# Deep Neural Networks for Natural Language Processing (AI6127)

JUNG-JAE KIM

TUTORIAL 1: TF-IDF

# Question 1: Consider these documents

- Doc1 breakthrough drug for schizophrenia
- Doc2 new schizophrenia drug
- Doc3 new approach for treatment of schizophrenia
- Doc4 new hopes for schizophrenia patients

- Draw a term-document count matrix for the document collection
- Calculate the inverse document frequencies of the terms in the document collection
- Draw a term-document TF-IDF weighted matrix for the document collection
- Find the document in the collection that is the closest to the new document "schizophrenia drug" by using cosine similarity between the TF-IDF vectors of the documents

Recall

# Term-document count matrices

- Consider the number of occurrences of a word in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 157 | 73 | 0 | 0 | 0 | 0 |
| Brutus | 4 | 157 | 0 | 1 | 0 | 0 |
| Caesar | 232 | 227 | 0 | 2 | 1 | 1 |
| Calpurnia | 0 | 10 | 0 | 0 | 0 | 0 |
| Cleopatra | 57 | 0 | 0 | 0 | 0 | 0 |
| mercy | 2 | 0 | 3 | 5 | 5 | 1 |
| worser | 2 | 0 | 1 | 1 | 1 | 0 |



Source: IR

Recall

# Inverse document frequency (idf)

- Rare terms are more informative than frequent terms
  - Stop words (e.g. the, of)
- $\mathrm{df}_t$ is the document frequency of $t$: the number of documents in training data that contain $t$
  - $\mathrm{df}_t$ is an inverse measure of the informativeness of $t$
  - $\mathrm{df}_t \leq N$
- We define the idf (inverse document frequency) of $t$ by

$$\mathrm{idf}_t = \log_{10}\,(N/\mathrm{df}_t)$$

  - We use log ($N/\mathrm{df}_t$) instead of $N/\mathrm{df}_t$ to "dampen" the effect of idf.

The base of the log is not important.



Source: IR

Recall

# tf-idf vector

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection



Source: IR

Recall

# Document similarity: Cosine similarity

- E.g. Find the document (d) in given collection that is the closest to a new document (q)
  - $\vec{d}$ and $\vec{q}$ are TF-IDF vectors

Dot product

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Unit vectors

# Question 2: Read and understand the example implementation of sentiment analysis of movie reviews by using binary or TF-IDF vectors

- https://drive.google.com/file/d/1DyDKHSg6yTpDNQgv5bMfZKi-VDvCHVkL/view?usp=sharing

# Hands-on

# Answer 1: Draw a term-document count matrix for the document collection

| | Doc1 | Doc2 | Doc3 | Doc4 |
|---|---|---|---|---|
| Approach | 0 | 0 | 1 | 0 |
| Breakthrough | 1 | 0 | 0 | 0 |
| Drug | 1 | 1 | 0 | 0 |
| For | 1 | 0 | 1 | 1 |
| Hopes | 0 | 0 | 0 | 1 |
| New | 0 | 1 | 1 | 1 |
| Of | 0 | 0 | 1 | 0 |
| Patients | 0 | 0 | 0 | 1 |
| Schirophrenia | 1 | 1 | 1 | 1 |
| treatment | 0 | 0 | 1 | 0 |

Doc1 breakthrough drug for schizophrenia
Doc2 new schizophrenia drug
Doc3 new approach for treatment of schizophrenia
Doc4 new hopes for schizophrenia patients

# Answer 1: Calculate the inverse document frequencies of the terms in the document collection

| | DF | IDF |
|---|---|---|
| Approach | 1 | 0.60 |
| Breakthrough | 1 | 0.60 |
| Drug | 2 | 0.30 |
| For | 3 | 0.12 |
| Hopes | 1 | 0.60 |
| New | 3 | 0.12 |
| Of | 1 | 0.60 |
| Patients | 1 | 0.60 |
| Schirophrenia | 4 | 0.00 |
| treatment | 1 | 0.60 |

Doc1 breakthrough drug for schizophrenia
Doc2 new schizophrenia drug
Doc3 new approach for treatment of schizophrenia
Doc4 new hopes for schizophrenia patients

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

# Answer 1: Draw a term-document TF-IDF weighted matrix for the document collection

| | Doc1 | Doc2 | Doc3 | Doc4 |
|---|---|---|---|---|
| Approach | 0 | 0 | 0.60 | 0 |
| Breakthrough | 0.60 | 0 | 0 | 0 |
| Drug | 0.30 | 0.30 | 0 | 0 |
| For | 0.12 | 0 | 0.12 | 0.12 |
| Hopes | 0 | 0 | 0 | 0.60 |
| New | 0 | 0.12 | 0.12 | 0.12 |
| Of | 0 | 0 | 0.60 | 0 |
| Patients | 0 | 0 | 0 | 0.60 |
| Schirophrenia | 0.00 | 0.00 | 0.00 | 0.00 |
| treatment | 0 | 0 | 0.60 | 0 |

$$\mathrm{w}_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

# Answer 1: Find the document in the collection that is the closest to the new document “schizophrenia drug”

| | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| Approach | 0 | 0 | 0.60 | 0 | 0 |
| Breakthrough | 0.60 | 0 | 0 | 0 | 0 |
| Drug | 0.30 | 0.30 | 0 | 0 | 0.30 |
| For | 0.12 | 0 | 0.12 | 0.12 | 0 |
| Hopes | 0 | 0 | 0 | 0.60 | 0 |
| New | 0 | 0.12 | 0.12 | 0.12 | 0 |
| Of | 0 | 0 | 0.60 | 0 | 0 |
| Patients | 0 | 0 | 0 | 0.60 | 0 |
| Schirophrenia | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| treatment | 0 | 0 | 0.60 | 0 | 0 |
| **Length** | **0.68** | **0.32** | **1.05** | **0.87** | **0.30** |

| | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| Approach | 0 | 0 | 0.57 | 0 | 0 |
| Breakthrough | 0.88 | 0 | 0 | 0 | 0 |
| Drug | 0.44 | 0.93 | 0 | 0 | 1 |
| For | 0.18 | 0 | 0.11 | 0.14 | 0 |
| Hopes | 0 | 0 | 0 | 0.69 | 0 |
| New | 0 | 0.37 | 0.11 | 0.14 | 0 |
| Of | 0 | 0 | 0.57 | 0 | 0 |
| Patients | 0 | 0 | 0 | 0.69 | 0 |
| Schirophrenia | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| treatment | 0 | 0 | 0.57 | 0 | 0 |

# Answer 1: Find the document in the collection that is the closest to the new document “schizophrenia drug”

cos(Doc2, Doc5) = 0.93

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|}$$

$$= \frac{\sum_{i=1}^{|V|} q_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

| | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| Approach | 0 | 0 | 0.57 | 0 | 0 |
| Breakthrough | 0.88 | 0 | 0 | 0 | 0 |
| Drug | 0.44 | 0.93 | 0 | 0 | 1 |
| For | 0.18 | 0 | 0.11 | 0.14 | 0 |
| Hopes | 0 | 0 | 0 | 0.69 | 0 |
| New | 0 | 0.37 | 0.11 | 0.14 | 0 |
| Of | 0 | 0 | 0.57 | 0 | 0 |
| Patients | 0 | 0 | 0 | 0.69 | 0 |
| Schirophrenia | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| treatment | 0 | 0 | 0.57 | 0 | 0 |

# Answer 2: Download and import movie review dataset

```
import nltk

nltk.download('movie_reviews')


from nltk.corpus import movie_reviews
```

# Answer 2: Data Processing in Boolen format

```
documents = [(list(movie_reviews.words(fileid)), category) \
        for category in movie_reviews.categories() \
        for fileid in movie_reviews.fileids(category)]

# print out the size of documents
print(len(documents))

# shuffle documents, reorganize the order of documents
random.shuffle(documents)

# split documents into training and test data
doc_train, doc_test = documents[100:], documents[:100]

# print out the size of training and test documents
print(len(doc_train), len(doc_test))
```

# Answer 2: Category set and examples

```
# display category set

categories = set([c for d, c in doc_train])

print(categories)


# display example movie review and its category

print(doc_train[0][0])

print(doc_train[0][1])
```

# Answer 2: Vocabulary

# get the word frequency distribution of training data

```
all_words = nltk.FreqDist(w.lower() for d, c in doc_train for w in d)
```

# get the top-2000 most common words

```
word_features = [word for word, num in all_words.most_common(2000)]
```

# display first 20 words in word_features

```
print(word_features[:20])
```

# display the size of all_words

```
all_word_num = len(all_words)
print(all_word_num)
```

# display top-k most common words, k in [0, 50) is generated randomly

```
k = random.randint(0, 50)
print(k)
all_words.pprint(k)
```

# plot the word frequency distribution of top-k most common words

```
all_words.plot(k, title='word distribution')
```

# Answer 2: Binary vectors

```
# define a function, return a dict, the type of key is str and value is Boolean

def document_features(document):
  document_words = set(document)
  features = {}
  for word in word_features:
    features[word] = (word in document_words)
  return features
```

```
# convert documents into Boolean vectors

train_set = [(document_features(d), c) for (d,c) in doc_train]

test_set = [(document_features(d), c) for (d,c) in doc_test]

# print example features

print(train_set[0][0])
```

# Answer 2: Training SVM with binary vectors

# train an SVM model with training set

```
import nltk.classify
from sklearn.svm import LinearSVC
```

# wrapper sklearn classifier using nltk

```
classifier = nltk.classify.SklearnClassifier(LinearSVC())
```

# train the LinearSVC classifier using training data

```
classifier.train(train_set)
```

# apply the trained SVM model for test set

```
accuracy = nltk.classify.accuracy(classifier, test_set)
print(accuracy)
```

# Answer 2: Training Random Forest with binary vectors

# train an RF model with training set

from sklearn.ensemble import RandomForestClassifier

classifier = nltk.classify.SklearnClassifier(RandomForestClassifier())

classifier.train(train_set)

# apply the trained RF model for test set

accuracy = nltk.classify.accuracy(classifier, test_set)

print(accuracy)

# Answer 2: TF-IDF vectors

```
# convert documents into tfidf vectors

from sklearn.feature_extraction.text
import TfidfVectorizer

tfidf =
TfidfVectorizer(stop_words='english')

train_text = [' '.join(d) for (d,c) in
doc_train]

train_tfidf =
tfidf.fit_transform(train_text)

train_cats = [c for d, c in doc_train]

test_text = [' '.join(d) for (d,c) in
doc_test]

test_tfidf = tfidf.transform(test_text)

test_cats = [c for d, c in doc_test]
```

# Answer 2: Training Random Forest with TF-IDF vectors

```
# train an RF model with training set
classifier = RandomForestClassifier()
classifier.fit(train_tfidf, train_cats)

# apply the trained RF model for test set
accuracy = classifier.score(test_tfidf, test_cats)

print(accuracy)
```