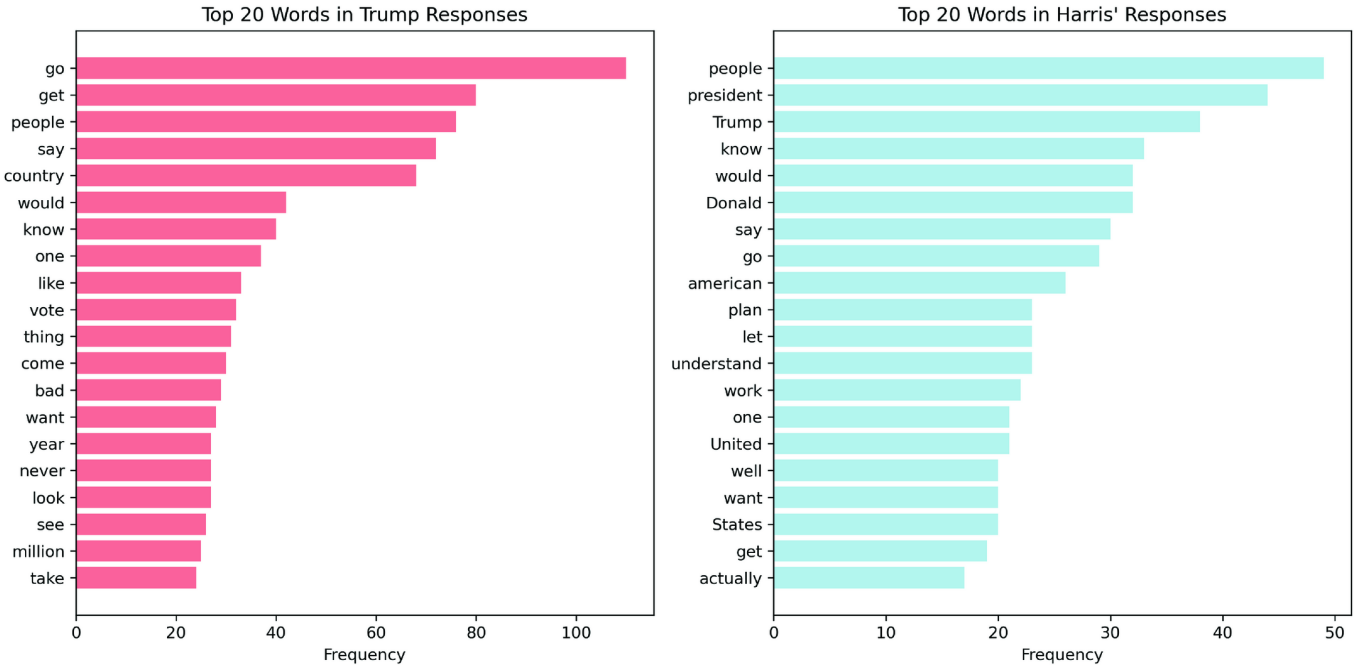
Textual Data: Vector Space Model and TF-IDF

Dennis Sun (modified by Jonathan Taylor)
Stanford University

DATASCI 112

April 20, 2026



Red and blue language: Word choices in the Trump and Harris 2024 presidential debate

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

1 Review

2 Vector Space Model

3 tf-idf

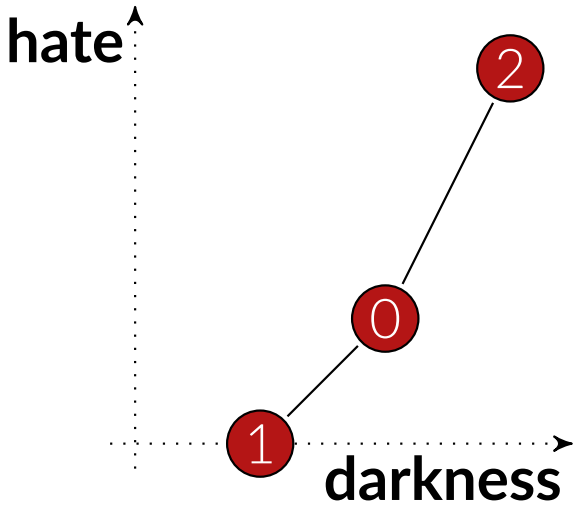
Textual Data

0. "Whoever has hate for his brother is in the darkness and walks in the darkness." —1 John 2:11
1. "Hello darkness, my old friend." —Simon & Garfunkel
2. "Returning hate for hate multiplies hate, adding deeper darkness to a night already devoid of stars. Darkness cannot drive out darkness; only light can do that." —MLK

$\Rightarrow$

	darkness	hate	…
0	2	1	…
1	1	0	…
2	3	3	…

Which document is most similar to document 0?



