Basic Bag-of-Words and Measuring Document Similarity

FEATURE SELECTION AND EXTRACTION

Feature Selection & Extraction

Feature: Any property in your data you think is useful for making predictions or explaining some relationship.

Possible useful features in NLP

Word count

Document age

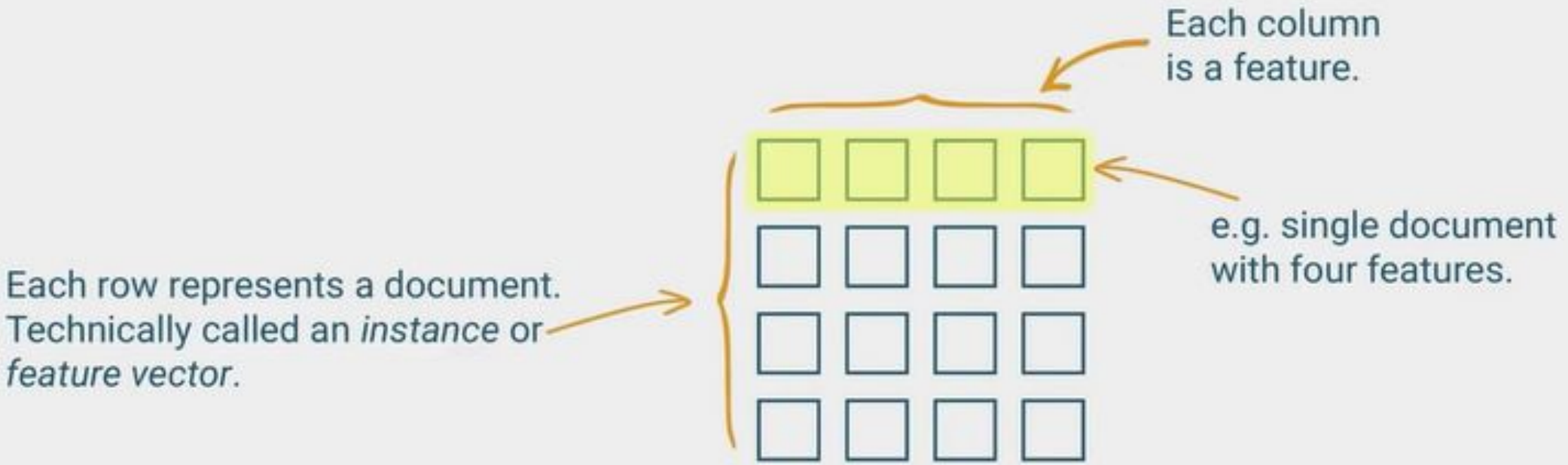
Author ID

Feature *selection*

Feature *extraction*

Combining features or reducing the number of features through *dimensionality reduction*.

What we want to end up with is a matrix



Selecting good features is critical in classic ML

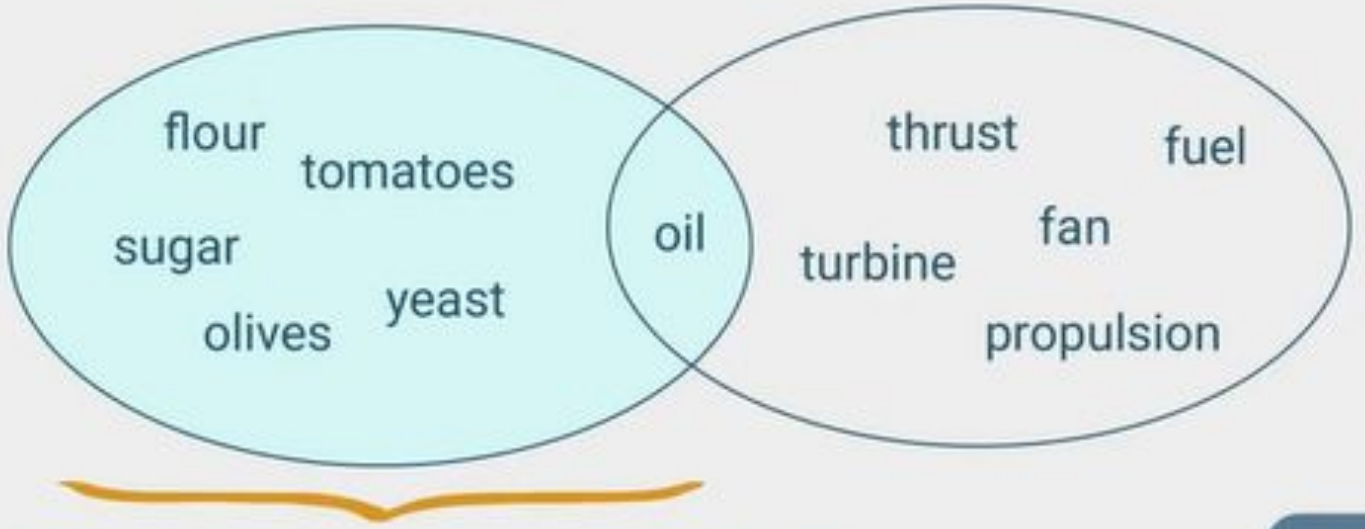
- Domain knowledge is an advantage. Good features with an ordinary algorithm can beat mediocre features with an advanced algorithm.
- Data cleaning and feature engineering is the majority of the work.

