DDR API

This is the function reference for DDR.

---

cosine_similarity

This module contains functions for calculating the cosine similarity between two vectors.

---

dot_product()

This function calculates the dot product between two vectors, v1 and v2.

- **dot_product** (**v1**, **v2**)

Parameters:

v1: A vector of numbers of length n.
v2: A vector of numbers of length n.

Returns

Dot product of the two input vectors.

cos_similarity()

This function calculates the cosine similarity between two vectors, v1 and v2.

- **cos_simiarity** (**v1**, **v2**)

Parameters:

v1: A vector of numbers of length n.
v2: A vector of numbers of length n.

Returns

Dot cosine similarity of the two input vectors.

file_length

This module contains a simple function for calculating the number of rows in a file.

---

file_len()

This function calculates the number of rows in a file.

- **file_len** (**fname**)

Parameters:

fname : string
Path to file for which rows should be counted.

Returns

Number of rows in the specified file.

get_loadings

This module contains a function for calculating the loadings of a corpus of documents on a set of constructs. The corpus of documents and the constructs must be represented in distributed form, which can be accomplished using the functions contained in the the get_vecs module. This function writes a tab delineated file where rows correspond to documents and columns correspond to constructs.

---

get_loadings()

This function calculates the similarities between aggregate document vectors and aggregate dictionary vectors and returns a tab delineated with rows as documents and columns as dictionary dimensions. The unique IDs contained in the document CSV input file are transferred to the output file.

- **get_loadings** (**agg_doc_vecs_path, agg_dic_vecs_path, out_path, num_features, delimiter = '\t'**)

Parameters:

agg_doc_vecs_path : string

Path to CSV file with aggregate document vectors as rows and a unique ID as the first column.

agg_dic_vecs_path : string

Path to CSV file with aggregated dictionary vectors as columns.

output_path : string

Path for output.

num_features : int/float

Dimensionality of the Word2Vec model used to generate vector representations.

delimiter : string

Delimiter to use for output CSV file. Default is tab delimited.

Returns NULL

get_vecs

This module contains functions for generating and writing distributed representations of dictionaries and documents from various formats.

---

make_agg_vec()

This function queries the distributed representations of a set of words, averages their representations, and returns this averaged vector.

- **make_agg_vec (words, model, num_features, model_word_set, filter_out =** [])

Parameters:

words : list

List of words that should be included in the returned vector representation.

model : Word2Vec model

Gensim.Word2Vec model containing vector representations to be used for aggregate vectors.

num_features : int/float

Dimensionality of the Word2Vec model being used.

model_word_set : list

List of words in vocabulary of Word2Vec model.

filter_out : list

Optional list of words to exclude from aggregation process.

Returns

Array containing the aggregate vector representation of the input words.

dic_vecs()

This function generates distributed representations of constructs represented by sets of words. We refer to these as distributed dictionary representations

- **dic_vecs** (**dic_terms, model, num_features, model_word_set, filter_out** = [])

  Parameters: **dic_terms** : Python dictionary object

  Python dictionary object with dimension names as keys and words as values

  model : Word2Vec model

  Gensim.Word2Vec model containing vector representations to be used for aggregate vectors.

  num_features : int/float

  Dimensionality of the Word2Vec model being used.

  model_word_set : list

  List of words in vocabulary of Word2Vec model.

  filter_out : list

  Optional list of words to exclude from aggregation process.

  Returns:

  agg_dic_vecs : Python dictionary object

  Python dictionary object with dimension names as keys and aggregate distributed dictionary representation vectors as values.

write_dic_vecs()

This function writes a python dictionary of distributed dictionary representations to a tab delineated file.

- **write_dic_vecs** (**dic_vecs, output_path, delimiter='\t'**)

Parameters:

dic_vecs : Python dictionary object

Dictionary object containing dimension names as keys and words as values.

output_path : string

Path to directory where output should be written.

delimiter : string

Delimiter to use for output CSV file. Default is tab delimited.

Returns NULL

doc_vecs_from_csv()

This function reads a CSV file with documents as rows and generates a CSV file containing aggregate distributed representations of each document in rows. If unique document identifiers are contained in the input file, they can be carried through to the output file. Otherwise, a new set of unique identifiers is output along with the vector representations.

- **doc_vecs_from_csv** (**input_path, output_path, model, num_features, model_word_set, text_col, filter_out = [], quotechar=None, delimiter = "\t", id_col = False, header = True**)

  Parameters:

  input_path : string

  Path to file or directory of files containing text to be analyzed.

  output_path : string

  Path to file or directory of files containing text to be analyzed.

  model : Word2Vec model

  Gensim.Word2Vec model containing vector representations to be used for aggregate vectors.

  num_features : int/float

  Dimensionality of the Word2Vec model being used.

  model_word_set : list

  List of words in vocabulary of Word2Vec model.

  text_col : string or int

  Column name or index number for column containing text to be analyzed.

filter_out : list

Optional list of words to exclude from aggregation process.

quotechar : string
If quote character is used to indicate text in the input file, specify what character is used.
delimiter : string

Delimiter to use for output CSV file. Default is tab delimited.

id_col : String, int, or False

Column name or index number for column containing unique identifier for documents. If input document does not contain a unique ID column, specify 'False' (which is the default value). If id_col is False, unique IDs will be automatically generated.

header : Boolean

Indicates whether input file contains a header or not.

