VECTORIZATION OF INPUT TEXT USING GloVe VECTORS 2

Introduction 2

Example 2

Input Data 4

1) Vocabulary File 4

2) Text String 5

Dependencies 5

Python dependency 5

Function Explanation 6

1) load\_bin\_vec 6

2) vector\_from\_line 6

Examples 6

Loading Python file 6

Calling the Function 6

Loading Individual function 7

Executing the code 7

Help 7

Processing text in Social Media Posts 8

Input Data 8

Dependencies 8

Python Dependency 8

Function Explanation 9

cleantext 9

Removing Stop words and Stemming 9

Python Dependency 9

NLTK Package 9

Function Explanation 9

removeStopwordswithStemming 10

Execution of Unit Test 10

# VECTORIZATION OF INPUT TEXT USING GloVe VECTORS

## Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

More information can be found in the paper by Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global Vectors for Word Representation](http://nlp.stanford.edu/pubs/glove.pdf)

The code shared has been utilized for sentiment analysis on Twitter data. The output of the code would be an n-dimensional vector representation of the input text. Dimensions of the output depend on the input Vocabulary file used. Each vector has a fixed number of values for a token; these values become the output dimensions. Vector files have varying number of dimensions, publically available data has dimensions in the range of 25 to 300.

### Example

An example text of "**A long time ago in a galaxy far, far away**" would look like below in a [300 dimension common crawl](#_Explanation_of_available) vocabulary file.

