

Natural Language Processing (AI615)

Prepared by Dr. Ashraf Elsayed

Lecture 8



Vector Semantics and Embeddings

Lecture 8

Prepared by
Dr. Ashraf Elsayed

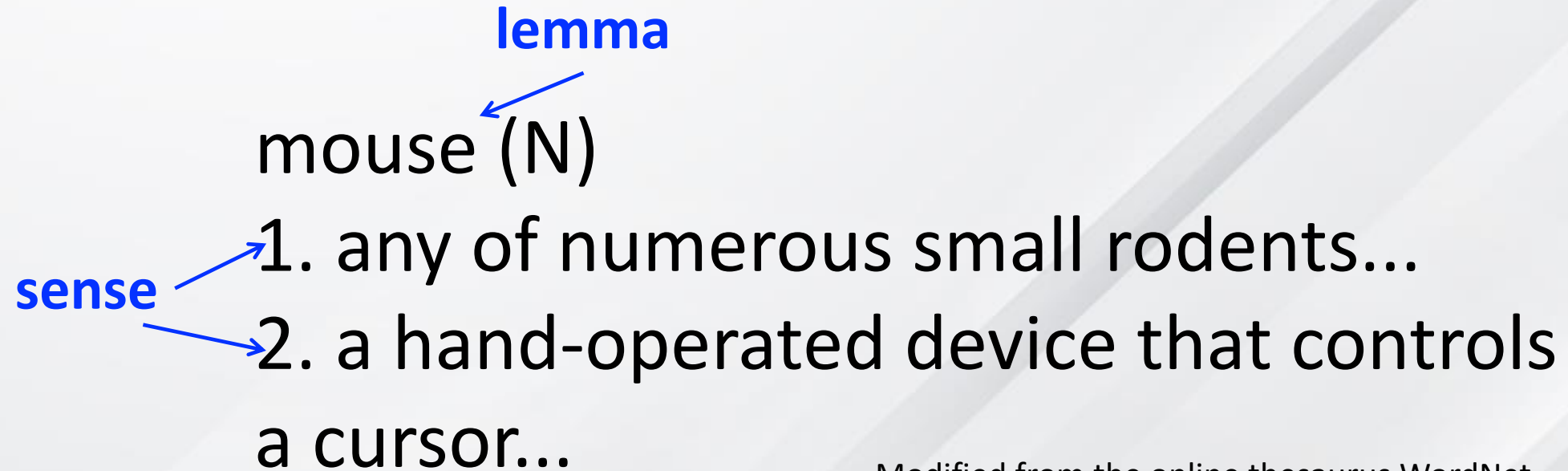
The objectives of this lecture is to:

- Understand different approaches to word representation, such as one-hot encoding, count-based methods, and predictive methods.
- Explore the concept of vector space models (VSMs) and how they are used to represent words or entities.
- Define word embeddings and their significance in natural language processing (NLP) tasks.
- Understand how word embeddings capture semantic and syntactic relationships between words.

What do words mean?

- N-gram or text classification methods we've seen so far
 - Words are just strings (or indices w_i in a vocabulary list)
 - That's not very satisfactory!
- Introductory logic classes:
 - The meaning of "dog" is DOG; cat is CAT
 $\forall x \text{ DOG}(x) \rightarrow \text{MAMMAL}(x)$

Lemmas and Senses



Modified from the online thesaurus WordNet

A **sense** or “**concept**” is the meaning component of a word
Lemmas can be **polysemous** (have multiple senses)

Relations between senses: Synonymy

- Synonyms have the same meaning in some or all contexts.
 - filbert / hazelnut
 - couch / sofa
 - big / large
 - automobile / car
 - vomit / throw up
 - water / H₂O

Relation: **Synonymy**?

water/H₂O

"H₂O" in a surfing guide?

big/large

my big sister != my large sister

Relation: Similarity

Words with similar meanings. Not synonyms, but sharing some element of meaning

car, bicycle

cow, horse

Ask humans how similar 2 words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

Relation: Word relatedness

- Also called "word association"
- Words can be related in any way, perhaps via a semantic frame or field
 - coffee, tea: **similar**
 - coffee, cup: **related**, not similar

Semantic field

- Words that
 - cover a particular semantic domain
 - bear structured relations with each other.

hospitals

surgeon, scalpel, nurse, anaesthetic, hospital

restaurants

waiter, menu, plate, food, chef

houses

door, roof, kitchen, family, bed

Relation: Antonymy

- Senses that are opposites with respect to only one feature of meaning

- Otherwise, they are very similar!