Returns NULL

doc_vecs_from_txt()

- **doc_vecs_from_txt** (**input_path, output_path, num_features, model, model_word_set, filter_out** = [], **delimiter** = '\t')

This function takes as input either the path to a text file where documents correspond to rows or a directory containing text files where each file contains one document. The output is a TSV file where rows represent the distributed representation of documents.

Parameters: input_path : string

Path to file or directory of files containing text to be analyzed.

output_path : string

Path to file or directory of files containing text to be analyzed.

model : Word2Vec model

Gensim.Word2Vec model containing vector representations to be used for aggregate vectors.

num_features : int/float

Dimensionality of the Word2Vec model being used.

model_word_set : list

List of words in vocabulary of Word2Vec model.

text_col : string or int

Column name or index number for column containing text to be analyzed.

filter_out : list

Optional list of words to exclude from aggregation process.

delimiter : string

Delimiter to use for output CSV file. Default is tab delimited.

id_col : String, int, or False

Column name or index number for column containing unique identifier for documents. If input document does not contain a unique ID column, specify 'False' (which is the default value). If id_col is False, unique IDs will be automatically generated.

Returns NULL

load_model

This module contains a helper function that loads a Word2Vec model and returns the model, model dimensionality, and model vocabulary.

---

load_model()

This function loads a Word2Vec model and returns the model object, model dimensionality, and model vocabulary.

- **load_model** (**model_path**)

  Parameters:

  model_path : string

Path to Word2Vec model.

Returns:

model : gensim.Word2Vec model object

A gensim.Word2Vec model object.

num__features : int

Model dimensionality.

model__word__set : list

List containing model vocabulary.

load__terms

This module contains functions for loading dictionary terms in various formats. Their output can serve as input to the dic_vecs() function.

---

get__files()

This function loads construct dictionary terms that are in text format.

- **get__files** (**input__path**)

Parameters:

input__path : string

Directory containing term file(s)

Returns: Dictionary object

A dictionary of file paths. Keys are file names and are used to name dimensions represented in other functions

terms__from__txt()

This function collects the file names of the files contained in the input directory. Note, the names of the files given as input will be used to specify the names of the constructs in the returned dictionary object.

- **terms__from__txt** (**input__path**)

Parameters:

input_path : string

Path to text file containg terms or directory containing multiple text files that each contain terms.

Returns: Dictionary object

A dictionary with dimension names as keys and words as values.

terms_from_liwc()

This function reads a LIWC format dictionary of words into the python dictionary form required for DDR.

- **terms_from_liwc** (**input_path**)

Parameters:

input_path : string

Path to LIWC dictionary

Returns: Dictionary object

A dictionary with dimension names as keys and words as values.

terms_from_csv()

This function reads a CSV file where columns correspond to dictionary constructs and rows contain words associated with the constructs. Column names should represent the dimension associated with the words in the column.

- **terms_from_csv** (**input_path, delimiter='\t'**)

Parameters:

input_path : string

Path to CSV file where columns correspond to constructs and rows contain construct relevant words.

delimiter : string

The delimiter used in the input file.

Returns: Dictionary object

A dictionary with dimension names as keys and words as values.

terms_to_csv()

This function writes to file a python dictionary where keys are construct names and values are the words associated with a given construct.

- **terms_to_csv** (**terms_dic, output_path, delimiter**)

Parameters:

input_path : string

Path to CSV file where columns correspond to constructs and rows contain construct relevant words.

output_path : string

The path to which the file should be written.

delimiter : string

The delimiter used in the input file.

Returns: NULL

nearest_neighbors

This module contains a function for identifying the nearest neighbors in a Word2Vec model to the distributed dictionary representations of given dictionaries of terms.

ddr_neighbors()

This function identifies the nearest neighbors in a Word2Vec model to the distributed dictionary representations of a given dictionaries of terms.

- **ddr_neighbors** (**dictionary_terms, model, n=2**)

Parameters:

dictionary_terms : Python dictionary object

This object should have the same form as the term dictionaries generated by the functions contained in the load_terms module. Specifically, keys are taken to represent construct names and values are taken to represent the words associated with a given construct.

model : string

A Word2Vec model object

n : int

The number of nearest neighbors to return for each construct.

Returns: Python dictionary object

Keys in the returned python dictionary are the construct names and values are the n words nearest to the distributed dictionary representation.

simple_progress_bar

This module contains a function implementing a simple progress bar.

update_progress()

This function implements the simple progress bar used by other DDR functions.

- update_progress(progress)

Parameters:

Progress : int

Indicator of progress through n iterations.

Returns NULL