Vector Semantics

Natalie Parde, Ph.D.

Department of Computer Science

University of Illinois at Chicago

CS 521: Statistical Natural Language Processing

Spring 2020

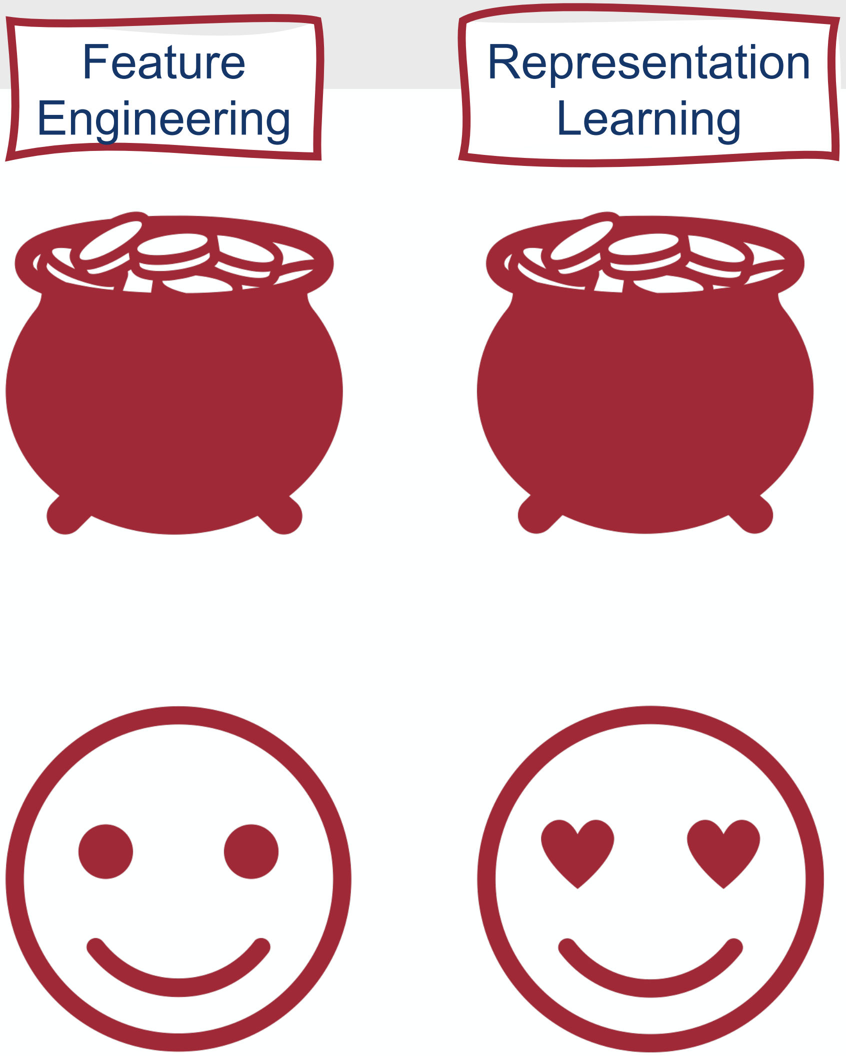
Many slides adapted from Jurafsky and Martin (https://web.stanford.edu/~jurafsky/slp3/).

What is vector semantics?

- A form of **representation learning** based on the notion that similar words tend to occur in similar environments
- Representations learned in this manner are typically referred to as **word embeddings**

Representation Learning

- The process of automatically learning useful representations of input text, in a self-supervised manner
- Recent trends have moved toward representation learning and away from creating representations by hand (i.e., by **feature engineering**)



The corresponding notion of encoding words based on their distribution is referred to as the distributional hypothesis.

- First formulated by linguists in the 1950s
  - Joos (1950)
  - Harris (1954)
  - Firth (1957)

Vector Semantics

For my assignment I'm writing a scathing **critique** of Dr. Parde's recent paper.