dark/light short/long fast/slow rise/fall
hot/cold up/down in/out

- More formally: antonyms can
 - define a binary opposition or be at opposite ends of a scale
 - long/short, fast/slow
 - Be *reversives*:
 - rise/fall, up/down

Connotation (sentiment)

- Words have **affective** meanings
 - Positive connotations (*happy*)
 - Negative connotations (*sad*)
- Connotations can be subtle:
 - Positive connotation: *copy, replica, reproduction*
 - Negative connotation: *fake, knockoff, forgery*
- Evaluation (sentiment!)
 - Positive evaluation (*great, love*)
 - Negative evaluation (*terrible, hate*)

So far

- **Concepts** or word senses
 - Have a complex many-to-many association with **words** (homonymy, multiple senses)
- Have relations with each other
 - Synonymy
 - Antonymy
 - Similarity
 - Relatedness
 - Connotation

Computational models of word meaning

- Can we build a theory of how to represent word meaning?
- We'll introduce **vector semantics**
 - The standard model in language processing!
 - Handles many of our goals!

Let's define words by their usages

- One way to define "usage":
 - words are defined by their environments (the words around them)
- Zellig Harris (1954):
- **If A and B have almost identical environments we say that they are synonyms.**

What does recent English borrowing *ongchoi* mean?

- Suppose you see these sentences:
 - Ong choi is delicious **sautéed with garlic**.
 - Ong choi is superb **over rice**
 - Ong choi **leaves** with salty sauces
- And you've also seen these:
 - ...spinach **sautéed with garlic over rice**
 - Chard stems and **leaves** are **delicious**
 - Collard greens and other **salty** leafy greens
- Conclusion:
 - Ongchoi is a leafy green like spinach, chard, or collard greens
 - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Ongchoi: *Ipomoea aquatica* "Water Spinach"

空心菜
kangkong
rau muống
...



Yamaguchi, Wikimedia Commons, public domain

Idea 1: Defining meaning by linguistic distribution

- Let's define the meaning of a word by its distribution in language use, meaning its neighboring words or grammatical environments.

Idea 2: Meaning as a point in space (Osgood et al. 1957)

- 3 affective dimensions for a word
 - **valence**: pleasantness
 - **arousal**: intensity of emotion
 - **dominance**: the degree of control exerted

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

NRC VAD Lexicon
(Mohammad 2018)

- Hence the connotation of a word is a vector in 3-space

Idea 1: Defining meaning by linguistic distribution

Idea 2: Meaning as a point in multidimensional space

Defining meaning as a point in space based on distribution

- Each word = a vector (not just "good" or " w_{45} ")
- Similar words are "**nearby in semantic space**"
- We build this space automatically by seeing which words are **nearby in text**



We define meaning of a word as a **vector**

- Called an **"embedding"** because it's embedded into a space
- The standard way to represent meaning in NLP
- **Every modern NLP algorithm uses embeddings as the representation of word meaning**

Intuition: why vectors?

- Consider sentiment analysis:
 - With **words**, a feature is a word identity
 - Feature 5: 'The previous word was "terrible"'
 - requires **exact same word** to be in training and test
 - With **embeddings**:
 - Feature is a word vector
 - 'The previous word was vector [35,22,17...]
 - Now in the test set we might see a similar vector [34,21,14]
 - We can generalize to **similar but unseen** words!!!

We'll discuss 2 kinds of embeddings

- **tf-idf**

- Information Retrieval workhorse!
- A common baseline model
- **Sparse** vectors
- Words are represented by (a simple function of) the **counts** of nearby words

- **Word2vec**

- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
- Later we'll discuss extensions called **contextual embeddings**

From now on: Computing with meaning representations instead of string representations

