COMP 6721 Applied Artificial Intelligence (Summer 2023)

Assignment #11: NLP, VSM & NER

Solutions

Question 1 Our first exercise is to convert natural language text into a vector representation that we can later use for further processing, for example, as input to a machine learning algorithm.

(a) Write a Python program that computes the binary sentence vectors for the three sentences: *"the big dog", "the big cat", "the big cat and dog".*

To compute the vectors:

```
import pandas as pd

sentences = """the big dog
the big cat
the big cat and dog"""
corpus = {}

for i, sent in enumerate(sentences.split('\n')):
   corpus['sent{}'.format(i)] = dict((tok, 1) for tok in
        sent.split())
df = pd.DataFrame.from_records(corpus).fillna(0).astype(int).T
print(df)
```

which will result in:

```
       and  big  cat  dog  the
sent0    0    1    0    1    1
sent1    0    1    1    0    1
sent2    1    1    1    1    1
```

(b) Now compute the dot product between sentence 0 and sentences 1 and 2.

You can print out the dot product with:

```
print(df.sent0.dot(df.sent1))
print(df.sent0.dot(df.sent2))
```

(c) What is the meaning of the dot product in case of binary vectors?

Since each word is only represented by 0 or 1, the vector dot product between two sentences tells us how many words they have in common.

Question 2 Instead of binary vectors, we will now compute tf-idf vectors.

(a) In Lab #4, you already used scikit-learn's `CountVectorizer()`. Modify your program to compute tf-idf vectors using the `TfidfTransformer` on the created count vectors.[1] Print out the tf-idf vectors, as well as the df values for each word in the corpus. To validate the calculations, make sure you understand what settings are being used by default (see the documentation on the parameters regarding idf reweighting, smoothing, and sublinear tf scaling).

What you will need to add after vectorizing the corpus is:

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer(sublinear_tf=True, use_idf=True)
tfidf_vectors = tfidf_transformer.fit_transform(X)
print(tfidf_vectors.toarray())
```

To print out the df values for each word:

```
import pandas as pd

print(pd.DataFrame(tfidf_transformer.idf_,
              index=vectorizer.get_feature_names(),
              columns=["idf_weights"]))
```

(b) Now compute the *cosine similarity* matrix between the documents and print it out.[2]

To compute the cosine similarity, use:

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_scores = cosine_similarity(tfidf_vectors)
print(similarity_scores)
```

which should give you a result like:

```
[[1.         0.29521695 0.         0.65997508 0.14760847]
 [0.29521695 1.         0.         0.12238056 0.41686221]
 [0.         0.         1.         0.16026063 0.13988101]
 [0.65997508 0.12238056 0.16026063 1.         0.12238056]
 [0.14760847 0.41686221 0.13988101 0.12238056 1.        ]]
```

[1] See https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer

[2] See https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity

Question 3 Let's now try to use the Python spaCy NLP library[3] for *Named Entity Recognition* (NER). NER is the task of identifying key information/entities in texts. Here, entities can be a single token or span a series of tokens. Every detected entity is then classified into a predetermined category (e.g., *Person, Organization, Date*). NER is an important step for various NLP tasks, for example, when you want extract the main subjects or concepts from text passages or get an overall idea of what a text is talking about.

In spaCy, named entities are available through the `ents` property of a `doc` object. Try the following code to extract named entities and some of their properties:

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for ent in doc.ents:
   print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

(a) Try playing a little with different sentences and visualize what labels are trained to be predicted by spaCy.

PERSON, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK_OF_ART, LAW, LANGUAGE, DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL

(b) Use *displaCy* to visualize the identified entities.

```
displacy.serve(doc, style='ent')
```

(c) Now try restricting your visualizer to only display specific categories of named entities for the following sentence. First try to visualize all the named entities to make sure you are only visualising the ones you wanted to see, e.g., only entities of type *Organization*:

```
text = "European authorities fined Google a record $5.1 billion on
Wednesday for abusing its power in the mobile phone market and ordered
the company to alter its practices."
```

You can control this with the `options` parameter:

```
options = {"ents": ["ORG"]}
displacy.serve(doc, style='ent', options=options )
```

[3] See https://spacy.io/