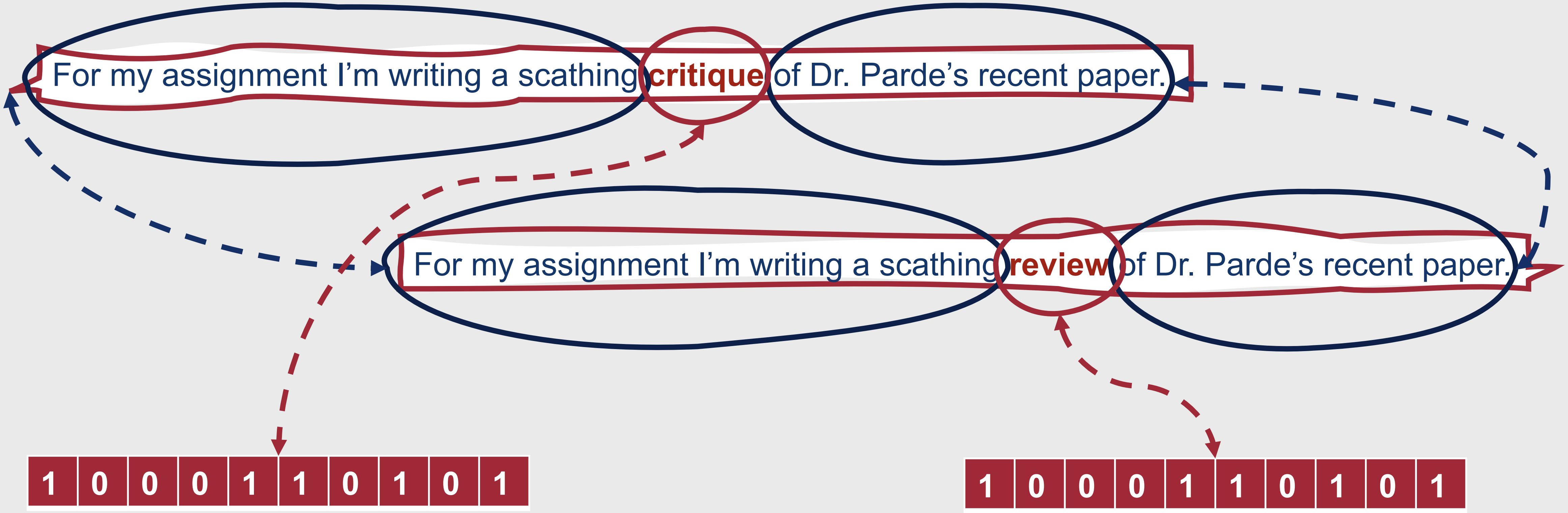
For my assignment I'm writing a scathing **review** of Dr. Parde's recent paper.

Vector Semantics

For my assignment I'm writing a scathing **critique** of Dr. Parde's recent paper.

For my assignment I'm writing a scathing **review** of Dr. Parde's recent paper.

Vector Semantics



There are many ways to make use of the distributional hypothesis!

- **Classical word vectors**
  - Bag of words representations
- **Non-contextual word embeddings**
  - Word2Vec
  - GloVe
- **Contextual word embeddings**
  - ELMo
  - BERT

There are many ways to make use of the distributional hypothesis!

- **Classical word vectors**
  - Bag of words representations
- **Non-contextual word embeddings**
  - Word2Vec
  - GloVe
- **Contextual word embeddings**
  - ELMo
  - BERT

This week

Later this semester

A brief foray into lexical semantics….

- Key linguistics concepts and terminology (and useful properties of words):
  - Lemmas and senses
  - Synonymy
  - Word similarity
  - Word relatedness
  - Frames and roles
  - Connotation

Lemmas and Senses

- **Lemma:** The base form of a word
  - Papers → paper
  - Mice → mouse
- **Word Sense:** Different aspects of meaning for a word
  - Mouse (1): A small rodent
  - Mouse (2): A device to control a computer cursor
- Words with the same lemma should (hopefully!) reside near one another in vector space
- Different senses of words should be represented as different vectors in **contextual word representations**, but not in **classic word vectors** or **non-contextual word representations**

Synonymy

- When a word sense for one word is (nearly) identical to the word sense for another word
- **Synonymy:** Two words are synonymous if they are substitutable for one another in any sentence without changing the situations in which the sentence would be true
  - This means that the words have the same **propositional meaning**

For my assignment I'm writing a scathing **critique** of Dr. Parde's recent paper.

For my assignment I'm writing a scathing **review** of Dr. Parde's recent paper.

