03_paragraph_vec_features_demo

April 13, 2022

1 Load training set and train paragraph vectors

Note: the paragraph vector model has been trained and is downloaded in the `prepare_feature_extraction()` function.

Retraining is therefore not needed, but optional

[1]:
```
#%load_ext autoreload
#%autoreload 2

# If you need fully deterministic results between runs, set the following␣
 ↪environment value prior to launching jupyter.
# See comment in sherlock.features.paragraph_vectors.
 ↪infer_paragraph_embeddings_features for more info.
%env PYTHONHASHSEED =13
```

```
env: PYTHONHASHSEED=13
```

[4]:
```
import multiprocessing as mp
import sys

from datetime import datetime

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from pyarrow.parquet import ParquetFile

from sherlock import helpers
from sherlock.features.paragraph_vectors import (
    initialise_nltk,
    tagcol_paragraph_embeddings_features,
    train_paragraph_embeddings_features
)
from sherlock.features.preprocessing import convert_string_lists_to_lists
from sherlock.functional import extract_features_to_csv
```

```
print(f'Started at {datetime.now()}')
```

```
Started at 2022-04-06 14:28:15.058996
```

1.1 Download and read in raw data

[19]:
```
helpers.download_data()
```

```
Downloading the raw data into ../data/data/.
Data was downloaded.
```

[5]:
```
train_samples = pd.read_parquet('../data/data/raw/train_values.parquet')
train_labels = pd.read_parquet('../data/data/raw/train_labels.parquet')
```

[6]:
```
print(train_samples.head(10))
print(train_labels.head(10))
```

```
                                                   values
55030              ['Global', 'United States', 'Australia']
167000  ['Fiction, Adult - Non-Floating', 'Fiction, Ad…
638282  ['', '', 'University of Puerto Rico - Rio Pied…
232298  ['Laughology', 'MTV', 'With Intent to Kill', '…
316158  ['Mare', 'Gelding', 'Gelding', 'Gelding', 'Gel…
467776  ['V.P., General Counsel & Sec.', 'V.P., Genera…
149640  ['GAJA', 'OREG', 'UCS', 'WCM', 'SLAM', 'ARIZ',…
23556   ['Applied Mathematics, University of Notre Dam…
263802  ['wakeup time in seconds for pbid to run its c…
476881     [35.0, 4.0, 52.0, 0.0, 30.0, 64.0, 84.0, None]
                type
55030           area
167000    collection
638282     team Name
232298        credit
316158        gender
467776      position
149640          club
23556    affiliation
263802   description
476881      position
```

[8]:
```
train_samples_converted, y_train = convert_string_lists_to_lists(train_samples,␣
↪train_labels, "values", "type")
```

```
100%|
          | 412059/412059 [00:55<00:00, 7464.94it/s]

types
<class 'pandas.core.series.Series'>
```

```
<class 'list'>
```

[10]:
```
print(train_samples_converted.shape)

print(train_samples_converted.iloc[0], y_train[0])
print(train_samples_converted.iloc[412058], y_train[412058])
```

```
(412059,)
['Global', 'United States', 'Australia'] area
['Norwegian Cod Liver Oil Cherry', 'Norwegian Cod Liver Oil Mint'] product
```

[11]:
```
#print(train_samples_converted.head)

print(train_samples_converted.head(10))
print(y_train[:10])
```

```
55030                    [Global, United States, Australia]
167000    [Fiction, Adult - Non-Floating, Fiction, Adult…
638282    [, , University of Puerto Rico - Rio Piedras, …
232298    [Laughology, MTV, With Intent to Kill, Comedy …
316158    [Mare, Gelding, Gelding, Gelding, Gelding, Mar…
467776    [V.P., General Counsel & Sec., V.P., General C…
149640    [GAJA, OREG, UCS, WCM, SLAM, ARIZ, NEM, VEN, M…
23556     [Applied Mathematics, University of Notre Dame…
263802    [wakeup time in seconds for pbid to run its ch…
476881           [35.0, 4.0, 52.0, 0.0, 30.0, 64.0, 84.0, ]
Name: values, dtype: object
['area', 'collection', 'team Name', 'credit', 'gender', 'position', 'club',
'affiliation', 'description', 'position']
```