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

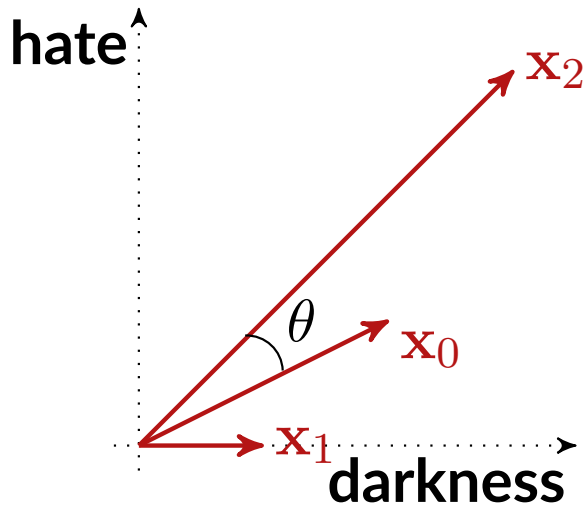
1 Review

2 Vector Space Model

3 tf-idf

Vector Space Model

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



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

$$||\mathbf{v}|| = \sqrt{\sum_{j=1}^{|V|} v_j^2}.$$

The distance between two vectors that we will use is inversely related to the angle between them:

$$d(\mathbf{x}, \mathbf{x}') = 1 - \cos\theta = 1 - \frac{\sum_{j=1}^{|V|} x_j x'_j}{||\mathbf{x}||\ ||\mathbf{x}'||}.$$

Using cosine distance, document 2 now appears closer!

Implementing the Vector Space Model

```
documents = [
    "whoever has hate for his brother is in the darkness and walks in the darkn
    "hello darkness my old friend",
    "returning hate for hate multiplies hate adding deeper darkness to a night
]
```

First, we use Pandas to get the term-frequency array (or matrix)

```
import pandas as pd
from collections import Counter

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

	whoever	has	hate	for	...	light	can	do	that
0	1.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	3.0	1.0	...	1.0	1.0	1.0	1.0

3 rows × 35 columns

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

Implementing the Vector Space Model

tf =

	whoever	has	hate	for	...	light	can	do	that
0	1.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	3.0	1.0	...	1.0	1.0	1.0	1.0

3 rows × 35 columns

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

$$d(\mathbf{x}, \mathbf{x}') = 1 - \frac{\sum_{j=1}^{|V|} x_j x'_j}{||\mathbf{x}||\ ||\mathbf{x}'||}.$$

```
import numpy as np

def cos_dist(v, w):
  return 1 - (v * w).sum() / (np.linalg.norm(v) * np.linalg.norm(w))

cos_dist(tf.loc[0], tf.loc[1]), cos_dist(tf.loc[0], tf.loc[2])
(0.8048199854102933, 0.646003837297605 6)
```

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"\w+")
vec.fit(documents)
tf_matrix = vec.transform(documents)
tf_array = tf_matrix.toarray()
```

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

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

```
array([[0.        , 0.80481999, 0.64600384]])
```

1 Review

2 Vector Space Model

3 tf-idf

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:

$$\mathrm{df}(D, t) = \frac{\#\text{ documents containing term } t}{\#\text{ documents}} = \frac{|\{d \in D : t \in d\}|}{|D|}$$

2. Invert and take a log to obtain **inverse document frequency**:

$$\mathrm{idf}(D, t) = 1 + \log \frac{1}{\mathrm{df}(D, t)}.$$

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

$$\text{tf-idf}(d, t, D) = \mathrm{tf}(d, t) \cdot \mathrm{idf}(D, t).$$

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

tf-idf by Hand

The term-frequency array for this corpus is:

0. "Whoever has hate for his brother is in the darkness and walks in the darkness."
1. "Hello darkness, my old friend."
2. "Returning hate for hate multiplies hate, adding deeper darkness to a night already devoid of stars. Darkness cannot drive out darkness; only light can do that."

$\Rightarrow$

	darkness	hate	…
0	2	1	…
1	1	0	…
2	3	3	…

Now let's calculate the tf-idf array!

1. Calculate the document frequencies:

$$\text{df}(D, \text{“darkness”}) = \frac{3}{3} = 1 \qquad \text{df}(D, \text{“hate”}) = \frac{2}{3}$$

2. Calculate the inverse document frequencies:

$$\text{idf}(D, \text{“darkness”}) = 1 + \log 1 = 1 \quad \text{idf}(D, \text{“hate”}) = 1 + \log \frac{3}{2} \approx 1.405$$

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

	darkness	hate	…
0	2	1.405	…
1	1	0	…
2	3	4.216	…

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(documents)
tfidf_array = vec.transform(documents).toarray()
```

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

```
pairwise_distances(tfidf_array[:1, :], tfidf_array,
                   metric="cosine")
array([[0.        , 0.94612045, 0.84453506]])
```

Dr. Seuss Example

Let's go into Colab and find the Dr. Seuss book that is most similar to *One Fish*, *Two Fish*, *Red Fish*, *Blue Fish*.