SimilarityLab

Release 0.0.1dev

Apr 07, 2020

Contents:

CHAPTER 1

Getting Started

En es parte aprenderemos a utilizar el modulo

class SimiLab.**tempName**(*matrices*, *yearDict*, *vocabularies*)

This class will allow you to track how words changes across time using embedding matrices of a given corpus. These matrices should be aligned with Procrustes.

Parameters

- **matrices** (*array_like*) – Embedding matrices
- **yearDict** (*dict*) – Stores the row index for a given word for each matrix
- **vocabularies** (*dict*) – Need to define

checkProjection()

Checks if the model's matrices have been projected by the user. Returning the state of projection.

Parameters **None** –

Returns **out** – False if the matrices are not projected. True if the matrices are projected.

Return type bool

findSimilars2Vec(*vector*, *year*, *threshold=0*, *maxWords=None*)

Finds the most similar words within a word2vec embedding matrix. This function computes the cosine similarities between embedding vectors and a given vector. Returning the most similar words within a given treshold.

Parameters

- **vector** (*array_like*) – Input Vector. Must match embedding dimension.
- **year** (*int*) – Choosen year.
- **threshold** (*float*) – Minimun cosine similarity allowed to consider a word 'close' to the given vector. If left blank, the default value is 0, which allows for all vectors to be considered.
- **maxWords** (*int*) – Maximum number of words to be returned as most similar. If left blank, the default value is 'None', which allows for all vectors to be considered.

Returns **out** – A dictionary containing the words found and its cosine similarities with respect to given input vector.

Return type dict

Raises `ValueError` – if 'treshold' is a negative floating point number, or it's value is greater than 1.0. if 'year' is not present in the current data.

Examples

```
>>> ma = [[-1,-2,-3],[4,5,6],[7,8,9]]
>>> mb = [[1,2.1,3],[4.2,4.8,6],[7.02,8,9.3]]
>>> mc = [[1.1,2.2,3.1],[4.23,5,6],[7.03,8,9.32]]
```

```
>>> matrices = [ma, mb, mc]
>>> yearDict = {1990:0, 1991:1, 1995:2}
>>> vocab1990 = {'martin':0, 'pablo':1, 'carlos':2}
>>> vocabularies = [vocab1990]
```

```
>>> tempObject = tempName(matrices, yearDict, vocabularies)
```

```
>>> newVec = tempObject.findSimilars([1,2,3], 3, 1990)
```

```
>>> print(newVec)
```

```
>>> {'pablo': 0.9746318461970761, 'carlos': 0.9594119455666702, 'martin': -1.
↪0}
```

findSimilars2Word(*word*, *year*, *threshold=0*, *maxWords=None*)

Finds the most similar words within a word2vec embedding matrix. This function computes the cosine similarities between embedding vectors and a given word. Returning the most similar words within a given treshold.

Parameters

- **word** (*string*) – Input Word. Must be part of selected year's vocabulary.
- **year** (*int*) – Choosen year.
- **threshold** (*float*) – Minimun cosine similarity allowed to consider a word 'close' to the given vector. If left blank, the default value is 0, which allows for all vectors to be considered.
- **maxWords** (*int*) – Maximum number of words to be returned as most similar. If left blank, the default value is 'None', which allows for all vectors to be considered.

Returns **out** – A dictionary containing the words found and its cosine similarities with respect to given input word.