1.2 Train Doc2Vec

[19]:
```
initialise_nltk()
```

```
Initialised NLTK, process took 0:00:00.209870 seconds.

[nltk_data] Downloading package punkt to /home/sunny/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/sunny/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[29]:
```
samples = train_samples_converted.dropna()
print(f'Samples: {type(samples)}, length={len(samples)}')

train_labels = train_labels.dropna()

print(f'Labels:  {type(train_labels)}, length={len(train_labels)}')
```

```
#print(train_labels) #df

labels = train_labels.values.flatten()
print(f'Labels:  {type(labels)}, length={len(labels)}')
```

```
Samples: <class 'pandas.core.series.Series'>, length=412059
Labels:  <class 'pandas.core.frame.DataFrame'>, length=412059
Labels:  <class 'numpy.ndarray'>, length=412059
```

[68]:
```
print(samples.head(10))
print(labels[:10])

samples = samples.head(10)
labels = labels[:10]
```

```
55030                    [Global, United States, Australia]
167000    [Fiction, Adult - Non-Floating, Fiction, Adult…
638282    [, , University of Puerto Rico - Rio Piedras, …
232298    [Laughology, MTV, With Intent to Kill, Comedy …
316158    [Mare, Gelding, Gelding, Gelding, Gelding, Mar…
467776    [V.P., General Counsel & Sec., V.P., General C…
149640    [GAJA, OREG, UCS, WCM, SLAM, ARIZ, NEM, VEN, M…
23556     [Applied Mathematics, University of Notre Dame…
263802    [wakeup time in seconds for pbid to run its ch…
476881           [35.0, 4.0, 52.0, 0.0, 30.0, 64.0, 84.0, ]
Name: values, dtype: object
['area' 'collection' 'team Name' 'credit' 'gender' 'position' 'club'
 'affiliation' 'description' 'position']
```

[ ]:
```
'''start = datetime.now()

print('Tagging columns')
cols = tagcol_paragraph_embeddings_features(samples, labels)

#print(cols)
print(f'Tagged Columns Doc2Vec Model, process took {datetime.now() - start}␣
 ↪seconds.')'''
```

[69]:
```
#paragraph_vectors.py
import random; import nltk; from nltk.corpus import stopwords; from gensim.
 ↪models.doc2vec import Doc2Vec, TaggedDocument
STOPWORDS_ENGLISH = stopwords.words("english")

def tokenise(values):
    joined = " ".join(s for s in values if len(s) >= 2)

    # stopwords need apostrophe
```

```
    filtered = "".join(
        e for e in joined if e.isalnum() or e.isspace() or e == "'"
    ).lower()

    return [
        word
        for word in nltk.word_tokenize(filtered)
        if len(word) >= 2 and word not in STOPWORDS_ENGLISH
    ]

def tagcol_paragraph_embeddings_features_nb(train_data: pd.Series, train_labels:
 ↪ list):
    random.seed(13)

    columns = []

    for i, col in enumerate(train_data):
        label = train_labels[i]
        values = random.sample(col, min(1000, len(col)))

        if len(values) > 0:
            values = list(map(lambda s: "" if s is None else str(s), values))

        tokens = tokenise(values)

        columns.append(TaggedDocument(tokens, label))

    return columns
```

[71]:
```
#Tagging

start = datetime.now()

cols = tagcol_paragraph_embeddings_features_nb(samples, labels[:10])

print(cols[0])
print(cols[0].words)
print(cols[0].tags, '\n')

print(f'Tagged Columns Doc2Vec Model, process took {datetime.now() - start}␣
 ↪seconds.', '\n')

print(cols)
```

```
TaggedDocument(['united', 'states', 'australia', 'global'], area)
['united', 'states', 'australia', 'global']
area
```

```
Tagged Columns Doc2Vec Model, process took 0:00:00.004365 seconds.

