subs2vec

Oct 22, 2020

Module and submodules

subs2vec is a set of word embeddings trained on large subtitle corpora in 50 languages, and the code accompanying this data, as published in Van Paridon & Thompson (2019).

The code is provided at github.com/jvparidon/subs2vec, with instructions for use as command line tools included.

This page serves to document the full subs2vecs module API in more detail for anyone who is interested and/or wishes to use/reuse the code.

CHAPTER 1

subs2vec

Provides:

- Routines for interacting with, evaluating, and training word embeddings as used in Van Paridon & Thompson (2020).
- Routines for extending lexical norms to unnormed words.
- Routines for retrieving word, bigram, and trigram frequencies.
- Command line tools to conveniently do any of the above, in 55 languages.

1.1 subs2vec.analogies

Evaluate word vectors by solving syntactic and semantic analogies.

`subs2vec.analogies.`**`evaluate_analogies`**(*lang*, *vecs_fname*, *method='multiplicative'*, *whole_matrix=False*)

Solve all available analogies for a set of word vectors in a given language.

Writes scores to tab-separated text file but also returns them.

Parameters

- **`vecs_fname`** – filename of a file containing a set of word vectors
- **`lang`** – language to evaluate word vectors in (uses two-letter ISO codes)
- **`method`** – solving method to use (options are *additive* and *multiplicative*, multiplicative is the default and usually performs best)
- **`whole_matrix`** – boolean determining whether to use whole matrix multiplication (faster, but uses more RAM than you may have available, *False* is the default)

Returns pandas DataFrame containing the analogies results

subs2vec.analogies.**novel_analogies**(*vecs_fname*, *analogies_fname*, *method='multiplicative'*, *whole_matrix=False*)

Solve novel analogies, using word vectors.

Writes predictions to tab-separated text file.

Parameters

- **vecs_fname** – file containing word vectors to use for prediction.
- **analogies_fname** – file containing analogies in tab-separated columns named 'a1', 'a2', and 'b1'
- **method** – solving method to use (options are *additive* and *multiplicative*, multiplicative is the default and usually performs best)
- **whole_matrix** – boolean determining whether to use whole matrix multiplication (faster, but uses more RAM than you may have available, *False* is the default)

subs2vec.analogies.**solve_analogies**(*vectors*, *analogies*, *novel=False*, *method='multiplicative'*, *whole_matrix=False*)

Solves analogies using specified methods.

Parameters

- **vectors** – Vectors object containing word vectors
- **analogies** – pandas DataFrame of analogies, columns labeled a1, a2, b1(, b2)
- **novel** – whether the task is to solve novel analogies (or alternatively, score the predictions relative to existing analogies)
- **method** – solving method to use (options are *additive* and *multiplicative*, multiplicative is the default and usually performs best)
- **whole_matrix** – boolean determining whether to use whole matrix multiplication (faster, but uses more RAM than you may have available, *False* is the default)

Returns dict containing score and predictions in separate pandas DataFrames

1.2 subs2vec.clean_subs

Clean and concatenate OpenSubtitles archives into a single training corpus.

subs2vec.clean_subs.**join_archive**(*lang*, *ioformat='txt'*, *years=(1900, 2050)*, *verbose=False*)

Concatenate a stripped OpenSubtitles archive into a single training corpus.

Parameters

- **lang** – language of the archive to join (selects the correct archive)
- **ioformat** – input/output format (default is txt)
- **years** – tuple of first and last year to select only subtitles from a range of years, default is 1900 to 2100.
- **verbose** – boolean setting whether or not to print a progress bar to the command line, default is False

Returns number of files that were concatenated

subs2vec.clean_subs.**strip_archive**(*lang*, *ioformat='txt'*, *years=(1900, 2100)*)

Strip xml from a compressed OpenSubtitles archive.

Writes stripped output directly to a new compressed archive.

