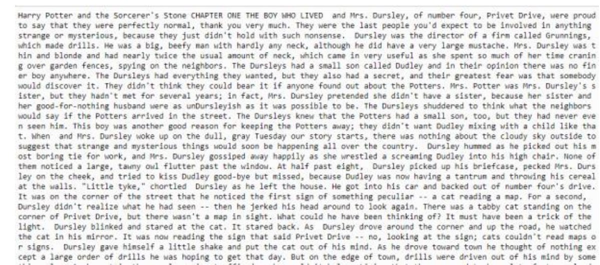
COMP30810
Intro to Text Analytics

Dr. Binh Thanh Le
thanhbinh.le@ucd.ie
Insight Centre for Data Analytics
School of Computer Science
University College Dublin

Recap of previous lecture

Corpus

Document



Harry Potter and the Sorcerer's Stone CHAPTER ONE THE BOY WHO LIVED and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that. When and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country. Dursley hummed as he picked out his most boring tie for work, and Mrs. Dursley gossiped away happily as she wrestled a screaming Dudley into his high chair. None of them noticed a large, tawny owl flutter past the window. At half past eight, Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Dursley as he left the house. He got into his car and backed out of number four's drive. It was on the corner of the street that he noticed the first sign of something peculiar -- a cat reading a map. For a second, Dursley didn't realize what he had seen -- then he jerked his head around to look again. There was a tabby cat standing on the corner of Privet Drive, but there wasn't a map in sight. What could he have been thinking of? It must have been a trick of the light. Dursley blinked and stared at the cat. It stared back. As Dursley drove around the corner and up the road, he watched the cat in his mirror. It was now reading the sign that said Privet Drive -- no, looking at the sign; cats couldn't read maps or signs. Dursley gave himself a little shake and put the cat out of his mind. As he drove toward town he thought of nothing except a large order of drills he was hoping to get that day. But on the edge of town, drills were driven out of his mind by some

Raw text

Document Summarization

<Meaning>

- Most common words
- Frequency
- N-gram
- POS

wordlist

Today Goals:

- Understanding TF-IDF
- Understanding ranking
- Understanding distance and similarity measurement
- Document summarization

Example

- Given the corpus of 4 documents as:

    Doc1: "The sky is blue."

    Doc2: "The sun is bright today."

    Doc3: "The sun in the sky is bright.“

    Doc4: "We can see the shining sun, the bright sun."

<u>Our task</u>: return the analysis for this corpus

1) TF-IDF

Term Frequency–Inverse Document Frequency

Term Frequency *: gives us the frequency of the word in each document in the corpus.*

It is the ratio of number of times the word appears in a document compared to the total number of words in that document.

term *document*

$$tf(t,d) = \frac{\boxed{f_d(t)}}{\boxed{\sum_{t' \in d} f_d(t')}}$$

Note:

- It increases as the number of occurrences of that word within the document increases.
- Each document has its own tf.

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

1) TF-IDF

Python code

First: build the wordlist for corpus:

```
import nltk
from nltk.tokenize import RegexpTokenizer

Doc1 = "The sky is blue."
Doc2 = "The sun is bright today."
Doc3 = "The sun in the sky is bright."
Doc4 = "We can see the shining sun, the bright sun."

## Retrieve the wordlist
alltext = ' '.join([Doc1,Doc2,Doc3,Doc4])
# tokenizer
pattern = r'\w+'
tokenizer = RegexpTokenizer(pattern)
corpus_tokens = tokenizer.tokenize(alltext)

# decapitalize
wordlist = list(set([word.lower() for word in corpus_tokens]))
wordlist
```

bag-of-words

```
['sky',
 'in',
 'bright',
 'sun',
 'today',
 'we',
 'can',
 'see',
 'shining',
 'is',
 'blue',
 'the']
```

1) TF-IDF

Python code <Term Frequency>

Easy code for understanding:

