Cairo University
Faculty of Engineering
Computer Engineering Department

Natural Language Processing

Dr. Sandra Wahid

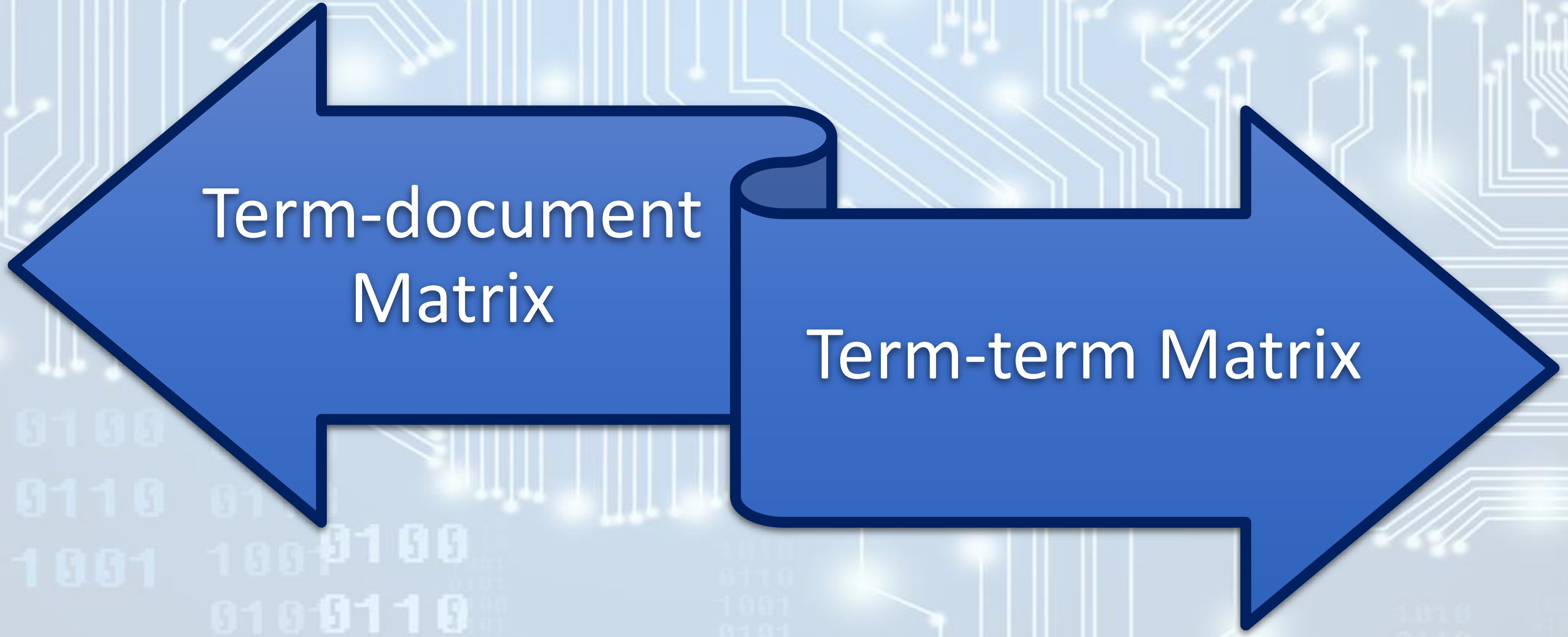
Words and Vectors

- **Distributional Hypothesis**: words that occur **in similar contexts** tend to have **similar meanings**.
- Words which are **synonyms** (like *oculist* and *eye-doctor*) tended to occur in the same environment (e.g., near words like eye or examined).
- **Vectors Semantics**: is the standard way to represent word meaning in NLP.
- **Word Embeddings/Word Vectors**: are learned representations of the meaning of words.
  - Vectors for representing words are called embeddings

Words and Vectors

- Distributional models of meaning are generally based on a **co-occurrence matrix**, a way of representing how often words co-occur.

- Two popular matrices:



Term-document Matrix

- Each row represents a word in the vocabulary and each column represents a document from some collection of documents

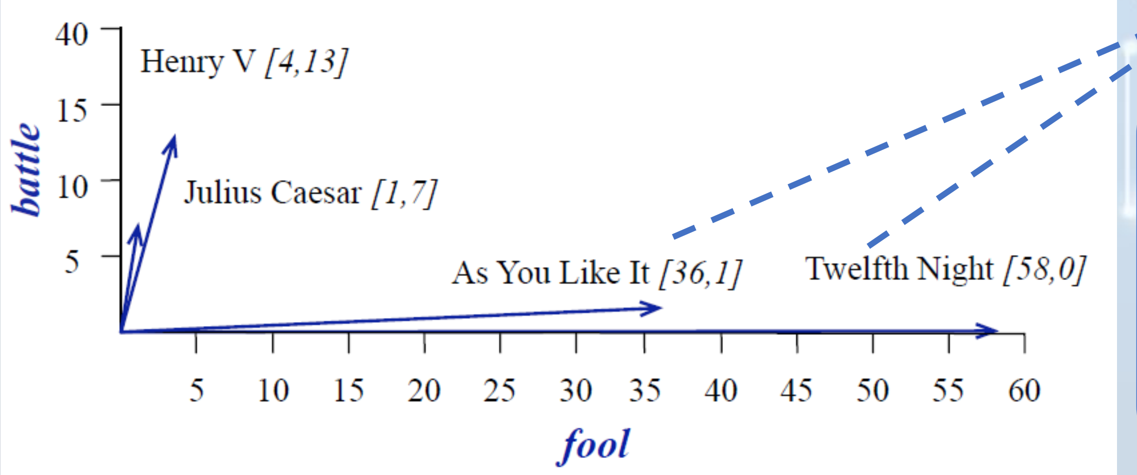
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