Word Similarity

- Words don't often have that many synonyms, but they do have a lot of **similar** words
  - Review ≈ summary
- Good way to check if two words are similar: Can word *Y* be commonly used in the same context as word *X*?
  - I'm writing a summary 🙂
  - Did you submit your summary yet? 🙂
  - That is a scathing summary 🤨

Word Relatedness

- Sometimes words are **related**, but not similar, to one another
- **Word Relatedness:** An association between words based on their shared participation in an event or **semantic field**
  - **Semantic Field:** A set of words covering a semantic domain
    - Restaurant: {waiter, menu, plate, food, …, chef}

Semantic Frames

- **Semantic Frame:** A set of words that denote perspectives or participants in a particular type of event
  - Commercial Transaction = {buyer, seller, goods, money}
- **Semantic Role:** A participant's underlying role with respect to the main verb in the sentence

Connotation

- Also referred to as **affective meaning**
- The aspects of a word's meaning that are related to a writer or reader's emotions, sentiment, opinions, or evaluations
- Generally three dimensions:
  - **Valence:** Positivity
    - High: Happy, satisfied
    - Low: Unhappy, annoyed
  - **Arousal:** Intensity of emotion
    - High: Excited, frenzied
    - Low: Relaxed, calm
  - **Dominance:** Degree of control
    - High: Important, controlling
    - Low: Awed, influenced

Connotation (Continued)

- Following this line of thought, each word can be represented by three numbers, corresponding to its value on each of the three affective dimensions

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Connotation (Continued)

- Following this line of thought, each word can be represented by three numbers, corresponding to its value on each of the three affective dimensions

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Word vector! (Osgood et al., 1957)

How, then, should we represent the meaning of a word?

- Two classic strategies:
  - **Bag of words representations:** A word is a string of letters, or an index in a vocabulary list
  - **Logical representation:** A word defines its own meaning (“dog” = DOG)

How, then, should we represent the meaning of a word?

- Two classic strategies:
  - **Bag of words representations:** A word is a string of letters, or an index in a vocabulary list
  - ~~**Logical representation:** A word defines its own meaning (“dog” = DOG)~~

Back to our discussion of vector semantics!

- Under the distributional hypothesis, we define a word by its **environment** or its **distribution** in language use
- This corresponds to the set of **contexts** in which the word occurs
  - **Context:** Neighboring words or grammatical environments
- **Two words with very similar sets of contexts (i.e., similar distributions) are assumed to have very similar meanings**

We do this to infer meaning in the real world all the time.

- Pretend you don't know what the Cantonese word *ongchoi* means
- However, you read the following sentences:
  - Ongchoi is delicious sautéed with garlic.
  - Ongchoi is superb over rice.
  - …ongchoi leaves with salty sauces…
- You've seen many of the other context words in these sentences previously:
  - …spinach sautéed with garlic over rice…
  - …chard stems and leaves are delicious…
  - …collard greens and other salty leafy greens…
- Your (correct!) conclusion?
  - Ongchoi is probably a leafy green similar to spinach, chard, or collard greens

Our goal in NLP is to do the same thing computationally.

- How would we do this in the sample case from the previous slide?
  - Count the words in the context of *ongchoi*
  - See what other words occur in those same contexts

We can represent a word's context using vectors.

- Define a word as a single vector point in an *n*-dimensional space
  - For bag of words representations, *n* = vocabulary size
- Represent the presence or absence of words in its surrounding context using numeric values
  - For bag of words representations, the value stored in a dimension *n* corresponds to the presence of a context word *c* in close proximity to the target word *w*

The goal is for the values in these vector representations to correspond with dimensions of meaning.

- Assuming this is the case, we should be able to:
  - Cluster vectors into semantic groups
  - Perform operations that are semantically intuitive

The goal is for the values in these vector representations to correspond with dimensions of meaning.