	Terms	Doc1	Doc2	Doc3	Doc4
0	sky	0.25	0.0	0.14	0.00
1	in	0.00	0.0	0.14	0.22
2	bright	0.00	0.2	0.14	0.11
3	sun	0.00	0.2	0.14	0.22
4	today	0.00	0.2	0.00	0.00
5	we	0.00	0.0	0.00	0.11
6	can	0.00	0.0	0.00	0.11
7	see	0.00	0.0	0.00	0.11
8	shining	0.00	0.0	0.00	0.11
9	is	0.25	0.2	0.14	0.00
10	blue	0.25	0.0	0.00	0.00
11	the	0.25	0.2	0.29	0.22

```
import pandas as pd
statistic_table = pd.DataFrame([])
statistic_table['Terms'] = wordlist
statistic_table['Doc1'] = [0]*len(statistic_table)
statistic_table['Doc2'] = [0]*len(statistic_table)
statistic_table['Doc3'] = [0]*len(statistic_table)
statistic_table['Doc4'] = [0]*len(statistic_table)
for idx,row in statistic_table.iterrows():
    statistic_table['Doc1'].iloc[idx] = round(Doc1.lower().count(row['Terms']) / len(Doc1.lower().split()),2)
    statistic_table['Doc2'].iloc[idx] = round(Doc2.lower().count(row['Terms']) / len(Doc2.lower().split()),2)
    statistic_table['Doc3'].iloc[idx] = round(Doc3.lower().count(row['Terms']) / len(Doc3.lower().split()),2)
    statistic_table['Doc4'].iloc[idx] = round(Doc4.lower().count(row['Terms']) / len(Doc4.lower().split()),2)

statistic_table
```

$\boxed{f_d(t)}$ $\boxed{\sum_{t' \in d} f_d(t')}$

1) TF-IDF

Inverse Document Frequency

- used to calculate the weight of rare words across all documents in the corpus.
- The words that occur rarely in the corpus have a high IDF score.

$$\boxed{idf(t, C) = log\left(\frac{|C|}{|\{d \in C : t \in d\}|}\right)}$$

$$= log\left(\frac{total\ number\ of\ documents}{the\ number\ of\ documents\ containing\ the\ term}\right)$$

1) TF-IDF

Python code

< Inverse Document Frequency >

```
## Compute
# total number of documents
n_docs = len(statistic_table.columns) -1  # = 4
#the number of documents containing the term
n_term = ((statistic_table['Doc1'] != 0)*1 +
          (statistic_table['Doc2'] != 0)*1 +
          (statistic_table['Doc3'] != 0)*1 +
          (statistic_table['Doc4'] != 0)*1)
import numpy as np
statistic_table['idf'] = round(np.log(n_docs / n_term),2)
statistic_table
```

Or

```
#the number of documents containing the term
n_term = np.sum((statistic_table.iloc[:,1:]!=0)*1,axis =1)
n_term
```

TF

Terms	Doc1	Doc2	Doc3	Doc4	idf
sky	0.25	0.0	0.14	0.00	0.69
in	0.00	0.0	0.14	0.22	0.69
bright	0.00	0.2	0.14	0.11	0.29
sun	0.00	0.2	0.14	0.22	0.29
today	0.00	0.2	0.00	0.00	1.39
we	0.00	0.0	0.00	0.11	1.39
can	0.00	0.0	0.00	0.11	1.39
see	0.00	0.0	0.00	0.11	1.39
shining	0.00	0.0	0.00	0.11	1.39
is	0.25	0.2	0.14	0.00	0.29
blue	0.25	0.0	0.00	0.00	1.39
the	0.25	0.2	0.29	0.22	0.00