[ 9.29011000e-01 5.29604000e-01 -6.65878000e-01 4.08702000e-01

1.12849000e+00 -2.27910800e+00 -1.80958300e+00 -1.02889600e+00

1.08694000e+00 1.90321800e+01 -1.25959500e+00 1.89732000e-01

-3.54128000e-01 -1.16974600e-01 -1.50833622e+00 1.62135000e-01

4.91749200e-01 1.04115700e+01 -1.18038100e+00 -2.52731000e-01

-5.67814200e-01 -8.28422000e-01 -5.57081000e-01 6.39157000e-01

-5.29896000e-01 9.36841300e-01 2.06823000e-01 8.27082000e-01

1.49097300e+00 1.43105230e+00 2.03604000e-01 7.51776800e-01

-6.15214000e-01 -1.59450800e+00 2.16575000e+00 -4.81026000e-01

1.21220700e+00 -5.49405000e-01 -1.33860300e+00 -1.20192700e+00

7.49733300e-01 9.45554200e-01 5.55220000e-01 -1.55007000e-01

-2.47663000e-01 -6.73540000e-03 -2.13899200e+00 -2.90519000e-01

-1.44375630e+00 -1.35387900e+00 -3.69843000e-01 7.96545000e-01

-4.60462000e-01 -1.02222100e+00 8.25063000e-01 1.15813900e+00

1.04170170e+00 -9.34856000e-01 2.82196500e+00 -1.14607000e+00

-5.56784400e-01 -1.08601000e-01 2.20990000e-02 7.81124000e-01

6.86984200e-01 -1.23929400e+00 -4.46473800e-01 1.52139600e+00

1.41570000e-02 2.24320190e+00 1.56899100e+00 1.01710120e+00

2.05691000e-01 9.16102800e-01 5.22148000e-01 -7.17086100e-01

1.09318100e+00 -1.09719400e+00 -1.64063600e+00 2.79256720e+00

-1.63740000e-01 8.57633000e-01 -6.69684000e-01 2.08267350e-01

-5.62067000e-01 -1.35810200e+00 -2.96729000e-01 -2.14325100e+00

1.89866000e+00 5.63275000e-02 -3.69283000e-01 1.24856500e+00

-2.76025320e+00 1.35055000e-01 -2.48730000e-02 1.40613000e+00

1.32286636e+00 -3.02227000e-01 4.79448000e-01 -7.58540000e-01

2.46278690e-01 6.91777400e-01 -2.33277000e-01 8.18845680e-01

-4.93840300e-01 -4.09708000e+00 1.44969900e+00 8.16190000e-01

1.14085900e+00 -7.61736000e-01 1.56074800e+00 -1.56114300e+00

1.29571400e+00 4.43117000e-01 1.06827900e+00 1.25113060e+00

-7.96920000e-01 1.66123000e+00 -9.26387000e-01 9.82556000e-01

1.48831100e-01 -6.92526000e-01 4.41994000e-01 -6.10335000e-01

2.23030000e-01 3.82146000e-01 1.57114000e-01 -5.33024500e-01

-9.05163500e-01 5.56395700e-01 5.68513000e-01 -4.58530300e-01

1.48410000e-02 3.22582000e-02 1.13558720e+00 -1.40991548e+00

-2.09572500e+00 7.74305000e-01 -7.59689000e-01 1.38177000e-01

-1.22752300e+01 1.35485200e+00 2.23664700e+00 -3.55558000e-01

6.22208000e-01 -1.91573600e+00 -2.10280970e+00 -6.08678000e-02

1.66150000e+00 -1.01111000e-01 1.77955100e+00 4.72449000e-01

9.23866000e-02 -8.38923000e-02 -1.37539000e+00 -3.46431000e-01

-2.55362000e-01 2.66827900e+00 9.29949000e-02 -9.53573100e-01

-5.85981500e-01 2.36690000e-02 1.09124300e+00 1.58859000e-01

-4.10897000e-01 8.41107000e-01 4.67770000e-01 3.07834120e-01

1.71357000e+00 4.41474000e-01 9.83849000e-01 -6.79229000e-01

-1.86529000e-01 -2.10371000e+00 -1.16732600e+00 7.87655000e-01

8.03023000e-01 -7.77200000e-01 3.08298000e-01 -5.15940000e-01

6.63339000e-01 -2.18409000e+00 -2.21785000e-01 -1.69873800e+00

1.03815300e+00 -1.12113062e+00 -2.63481800e+00 2.01501000e-01

1.49999700e+00 3.83799000e-01 -1.00421500e+00 2.88102000e-01

-1.15085200e+00 5.89758000e-01 3.30608000e-01 2.11133000e-01

-1.39134000e+00 -1.15050100e+00 -1.83299260e+00 8.43875000e-01

-3.75354000e-01 -2.80944000e-01 -1.03559100e+00 1.27389800e+00

1.01873940e+00 1.00038339e+00 1.14976400e+00 -1.20577200e+00

-5.05070000e-01 -5.42464000e-01 -8.41237650e-01 -8.12729000e-01

-5.83593000e-01 -2.52843030e+00 1.72766000e+00 8.08340000e-03

-1.64380000e-01 7.64562100e-01 -1.73744000e-01 -7.86905000e-01

1.24417000e+00 -1.86068400e+00 -1.92887700e+00 -6.46824000e-01

-7.47732000e-01 -3.13541000e-01 3.01025000e-01 1.48315780e+00

-4.86510000e-01 8.42178000e-01 1.05389100e+00 2.10914500e+00

1.39273440e+00 9.79612100e-01 -1.28534700e+00 -1.18483300e+00

-6.24000000e-03 -6.93508000e-01 -2.32093000e+00 1.06052100e+00

1.26123500e+00 -3.19037000e-01 1.38393400e+00 1.93045700e+00

-3.71880700e-01 3.42005500e-01 -1.39996400e+00 1.22553000e-01

-1.01510500e+00 1.08634900e+00 -1.94782000e-01 -1.28204800e+00

1.25536700e+00 1.51870000e+00 1.00346900e+00 3.29835000e+00

2.63650000e-02 2.24058200e-01 -1.55732100e-01 -3.88500000e-03

9.85360500e-01 3.65781000e+00 2.81244000e-01 1.52443760e+00