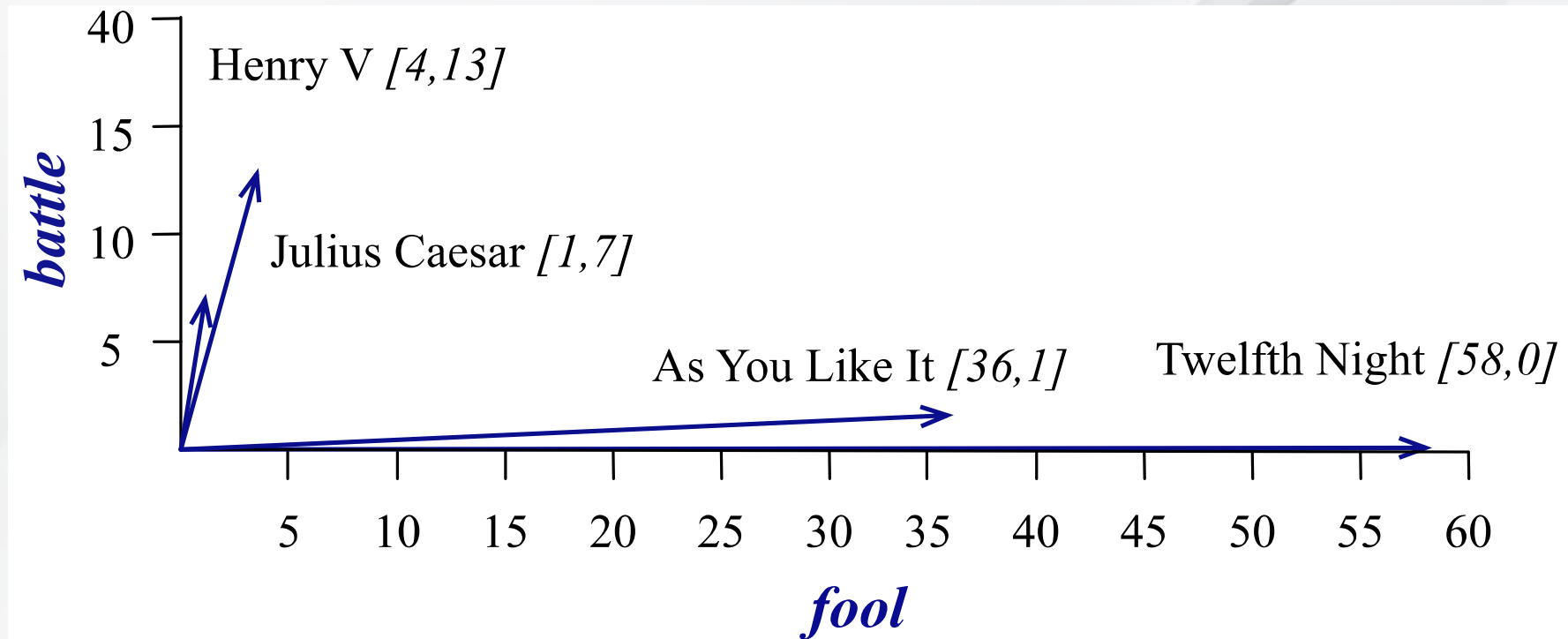
荃者所以在鱼，得鱼而忘荃 Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言 Words are for meaning;
Once you get the meaning, you can forget the words
庄子(Zhuangzi), Chapter 26

Term-document matrix

Each document is represented by a vector of words

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Visualizing document vectors



Vectors are the basis of information retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Vectors are similar for the two comedies

But comedies are different than the other two

Comedies have more *fools* and *wit* and fewer *battles*.

Idea for word meaning: Words can be vectors too!!!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