1) TF-IDF

TF-IDF Combining these two we come up with the TF-IDF score for a word in a document in the corpus. It is the product of *TF* and *IDF*

$$\boxed{tfidf(t,d,C) = tf(t,d) \times idf(t,C)}$$

Python code

```
statistic_table['tfidf-Doc1'] = statistic_table['Doc1']*statistic_table['idf']
statistic_table['tfidf-Doc2'] = statistic_table['Doc2']*statistic_table['idf']
statistic_table['tfidf-Doc3'] = statistic_table['Doc3']*statistic_table['idf']
statistic_table['tfidf-Doc4'] = statistic_table['Doc4']*statistic_table['idf']
statistic_table
```

1) TF-IDF

	TF				IDF	TF-IDF			
Terms	**Doc1**	**Doc2**	**Doc3**	**Doc4**	**idf**	**tfidf-Doc1**	**tfidf-Doc2**	**tfidf-Doc3**	**tfidf-Doc4**
sky	0.25	0.0	0.14	0.00	0.69	0.1725	0.000	0.0966	0.0000
in	0.00	0.0	0.14	0.22	0.69	0.0000	0.000	0.0966	0.1518
bright	0.00	0.2	0.14	0.11	0.29	0.0000	0.058	0.0406	0.0319
sun	0.00	0.2	0.14	0.22	0.29	0.0000	0.058	0.0406	0.0638
today	0.00	0.2	0.00	0.00	1.39	0.0000	0.278	0.0000	0.0000
we	0.00	0.0	0.00	0.11	1.39	0.0000	0.000	0.0000	0.1529
can	0.00	0.0	0.00	0.11	1.39	0.0000	0.000	0.0000	0.1529
see	0.00	0.0	0.00	0.11	1.39	0.0000	0.000	0.0000	0.1529
shining	0.00	0.0	0.00	0.11	1.39	0.0000	0.000	0.0000	0.1529
is	0.25	0.2	0.14	0.00	0.29	0.0725	0.058	0.0406	0.0000
blue	0.25	0.0	0.00	0.00	1.39	0.3475	0.000	0.0000	0.0000
the	0.25	0.2	0.29	0.22	0.00	0.0000	0.000	0.0000	0.0000

Doc1: "The sky is blue"

2) Information Retrieval (IR)

1. What is the document having a good description for the "sun"?

Doc1: "The sky is blue."
Doc2: "The sun is bright today."
Doc3: "The sun in the sky is bright."
Doc4: "We can see the shining sun, the bright sun."

Query: "sun"

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sun	0.0000	0.058	0.0406	0.0638

TF-IDF

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sky	0.1725	0.000	0.0966	0.0000
in	0.0000	0.000	0.0966	0.1518
bright	0.0000	0.058	0.0406	0.0319
sun	0.0000	0.058	0.0406	0.0638
today	0.0000	0.278	0.0000	0.0000
we	0.0000	0.000	0.0000	0.1529
can	0.0000	0.000	0.0000	0.1529
see	0.0000	0.000	0.0000	0.1529
shining	0.0000	0.000	0.0000	0.1529
is	0.0725	0.058	0.0406	0.0000
blue	0.3475	0.000	0.0000	0.0000
the	0.0000	0.000	0.0000	0.0000

Word vector

Document vector

2) Information Retrieval (IR)

1. What is the document having a good description for the "sun"?

Doc1: "The sky is blue."
Doc2: "The sun is bright today."
Doc3: "The sun in the sky is bright."
Doc4: "We can see the shining sun, the bright sun."

Query: "sun"

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sun	0.0000	0.058	0.0406	0.0638

2. What is the document having a good description for the "sun" or "sky"?

Query: "sun" or "sky"

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sky	0.1725	0.000	0.0966	0.0000
sun	0.0000	0.058	0.0406	0.0638
	0.1725	0.058	0.1372	0.0638

Why?: the description is stronger in Doc1, because the word "blue" has high information content.

2) Information Retrieval (IR)

1. What is the document having a good description for the "sun"?

Doc1: "The sky is blue."
Doc2: "The sun is bright today."
Doc3: "The sun in the sky is bright."
Doc4: "We can see the shining sun, the bright sun."