1.10965000e-01 -1.37811300e+00 1.04129790e+00 -2.58604000e-02

-4.83545000e-01 -8.80727070e-01 -1.77646500e-01 2.18830000e-01

-8.10095000e-01 -4.44029000e+00 -1.61950700e+00 2.43230000e-01

-1.78953000e+00 -6.43274000e-01 1.13155020e+00 1.63047596e+00

1.20377200e+00 1.21790900e+00 -1.39710600e+00 -1.77789000e-01

-4.29360000e-01 -2.53984580e+00 8.23061000e-01 -1.89326300e+00

1.15341460e+00 -5.18060000e-01 -1.93759800e+00 6.18098400e-01

2.87931000e-01 -1.06683200e+00 1.36519900e+00 7.03638000e-01

-2.84436500e-01 2.00400000e-02 -1.95623124e+00 9.32337600e-01]

## Input Data

The code requires two inputs:

### Vocabulary File

Need to provide path to a valid vocabulary file.

Vocabulary file contains the mapping of the word to the feature space. Pre-trained word vectors are available and can be used. They can be found at <http://nlp.stanford.edu/projects/glove/>

#### Explanation of available vocabulary files

There are 4 publically available GloVe vocabulary files

1. Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): [glove.6B.zip](http://nlp.stanford.edu/data/glove.6B.zip)
   1. This vocabulary file uses a web crawl of Wikipedia data and the GIgaword corpus. English Gigaword is a comprehensive archive of newswire text data in English that has been acquired over several years by the LDC. (Linguistic Data Consortium).
   2. This data should be used where the proper usage of English language is expected.
2. Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): [glove.42B.300d.zip](http://nlp.stanford.edu/data/glove.42B.300d.zip)
   1. Common Crawl dataset is created by a web crawl. This dataset contains 42 Billion tokens.
   2. This dataset should be used for generic English language usage. It would contain a mix of proper English and web slangs.
3. Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): [glove.840B.300d.zip](http://nlp.stanford.edu/data/glove.840B.300d.zip)
   1. This is a bigger corpus of 840 billion tokens, everything else is similar to the above corpus.
4. Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): [glove.twitter.27B.zip](http://nlp.stanford.edu/data/glove.twitter.27B.zip)
   1. This dataset is created using 2 Billion tweets and 27 billion tokens from it.
   2. It has multiple dimensions. Usage is dependent on the how comprehensively is the feature space to be mapped. Higher dimensions will give more granularity of the feature space.
   3. This dataset should be used for usage with text from twitter or Social media.

#### Sample of Vocabulary File

A sample of the GloVe Vocabulary file

the 0.04656 0.21318 -0.0074364 [...] 0.053913

, -0.25539 -0.25723 0.13169 [...] 0.35499

. -0.12559 0.01363 0.10306 [...] 0.13684

of -0.076947 -0.021211 0.21271 [...] -0.046533

to -0.25756 -0.057132 -0.6719 [...] -0.070621 [...]

sandberger 0.429191 -0.296897 0.15011 [...] -0.0590532

Each new line contains a token followed by N-dimensional signed floats, depending on the length of features in each vector

#### Valid File

The input file should be a whitespace (\t, \n, \f, \s, \r) delimited file, in which the word before the first delimiter specifies the vector space for that word.

### Text String

The text string is the input text, which needs to be vectorized. Input string is separated using a whitespace delimiter.

## Dependencies

The code has the following dependencies:

### Python dependency

#### NumPy package

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

#### Argparse package

Command line option and argument parsing.

## Function Explanation

The code contains two functions:

### load\_bin\_vec

This function is used to load the vocabulary file. It has been tested on GloVe vocabulary file. Any glove vocabulary file either publically available or manually trained according to the format [specified above](#_Valid_File) is valid.

#### Input

vocab\_file: Path to the vocabulary file

#### Output

glove\_vec : Python dict object containing the vectors from the vocabulary file

vocab\_size : Integer object containing the number of feature dimensions

### vector\_from\_line

This function vectorizes the input text. It returns a numeric value associated with the input string.