[TaggedDocument(words=['united', 'states', 'australia', 'global'], tags='area'),
TaggedDocument(words=['fiction', 'adult', 'fiction', 'adult', 'fiction',
'adult', 'nonfloating', 'fiction', 'adult', 'fiction', 'adult'],
tags='collection'), TaggedDocument(words=['university', 'puerto', 'rico', 'rio',
'piedras', 'parck', 'place', 'dealerships', 'sun', 'university', 'puerto',
'ricorio', 'piedras', 'university', 'puerto', 'ricorio', 'piedras', 'park',
'place', 'dealerships', 'university', 'puerto', 'rico', 'rio', 'piedras',
'park', 'place', 'dealerships', 'university', 'puerto', 'ricorio', 'piedras',
'parck', 'place', 'dealerships', 'sun', 'park', 'place', 'dealerships', 'park',
'place', 'dealerships', 'park', 'place', 'dealerships', 'park', 'place',
'dealerships', 'carolina', 'tri', 'university', 'puerto', 'ricorio', 'piedras',
'park', 'place', 'dealerships', 'carolina', 'tri', 'park', 'place',
'dealerships', 'park', 'place', 'dealerships', 'university', 'puerto', 'rico',
'rio', 'piedras', 'university', 'puerto', 'ricorio', 'piedras', 'dallas',
'racing', 'university', 'puerto', 'ricorio', 'piedras', 'parck', 'place',
'dealerships', 'sun'], tags='team Name'), TaggedDocument(words=['intent',
'kill', 'laughology', 'mtv', 'comedy', 'hideaway', 'comedy', 'time', 'radio'],
tags='credit'), TaggedDocument(words=['gelding', 'mare', 'mare', 'colt', 'mare',
'mare', 'gelding', 'mare', 'mare', 'gelding', 'gelding', 'gelding', 'gelding',
'gelding', 'mare', 'gelding', 'mare', 'gelding', 'mare', 'mare', 'filly',
'gelding', 'filly', 'gelding', 'mare', 'mare', 'mare', 'mare', 'gelding',
'gelding', 'mare', 'gelding', 'mare', 'mare', 'gelding', 'mare', 'mare',
'gelding', 'gelding', 'gelding', 'gelding', 'mare', 'mare', 'mare', 'filly',
'gelding', 'gelding', 'gelding', 'mare', 'mare'], tags='gender'),
TaggedDocument(words=['vp', 'general', 'counsel', 'sec', 'vp', 'general',
'counsel', 'sec', 'vp', 'general', 'counsel', 'sec'], tags='position'),
TaggedDocument(words=['vmst', 'wcm', 'ven', 'oreg', 'slam', 'gaja', 'ariz',
'nem', 'ucs', 'mcm'], tags='club'), TaggedDocument(words=['developmental',
'biology', 'stowers', 'institute', 'medical', 'research', 'developmental',
'biology', 'program', 'memorial', 'sloankettering', 'cancer', 'center', 'loci',
'university', 'wisconsin', 'madison', 'engineering', 'auckland',
'bioengineering', 'institute', 'cell', 'biology', 'duke', 'university',
'department', 'mathematics', 'florida', 'state', 'university', 'mathematics',
'statistics', 'smith', 'college', 'systems', 'computational', 'biology', 'lane',
'center', 'computational', 'biology', 'carnegiemellon', 'university',
'electrical', 'computer', 'engineering', 'university', 'houston', 'division',
'mathematics', 'university', 'dundee', 'oncology', 'oxford', 'university',
'mathematics', 'university', 'exeter', 'electrical', 'computer', 'engineering',
'university', 'california', 'santa', 'barbara', 'mathematics', 'university',
'california', 'irvine', 'cell', 'regenerative', 'biology', 'university',
'wisconsin', 'applied', 'mathematics', 'university', 'notre', 'dame', 'tumor',
'cell', 'biology', 'lab', 'london', 'research', 'institute', 'mathematics',
'department', 'university', 'british', 'columbia', 'developmental', 'biology',
'genetics', 'stanford', 'university', 'biology', 'ecology', 'university',
'maine', 'department', 'computer', 'science', 'molecular', 'physiology',
```

