Text Similarity

SCHOOL OF INFOCOMM

How similar are these documents?

Hundreds of passengers who tested negative for the new coronavirus have begun leaving a quarantined cruise ship in Japan amid heavy criticism over the country's handling of the outbreak.

One Japanese health expert who visited the Diamond Princess at the port in Yokohama said the situation on board was "completely chaotic".

US officials said moves to contain the virus "may not have been sufficient".

Passengers have described **the difficult quarantine situation on the vessel**.

At least 621 passengers and crew on the Diamond Princess have so far been infected by the Covid-19 virus - the biggest cluster outside mainland China.

The ship was carrying 3,700 people in total.

Hundreds of passengers have begun leaving the stricken Diamond Princess in Japan after testing negative for the coronavirus Covid-19, ending two weeks of quarantine that experts say failed to prevent the virus spreading onboard.

Japanese TV showed passengers – who spent quarantine largely confined to their cabins – leaving the ship on Wednesday morning to board waiting buses, while others left the pier in Yokohama, near Tokyo, by taxi.

Local health authorities said just over 500 passengers were expected to disembark on Wednesday with another 2,500 to follow over the next two days. About half the passengers were Japanese, media reports said.

Those living or staying in Japan were given contact details in case they develop symptoms of Covid-19, which has killed more than 2,000 people in China and infected more than 74,000 others. Hundreds of infections have been reported in other countries, along with five deaths. Japan has 615 cases confirmed, including 542 on the Diamond Princess.

Sources BBC, Guardian

Text Similarity Measures

Metrics that measure the similarity or distance between two texts

Measure based on:

surface closeness (lexical similarity)

meaning closeness (semantic similarity)

- In this class, we will be discussing **lexical** documents similarities
- Measuring similarity between documents is fundamental to document analysis. Some of the applications that use document similarity measures include; information retrieval, text classification, document clustering, topic modeling, topic tracking, matrix decomposition

Ways to Measure Text Similarities

Category	Approach	Features	Applications
Edit based similarities	Compare two strings by counting the number of operations need to transform one to the other	Good for short strings or tokens Does not take into account semantics	Spelling corrections
Token based similarities	Compare two strings by comparing the tokens between them	Good for long text Computationally simple Takes in account semantics	Text mining information retrieval
Sequence based	Compare two strings by comparing the sub-sequences of tokens between them	Good for short strings or tokens Does not take into account semantics	Bioinformatics Version control systems

Measuring Document Terms Matrix

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Document term matrix contains vectors that are typically very long and sparse

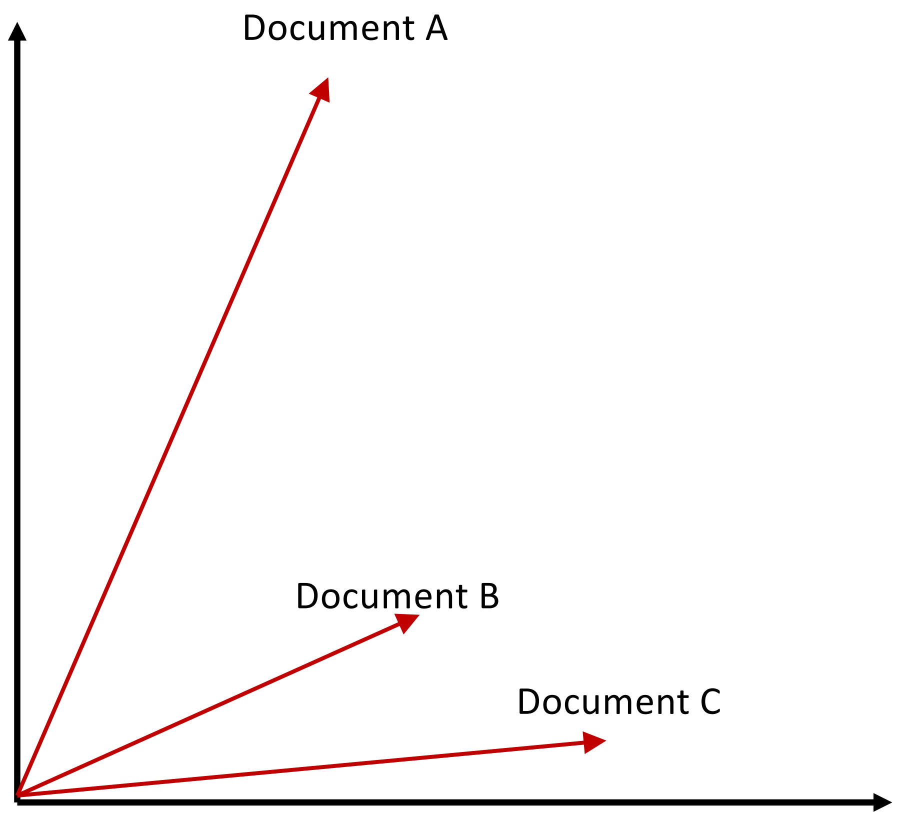
Two documents have many 0 values in common, meaning that they do not share many words, but this does not make them similar

