_1d`

TWO_MEANS_1D - returns the clustering that optimizes the
objective function of the k-means algorithm for the input
vector.

    [CUTOFF, CLUSTERS, DISTANCE, OF, MEAN1, MEAN2]=
    TWO_MEANS_1D(A) returns the cutoff value of the clustering,
    the cluster structure, the separation distance, the value
    of the objective function and the two mean values.

unique_elements

```
UNIQUE_ELEMENTS - detects all distinct elements of a vector
    [ELEMENTS, N] = UNIQUE_ELEMENTS(X) returns in ELEMENTS all
    distinct elements of vector X, and in N the number of times
    each element appears in X. A value is repeated if it appears
    in non-consecutive elements. For no repetitive elements sort
    the input vector.
```

unique_words

UNIQUE_WORDS - detects all distinct elements of a cell array of chars (used by tmg.m, tmg_query.m, tdm_update.m)

[NEW_WORDS, NEW_DOC_IDS]=UNIQUE_WORDS(WORDS, DOC_IDS, N_DOCS)

returns in NEW_WORDS all distinct elements of the cell array of chars WORDS. DOC_IDS is the vector of the document identifiers containing the corresponding words, while N_DOCS is the total number of documents contained to the collection. NEW_DOC_IDS contains the inverted index of the collection as a cell array of 2 x N_DOCS arrays.

`vsm`

VSM - Applies the Vector Space Model to a document collection

[SC, DOCS_INDS] = VSM(D, Q, NORMALIZE_DOCS) applies the Vector Space Model to the text collection represented by the term - document matrix D for the query defined by the vector Q [1]. NORMALIZE_DOCS defines if the document vectors are to be normalized (1) or not (0). SC contains the sorted similarity coefficients, while DOC_INDS contains the corresponding document indices.

REFERENCES:

[1] M.Berry and M.Browne, Understanding Search Engines, Mathematical Modeling and Text Retrieval, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.