Bag-of-Words

Describing documents by word occurrences (multiple approaches).

Primary Idea

Meaning and similarity are encoded in vocabulary.



The more two documents share the same vocabulary, the more likely they belong to the same class.

Called a "bag" because we throw out word order.

Bag-of-Words

Describing documents by word occurrences (multiple approaches).

Binary BOW

Each row represents one document. A document could be a sentence, tweet, or entire book.

Each column (*feature*) represents a word in the *vocabulary*.

```
[0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0]
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0]
[0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0]
```

The number of rows is equal to the number of documents in the corpus.

The number of columns is equal to the size of the vocabulary.

- "1" means a token occurs in a document.
- "0" means it does not.

Binary BOW Example

d_1 : ['Red', 'Bull', 'drops', 'hint', 'on', 'F1', 'engine']

d_2 : ['Honda', 'exits', 'F1', 'leaving', 'F1', 'partner', 'Red', 'Bull']

d_3 : ['Hamilton', 'eyes', 'record', 'eighth', 'F1', 'title']

d_4 : ['Aston', 'Martin', 'announces', 'sponsor']

Let's say we have four documents...

['Aston', 'Bull', 'F1', 'Hamilton', 'Honda', 'Martin', 'Red', 'announces', 'drops', 'eighth', 'engine', 'exits', 'eyes', 'hint', 'leaving', 'on', 'partner', 'record', 'sponsor', 'title']

Our resulting 20-word vocabulary (sorted)

Binary BOW Example

```
{
    'Aston':     0,
    'Bull':      1,
    'F1':        2,
    'Hamilton':  3,
    'Honda':     4,
    'Martin':    5,
    'Red':       6,
    'announces': 7,
    'drops':     8,
    'eighth':    9,
    'engine':    10,
    'exits':     11,
    'eyes':      12,
    'hint':      13,
    'leaving':   14,
    'on':        15,
    'partner':   16,
    'record':    17,
    'sponsor':   18,
    'title':     19
}
```

d_1 : ['Red', 'Bull', 'drops', 'hint', 'on', 'F1', 'engine']

d_2 : ['Honda', 'exits', 'F1', 'leaving', 'F1', 'partner', 'Red', 'Bull']

d_3 : ['Hamilton', 'eyes', 'record', 'eighth', 'F1', 'title']

d_4 : ['Aston', 'Martin', 'announces', 'sponsor']

Resulting binary BOW encoding

D_1: [0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0]

D_2: [0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0]

D_3: [0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1]