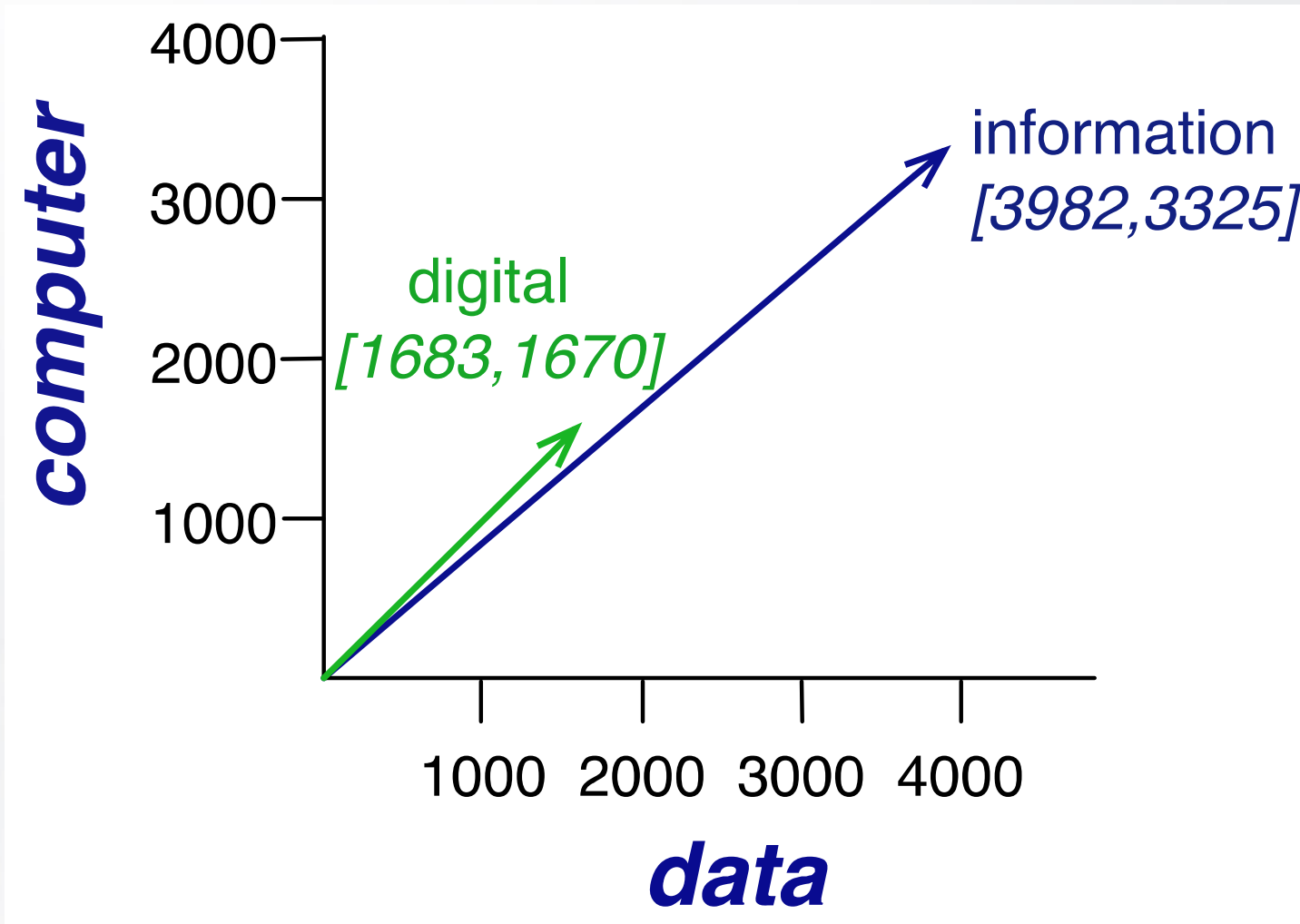
fool is "the kind of word that occurs in comedies, especially Twelfth Night"

More common: word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Computing word similarity: Dot product and Cosine

- The dot product between two vectors is a scalar:

$$\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$$

- The dot product tends to be high when the two vectors have large values in the same dimensions
- Dot product can thus be a useful similarity metric between vectors

Problem with raw dot-product

- Dot product favors long vectors
- Dot product is higher if a vector is longer (has higher values in many dimension)
- Vector length:

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

- Frequent words (of, the, you) have long vectors (since they occur many times with other words).
- We'd like a similarity metric that tells us how similar two words are regardless of their frequency.

Alternative: cosine for computing word similarity

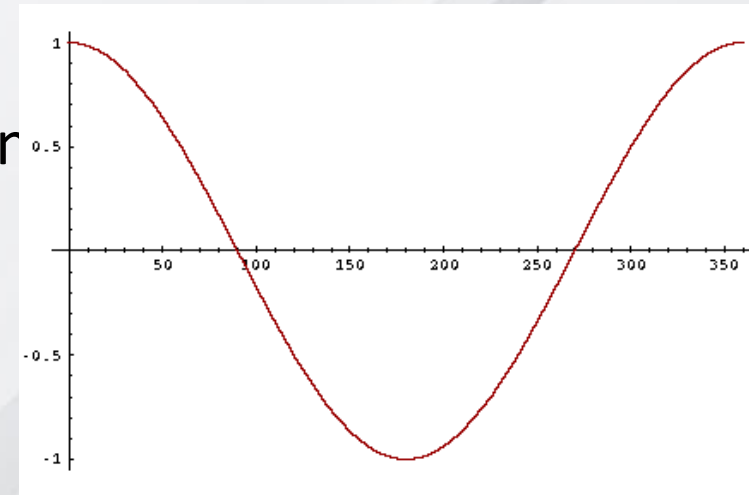
$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

Based on the definition of the dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$

$$\begin{aligned}\mathbf{a} \cdot \mathbf{b} &= |\mathbf{a}| |\mathbf{b}| \cos \theta \\ \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} &= \cos \theta\end{aligned}$$

Cosine as a similarity metric

- -1: vectors point in opposite direction
- +1: vectors point in same directions
- 0: vectors are orthogonal



