Lecture 12
Textual Data: Vector Space Model and TF-IDF

Dennis Sun
Stanford University
DATASCI / STATS 112

February 8, 2023

1. Review
2. Vector Space Model
3. tf-idf
4. Reminders

1. Review
2. Vector Space Model
3. tf-idf
4. Reminders

Textual Data

0 "A Rose Is Still a Rose"

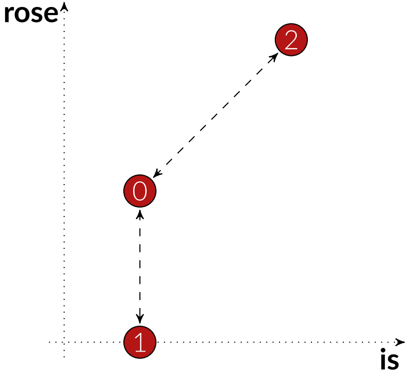
1 "There is no there there."

2 "Rose is a rose is a rose is a rose."

$\Rightarrow$

	is	rose	…
0	1	2	…
1	1	0	…
2	3	4	…

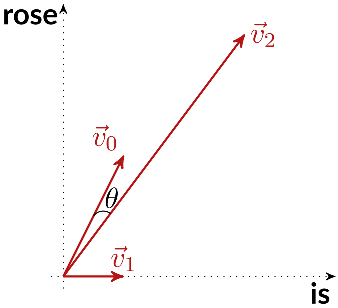
Which document is most similar to document 0?



Using Euclidean distance, document 1 appears closer than document 2!

Vector Space Model

In the **vector space model**, documents are represented as *vectors* instead of points.



The **length of a vector** is its distance from the origin $\vec{0}$:

$$||\vec{v}|| = \sqrt{\sum_{j=1}^{D} v_j^2}.$$

The distance between two vectors is the angle between them:

$$d(\vec{v}, \vec{w}) = 1 - \cos\theta = 1 - \frac{\text{sum of } v_j \cdot w_j}{||\vec{v}|| \cdot ||\vec{w}||}.$$

Using cosine distance, document 2 now appears closer!

Implementing the Vector Space Model

```
corpus = ["a rose is still a rose",
          "there is no there there",
          "rose is a rose is a rose is a rose"]
```

First, we use Pandas to get the term-frequency matrix.

```
import pandas as pd
from collections import Counter

tf = pd.DataFrame(
    [pd.Series(Counter(doc.split())) for doc in corpus],
).fillna(0)
tf
```

	a	rose	is	still	there	no
0	2.0	2.0	1.0	1.0	0.0	0.0
1	0.0	0.0	1.0	0.0	3.0	1.0
2	3.0	4.0	3.0	0.0	0.0	0.0

Now we just have to implement the formula for cosine distance.

Implementing the Vector Space Model

tf =

	a	rose	is	still	there	no
0	2.0	2.0	1.0	1.0	0.0	0.0
1	0.0	0.0	1.0	0.0	3.0	1.0
2	3.0	4.0	3.0	0.0	0.0	0.0

Now we just have to implement the formula for cosine distance.

$$d(\vec{v}, \vec{w}) = 1 - \frac{\text{sum of } v_j \cdot w_j}{||\vec{v}|| \cdot ||\vec{w}||}.$$

```
import numpy as np

def length(v):
  return np.sqrt((v ** 2).sum())

def cos_dist(v, w):
  return 1 - (v * w).sum() / (length(v) * length(w))

cos_dist(tf.loc[0], tf.loc[1]), cos_dist(tf.loc[0], tf.loc[2])
```

```
(0.9046537410754407, 0.07804555427071147)
```

Vector Space Model in Scikit-Learn

It's always easier to do it in Scikit-Learn.

```
from sklearn.feature_extraction.text import CountVectorizer

vec = CountVectorizer(token_pattern=r"(?u)\b\w+\b")
vec.fit(corpus)
tf_mat = vec.transform(corpus)
tf_mat.todense()
```

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

```
from sklearn.metrics import pairwise_distances
pairwise_distances(tf_mat[0, :], tf_mat[1:, :], metric="cosine")
```

```
array([[0.90465374, 0.07804555]])
```

tf-idf

So far, we've simply counted the **term frequency** $\mathrm{tf}(d, t)$: how many times each term t appears in each document d.

Problem: Common words like "is" or "the" tend to dominate because they have high counts.

We need to adjust for how common each word is:

1. Count the fraction of documents the term appears in:
$$\mathrm{df}(t, D) = \frac{\#\text{ documents containing term } t}{\#\text{ documents}} = \frac{|d \in D : t \in d|}{|D|}$$
2. Invert and take a log to obtain **inverse document frequency**:
$$\mathrm{idf}(t, D) = 1 + \log \frac{1}{\mathrm{df}(t, D)}.$$
3. Multiply tf by idf to get tf-idf:
$$\text{tf-idf}(d, t, D) = \mathrm{tf}(d, t) \cdot \mathrm{idf}(t, D).$$

Now we can use the **tf-idf matrix** just like we used the term-frequency matrix.

tf-idf by Hand

Previously, we saw the term-frequency matrix for this corpus:

0. `"A Rose Is Still a Rose"`
1. `"There is no there there."`
2. `"Rose is a rose is a rose is a rose."`

$\Rightarrow$

	is	rose	…
0	1	2	…
1	1	0	…
2	3	4	…

Now let's calculate the tf-idf matrix!

1. Calculate the document frequencies:

$$\mathrm{df}(\text{“is”}, D) = \frac{3}{3} = 1 \qquad \mathrm{df}(\text{“rose”}, D) = \frac{2}{3}$$

2. Calculate the inverse document frequencies:

$$\mathrm{idf}(\text{“is”}, D) = 1 + \log 1 = 1 \quad \mathrm{idf}(\text{“rose”}, D) = 1 + \log 1.5 \approx 1.176$$

3. Multiply tf by idf to get tf-idf:

	is	rose	…
0	1	2.81	…
1	1	0	…
2	3	5.62	…

tf-idf in Scikit-Learn

```
from sklearn.feature_extraction.text import TfidfVectorizer

# The options ensure that the numbers match our example above.
vec = TfidfVectorizer(smooth_idf=False, norm=None)
vec.fit(corpus)
tfidf_mat = vec.transform(corpus)
tfidf_mat.todense()
```

```
matrix([[1.        , 0.        , 2.81093022, 2.09861229, 0.        ],
        [1.        , 2.09861229, 0.        , 0.        , 6.29583687],
        [3.        , 0.        , 5.62186043, 0.        , 0.        ]])
```

Now we can use this tf-idf matrix just as we used the term frequency matrix!

```
pairwise_distances(tfidf_mat[0, :], tfidf_mat[1:, :], metric="cosine")
```

```
array([[0.95915143, 0.19106774]])
```

In-Class Exercise

Let’s do another example in Colab.

1. Review
2. Vector Space Model
3. tf-idf
4. Reminders

Reminders

- I will start hosting office hours in-person and on Zoom.
- Keep working on Assignment 3! It is due Friday.
- Start looking for data sets for the final project. Come talk to me about your ideas!