We need a measure that focus on the words that the two documents do have in common, the occurrence frequency of such words, and ignores zero

Cosine Similarity

Measures the similarity between two **vectors** based on orientation in the vector space

- Two vectors with the same orientation have a cosine similarity of 1
- Two vectors oriented at 90° relative to each other have a similarity of 0
- Two vectors oriented at 180° relative to each other have a similarity of -1



Cosine Similarity

Each row of the document-term matrix can be considered as a vector representing a given document.

To compare 2 documents represented by vectors A and B,

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine Similarity

A = [3, 8, 7, 5, 2, 9]

B = [10, 8, 6, 6, 4, 5]

$$similarity(A,B) = \frac{3\cdot 10 + 8\cdot 8 + 7\cdot 6 + 5\cdot 6 + 2\cdot 4 + 9\cdot 5}{\sqrt{3^2 + 8^2 + 7^2 + 5^2 + 2^2 + 9^2} \times \sqrt{10^2 + 8^2 + 6^2 + 6^2 + 4^2 + 5^2}}$$

Scikit-Learn Pairwise Metric

Let's take a look at
https://scikit-learn.org/stable/modules/metrics.html#metrics

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Scikit-Learn Cosine Similarity

```
corpus = ["NLP love you",
          "You love NLP You Love NLP",
          "Artificial intellience",
          "artificial sugar"]
```

```
[[0 0 1 1 0 1]
 [0 0 2 2 0 2]
 [1 1 0 0 0 0]
 [1 0 0 0 1 0]]
```

```
cv = CountVectorizer( lowercase=True, ngram_range=(1,1))
dtm = cv.fit_transform(corpus).toarray()
```

```
print(cosine_similarity([dtm[1]], [dtm[0]]))
print(cosine_similarity([dtm[1]], [dtm[1]]))
print(cosine_similarity([dtm[1]], [dtm[2]]))
print(cosine_similarity([dtm[1]], [dtm[3]]))
```

```
[[1.]]
[[1.]]
[[0.]]
[[0.]]
```

Keeping Scores

If you have a corpus of 10,000 documents to compare

Document Index	Document Index	Cosine Score
0	0	
0	1	
…	…	
0	9999	
1	2	
1	3	
…	…	
9998	9999	

Python Tips

```python
from itertools import combinations
pairs = list(combinations(range(len(corpus)),2))
pairs
# pairs contains all possible combination of index
```

```
[(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]
```

Python Tips

```
dtm
```

```
array([[0, 0, 1, 1, 0, 1],
       [0, 0, 2, 2, 0, 2],
       [1, 1, 0, 0, 0, 0],
       [1, 0, 0, 0, 1, 0]])
```

```
for (idx_a, idx_b) in pairs:
    cs = round(cosine_similarity([dtm[idx_a]], [dtm[idx_b]])[0][0], 5)
    print(f"""Consine Similarity Score for document pairs ({ idx_a }, {idx_b}) is {cs}""")
```

```
Consine Similarity Score for document pairs (0, 1) is 1.0
Consine Similarity Score for document pairs (0, 2) is 0.0
Consine Similarity Score for document pairs (0, 3) is 0.0
Consine Similarity Score for document pairs (1, 2) is 0.0
Consine Similarity Score for document pairs (1, 3) is 0.0
Consine Similarity Score for document pairs (2, 3) is 0.5
```

Scikit-Learn Cosine Similarity

```
[[0 0 1 1 0 1]
 [0 0 2 2 0 2]
 [1 1 0 0 0 0]
 [1 0 0 0 1 0]]
```

```
orpus = ["NLP love you",
         "You love NLP You Love NLP",
         "Artificial intellience",
         "artificial sugar"]
```

```
cv = CountVectorizer( lowercase=True, ngram_range=(1,1))
dtm = cv.fit_transform(corpus).toarray()
```

```
print(cosine_similarity(dtm, dtm))
```

```
[[1.  1.  0.  0. ]
 [1.  1.  0.  0. ]
 [0.  0.  1.  0.5]
 [0.  0.  0.5 1. ]]
```

(i,j) represents the cosine similarity between document i and document j

Document Similarity: Full Example

Here are five documents.