The red boxes show that each document is represented as a column vector of length 4.

In real term-document matrices, the vectors representing each document would have dimensionality **|V|→the vocabulary size**. We can think of the vector for a document as a point in |V| dimensional space.

- A visualization in two dimensions:



The comedies have high values for the fool dimension and low values for the battle dimension

Term-document matrices were originally defined as a means of finding **similar documents** for the task of document **information retrieval**.
Two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar.

Term-document Matrix

- Using this matrix, we can also think of representing words as vectors using document dimensions.

	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

The red boxes show that each word is represented as a row vector of length 4.

- For documents, we saw that similar documents had similar vectors, because similar documents tend to have similar words.

→This same principle applies to words:

- Similar words have similar vectors because they tend to occur in similar documents.
- The term-document matrix thus lets us represent the meaning of a word by the documents it tends to occur in.

Term-term Matrix

- Also called the **word-word matrix** or the **term-context matrix**, in which the columns are labeled by words rather than documents.
- This matrix is of dimensionality $|V| \times |V|$.
- Each cell records the number of times the **row (target) word** and the **column (context) word** co-occur in some context in some training corpus.
- The context is mostly a **window** around the word, for example *4 words* to the left and *4 words* to the right, the cell represents the number of times (in some training corpus) the column word occurs in such a ±4 word window around the row word.
- Example: using a context window of 2:

Corpus: He is not lazy. He is intelligent. He is smart. Unique Words: ['He', 'is', 'not', 'lazy', 'intelligent', 'smart']

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1
not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

The word vector for "He"

The word vector for "intelligent"

The dimensionality of the vector, is generally the **size of the vocabulary** → most of these numbers are zeros so these are **sparse** vector representations

Cosine Similarity Measure

- To measure similarity between two target words *v* and *w*, we compute the dot product between the vectors of these two words.

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

- The dot product acts as a similarity metric because it will tend to be high just when the two vectors have large values in the same dimensions. Alternatively, vectors that have zeros in different dimensions—orthogonal vectors—will have a dot product of 0, representing their strong dissimilarity.

What is the problem with this raw dot product measure?

It favors long vectors. The vector length is defined as:

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

More frequent words have longer vectors, since they tend to co-occur with more words and have higher co-occurrence values with each of them.

→ The raw dot product will be higher for frequent words. We'd like a similarity metric that tells us how similar two words are regardless of their frequency.

Cosine Similarity Measure

- **Solution:** modify the dot product to **normalize** for the vector length by dividing the dot product by the lengths of each of the two vectors.

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

The normalized dot product turns out to be the same as the **cosine of the angle** between the two vectors

- The **cosine similarity** metric between two vectors *v* and *w* thus can be computed as:

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

- For some applications we pre-normalize each vector, by dividing it by its length, creating a unit vector of length 1→For unit vectors, the dot product is the same as the cosine.

Cosine Similarity Measure

- The cosine value ranges from 1 for vectors pointing in the same direction, through 0 for orthogonal vectors, to -1 for vectors pointing in opposite directions.

→But since raw frequency values are non-negative, the cosine for these vectors ranges from 0–1.

- Example:

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

$$\cos(\text{cherry}, \text{information}) = \frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .018$$

$$\cos(\text{digital}, \text{information}) = \frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

The model decides that information is way closer to digital than it is to cherry

Weighting Functions

- The co-occurrence matrices represent each cell by **frequencies**. But raw frequency is not the best measure of association between words.

- If we want to know what kinds of contexts are shared by *cherry* and *strawberry* but not by *digital* and *information*, we're not going to get good discrimination from words like *the*, *it*, or *they*, which occur frequently with all sorts of words and **aren't informative** about any particular word.

Words that **occur nearby frequently** (maybe *pie* nearby *cherry*) are more important than words that only appear once or twice.
Yet words that are **too frequent**, like *the* or *they* are unimportant.

How can we balance these two conflicting constraints?

Two common solutions:

TF-IDF Weighting

PPMI Algorithm

TF-IDF

- Stands for **Term Frequency-Inverse Document Frequency.**

- **Term Frequency**: the frequency of the word *t* in the document *d:* $\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$
- More commonly we squash the raw frequency a bit, by using the $\log_{10}$ of the frequency instead. The intuition is that a word appearing 100 times in a document doesn't make that word 100 times more likely to be relevant to the meaning of the document. Because we can't take the log of 0, we normally add 1 to the count: $\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1)$
- We can use normalized $\boldsymbol{tf_{t,d} = \frac{count(t,d)}{number\ of\ words\ in\ d}}$ (this will be our default)

- The second factor in tf-idf is used to give a higher weight to words that occur only in a few documents.
  - Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection while terms that occur frequently across the entire collection aren't as helpful.

