DOC2VEC gensim tutorial

Deepak Mishra Mar 17, 2018 · 2 min read

Today I am going to demonstrate a simple implementation of nlp and doc2vec. The idea is to implement **doc2vec model training and testing** using **gensim 3.4** and **python3.** The new updates in gensim makes the implemention of doc2vec easier.

Let's start implementing

```
#Import all the dependencies
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
```

Let's prepare data for training our doc2vec model

```
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)]
```

Here we have a list of four sentences as training data. Now I have tagged the data and its ready for training. Lets start training our model.

```
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.save("d2v.model")
print("Model Saved")
```

Note: **dm defines the training algorithm. If dm=1 means 'distributed memory' (PV-DM)** ***and dm =0 means*** **'distributed bag of words' (PV-DBOW). Distributed Memory model preserves the word order in a document whereas Distributed Bag of words just uses the bag of words approach, which doesn't preserve any word order.**

So we have saved the model and it's ready for implementation. Lets play with it.

```
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'])
```

So doc2vec is as simple as this. Hope you guys liked it. Feel free to comment . Here is link to my blog for older version of gensim, you guys can also view that. https://medium.com/@mishra.thedeepak/doc2vec-in-a-simple-way-fa80bfe81104

Doc2vec Gensim NLP Python Deep Learning