```
'biophysics', 'baylor', 'college', 'medicine', 'college', 'life', 'sciences',
'university', 'dundee', 'pathology', 'laboratory', 'medicine', 'university',
'california', 'davis', 'developmental', 'biology', 'sloankettering',
'mathematics', 'statistics', 'boston', 'university', 'lee', 'moffitt', 'cancer',
'center', 'research', 'institute', 'lee', 'moffitt', 'cancer', 'center',
'research', 'institute', 'systems', 'computational', 'biology', 'albert',
'einstein', 'college', 'medicine', 'centre', 'mathematical', 'biology',
'mathematical', 'institute', 'oncology', 'oxford', 'university', 'department',
'cell', 'biology', 'university', 'virginia', 'health', 'system', 'cancer',
'biophyics', 'hubrecht', 'institute', 'university', 'medical', 'center',
'utrecht', 'mathematics', 'case', 'western', 'reserve', 'university',
'developmental', 'biology', 'memorial', 'sloankettering', 'cancer', 'center',
'mathematics', 'statistics', 'georgia', 'state', 'university', 'heart', 'lung',
'research', 'institute', 'davis', 'heart', 'lung', 'research', 'institute',
'department', 'systems', 'biology', 'harvard', 'medical', 'school',
'department', 'anatomy', 'structural', 'biology', 'albert', 'einstein',
'college', 'medicine', 'nuclear', 'dynamics', 'babraham', 'institute',
'imaging', 'stowers', 'institute', 'medical', 'research', 'mathematics',
'university', 'british', 'columbia', 'mathematics', 'university', 'college',
'london', 'molecular', 'genetics', 'ohio', 'state', 'university'],
tags='affiliation'), TaggedDocument(words=['password', 'used', 'authenticate',
'proxy', 'server', 'wakeup', 'time', 'seconds', 'pbid', 'run', 'checks',
'proxy', 'server', 'port', 'number', 'number', 'hours', 'representing', 'often',
'pbid', 'refreshes', 'index', 'meta', 'files', 'repos', 'default', 'every',
'24', 'hours', 'proxy', 'server', 'ip', 'address', 'username', 'used',
'authenticate', 'proxy', 'server', 'http', 'socks5'], tags='description'),
TaggedDocument(words=['840', '640', '40', '520', '00', '300', '350'],
tags='position')]
```

[92]:
```python
# need to save pkl + 3 npy files but missing one npy file

from gensim.models.doc2vec import Doc2Vec, TaggedDocument
import multiprocessing


# Train Doc2Vec model - train_paragraph_embeddings_features()

train_model = Doc2Vec(
    cols,
    dm=0,
    negative=3,
    workers=multiprocessing.cpu_count(),
    vector_size=vec_dim,
    epochs=2,
    min_count=2,
    seed=13,
)
```

```
# Save trained model
model_file = f"../sherlock/features/par_vec_trained_{vec_dim}.pkl"

train_model.save(model_file)
train_model.delete_temporary_training_data(
    keep_doctags_vectors=True, keep_inference=True
)

#print(train_model.docvecs.most_similar(0))
```

[93]:
```
#Training

start = datetime.now()

vec_dim = 400
print(f'Training Doc2Vec model in {vec_dim} dimensions')

train_paragraph_embeddings_features(cols, vec_dim)

print(f'Trained Doc2Vec Model, {vec_dim} dim, process took {datetime.now() -␣
 ↪start} seconds.')
```

```
Training Doc2Vec model in 400 dimensions
Trained Doc2Vec Model, 400 dim, process took 0:00:00.133444 seconds.
```

[74]:
```

```

[89]:
```
doc_words1 = ["last", "Deployment" ,"early" ,"other" ,"the work", "impact",␣
 ↪"receive" ,"Behind the back" ,"Tsukuri", "trick" ,"Every time" ,"thing",␣
 ↪"Take off your hat", "To do", "Read", "Cheap" ,"Me" ,"Mystery"]
doc_words2 = [ "Initiation love", "Similarly" ,"last", "A few lines", "Plot␣
 ↪twist", "Go", "Time", "Time", "various", "scene", "To do" ,"To be",␣
 ↪"Foreshadowing" ,"Sprinkle", " To be" ,"Is", "thing", "notice"]

doc_words3 = ["computer", "it", "science","python", "data", "database"]
doc_words4 = ["python", "data", "database","computer", "it", "science"]

