NLP_lab-4_mahalakshmi18

March 23, 2021

0.0.1 Lab4. Computing Document Similarity using Doc2Vec Model

0.0.2 EXERCISE-1

0.0.3 1. Import dependencies

[3]:
```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
from sklearn import utils
```

0.0.4 2. Create dataset

[5]:
```
data = ["I love machine learning. Its awesome.",
 "I love coding in python",
 "I love building chatbots",
 "they chat amagingly well"]
```

[8]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[8]:
```
True
```

0.0.5 3. Create TaggedDocument

[11]:
```
tagged_data = [TaggedDocument(words=word_tokenize(d.lower()),
tags=[str(i)]) for i, d in enumerate(data)]
```

0.0.6 4. Train Model

[12]:
```
vec_size = 20
alpha = 0.025
```

[14]:
```
model = Doc2Vec(vector_size=vec_size,
 alpha=alpha,
 min_alpha=0.00025,
 min_count=1,
 dm =1)
# build vocabulary
```

```
model.build_vocab(tagged_data)
# shuffle data
tagged_data = utils.shuffle(tagged_data)
# train Doc2Vec model
model.train(tagged_data,
 total_examples=model.corpus_count,
 epochs=30)
model.save("d2v.model")
print("Model Saved")
```

```
Model Saved
```

0.0.7 5. Find Similar documents for the given document

[15]:
```
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)
```

```
V1_infer [-0.00869065 -0.01238877  0.01076634  0.01837858 -0.01185473  0.0152639
  0.00312291  0.00715544 -0.01393549 -0.02041687  0.00920063 -0.0127892
 -0.00577795 -0.00978915 -0.00954527  0.00275575  0.0105227  -0.01635379
  0.02378107  0.00102777]
```

[16]:
```
similar_doc = model.docvecs.most_similar('1')
print(similar_doc)
```

```
[('3', 0.2867772579193115), ('0', 0.14935939013957977), ('2',
-0.17948105931282043)]
```

[18]:
```
print(model.docvecs['1'])
```

```
[ 4.4631111e-03  2.0471280e-02  1.2875644e-02  2.3986014e-02
 -8.5967574e-03  7.7667716e-03  8.9874053e-03  2.4314741e-02
  1.1871044e-02 -2.1266585e-02 -2.6052014e-03 -5.8996924e-03
  1.7793141e-02 -5.5611712e-05  1.4491647e-02  7.6084463e-03
  1.8754840e-03  2.2718612e-02 -1.9271666e-02  1.0655354e-02]
```

0.0.8 EXERCISE-2

0.0.9 Question1. Train the following documents using Doc2Vec model

[39]:
```
docs=["the house had a tiny little mouse",
"the cat saw the mouse",
"the mouse ran away from the house",
"the cat finally ate the mouse",
"the end of the mouse story"
]
```

[40]:
```
tagged_data = [TaggedDocument(words=word_tokenize(d.lower()),
tags=[str(i)]) for i, d in enumerate(docs)]
```

[41]:
```
vec_size = 20
alpha = 0.025
# create model
model = Doc2Vec(vector_size=vec_size, alpha=alpha, min_alpha=0.
 ↪00025,min_count=1,dm =1)
```

[42]:
```
model.build_vocab(tagged_data)
```

[43]:
```
tagged_data = utils.shuffle(tagged_data)
```

[44]:
```
model.train(tagged_data,total_examples=model.corpus_count,epochs=30)
model.save("d2v.model")
print("Model Saved")
```

```
Model Saved
```

0.0.10 Question2. Find the most similar TWO documents for the query document "cat stayed in the house".

[45]:
```
from gensim.models.doc2vec import Doc2Vec
model= Doc2Vec.load("d2v.model")
```

[46]:
```
test_data = word_tokenize("cat stayed in the house".lower())
v1 = model.infer_vector(test_data)
print("V1_infer", v1)
```

```
V1_infer [-0.01994306 -0.00365496 -0.02367386 -0.00445433  0.00579851
0.01591771
 -0.01106277  0.00164189 -0.00461278 -0.00962676  0.00456066 -0.02325257
  0.01938994 -0.00737341 -0.02321905 -0.02015355 -0.0067498  -0.00298602
  0.00556086 -0.00357911]
```

[47]:
```
similar_doc = model.docvecs.most_similar('2')
print(similar_doc)
```

```
[('3', 0.0957995355129242), ('0', 0.0746668130159378), ('4',
0.0397355742752552), ('1', -0.15176615118980408)]
```