Parameters

- **lang** – language of the archive to strip (selects the correct archive)
- **ioformat** – input/output format, default is txt
- **years** – tuple of first and last year to select only subtitles from a range of years, default is 1900 to 2100

1.3 subs2vec.clean_wiki

Clean Wikipedia dumps for use as a training corpus.

subs2vec.clean_wiki.**big_strip_wiki_file**(*fname*, *lines_per_chunk=1000000.0*)
Strip xml and other tags from a Wikipedia dump that doesn't fit into RAM.

Processes Wikipedia dump in chunks and then concatenates the junks into a single text file.

Parameters

- **fname** – Wikipedia dump file, in xml or bzip2 format.
- **lines_per_chunk** – number of lines in each chunk (default is 1e6, one million lines)

subs2vec.clean_wiki.**strip_wiki_file**(*fname*)
Strip xml and other tags from Wikipedia dump.

Writes stripped Wikipedia text directly to text file.

Parameters **fname** – Wikipedia dump file, in xml or bzip2 format.

1.4 subs2vec.count_words

Count words in a training corpus or other file or directory of files.

subs2vec.count_words.**count_words**(*fname*)
Count words in a file.

Parameters **fname** – name of file to count words in

Returns number of words in the file

subs2vec.count_words.**count_words_in_path**(*path*)
Count words in text files on a given path.

Parameters **path** – either filename or directory name to count words in

1.5 subs2vec.deduplicate

Remove duplicate lines from a text file.

subs2vec.deduplicate.**big_dedup_file**(*in_fname*, *out_fname*, *n_bins*)
Remove duplicate lines from a text file that does not fit into RAM.

Because of *itertools.cycle()* this is only pseudorandomized and pseudodeduplicated (i.e.: consecutive lines of input cannot end up as consecutive in the output and up to n_bins duplicates of an item may remain). If your

file fits into RAM afterward, you may consider running it through normal deduplication (which includes true randomization). Writes directly to text file.

Parameters

- **in_fname** – file to read text from
- **out_fname** – file to write text to
- **n_bins** – number of bins to split files into (note that up to n_bins duplicates of any given line may remain after applying this function.

subs2vec.deduplicate.**dedup_file**(*in_fname*, *out_fname*)

Removes duplicate lines from a text file.

Writes directly to text file.

Parameters

- **in_fname** – file to read text from
- **out_fname** – file to write text to

1.6 subs2vec.download

Download datasets for training subs2vec models (not Windows-compatible).

subs2vec.download.**download_corpus**(*lang*, *source*)

Convenient method for downloading corpora.

Uses the subprocess module and curl to download corpora. Not Windows-compatible.

Parameters

- **lang** – language to download (use 2-letter ISO code)
- **source** – corpus to download, (options are "subs" or "wiki")

subs2vec.download.**download_vecs**(*lang*, *source*, *binaries=False*)

Convenient method for downloading vectors.

Uses the subprocess module and curl to download corpora. Not Windows-compatible.

Parameters

- **lang** – language to download (use 2-letter ISO code)
- **source** – corpus to download, (options are "subs", "wiki", or "wiki+subs")
- **binaries** – download binaries instead of vecs, boolean (set to False by default)

1.7 subs2vec.frequencies

Extract unigram, bigram, and trigram frequencies, either from a text corpus or from a pre-existing frequencies file.

subs2vec.frequencies.**count_ngrams**(*fname*, *kind='words'*, *min_freq=1*, *no_bigrams=False*, *no_trigrams=False*)

Counts unigrams, bigrams, and trigrams line-by-line in a text corpus.

Parameters

- **fname** – text corpus to count in

- **kind** – kind of items to count (options are 'words' or 'letters')
- **min_freq** – minimum frequency threshold for an item to be included in the output
- **no_bigrams** – whether to skip counting bigrams for improved speed/memory footprint (default False)
- **no_trigrams** – whether to skip counting trigrams for improved speed/memory footprint (default False)

Returns tuple of pandas DataFrames containing frequencies for unigrams, bigrams, and trigrams, respectively

1.8 subs2vec.lang_evaluate

Evaluate a set of vectors for a given language.

subs2vec.lang_evaluate.**evaluate_lang**(*lang*, *vecs_fname*)
Evaluate a set of vectors on all available metrics.