“The weather is hot under the sun”

“I make my hot chocolate with milk”

“One hot encoding”

“I will have a chai latte with milk”

“There is a hot sale today”

Let’s see which pairs are most similar from a lexical standpoint

General Approach

```
corpus = ['The weather is hot under the sun',
          'I make my hot chocolate with milk',
          'One hot encoding',
          'I will have a chai latte with milk',
          'There is a hot sale today']
```

All possible pairs of the corpus

```
[(0, 1),
 (0, 2),
 (0, 3),
 (0, 4),
 (1, 2),
 (1, 3),
 (1, 4),
 (2, 3),
 (2, 4),
 (3, 4)]
```

Extract features

```
array([[0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1],
       [0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0]])
```

X : Document Term matrix

For every pair
- Calculate cosine_similarity by using the tuples to refer back to the document-term matrix

Example: cosine_similarity(X[3], X[4])

Implementation Walk-Through

Ref to :
document-example-4-cosine-similarity-workflow-count.pynb

Results (using Count Vectorizer)

```
0.408 ('The weather is hot under the sun', 'One hot encoding')
0.408 ('One hot encoding', 'There is a hot sale today')
0.354 ('I make my hot chocolate with milk', 'One hot encoding')
0.333 ('The weather is hot under the sun', 'There is a hot sale today')
0.289 ('The weather is hot under the sun', 'I make my hot chocolate with milk')
0.289 ('I make my hot chocolate with milk', 'There is a hot sale today')
0.289 ('I make my hot chocolate with milk', 'I will have a chai latte with milk')
0.0 ('The weather is hot under the sun', 'I will have a chai latte with milk')
0.0 ('One hot encoding', 'I will have a chai latte with milk')
0.0 ('I will have a chai latte with milk', 'There is a hot sale today')
```

Documents with "hot" are most similar, but it’s just because the term “hot” is a popular word

Semantically, “Milk” seems to be a better differentiator, so how we can mathematically highlight that?

Exercise

Document Similarities

Refer to

document-exercise-2-documents-similarity.ipynb

Exercise Results (using TFIDF Vectorizer)

```
0.232 ('I make my hot chocolate with milk', 'I will have a chai latte with milk')
0.182 ('The weather is hot under the sun', 'One hot encoding')
0.182 ('One hot encoding', 'There is a hot sale today')
0.161 ('I make my hot chocolate with milk', 'One hot encoding')
0.137 ('The weather is hot under the sun', 'There is a hot sale today')
0.121 ('The weather is hot under the sun', 'I make my hot chocolate with milk')
0.121 ('I make my hot chocolate with milk', 'There is a hot sale today')
0.0 ('The weather is hot under the sun', 'I will have a chai latte with milk')
0.0 ('One hot encoding', 'I will have a chai latte with milk')
0.0 ('I will have a chai latte with milk', 'There is a hot sale today')
```

Application: A Simple Content Based Recommendation System

(1) Collect dataset of contents (e.g. movie transcripts, movie description)

(2) Create document-term matrix using TF-IDF

(3) Calculate cosine similarities between all possible pairs of documents

(4) Given an input (e.g. movie title), find 10 songs which descriptions are the most similar to the input movie title

Exercise

Document Similarities

Refer to

document-exercise-3-documents-similarity.ipynb

References

- Intel AI Developer Program, https://software.intel.com/en-us/ai
- What is text similarity, https://kavita-ganesan.com/what-is-text-similarity/#.XbuiLEUzb3Q
- Bag of Words - Intro to Machine Learning, https://youtu.be/OGK9SHt8SWg
- What is TF-IDF? https://monkeylearn.com/blog/what-is-tf-idf/
- How to Prepare Text Data for Machine Learning with scikit-learn, https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/