# convert test/unseen paragraph to vector
print(train_model.infer_vector(doc_words1))
```

```
[-1.04569527e-03  2.02358613e-04 -3.77287070e-04 -7.85768207e-04
 -9.25735221e-04 -2.93455407e-04 -4.31377062e-04  5.61634115e-05
  1.11719687e-03  3.19446175e-04  2.16185450e-04  1.10368454e-03
  1.35044073e-04 -4.74251516e-04 -9.38101264e-04 -8.97773134e-05
 -4.60252304e-05 -9.20769380e-05 -1.13637152e-03 -2.06918659e-04
  1.13180722e-03 -1.18021003e-03 -3.15152050e-04 -1.27784051e-05
```

```
-7.88875914e-04  2.74773720e-05 -1.05066271e-03  7.21336517e-04
-7.51639192e-04  8.22321163e-04  2.13560008e-04  5.86735609e-04
-1.33305846e-04  9.03232838e-04 -9.11312178e-04 -6.78860408e-04
 1.10439541e-04 -1.40898352e-04 -8.43701593e-04  8.35611427e-04
-1.21726771e-03  4.59525996e-04 -1.45541664e-04 -5.73182944e-04
-4.41489217e-04  7.74979009e-04  2.59373803e-04 -1.13602646e-03
-1.12703885e-03  1.21715933e-03  1.01167907e-03  7.39598996e-04
-5.02880372e-04 -2.93223711e-04  1.12231693e-03 -2.73212354e-04
 7.65903562e-04 -5.32661215e-04  3.47405963e-04 -2.90148368e-04
-6.29651331e-05 -8.42849724e-04 -1.18696212e-03  4.23624675e-04
 2.30472695e-04  2.14179381e-04 -1.16537977e-03  3.69126850e-04
-1.18487072e-03 -2.17929264e-04 -1.08599046e-03 -1.14708592e-03
 1.75378882e-04 -1.10128615e-03  7.18792377e-04  6.06521033e-04
 5.80867636e-04  5.28467353e-04 -6.80933939e-04  1.20578660e-03
 2.48394033e-04  1.01120735e-03 -5.46673022e-04  2.81009095e-04
 1.15134486e-03 -8.60089145e-04 -1.58941504e-04 -1.03429728e-03
 5.61338558e-04 -5.20080561e-04  1.04506849e-03 -2.38508321e-04
 9.22423787e-04  3.60771111e-04 -1.02617871e-03 -1.15842174e-03
 9.31078161e-04 -1.50128020e-04  1.80522053e-04 -9.41479870e-04
-1.57749717e-04 -1.17634831e-03  1.16310327e-03  6.65736268e-04
-1.22204423e-03 -6.17392187e-04  9.84246843e-04 -1.33654539e-04
 3.03670444e-04 -2.07195801e-04  1.02654868e-03  8.78375256e-04
-9.99493757e-04 -6.60572405e-05  8.57503153e-04  8.71243188e-04
 1.04723230e-03 -1.46265622e-04  2.75952771e-04 -1.10554683e-03
-7.56719557e-04  6.66551816e-04 -2.38844950e-04 -1.12766377e-03
 7.94550462e-04  3.88753338e-04  1.17096177e-03  1.01817551e-03
-5.04618802e-04 -1.12290774e-03  5.48311546e-05 -2.84302314e-05
-9.22885025e-04  6.99591008e-04 -7.25944934e-04 -9.01155232e-04
 2.73266050e-04 -1.15628762e-03 -1.20711059e-03  3.68370325e-04
-3.53883457e-04 -2.33463914e-04  1.14592619e-03 -9.90387285e-04
 1.08020136e-03 -7.64360186e-04 -1.05361547e-03  7.84554519e-04
 3.38253652e-04 -1.24110468e-03 -1.16133620e-03  4.30076936e-04
 4.80189570e-04  3.26515554e-04 -1.06937008e-03 -4.23881516e-04
 1.00753503e-03  2.11788123e-04  1.19360280e-03  8.19464622e-05
-1.22187845e-03 -8.60517612e-04 -7.23242047e-05  4.69033752e-04
-1.24937925e-03 -9.60765057e-04 -7.82008749e-04  7.49604078e-04
-7.31245964e-04  2.21471288e-04  6.91586407e-04 -1.88871840e-04
 6.61762140e-04  9.41883365e-04 -8.11732432e-04 -1.09076733e-03
 3.09664232e-04 -4.25946637e-04  4.11291228e-04  5.25962096e-04
-3.51283757e-04 -8.31304875e-04  1.22661109e-03  5.97133534e-04
-1.11727382e-03 -2.79017666e-04 -2.22787610e-04 -3.95307434e-04
-8.66050192e-04  3.47964582e-04 -2.56497500e-04  2.42639770e-04
 5.77456201e-04  1.01151201e-03 -4.95793764e-04  7.21301476e-04
-4.46542217e-05  4.04024941e-05 -2.47449061e-04  2.13502004e-04
 5.45262475e-04 -2.49897159e-04 -9.65535291e-04 -6.46600442e-04
 1.04889635e-03 -1.17487507e-04 -9.46517277e-04 -5.13180741e-04
 7.83555734e-05 -3.22283071e-04 -6.09494047e-04  1.05427217e-03
 1.75497218e-04  1.06620020e-03 -4.70531260e-04 -8.04665207e-04
```