- But since raw frequency values are non-negative, the cosine for term-term matrix vectors ranges from 0–1

Cosine examples

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

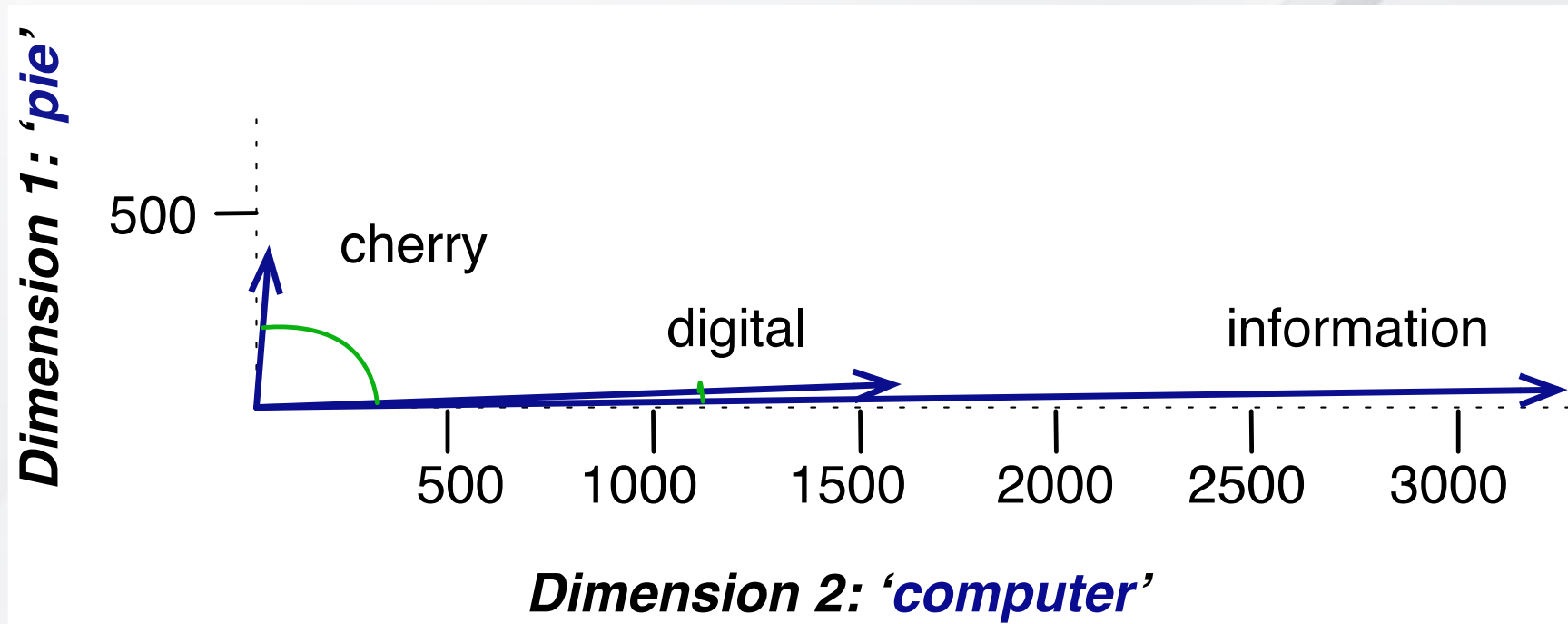
$$\cos(\text{cherry}, \text{information}) =$$

$$\frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital}, \text{information}) =$$

$$\frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

Visualizing cosines (well, angles)



The Three Documents and Similarity Metrics



Considering only the 3 words from the above documents: 'sachin', 'dhoni', 'cricket'

Doc Sachin: Wiki page on Sachin Tendulkar

Dhoni - 10
Cricket - 50
Sachin - 200

Doc Dhoni: Wiki page on Dhoni

Dhoni - 400
Cricket - 100
Sachin - 20

Doc Dhoni_Small: Subsection of wiki on Dhoni

Dhoni - 10
Cricket - 5
Sachin - 1

Document - Term Matrix (Word Counts)