Query: "sun"

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sun	0.0000	0.058	0.0406	0.0638

2. What is the document having a good description for the "sun" or "sky"?

Query: "sun" or "sky"

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sky	0.1725	0.000	0.0966	0.0000
sun	0.0000	0.058	0.0406	0.0638
	0.1725	0.058	0.1372	0.0638

Why?: the description is stronger in Doc1, because the word "blue" has high information content.

3) Documents as vectors

- Each document is now represented by a real-valued vector of tf-idf $\in \mathcal{R}^{|V|}$
- So, we have a $|V|$- dimension vector space
- Terms are axes of the space
- Documents are samples in this space

→This is a very high-dimensional space: easily to get tens of thousands (millions) of dimensions.

→These are very sparse vectors – most entries are zero.

TF-IDF

Terms	tfidf-Doc1	tfidf-Doc2	tfidf-Doc3	tfidf-Doc4
sky	0.1725	0.000	0.0966	0.0000
in	0.0000	0.000	0.0966	0.1518
bright	0.0000	0.058	0.0406	0.0319
sun	0.0000	0.058	0.0406	0.0638
today	0.0000	0.278	0.0000	0.0000
we	0.0000	0.000	0.0000	0.1529
can	0.0000	0.000	0.0000	0.1529
see	0.0000	0.000	0.0000	0.1529
shining	0.0000	0.000	0.0000	0.1529
is	0.0725	0.058	0.0406	0.0000
blue	0.3475	0.000	0.0000	0.0000
the	0.0000	0.000	0.0000	0.0000

Word vector

Document vector

$|V|$: size of terms (=12)

4) What can we do with vectors?

- Assume we have a story (book) → corpus

Task: get the most informative sentence and word in that story

➔Book is a corpus

➔Sentences are documents

Document Summarization

Document Ranking

	Word 1	Word 2	Word 3	SUM
Sentence 1	tf-idf	tf-idf	tf-idf	0.124
Sentence 2	tf-idf	tf-idf	tf-idf	0.235
Sentence 3	tf-idf	tf-idf	tf-idf	0.254
Sentence 4	tf-idf	tf-idf	tf-idf	0.568
…				
SUM	0.256	1.568	1.245	

This is the most informative sentence

This is the most informative word in story

5) Improvement for ranking

- To rank each sentence ➔ use the tf-idf values.
- However, rather than simply taking the summation of all the values for a given sentence, it will be useful when using some additional techniques as follow:
  1. Only summing tf-idf values where the underling word is a noun.
  2. Add an additional value to a given sentence if it has any words that are included in the **title** of the document (title can be the first sentence of document).
  3. Apply a position weighting. For example if there are 10 sentences in a document, sentence nine's "position weighting" would be 0.9. This weighting is then multiplied by the value calculated in point 2.
  4. Apply n-gram to check the tf-idf of multiword in sentence.

https://medium.com/@acrosson/summarize-documents-using-tf-idf-bdee8f60b71

6) Distance and Similarity

We will cover three basic distance measurements in Text Mining:

- Euclidean Distance
- Cosine Similarity
- Jaccard Similarity

Consider these three sentences:

s1 = “David loves dogs”

s2 = “Dogs are ok with David”

s3 = “Cats love rain”

6) Distance

1. We generate tokens for sentences:

   s1 = (“david”, “love”, “dog”)
   s2 = (“dog”,”ok”,”david”)
   s3 = (“cat”,”love”,”rain”)