```
-1.17619184e-03 -8.34473409e-04  1.01906795e-03 -6.51090886e-05
 4.23366379e-04  1.18884153e-03  1.17380137e-03  1.21293240e-03
 5.71998535e-04 -7.71814724e-04  7.16363429e-04  6.93277107e-04
-1.20434072e-03 -1.24604523e-03  1.58911978e-04 -2.55777759e-05
-9.40238649e-04  7.25976715e-05 -7.33928813e-04 -1.95505258e-04
 2.65310140e-04  9.92397545e-04 -1.07603124e-03  7.40334217e-04
-3.34817523e-05  1.19349884e-03 -1.19446230e-03  1.13862252e-03
 2.71461613e-04  2.23874158e-04 -1.12078467e-03  5.75024111e-04
 1.03174686e-03  1.15992152e-03  3.90524074e-04 -5.45238254e-05
 5.14553511e-04  1.10064016e-03 -3.62039515e-04 -1.69803563e-04
 1.24232355e-03  2.66998133e-04  3.36352881e-04  8.98102880e-04
-1.13353727e-03 -1.70447936e-04  4.41005686e-04 -7.31521111e-04
 9.83413542e-04  3.23517626e-04  6.32272568e-04  7.15112721e-04
-4.78403235e-04  3.84561863e-04  8.11340346e-04  5.60958055e-04
 1.02557102e-03  7.92041305e-04  2.17235298e-04  4.15863004e-04
-9.48385394e-04 -8.51605786e-04  1.25211984e-04  4.58416966e-04
 4.41806827e-04 -6.38347701e-04  1.75363384e-04  1.21346954e-03
-8.93431366e-04  9.82510741e-04  1.09821989e-03  3.89102097e-05
 3.66752647e-04 -1.20023347e-03 -6.17260521e-05 -2.65721319e-04
-2.76242730e-07  7.91242521e-04  9.32318741e-04  5.39786706e-04
-2.73562237e-05  9.74806841e-04 -1.02668838e-03  9.70886380e-04
 1.10011338e-03 -1.04057987e-03 -7.60799274e-04  1.67967723e-04
-1.22770245e-04  5.51629229e-04 -1.17288867e-03 -1.19527092e-03
-6.35313743e-04 -1.02347077e-03  3.80391197e-04 -5.93543227e-04
 8.36609281e-04 -1.08183164e-03 -2.75870570e-05 -8.79276718e-04
-7.85536322e-05 -6.22805528e-05  1.08298566e-03  4.33517445e-04
-8.30268080e-04 -5.14091698e-05 -8.09507561e-04 -2.47146789e-04
-4.55400470e-04  3.97917640e-04 -6.36709330e-04 -1.20495561e-05
 4.48801438e-04 -2.58375949e-04  1.17038086e-03  8.70009942e-04
-6.18262915e-04 -4.82269999e-04 -9.84001090e-04  2.17879264e-04
 1.22700282e-03  3.16997146e-04  7.43419689e-04  6.48794754e-04
 8.04923999e-04  3.98776057e-04 -8.13062215e-05 -9.97240422e-04
-1.16917712e-03  1.02603971e-03  3.46297864e-04 -5.26604534e-04
-1.22477137e-03 -4.73621330e-04  1.15760521e-03 -1.10240874e-03
-3.55270517e-04  5.62618370e-04  2.79903179e-04 -4.77300346e-04
 6.19594357e-04  1.03359076e-03  7.27134990e-04  3.53988493e-04
 8.83608765e-04 -8.08440789e-04 -5.17039094e-04 -4.30083834e-04
-1.14987185e-03 -3.06922942e-04  6.13877055e-05  5.10488389e-05
-1.21016354e-04  1.26709201e-04  7.93660351e-04 -3.42991465e-04
-5.70271339e-04 -4.55208792e-04 -1.61173579e-04 -1.37950396e-04
-2.01515577e-04  1.02058018e-03 -1.01951162e-04 -7.26233062e-04
 1.08152430e-03 -4.90243954e-04 -5.04067575e-04  1.32226167e-04
-6.43006817e-04  6.25655870e-04 -6.12331613e-04  6.30986935e-04
-1.15769519e-03  1.11467729e-03 -4.47940518e-04 -5.72708610e-04
-1.36314600e-04 -6.60063815e-04 -9.67860746e-04 -3.65665415e-04
 7.58290291e-04  5.38481727e-05 -5.54165104e-04  1.12028455e-03]
```

[ ]:
```
# inference reference in paragraph_vectors.py