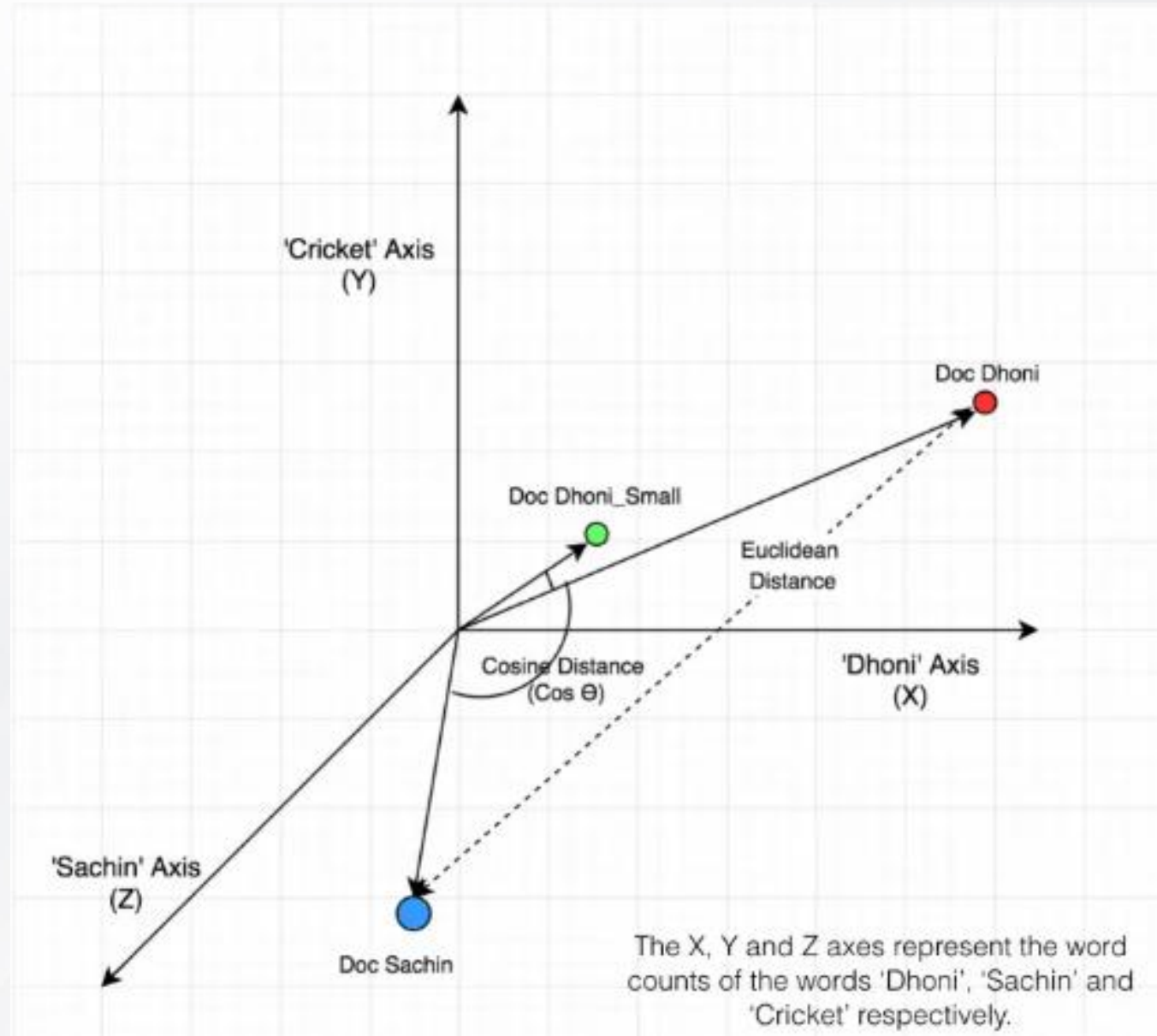
Word Counts	"Dhoni"	"Cricket"	"Sachin"
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1



Similarity Metrics

Similarity or Distance Metrics	Total Common Words	Euclidean distance	Cosine Similarity
Doc Sachin & Doc Dhoni	$10 + 50 + 10 = 70$	432.4	0.15
Doc Dhoni & Doc Dhoni_Small	$20 + 10 + 7 = 37$	204.0	0.23
Doc Sachin & Doc Dhoni_Small	$10 + 10 + 7 = 27$	401.85	0.77

As you can see, all three documents are connected by a common theme – the game of Cricket.