Parameters

- **lang** – language to evaluate
- **vecs_fname** – .vec file containing word vectors

1.9 subs2vec.lookup

Look up items in large lists of frequencies, norms, or vectors.

subs2vec.lookup.**lookup**(*big_fname*, *items_fname*)
Line-wise Lookup of a list of items in a large file.

Parameters

- **big_fname** – filename of large file to look up items in
- **items_fname** – filename of list of items to look up

1.10 subs2vec.norms

Predict lexical norms, either to evaluate word vectors, or to get norms for unnormed words.

subs2vec.norms.**evaluate_norms**(*lang*, *vecs_fname*, *alpha=1.0*)
Predict lexical norms to evaluate a set of word vectors in a given language.

Writes scores to tab-separated text file but also returns them.

Parameters

- **lang** – language to evaluate word vectors in (uses two-letter ISO codes)
- **vecs_fname** – word vectors to evaluate
- **alpha** – regularization strength, default 1.0, set higher for small datasets

Returns pandas DataFrame containing the norms results

subs2vec.norms.**extend_norms**(*vecs_fname*, *norms_fname*, *alpha=1.0*)
Extend lexical norms to unobserved words, using word vectors.

Writes predictions to tab-separated text file.

Parameters

- **vecs_fname** – file containing word vectors to use for prediction.
- **norms_fname** – file containing norms in tab-separated columns, first column should contain words,

first line should contain column names, unobserved cells should be left empty :param alpha: regularization strength, default 1.0, set higher for small datasets

subs2vec.norms.**predict_norms**(*vectors*, *norms*, *alpha=1.0*)
Predict lexical norms and return score.

Parameters

- **vectors** – Vectors object containing word vectors
- **norms** – pandas DataFrame of lexical norms
- **alpha** – regularization strength, default 1.0, set higher for small datasets

Returns dict containing scores and predictions in separate pandas DataFrames

1.11 subs2vec.plots

1.12 subs2vec.similarities

Compute rank correlations between word vector cosine similarities and human ratings of semantic similarity.

subs2vec.similarities.**compare_similarities**(*vectors*, *similarities*)
Correlate vector similarities to human ratings of semantic similarity.

Computes cosine similarities, and uses rank (Spearman) correlation as a measure of similarity to the specified human ratings.

Parameters

- **vectors** – Vectors object containing word vectors.
- **similarities** – pandas DataFrame of similarities, labeled word1, word2, and similarity

Returns dict containing score and predictions in separate pandas DataFrames

subs2vec.similarities.**compute_similarities**(*vectors*, *df_words*)
Compute semantic similarities for novel word pairs.

Parameters

- **vectors** – Vectors object containing word vectors
- **df_words** – pandas DataFrame containing word pairs in columns labeled 'word1' and 'word2'

Returns pandas DataFrame containing word pairs and cosine similarities in a column labeled 'similarity'

subs2vec.similarities.**evaluate_similarities**(*lang*, *vecs_fname*)
Compute similarities for all available ratings datasets for a set of word vectors in a given language.

Writes scores to tab-separated text file but also returns them.

Parameters

- **lang** – language to evaluate word vectors in (uses two-letter ISO codes)
- **vecs_fname** – word vectors to evaluate

Returns pandas DataFrame containing the similarities results

subs2vec.similarities.**novel_similarities**(*vecs_fname*, *words_fname*)
Predict semantic similarities for novel word pairs, using word vectors.

Writes predictions to tab-separated text file.

Parameters

- **vecs_fname** – file containing word vectors to use for prediction.
- **similarities_fname** – file containing word pairs in tab-separated columns named 'word1' and 'word2'

1.13 subs2vec.train_model

Train a model using the default parameters from Van Paridon & Thompson (2019).

