Tf-idf with gensim

Puteaux, Fall/Winter 2020-2021

```
##############################################
##                                          ##
##  Natural Language Processing in Python   ##
##                                          ##
##############################################
```

```
§1 Introduction to Natural Language Processing in Python
```

```
§1.2 Simple topic identification
```

1 Tf-idf with gensim

1.1 What is tf-idf?

- Tf-idf means term frequency - inverse document frequency.
- Allow determining the most important words in each document.
- Each corpus may have shared words beyond just stopwords.
- These words should be down-weighted in importance.
- Example:
  - *"sky" from the theme of astronomy*
- Ensures most common words don't show up as keywords.
- Keep document specific frequent words weighted high.

1.2 What is the tf-idf formula?

- $w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$
  - $w_{i,j}$ = tf-idf weight for token i in document j
  - $tf_{i,j}$ = number of occurences of token i in document j
  - df_i = number of documents that contain token i
  - N = total number of documents

1.3 Code of tf-idf with gensim:

[1]:
```
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize

my_documents = [
    'The movie was about a spaceship and aliens.',
    'I really liked the movie!',
    'Awesome action scenes, but boring characters.',
    'The movie was awful! I hate alien films.',
    'Space is cool! I liked the movie.',
    'More space films, please!',
]

tokenized_docs = [word_tokenize(doc.lower()) for doc in my_documents]
dictionary = Dictionary(tokenized_docs)
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
```

[2]:
```
from gensim.models.tfidfmodel import TfidfModel

tfidf = TfidfModel(corpus)
tfidf[corpus[1]]
```

[2]:
```
[(5, 0.1746298276735174),
 (7, 0.1746298276735174),
 (9, 0.1746298276735174),
 (10, 0.29853166221463673),
 (11, 0.47316148988815415),
 (12, 0.7716931521027908)]
```

1.4 Practice question for what is tf-idf:

- To calculate the tf-idf weight for the word “computer”, which appears five times in a document containing 100 words. Given a corpus containing 200 documents, with 20 documents mentioning the word “computer”, so tf-idf can be calculated by multiplying term frequency with inverse document frequency.
- Notes:
  - term frequency = percentage share of the word compared to all tokens in the document
  - inverse document frequency = logarithm of the total number of documents in a corpus divided by the number of documents containing the term
- Which of the below options is correct?

  ☒ (5 / 100) * log(200 / 20)

  ☐ (5 * 100) / log(200 * 20)

  ☐ (20 / 5) * log(200 / 20)

☐ (200 * 5) * log(400 / 5)

1.5 Practice exercises for tf-idf with gensim:

▶ **Package pre-loading:**

[3]:
```
import zipfile

from nltk import word_tokenize
from nltk.corpus import stopwords

from gensim.corpora.dictionary import Dictionary
from gensim.models.tfidfmodel import TfidfModel
```

▶ **Data pre-loading:**

[4]:
```
file_name = 'ref3. Wikipedia articles.zip'
with zipfile.ZipFile(file_name, 'r') as archive:
    files = [
        archive.read(name) for name in archive.namelist()
        if name.endswith('.txt')
    ]

doc_tokens = [word_tokenize(file.decode("utf-8")) for file in files]
articles = []
english_stops = stopwords.words('english')
for i in range(len(doc_tokens)):
    lower_tokens = [t.lower() for t in doc_tokens[i]]
    alphanumeric_only = [t for t in lower_tokens if t.isalnum()]
    no_stops = [t for t in alphanumeric_only if t not in english_stops]
    articles.append(no_stops)

dictionary = Dictionary(articles)
corpus = [dictionary.doc2bow(article) for article in articles]

doc = corpus[4]
```

▶ **Wikipedia tf-idf practice:

[5]:
```
# Create a new TfidfModel using the corpus: tfidf
tfidf = TfidfModel(corpus)

# Calculate the tfidf weights of doc: tfidf_weights
tfidf_weights = tfidf[doc]

# Print the first five weights
print(tfidf_weights[:5])
```

```
[(13, 0.021411676334320492), (24, 0.01738903055915624), (43,
```

```
0.00805356588388867), (45, 0.021821227698039212), (50, 0.01376766181415054)]
```

[6]:
```
# Create a new TfidfModel using the corpus: tfidf
tfidf = TfidfModel(corpus)

# Calculate the tfidf weights of doc: tfidf_weights
tfidf_weights = tfidf[doc]

# Print the first five weights
print(tfidf_weights[:5])

# Sort the weights from highest to lowest: sorted_tfidf_weights
sorted_tfidf_weights = sorted(tfidf_weights, key=lambda w: w[1], reverse=True)

# Print the top 5 weighted words
for term_id, weight in sorted_tfidf_weights[:5]:
    print(dictionary.get(term_id), weight)
```

```
[(13, 0.021411676334320492), (24, 0.01738903055915624), (43,
0.00805356588388867), (45, 0.021821227698039212), (50, 0.01376766181415054)]
compiled 0.2182122769803921
compilation 0.21353333707313848
eiffel 0.17794444756094874
abstraction 0.1745698215843137
intermediate 0.16521194176980647
```