TF-IDF

- **Inverse Document Frequency**: the *idf* is defined using the fraction N/df_t
  - *N* is the total number of documents in the collection.
  - df_t is the number of documents in which term *t* occurs.
  - The fewer documents in which a term occurs, the higher this weight.
  - The lowest weight of 1 is assigned to terms that occur in all the documents.
- Because of the large number of documents in many collections, this measure too is usually squashed with a log function: $\boxed{\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)}$ (this will be our default)
- The tf-idf weighted value $w_{t,d}$ for word *t* in document *d*: $w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$
- Example:

doc A: The car is driven on the road.

doc B: The truck is driven on the highway.

Word	TF A	TF B	IDF	TF*IDF A	TF*IDF B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Think of it as: Terms that occur frequently in a particular document are likely more important than terms that occur frequently in all documents in the corpus.

Pointwise Mutual Information (PMI)

- PPMI (positive pointwise mutual information) is an alternative weighting function to tf-idf.
  - It is used for **term-term-matrices**, when the vector dimensions correspond to words rather than documents.
- Pointwise mutual information is a measure of how often two events x and y occur, compared with what we would expect if they were independent.
- The pointwise mutual information between a target word *w* and a context word *c:*

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

PMI is a useful tool whenever we need to find words that are strongly associated.

- PMI values range from negative to positive infinity → Negative PMI values (which imply things are co-occurring less often than we would expect by chance) tend to be unreliable unless our corpora are enormous.

Pointwise Mutual Information (PMI)

- **<u>Solution:</u>** it is more common to use **Positive PMI** (called PPMI) which replaces all negative PMI values with zero:

$$\text{PPMI}(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0\right)$$

- Formally: we have a co-occurrence matrix *F* with *W* rows (words) and *C* columns (contexts), where f_{ij} gives the number of times word w_i occurs in context c_j.
- This can be turned into a PPMI matrix where $PPMI_{ij}$ gives the PPMI value of word w_i with context c_j referred to as: $PPMI(w_i,c_j)$ or $PPMI(w = i, c = j)$:

Number of co-occurrences of w and c

Number of instances of w

Number of instances of c

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

Total words count

$$\text{PPMI}_{ij} = \max\left(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0\right)$$

Pointwise Mutual Information (PMI)

- **Example** (for ease of calculations these are the only words/contexts that matter)

co-occurrence counts:

	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

PPMI(information,data):

$$P(\text{w=information, c=data}) = \frac{3982}{11716} = .3399$$

$$P(\text{w=information}) = \frac{\boxed{7703}}{11716} = .6575$$

$$P(\text{c=data}) = \frac{\boxed{5673}}{11716} = .4842$$

$$\text{PPMI(information,data)} = \log 2(.3399/(.6575 * .4842)) = .0944$$

Probabilities computed from the counts:

	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	

PPMI matrix:

	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

cherry and *strawberry* are highly associated with both *pie* and *sugar*, and *data* is mildly associated with *information*.

Pointwise Mutual Information (PMI)

- PMI has the problem of being **biased toward infrequent events** → very rare words tend to have very high PMI values.
- **<u>Solutions:</u>**

1. **Slightly change the computation for *P(c)***, using a different function $P_\alpha(c)$that raises the probability of the context word to the power of α: (a setting of α=0.75 improved performance)

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

$$\boxed{P_\alpha(c) > P(c) \text{ when } c \text{ is rare}}$$

To illustrate this intuition, consider two events, P(a) = 0.99 and P(b) = 0.01

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

2. **Laplace smoothing**: Before computing PMI, a small constant *k* (values of 0.1-3 are common) is added to each of the counts, shrinking (discounting) all the non-zero values. The larger the *k*, the more the non-zero counts are discounted.

Applications of the TF-IDF & PPMI Vector Models

- A target word is represented as a vector:
  - of dimension=the number of **documents** in a large collection (the **term-document** matrix).
  - of dimension=the number of **words** in vocabulary (the **term-term** matrix).
- The elements of the vectors are weights: **tf-idf** (for **term-document** matrices) or **PPMI** (for **term-term** matrices).
- The **vectors are sparse** (since most values are zero).

- The **document vector** can be computed by taking the vectors of all the words in the document and computing the centroid of all those vectors: $d = \frac{w_1 + w_2 + ... + w_k}{k}$

- Documents' similarity can be computed as $cos(d_1,d_2)$
  - Useful applications: information retrieval, plagiarism detection, news recommender systems, …
- Words' similarity can be computed as $cos(w_1,w_2)$
  - For example, we can find the 10 most similar words to any target word *w* by computing the cosines between *w* and each of the *V-1* other words, sorting, and looking at the top 10.


Thank You