Let's see the example of how to calculate the cosine similarity between two text document.

```
doc_1 = "Data is the oil of the digital economy"
doc_2 = "Data is a new oil"
```

Vector representation of the document

```
doc_1_vector = [1, 1, 1, 1, 0, 1, 1, 2]
doc_2_vector = [1, 0, 0, 1, 1, 0, 1, 0]
```

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

$$A \cdot B = \sum_{i=1}^n A_i B_i$$

$$= (1 * 1) + (1 * 0) + (1 * 0) + (1 * 1) + (0 * 1) + (1 * 0) + (1 * 1) + (2 * 0)$$

$$= 3$$

$$\sqrt{\sum_{i=1}^n A_i^2} = \sqrt{1 + 1 + 1 + 1 + 0 + 1 + 1 + 4} = \sqrt{10}$$

$$\sqrt{\sum_{i=1}^n B_i^2} = \sqrt{1 + 0 + 0 + 1 + 1 + 0 + 1 + 0} = \sqrt{4}$$

$$\text{cosine similarity} = \cos\theta = \frac{A \cdot B}{|A||B|} = \frac{3}{\sqrt{10} * \sqrt{4}} = 0.4743$$

The **Cosine Similarity** is a better metric than **Euclidean distance** because if the two text document far apart by *Euclidean* distance, there are still chances that they are close to each other in terms of their context.

Compute Cosine Similarity in Python

Let's compute the Cosine similarity between two text document and observe how it works.

The common way to compute the Cosine similarity is to first we need to count the word occurrence in each document. To count the word occurrence in each document, we can use **CountVectorizer** or **TfidfVectorizer** functions that are provided by Scikit-Learn library.

Define the Data

Let's define the sample text documents and apply CountVectorizer on it.

```
doc_1 = "Data is the oil of the digital economy"  
doc_2 = "Data is a new oil"  
  
data = [doc_1, doc_2]
```

Call CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

count_vectorizer = CountVectorizer()
vector_matrix = count_vectorizer.fit_transform(data)
vector_matrix
```

```
<2x8 sparse matrix of type '<class 'numpy.int64'>'
  with 11 stored elements in Compressed Sparse Row format>
```

The generated vector matrix is a sparse matrix, that is not printed here. Let's convert it to numpy array and display it with the token word.
Here, is the unique tokens list found in the data.

```
tokens = count_vectorizer.get_feature_names()  
tokens
```

```
['data', 'digital', 'economy', 'is', 'new', 'of', 'oil', 'the']
```

Convert sparse vector matrix to numpy array to visualize the vectorized data of doc_1 and doc_2.

```
vector_matrix.toarray()
```

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

Let's create the pandas DataFrame to make a clear visualization of vectorize data along with tokens.

```
import pandas as pd

def create_dataframe(matrix, tokens):

    doc_names = [f'doc_{i+1}' for i, _ in enumerate(matrix)]
    df = pd.DataFrame(data=matrix, index=doc_names, columns=tokens)
    return(df)
```

```
create_dataframe(vector_matrix.toarray(),tokens)
```

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

Find Cosine Similarity

Scikit-Learn provides the function to calculate the [Cosine similarity](#). Let's compute the Cosine Similarity between **doc_1** and **doc_2**.

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix, ['doc_1', 'doc_2'])
```

	doc_1	doc_2
doc_1	1.000000	0.474342
doc_2	0.474342	1.000000

By observing the above table, we can say that the Cosine Similarity between **doc_1** and **doc_2** is **0.47**

But raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies.
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it*, or *they* are not very informative about the context
- It's a paradox! How can we balance these two conflicting constraints?

Two common solutions for word weighting

- **tf-idf:** tf-idf value for word t in document d :

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

Words like "the" or "it" have very low idf

- **PMI:** (Pointwise mutual information)

- $\text{PMI}(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$

See if words like "good" appear more often with "great" than we would expect by chance

Term frequency (tf)

- $tf_{t,d} = \text{count}(t,d)$
- Instead of using raw count, we squash a bit:
- $tf_{t,d} = \log_{10}(\text{count}(t,d)+1)$

Document frequency (df)

- df_t is the number of documents t occurs in.
- (note this is not collection frequency: total count across all documents)
- "*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left(\frac{N}{\text{df}_t} \right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

What is a document?

- Could be a play or a Wikipedia article
- But for the purposes of tf-idf, documents can be **anything**; we often call each paragraph a document!

Final tf-idf weighted value for a word

- Raw counts:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- tf-idf:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Solved Example

Corpus D

d_1 A quick brown fox jumps over the lazy dog. What a fox!

d_2 A quick brown fox jumps over the lazy fox. What a fox!

Question: How word fox is relevant to corpus D documents?

Corpus D

d_1 A quick brown fox jumps over the lazy dog. What a fox!

d_2 A quick brown fox jumps over the lazy fox. What a fox!

Question: How word fox is relevant to corpus D documents?

Solution:

TF-IDF

TF is the frequency of any "term" in a given "document".