def infer_paragraph_embeddings_features(
    col_values: list, features: OrderedDict, dim, reuse_model
):
    if not reuse_model or model is None:
        # Load pretrained paragraph vector model
        initialise_pretrained_model(dim)

    # Resetting the random seed before inference keeps the inference vectors␣
↪deterministic. Gensim uses random values
    # in the inference process, so setting the seed just beforehand makes the␣
↪inference repeatable.
    # https://github.com/RaRe-Technologies/gensim/issues/447

    # To make the inference repeatable across runtime launches, we also need to␣
↪set PYTHONHASHSEED
    # prior to launching the execution environment (i.e. jupyter notebook).  E.
↪g. export PYTHONHASHSEED=13
    # See above Github thread for more information.
    model.random.seed(13)

    tokens = tokenise(col_values)

    # Infer paragraph vector for data sample.
    inferred = model.infer_vector(tokens, steps=20, alpha=0.025)

    if is_first():
        # the first output needs fully expanded keys (to drive CSV header)
        for idx, value in enumerate(inferred):
            features["par_vec_" + str(idx)] = value
    else:
        # subsequent lines only care about values, so we can pre-render a block␣
↪of CSV. This
        # cuts overhead of storing granular values in the features dictionary
        features["par_vec-pre-rendered"] = ",".join(map(lambda x: "%g" % x,␣
↪inferred))
```

[ ]:

[ ]:

[108]:
```
#Simpler version

from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
```

```
data = ["I love machine learning. Its awesome.",
        "I love coding in python",
        "I love building chatbots",
        "they chat amagingly well"]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)])␣
 ↪for i, _d in enumerate(data)]



max_epochs = 100
vec_size = 20
alpha = 0.025

model = Doc2Vec(size=vec_size,
                alpha=alpha,
                min_alpha=0.00025,
                min_count=1,
                dm =1)

model.build_vocab(tagged_data)

for epoch in range(max_epochs):
    print('iteration {0}'.format(epoch))
    model.train(tagged_data,
                total_examples=model.corpus_count,
                epochs=model.iter)
    # decrease the learning rate
    model.alpha -= 0.0002
    # fix the learning rate, no decay
    model.min_alpha = model.alpha


#model_file = f"../sherlock/features/par_vec_trained_{vec_dim}.pkl"
model.save("d2v.model")
print("Model Saved")
```

```
/home/sunny/miniconda3/envs/env5/lib/python3.7/site-
packages/gensim/models/doc2vec.py:574: UserWarning: The parameter `size` is
deprecated, will be removed in 4.0.0, use `vector_size` instead.
  warnings.warn("The parameter `size` is deprecated, will be removed in 4.0.0,
use `vector_size` instead.")
/home/sunny/miniconda3/envs/env5/lib/python3.7/site-
packages/ipykernel_launcher.py:29: DeprecationWarning: Call to deprecated `iter`
(Attribute will be removed in 4.0.0, use self.epochs instead).