2. We build a vocabulary with all the words in our corpus.

   bag-of-words

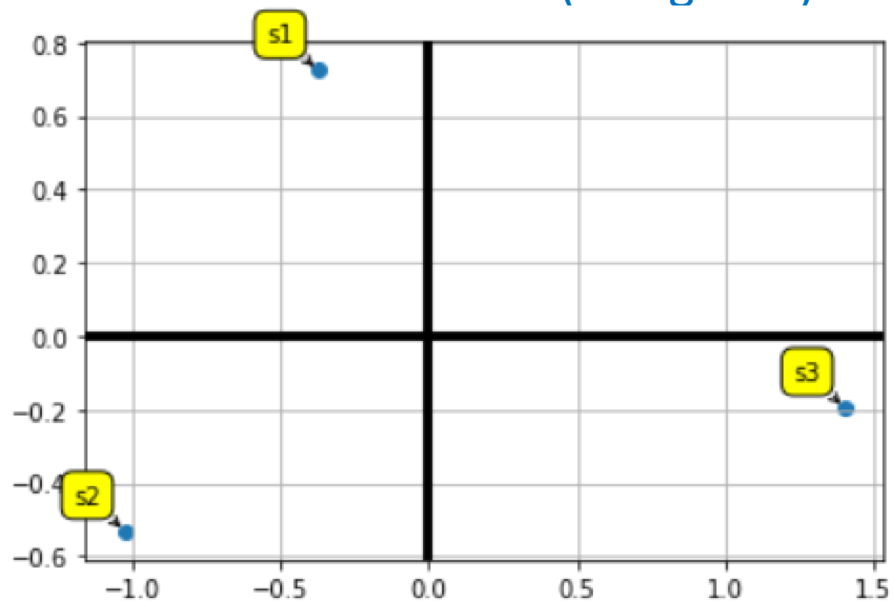
{'cat': 0, 'david': 1, 'dog': 2, 'love': 3, 'ok': 4, 'rain': 5}

3. Then, we generate the vectors (6-dimensional space) by using counting

   s1 = [0, 1, 1, 1, 0, 0]
   s2 = [0, 1, 1, 0, 1, 0]
   s3 = [1, 0, 0, 1, 0, 1]

In here, you can use tf-idf vectors, but for the example, I simply use counting

4. Reduce 6-d to 2-d for visualization (using PCA)

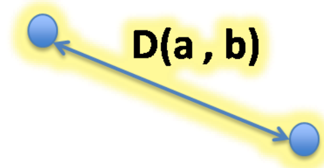


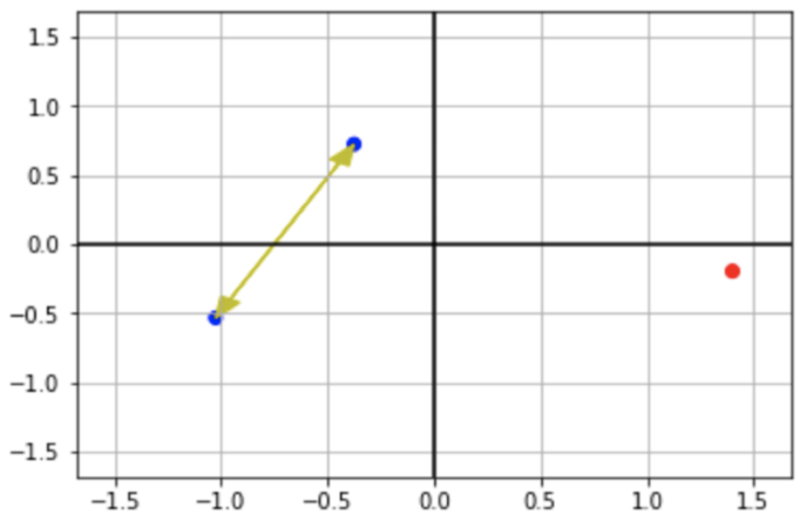

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X = np.array([s1,s2,s3])
v = pca.fit_transform(X)
```

6) Distance

- So, the Euclidean distance is good in this case

$$D(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$





Euclidean distance matrix

	S1	S2	S3
S1	0	1.41	2
S2	1.41	0	2.44
S3	2	2.44	0

S1 to S2 are closer than S1 to S3, or S2 to S3