TF("fox", d_1) =

TF("fox", d_2) =

Corpus D

d_1 A quick brown **fox** jumps over the lazy dog. What a **fox**!
 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
 1 2 3 4 5 6 7 8 9 10 11 12

d_2 A quick brown **fox** jumps over the lazy **fox**. What a **fox**!
 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
 1 2 3 4 5 6 7 8 9 10 11 12

Question: How word **fox** is relevant to corpus D documents?

Solution:

TF-IDF

TF is the frequency of any "term" in a given "document".

$$TF(\text{"fox"}, d1) = 2 / 12 = 0.17$$

$$TF(\text{"fox"}, d2) = 3 / 12 = 0.25$$

Corpus D

d_1 A quick brown fox jumps over the lazy dog. What a fox!

d_2 A quick brown fox jumps over the lazy fox. What a fox!

Question: How word fox is relevant to corpus D documents?

Solution:

TF-IDF

TF is the frequency of any "term" in a given "document".

IDF is constant per corpus, and accounts for the ratio of documents that include that specific "term".

$TF(\text{"fox"}, d_1) = 2 / 12 = 0.17$

$TF(\text{"fox"}, d_2) = 3 / 12 = 0.25$

$IDF(\text{"fox"}, D) = \log(2/2) = 0$

Corpus D

$+1 \rightarrow d_1$	A quick brown fox jumps over the lazy dog. What a fox!	TF-IDF = $0.17 \times 0 = 0$ ("fox" , d1, D)
$+1 \rightarrow d_2$	A quick brown fox jumps over the lazy fox. What a fox!	TF-IDF = $0.25 \times 0 = 0$ ("fox" , d2, D)

Question: How word **fox** is relevant to corpus D documents?

Solution:

TF-IDF

TF is the frequency of any "term" in a given "document".

IDF is constant per corpus, and accounts for the ratio of documents that include that specific "term".

TF(**"fox"**, d1) = $2 / 12 = 0.17$

TF(**"fox"**, d2) = $3 / 12 = 0.25$

IDF(**"fox"**, D) = $\log(2/2) = 0$

Corpus D	
+1 → d ₁	A quick brown fox jumps over the lazy dog. What a fox!
+1 → d ₂	A quick brown fox jumps over the lazy fox. What a fox!

$$TF-IDF = 0.17 \times 0 = 0$$

$TF-IDF("fox", d1, D)$

$$TF-IDF = 0.25 \times 0 = 0$$

$TF-IDF("fox", d2, D)$

Answer: Using TF-IDF, the word "fox" is equally relevant for both document d1 and document d2

Question: How word fox is relevant to corpus D documents?

Solution:

TF is the frequency of any "term" in a given "document".

TF-IDF

IDF is constant per corpus, and accounts for the ratio of documents that include that specific "term".

$$TF("fox", d1) = 2 / 12 = 0.17$$

$$TF("fox", d2) = 3 / 12 = 0.25$$

$$IDF("fox", D) = \log(2/2) = 0$$

Pointwise Mutual Information

- **Pointwise mutual information:**

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X, Y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
 - Things are co-occurring **less than** we expect by chance
 - Unreliable without enormous corpora
 - Imagine w_1 and w_2 whose probability is each 10^{-6}
 - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}
- So we just replace negative PMI values by 0
- Positive PMI (**PPMI**) between word1 and word2:
$$\text{PPMI}(\text{word}_1, \text{word}_2) = \max\left(\log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}, 0\right)$$

Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^C f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^W f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*} p_{*j}} \quad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

- $p(w=\text{information}, c=\text{data}) = 3982/11716 = .3399$
- $p(w=\text{information}) = 7703/11716 = .6575$
- $p(c=\text{data}) = 5673/11716 = .4842$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N} \quad p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*} p_{*j}}$$

- $pmi(\text{information}, \text{data}) = \log_2 (.3399 / (.6575 * .4842)) = .0944$

Resulting PPMI matrix (negatives replaced by 0)

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Sparse versus dense vectors

- tf-idf (or PMI) vectors are
 - **long** (length $|V| = 20,000$ to $50,000$)
 - **sparse** (most elements are zero)
- Alternative: learn vectors which are
 - **short** (length 50-1000)
 - **dense** (most elements are non-zero)



Word Embeddings

Summary

- Understand different approaches to word representation, such as one-hot encoding, count-based methods, and predictive methods.
- Explore the concept of vector space models (VSMs) and how they are used to represent words or entities.
- Define word embeddings and their significance in natural language processing (NLP) tasks.
- Understand how word embeddings capture semantic and syntactic relationships between words.



Thank You