D_4: [1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

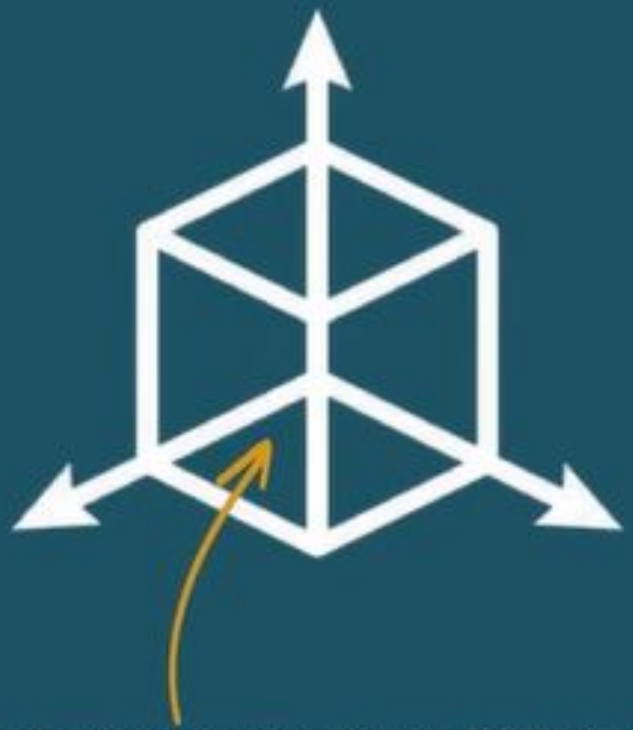
We have shifted our thinking about text...

From a sequence of symbols...

to points in a multidimensional space that encodes some meaning of the text.

With BOW, we assume meaning is encoded in the presence of certain words.

We have shifted our thinking about text...



Each feature vector in our BOW is now a point in this multidimensional* space.

Vector Space Model (VSM)

*The number of dimensions is equal to the number of features (i.e. size of vocabulary).

Vector Space Model

What we can now do

Two documents which have similar vocabulary should be closer together in this space. This allows us to start measuring document similarity which is useful for:

- Relevance ranking
- Plagiarism detection
- Document classification

And other applications...

How do we measure similarity?

Dot Product

The basis for most vector similarity metrics

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

Two vectors of equal length (*a*, *b*)

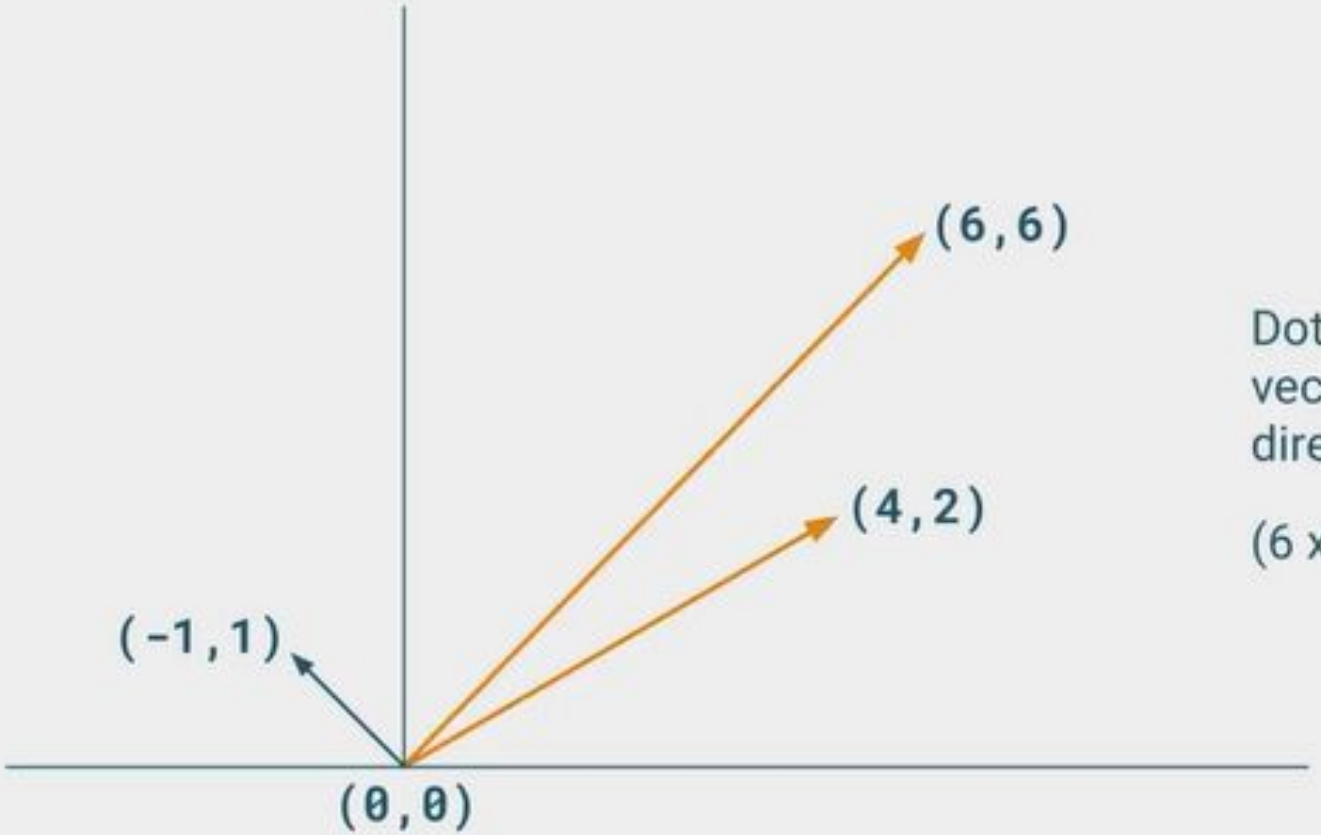
Multiply their corresponding values and add the products together.