Use of this module requires working binaries for fastText and word2vec on the path and plenty of RAM. Please note that even on a very fast desktop, training could take hours or days.

subs2vec.train_model.**build_phrases**(*training_data*, *phrase_pass=5*)
Use word2phrase to connect common phrases.

Uses the word2phrase tool to connect high mutual information phrases such as "New York" using underscores, so fastText will treat them as a single lexical item. Requires a working word2phrase (included in word2vec) binary on the path.

Parameters

- **training_data** – text file containing the training corpus
- **phrase_pass** – number of passes to do over the training file (default is 5)

Returns filename of the phrase-joined corpus

subs2vec.train_model.**fix_encoding**(*training_data*)
Fix utf-8 file encoding after word2phrase mangles it (happens sometimes).

Parameters **training_data** – file containing text with encoding that needs fixing

Returns filename of repaired text file

subs2vec.train_model.**generate**(*lang*, *prefix*, *training_data*, *lowercase=False*)
Generate a fastText model using default parameters from Van Paridon & Thompson (2019).

Parameters

- **lang** – language tag to use for the model binary and vector filenames
- **training_data** – text file containing the training corpus
- **prefix** – prefix to use for the model binary and vector filenames

- **lowercase** – boolean setting whether to cast training corpus to lower case (default is *False*)

subs2vec.train_model.**lowercase**(*training_data*)

Cast a text file to lower case.

Parameters **training_data** – file containing text to cast to lower case

Returns filename of lower cased text file

subs2vec.train_model.**train_fasttext**(*training_data*, *prefix*, *lang*, *d=300*, *neg=10*, *epoch=10*, *t=0.0001*)

Train a fastText model on a given training corpus.

Requires a working fastText binary on the path.

Parameters

- **training_data** – text file containing the training corpus
- **prefix** – prefix to use for the model binary and vector filenames
- **lang** – language tag to use for the model binary and vector filenames
- **d** – number of dimensions in the vector (default is 300)
- **neg** – number of negative samples (fastText default is 5, subs2vec default used here is 10)
- **epoch** – number of training epochs (fastText default is 5, subs2vec default used here is 10)
- **t** – sampling threshold (default is .0001)

Returns tuple of filenames for model binary and vectors

1.14 subs2vec.utensils

subs2vec.utensils.**log_timer**(*func*)

Decorator to add logging timer to other functions.

Use by prepending *@log_timer* to the target function definition. Logs function name and duration in seconds to level *INFO*.

Parameters **func** – any function

Returns func with logging timer

subs2vec.utensils.**timer**(*func*)

Decorator to add timing wrapper to other functions.

Use by prepending *@timer* to the target function definition. Logs start and finish time in *y/m/d h:m:s* and keeps track of duration in seconds. Wrapper returns a tuple containing the original results and a dictionary containing start, finish, and duration.

Parameters **func** – any function

Returns func with timing wrapper

1.15 subs2vec.vecs

Provides Vectors object and methods to load, write, and interact with word vectors.

class subs2vec.vecs.**Vectors**(*fname*, *normalize=False*, *n=1000000.0*, *d=300*)

Bases: object

Creates a Vectors object containing numpy arrays of words and word vectors.

as_df()

Casts word vectors to pandas DataFrame.

Each row contains a vector, each column corresponds with a vector dimension. Rows are indexed by word.

Returns pandas DataFrame containing word vectors

as_dict()

Casts word vectors to Python dict.

The dict is indexed by word, with the items being word vectors in the form of numpy arrays.

Returns Python dict containing word vectors

write_vecs(*vecs_fname*)

Writes word vectors to .vec file.

Parameters **vecs_fname** – filename to write vectors to

CHAPTER 2

Indices and tables

- genindex
- modindex

Python Module Index

S

Index

T

V

W