```
from sklearn.metrics.pairwise import euclidean_distances
import numpy as np

np.round(euclidean_distances([s1,s2,s3],[s1,s2,s3]),2)
```

6) Distance

Now, consider these sentences:

s1 = “David loves dogs dogs dogs dogs dogs”

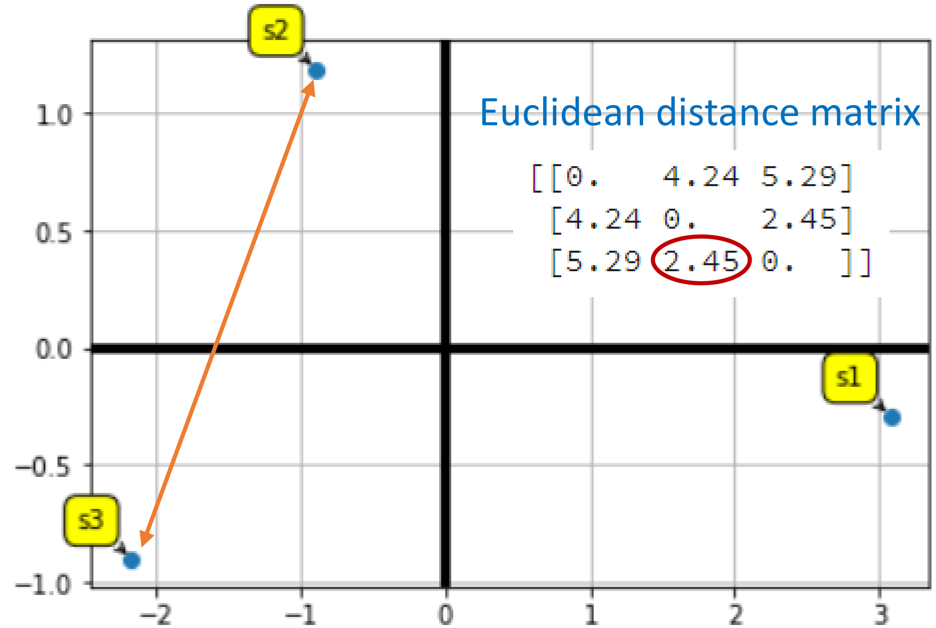
s2 = “Dogs are ok with David”

s3 = “Cats love rain”

{'cat': 0, 'david': 1, 'dog': 2, 'love': 3, 'ok': 4, 'rain': 5}

s1 = [0, 1, 5, 1, 0, 0]
s2 = [0, 1, 1, 0, 1, 0]
s3 = [1, 0, 0, 1, 0, 1]

Let’s plot those vectors using PCA again



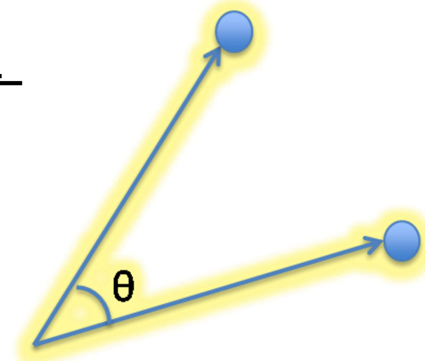
See what happened: now, Sentence 2 and 3 are closer than 1 and 2.

This is very bad for us

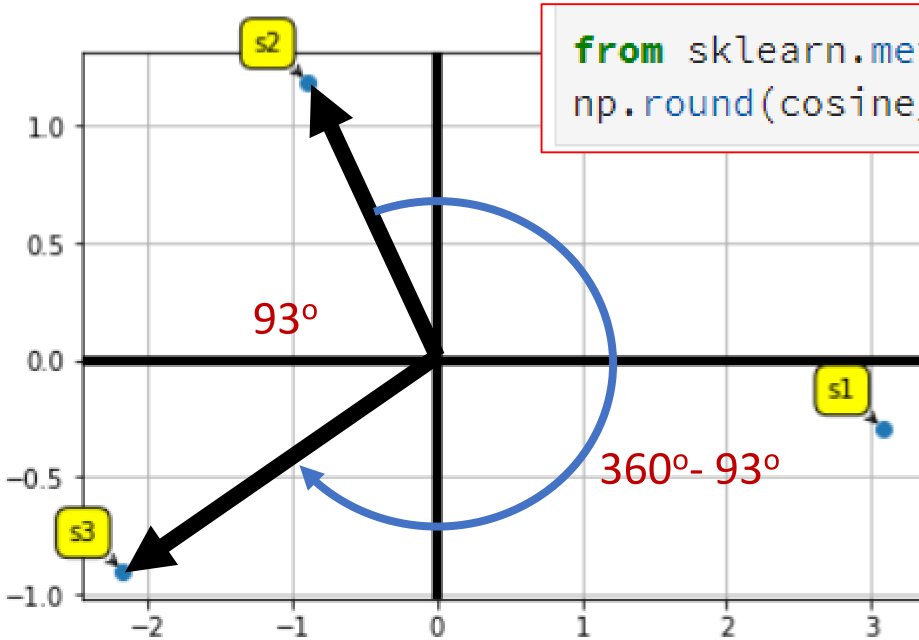
6) Similarity

Cosine Similarity

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$