a = [1, 2, 3]
b = [4, 5, 6]

a · **b** = (1x4) + (2x5) + (3x6) = **32**

Dot Product

The basis for most vector similarity metrics

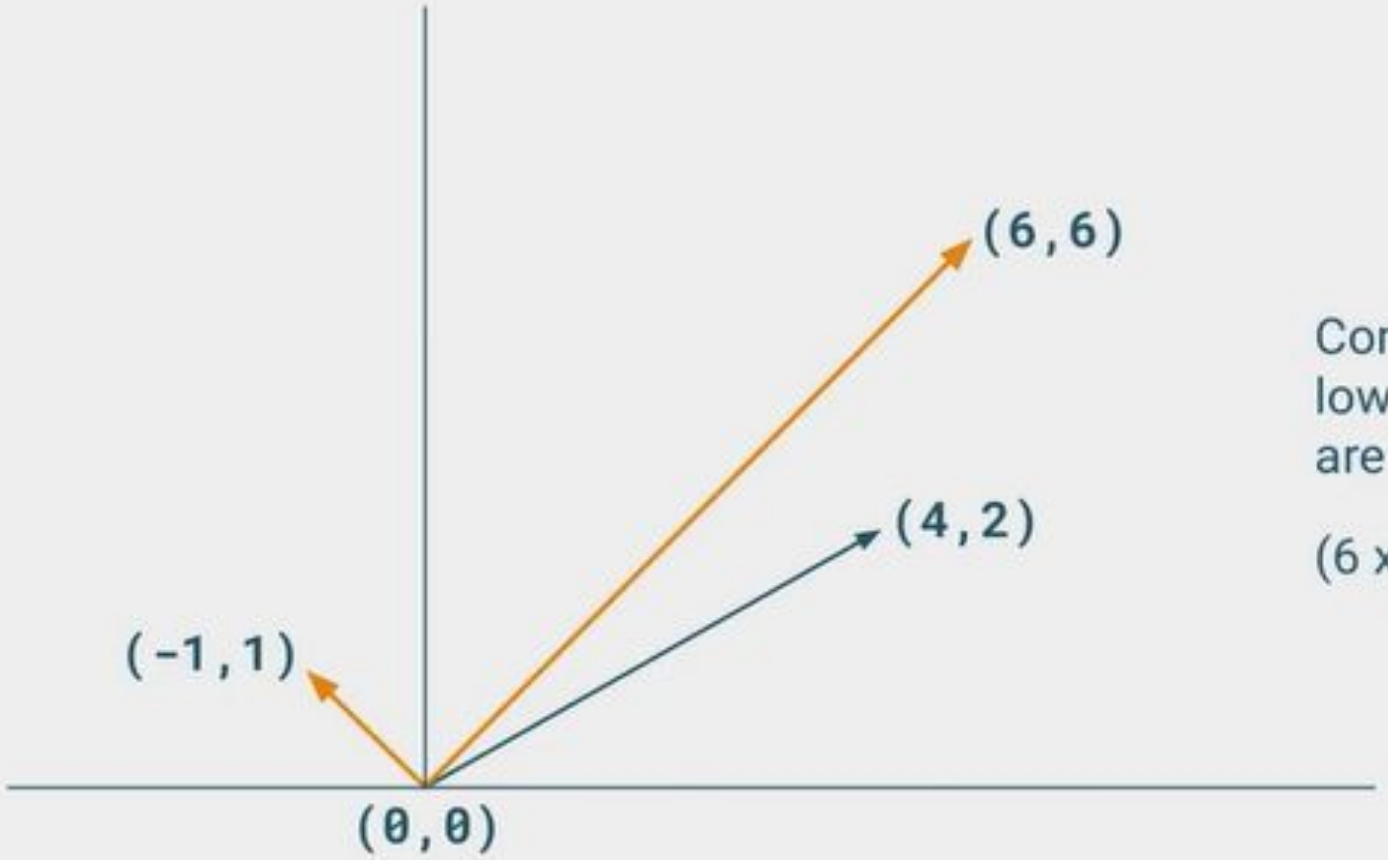


Dot product will be high when two vectors share a similar or same direction, and with large magnitudes.

(6 x 4) + (4 x 2) = 30

Dot Product

The basis for most vector similarity metrics



Conversely, the dot product will be low or 0 (orthogonal vectors) if they are dissimilar.

(6 x -1) + (6 x 1) = 0

Dot Product

The basis for most vector similarity metrics

But high frequency words will lead to larger dot products...

So we normalize by vector length:

$$\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

Also known as the *L^2 norm* or *Euclidean norm*.

Which leads to

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} = \cos\theta$$

The same as the cosine of the angle between two vectors

Cosine Similarity

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

The value of **cos θ** ranges from 0 to 1.*

- 1 when the vectors point in the same direction.
- 0 when the vectors are orthogonal (dissimilar).

*won't be lower than zero because we won't have negative frequencies.

Cosine Similarity Example

d_1 : ['Red', 'Bull', 'drops', 'hint', 'on', 'F1', 'engine']

[0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0]

d_2 : ['Honda', 'exits', 'F1', 'leaving', 'F1', 'partner', 'Red', 'Bull']

[0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0]

d_3 : ['Hamilton', 'eyes', 'record', 'eighth', 'F1', 'title']

[0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1]

d_4 : ['Aston', 'Martin', 'announces', 'sponsor']

[1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

$$\cos\theta = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\| \, \|\mathbf{d}_2\|} = 0.43$$

$$\cos\theta = \frac{\mathbf{d}_1 \cdot \mathbf{d}_3}{\|\mathbf{d}_1\| \, \|\mathbf{d}_3\|} = 0.15$$