Return type dict

Examples

```
>>> ma = [[-1,-2,-3],[4,5,6],[7,8,9]]
>>> mb = [[1,2.1,3],[4.2,4.8,6],[7.02,8,9.3]]
>>> mc = [[1.1,2.2,3.1],[4.23,5,6],[7.03,8,9.32]]
```

```
>>> matrices = [ma, mb, mc]
>>> yearDict = {1990:0, 1991:1, 1995:2}
>>> vocab1990 = {'martin':0, 'pablo':1, 'carlos':2}
>>> vocabularies = [vocab1990]
```

```
>>> tempObject = tempName(matrices, yearDict, vocabularies)
>>> newVec = tempObject.findSimilars2Word('pablo', 3, 1990)
>>> print(newVec)
    {'pablo': 1.0, 'carlos': 0.9981908926857268, 'martin': -0.974631846197076}
```

getEvol (*w1*, *y1*, *y2*)

Finds a given word's evolution between two years. This function computes the cosine similarity between the vectorized representation of a given word in one year and the same word in a different year. Returning said cosine similarity.

Parameters

- **y1** (*int*) – First choosen year.
- **w1** (*string*) – Input word. Must be present in y1 and y2.
- **y2** (*int*) – Second choosen year.

Returns **out** – Cosine similarity between the obtained vectors from w1 in both years.

Return type float

getEvolByStep (*word*)

Finds one word's evolution throughout all years. This function computes the cosine similarities between the vectorized representation of a given word from one year to the next, covering all years.

Parameters **word** (*string*) – Choosen word.

Returns **out** – A list containing all cosine similarities from one year to the next. If the word is missing from one of the two compared years, 'missing' is returned in the list's corresponding position.

Return type array_like

getSim (*w1*, *y1*, *w2*, *y2*)

Finds the similarity between two selected words in two given years. This function computes the cosine similarity between the vectorized representation of a first word in a selected year, and the vector for a second word in another year. Returning the cosine similarity between the two vectors.

Parameters

- **y1** (*int*) – First choosen year.
- **w1** (*string*) – First input word. Must be present in y1.
- **y2** (*int*) – Second choosen year.
- **w2** (*string*) – Second input word. Must be present in y2.

Returns **out** – Cosine similarity between the obtained vectors from w1 and w2.

Return type float

getVector (*word*, *year*)

Finds the vectorized representation of a given word within a word2vec embedding matrix. This function looks for a chosen word in a given year's vocabulary and, if posible, identifies it's corresponding vector. Returning the vectorized representation of the word.

Parameters

- **word** (*string*) – Input Word. Must be part of selected year's vocabulary.
- **year** (*int*) – Choosen year.

Returns **out** – The selected word's vectorized representation.

Return type array_like

Raises `ValueError` – if 'word' is not present in the year's vocabulary. if 'year' is not present in the current data.

Examples

```
>>> ma = [[-1,-2,-3],[4,5,6],[7,8,9]]
>>> mb = [[1,2.1,3],[4.2,4.8,6],[7.02,8,9.3]]
>>> mc = [[1.1,2.2,3.1],[4.23,5,6],[7.03,8,9.32]]
>>> matrices = [ma, mb, mc]
>>> yearDict = {1990:0, 1991:1, 1995:2}
>>> vocab1990 = {'martin':0, 'pablo':1, 'carlos':2}
>>> vocabularies = [vocab1990]
>>> tempObject = tempName(matrices, yearDict, vocabularies)
>>> newVec = tempObject.getVector('pablo', 1990)
>>> print(newVec)
    [4, 5, 6]
```

getVectorPosNeg(*positives*, *negatives*, *year*)

This function obtains a vector by computing the sum of all the vectorized words in 'positives' and subtracting all vectorized words from 'negatives'. Returning said vector.

Parameters

- **positives** (*array_like*) – All words to add to resulting vector.
- **negatives** (*array_like*) – All words to subtract from resulting vector.
- **year** (*int*) – Chosen year.

Returns **out** – Resulting vector from adding all positives and subtracting all negatives

Return type array_like

Examples

projectMatrices()

Proyects all matrices and vocabularies from oldest to newest. This function progressively adds missing words from the oldest matrices and vocabularies to the newer ones, resulting in a complete vocabulary for the final year and a matrix with all the word's vectorized representations.

Parameters **None** –

Returns **out** – Nothing.

Return type None

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index