```
from sklearn.metrics.pairwise import cosine_similarity
np.round(cosine_similarity([s1,s2,s3], [s1,s2,s3]),3)
```



	S1	S2	S3
S1	[[1.	0.667	0.111]
S2	[0.667	1.	0.]
S3	[0.111	0.	1.]]

S1 to S2 are more similar than S1 to S3, or S2 to S3

```
[round(np.cos(np.deg2rad(360-93)),3),round(np.cos(np.deg2rad(93)),3)]
```

```
[-0.052, -0.052]
```

6) Similarity

Jaccard Similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{size(Intersection\ of\ A\ and\ B)}{size(Union\ of\ A\ and\ B)}$$

```
from sklearn.metrics import jaccard_similarity_score
J = np.zeros((3,3))
for idx1,tmps1 in enumerate([s1,s2,s3]):
    for idx2,tmps2 in enumerate([s1,s2,s3]):
        J[idx1,idx2] = round(jaccard_similarity_score(tmps1,tmps2),2)
        J[idx2,idx1] = J[idx1,idx2]
print(J)
```

```
s1 = [0, 1, 5, 1, 0, 0]
s2 = [0, 1, 1, 0, 1, 0]
```

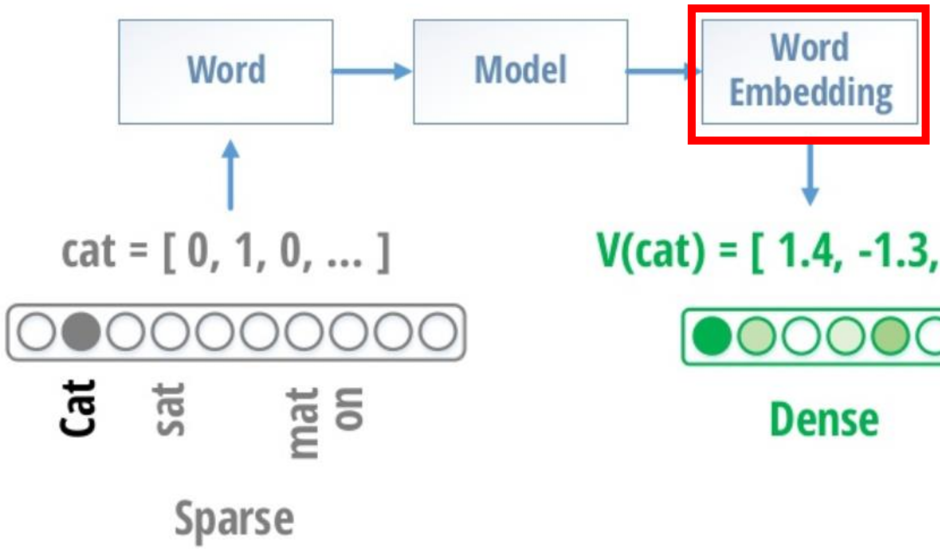
$\rightarrow J(s1, s2) = \frac{3}{6} = 0.5$

	S1	S2	S3
S1	[[1.	0.5	0.33]
S2	[0.5	1.	0.]
S3	[0.33	0.	1.]]

S1 to S2 are more similar than S1 to S3, or S2 to S3

7) Conclusion

- The techniques are simple for finding the similarity of two documents if having common words
- *Advantage*: text is represented as a **vector of numbers**
- *Limitation*:
  - The techniques do not cover the synonym scenario (e.g. [dog, puppy] , [buy, purchase], [funny, amusing], etc.)
  - The vectors of documents are sparse vectors (many zeros)

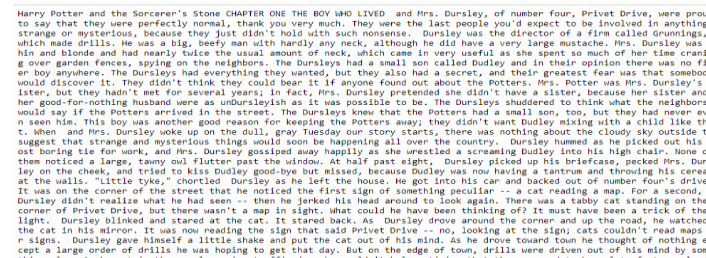


Advanced Text Analytics

Summary

Corpus

Document