$$\cos\theta = \frac{\mathbf{d}_1 \cdot \mathbf{d}_4}{\|\mathbf{d}_1\| \, \|\mathbf{d}_4\|} = 0.0$$

The threshold you select for “similar” will depend on your data.

Binary or raw frequency BOW can be a good start but...

Drawbacks

- Doesn't capture similarity between synonyms.
- No way to handle Out-of-Vocabulary (OOV) words.
- Creates sparse vectors which can be inefficient.
- Word order information is lost. *"Chelsea beats Barcelona"* is different from *"Barcelona beats Chelsea"*.

N-grams

Chunks of continuous tokens

- 2-gram or *bigram* has two tokens per chunk
- 3-gram or *trigram* has three tokens per chunk

.
.
.

"Chelsea beats Barcelona"

["Chelsea beats", "beats Barcelona"]

"Barcelona beats Chelsea"

["Barcelona beats", "beats Chelsea"]

Helps capture some context our previous approach didn't.

The vocabulary is the collection of bigrams across the corpus.

Code implementation of Basic Bag of word

Basic Bag-of-Words (BOW)

```
import spacy

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

```
# A corpus of sentences.
corpus = [
  "Red Bull drops hint on F1 engine.",
  "Honda exits F1, leaving F1 partner Red Bull.",
  "Hamilton eyes record eighth F1 title.",
  "Aston Martin announces sponsor."
]
```

```
vectorizer = CountVectorizer()
```

The *fit_transform* method does two things:

1. It learns a vocabulary dictionary from the corpus.
2. It returns a matrix where each row represents a document and each column represents a token (i.e. term).

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer.fit_transform

```
[ ] bow = vectorizer.fit_transform(corpus)
```

We can take a look at the features and vocabulary dictionary. Notice the **CountVectorizer** took care of tokenization for us. It also removed punctuation and lower-cased everything.

Thank you