#### Input

line: Text String that needs to be vectorized

glove\_vec : Python dict object containing the vectors from the vocabulary file

vocab\_size : Integer object containing the number of feature dimensions

#### Output

vec : Numpy array containing the numeric vector representation of the sentence

## Examples

### Loading Python file

import sys

sys.path.append(<path to the Featurization.py directory>

import Featurization

### Calling the Function

After loading the python file:

#### Load GloVe vectors by calling

glove\_vec, vocab\_size = load\_bin\_vec(vocab\_file)

glove\_vec : Python dict object containing the vectors from the vocabulary file

vocab\_size : Integer object containing the number of feature dimensions

vocab\_file: Path to the vocabulary file

#### Vectorize Input Text

line\_vector = vector\_from\_line(text,glove\_vec,vocab\_size)

text : Text String that needs to be vectorized

glove\_vec : Python dict object containing the vectors from the vocabulary file

vocab\_size : Integer object containing the number of feature dimensions

line\_vector : Numpy array containing the numeric vector representation of the sentence

### Loading Individual function

import sys

sys.path.append(<path to the Featurization.py directory>

from Featurization import <Function name>

### Executing the code

The given code file can be executed as it is to find vectors for an input text. It utilizes input parameters which are explained below

Python FeaturizationExecutor.py -v VOCAB\_FILE -t TEXT

**VOCAB\_FILE : Path to the vocabulary file**

**TEXT : String input that needs to be vectorized**

The above command will print the word vector for the passed input text

### Help

Help file available with the code can be accessed using the below command

Python FeaturizationExecutor.py –h

**Ouptut**

usage: FeaturizationExecutor.py [-h] -v VOCAB\_FILE -t TEXT

This Script is used to create the Vector Representation for Words

optional arguments:

-h, --help show this help message and exit

Required Named Arguments:

-v VOCAB\_FILE, --vocab\_file VOCAB\_FILE

Path of Pre-Trained Word Vectors in tab seperated

format. Further information about the format is

available in the Readme file

-t TEXT, --text TEXT Input text for which the Vectors need to be formed

# Processing text in Social Media Posts

Text that is typed in social media posts is unique in the way that it has its own set of grammar. We are trying to extract information and reduce noise by using some of the rules that are known by us.

The code to process Social Media posts does the following

Identifies the emoticon being used and replaces them with text. This would help in extracting information from emoticons. The code

* Classifies emoticons into 4 major categories, as ***“Joking”***, ***“Happy”***, ***“Sad”*** and blank (***“ “***)
* Removes URL’s
* Removes hash (#) from hash tags
* Replaces user references AT\_USER ( @Barclays -> AT\_USER)
* Duplicate letter beyond two letters are removed
* Removes ascii characters.

The function that performs these operations is named as ***“cleantext”***

## Input Data

The code requires a text string as input.

## Dependencies

### Python Dependency

#### NLTK Package

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](http://nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries

#### ITERTOOLS Package

This module provides functions for ‘iterator algebra’. The module standardizes a core set of fast, memory efficient tools that are useful by them or in combination.

#### Math Package

This module provides access to the mathematical functions defined by the C standard.

## Function Explanation

### cleantext

This function is used to process text from Social media posts and convert emoticons to a text value. The emoticons are classified into four values, “Joking”, “Happy”, “Sad” and blank (“ “).

#### Input

text : Input text that needs to be cleaned in String format

#### Example

##### Clean Input Text

text\_clean = cleantext(text)

# Removing Stop words and Stemming

Using the NLTK library a Porter-Stemming algorithm is executed on the input text. Also commonly used terms referred as stop words are removed

## Python Dependency

### NLTK Package

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](http://nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries

## Function Explanation

### removeStopwordswithStemming

Function to execute the stop word removal and porter-stemming algorithm

#### Input

text : Input text that needs to be processed. The text should be in String format

#### Example

##### Execute stemming and stop word removal

text\_clean = removeStopwordswithStemming(text)

# Execution of Unit Test

Unit tests for this module have been added in the file **UnitTest.py**.

Call the **UnitTest.py** file from terminal or command prompt to execute the unit test

#### Example

##### Execution of the unit test

Python UnitTest.py