Harry Potter and the Sorcerer's Stone CHAPTER ONE THE BOY WHO LIVED and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that. When and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country. Dursley hummed as he picked out his most boring tie for work, and Mrs. Dursley gossiped away happily as she wrestled a screaming Dudley into his high chair. None of them noticed a large, tawny owl flutter past the window. At half past eight, Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Dursley as he left the house. He got into his car and backed out of number four's drive. It was on the corner of the street that he noticed the first sign of something peculiar -- a cat reading a map. For a second, Dursley didn't realize what he had seen -- then he jerked his head around to look again. There was a tabby cat standing on the corner of Privet Drive, but there wasn't a map in sight. What could he have been thinking of? It must have been a trick of the light. Dursley blinked and stared at the cat. It stared back. As Dursley drove around the corner and up the road, he watched the cat in his mirror. It was now reading the sign that said Privet Drive -- no, looking at the sign; cats couldn't read maps or signs. Dursley gave himself a little shake and put the cat out of his mind. As he drove toward town he thought of nothing except a large order of drills he was hoping to get that day. But on the edge of town, drills were driven out of his mind by some

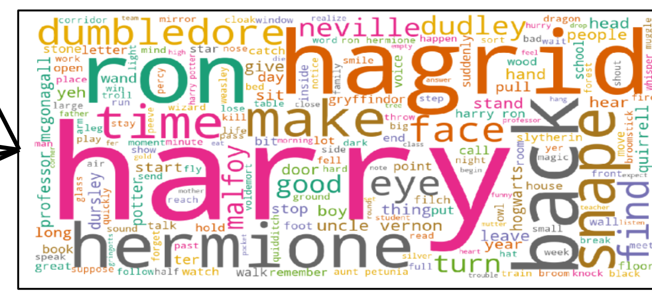
Raw text

Most common words

Frequency

N-gram

POS



wordlist

Text → Vector

TF-IDF

Information Retrieval

Ranking

Distance & Similarity