iteration 0
iteration 1
```

```
iteration 2
iteration 3
iteration 4
iteration 5
iteration 6
iteration 7
iteration 8
iteration 9
iteration 10
iteration 11
iteration 12
iteration 13
iteration 14
iteration 15
iteration 16
iteration 17
iteration 18
iteration 19
iteration 20
iteration 21
iteration 22
iteration 23
iteration 24
iteration 25
iteration 26
iteration 27
iteration 28
iteration 29
iteration 30
iteration 31
iteration 32
iteration 33
iteration 34
iteration 35
iteration 36
iteration 37
iteration 38
iteration 39
iteration 40
iteration 41
iteration 42
iteration 43
iteration 44
iteration 45
iteration 46
iteration 47
iteration 48
iteration 49
```

```
iteration 50
iteration 51
iteration 52
iteration 53
iteration 54
iteration 55
iteration 56
iteration 57
iteration 58
iteration 59
iteration 60
iteration 61
iteration 62
iteration 63
iteration 64
iteration 65
iteration 66
iteration 67
iteration 68
iteration 69
iteration 70
iteration 71
iteration 72
iteration 73
iteration 74
iteration 75
iteration 76
iteration 77
iteration 78
iteration 79
iteration 80
iteration 81
iteration 82
iteration 83
iteration 84
iteration 85
iteration 86
iteration 87
iteration 88
iteration 89
iteration 90
iteration 91
iteration 92
iteration 93
iteration 94
iteration 95
iteration 96
iteration 97
```

```
iteration 98
iteration 99
Model Saved
```

[107]:
```
from gensim.models.doc2vec import Doc2Vec

model= Doc2Vec.load("d2v.model")


#to find the vector of a document which is not in training data
test_data = word_tokenize("I love chatbots".lower())
v1 = model.infer_vector(test_data)
print("V1_infer", v1)

# to find most similar doc using tags
similar_doc = model.docvecs.most_similar('1')
print(similar_doc)


# to find vector of doc in training data using tags or in other words, printing␣
 ↪the vector of document at index 1 in training data
print(model.docvecs['1'])
```

```
V1_infer [-0.01636579  0.01370141  0.00942792  0.03040449  0.01499906
-0.01971995
 -0.02642728 -0.00770284  0.00807515  0.01649123  0.00551462 -0.0012941
  0.02232379  0.00538251 -0.00047591  0.01730111 -0.01723027  0.02204072
 -0.01876928  0.01507904]
[('0', 0.9922033548355103), ('3', 0.9920569062232971), ('2',
0.9906700849533081)]
[-0.48180726  0.36426634 -0.28672537  0.37891588 -0.12584305 -0.16265571
 -0.2606042   0.03638057 -0.06310781 -0.07571291  0.19111769 -0.20922482
  0.42656934  0.08407371 -0.10172732  0.34621498  0.14652233  0.02171521
 -0.29588342 -0.26338166]
```

[105]:
```
#Misc
sim_value =train_model.docvecs.similarity_unseen_docs(train_model, doc_words1,␣
 ↪doc_words4, alpha=1, min_alpha=0.0001, steps=5)
print(sim_value)

sim_value =train_model.docvecs.similarity_unseen_docs(train_model, doc_words3,␣
 ↪doc_words4, alpha=1, min_alpha=0.0001, steps=5)
print(sim_value, '\n')


# sherlock tagging
print(train_model)
```

```
print(train_model.docvecs.most_similar(0), '\n')#tagging res is weird.. could␣
 ↪be syntax problem here?

# library tagging
print(model)
print(model.docvecs.most_similar('0'))


#also not an ideal result
#print(model.most_similar(positive=['woman', 'king'], negative=['man']))
#print(model.most_similar(positive=['country']))
```

```
-0.04946377
0.5973945

Doc2Vec(dbow,d400,n3,mc2,s0.001,t8)
[('i', 0.8961265087127686), ('n', 0.7977637052536011), ('f',
0.7976861000061035), ('o', 0.7767428159713745), ('t', 0.7520517110824585), ('l',
0.6260175704956055), ('e', 0.5449905395507812), ('m', 0.3726497292518616), ('c',
0.37263673543930054), ('r', 0.3354871869087219)]

Doc2Vec(dm/m,d20,n5,w5,s0.001,t3)
[('3', 0.9951073527336121), ('1', 0.9922033548355103), ('2',
0.9865337610244751)]
```