Advanced NLP

August 16, 2024

[2]:
```
# import necessary libraries
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[2]:
```
True
```

[3]:
```
# sample text data
corpus =[
    'word embeddings are essential for natural language processing tasks.',
    'They capture semantic relationship between words',
    'Word2Vec is a popular algorithm for learning word embeddings'
]
```

[4]:
```
# Tokenize the text data
tokenized_corpus = [word_tokenize(sentence.lower()) for sentence in corpus]
print(tokenized_corpus)
```

```
[['word', 'embeddings', 'are', 'essential', 'for', 'natural', 'language',
'processing', 'tasks', '.'], ['they', 'capture', 'semantic', 'relationship',
'between', 'words'], ['word2vec', 'is', 'a', 'popular', 'algorithm', 'for',
'learning', 'word', 'embeddings']]
```

[5]:
```
# word2vec model training (using skip-gram as an example)
model = Word2Vec(sentences = tokenized_corpus,
                 vector_size = 100, window = 5, sg = 1, min_count = 1)

# save the trained model
model.save("Word2Vecmodel.model")
```

[6]:
```
# get the learned word embeddings
word_embeddings = model.wv

# use the learned word embeddings
```

```
similar_words = word_embeddings.most_similar("word", topn =  3)

print("words similar to 'word':", similar_words)
```

```
words similar to 'word': [('words', 0.21889080107212067), ('tasks',
0.21592096984386444), ('capture', 0.0934029147028923)]
```

[7]:
```
# NLTK library "reuters" dataset
from nltk.corpus import reuters
```

[8]:
```
# Download the reuters dataset if not available
data = nltk.download("reuters")
print(data)
```

```
True
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data…
[nltk_data]   Package reuters is already up-to-date!
```

[9]:
```
# Load and tokenize the reuters dataset
tokenized_sentences = reuters.sents()

# word2vec model training (using skip-gram)
model = Word2Vec(sentences=tokenized_sentences,
                 vector_size=100,  # Corrected from 'windows' to 'window'
                 window=5,
                 sg=1,
                 min_count=1)
# Save the trained model
model.save("Word2Vec_reuters_model.model")
```

[10]:
```
# Get the learned word embeddings
word_embeddings = model.wv

# Using the learned word embeddings
similar_words = word_embeddings.most_similar('market', topn =3)

print("words similar to 'market':", similar_words)
```

```
words similar to 'market': [('markets', 0.7742810845375061), ('buying',
0.6610605716705322), ('exchanges', 0.6609679460525513)]
```

Ruth.O.Ajagunna

[ ]:
```

```