- Assuming this is the case, we should be able to:
  - Cluster vectors into semantic groups
  - Perform operations that are semantically intuitive

$\boxed{\text{summary}}$

+

$\boxed{\text{analysis}}$

=

$\boxed{\text{critique}}$

How do we build vector representations of meaning in a bag of words model?

critique

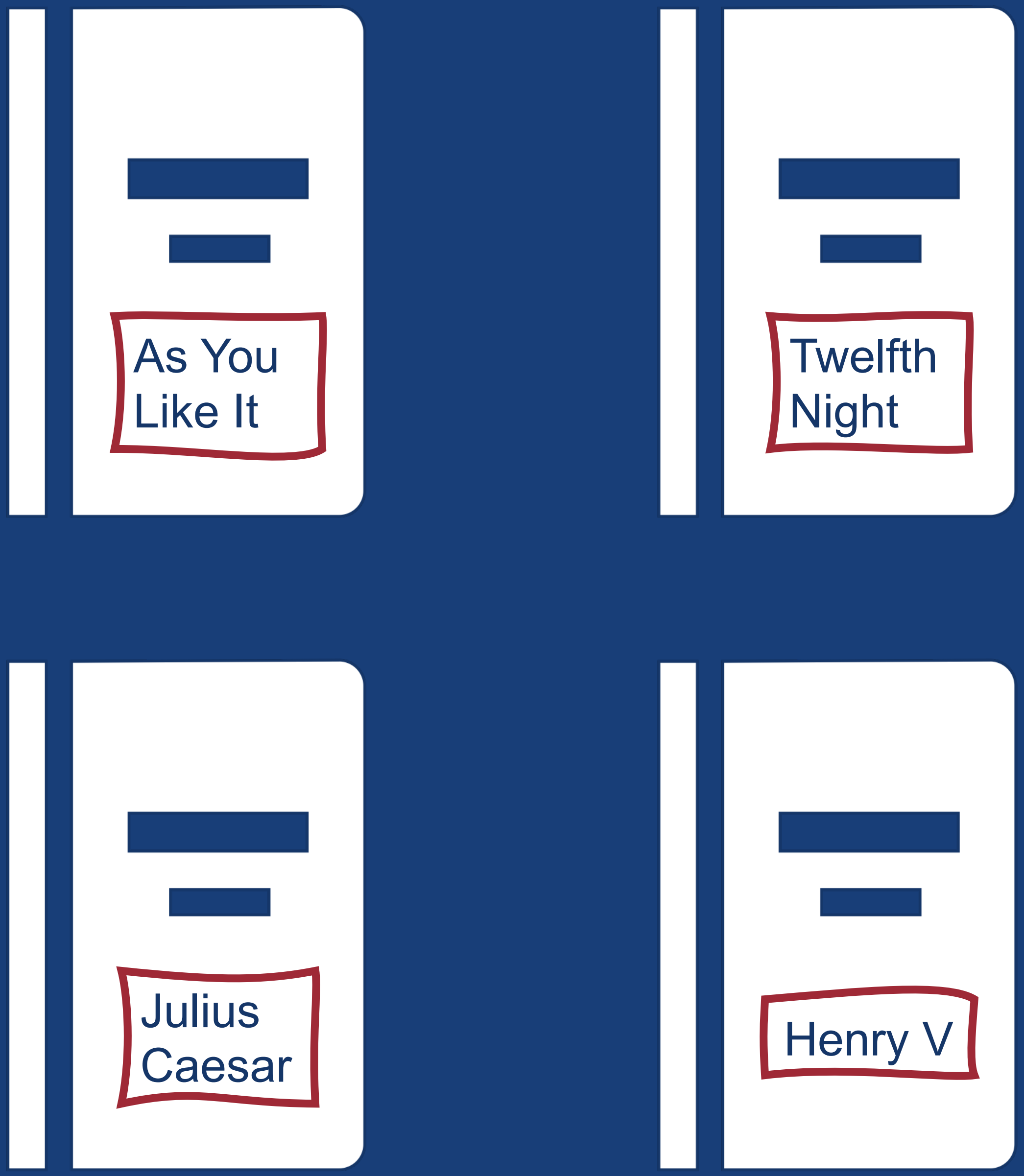
	c_1	…	critique	…	c_n
w_1	…	…	…	…	…
…	…	…	…	…	…
critique	?	?	?	?	?
…	…	…	…	…	…
w_n	…	…	…	…	…

One Approach: TF*IDF

- Term Frequency * Inverse Document Frequency
- Meaning of a word is defined by the counts of nearby words
- To do this, a **co-occurrence matrix** is needed

Word co-occurrence matrices originated from term-document matrices for information retrieval.

- Rows: Words in a vocabulary
- Columns: Documents in a selection

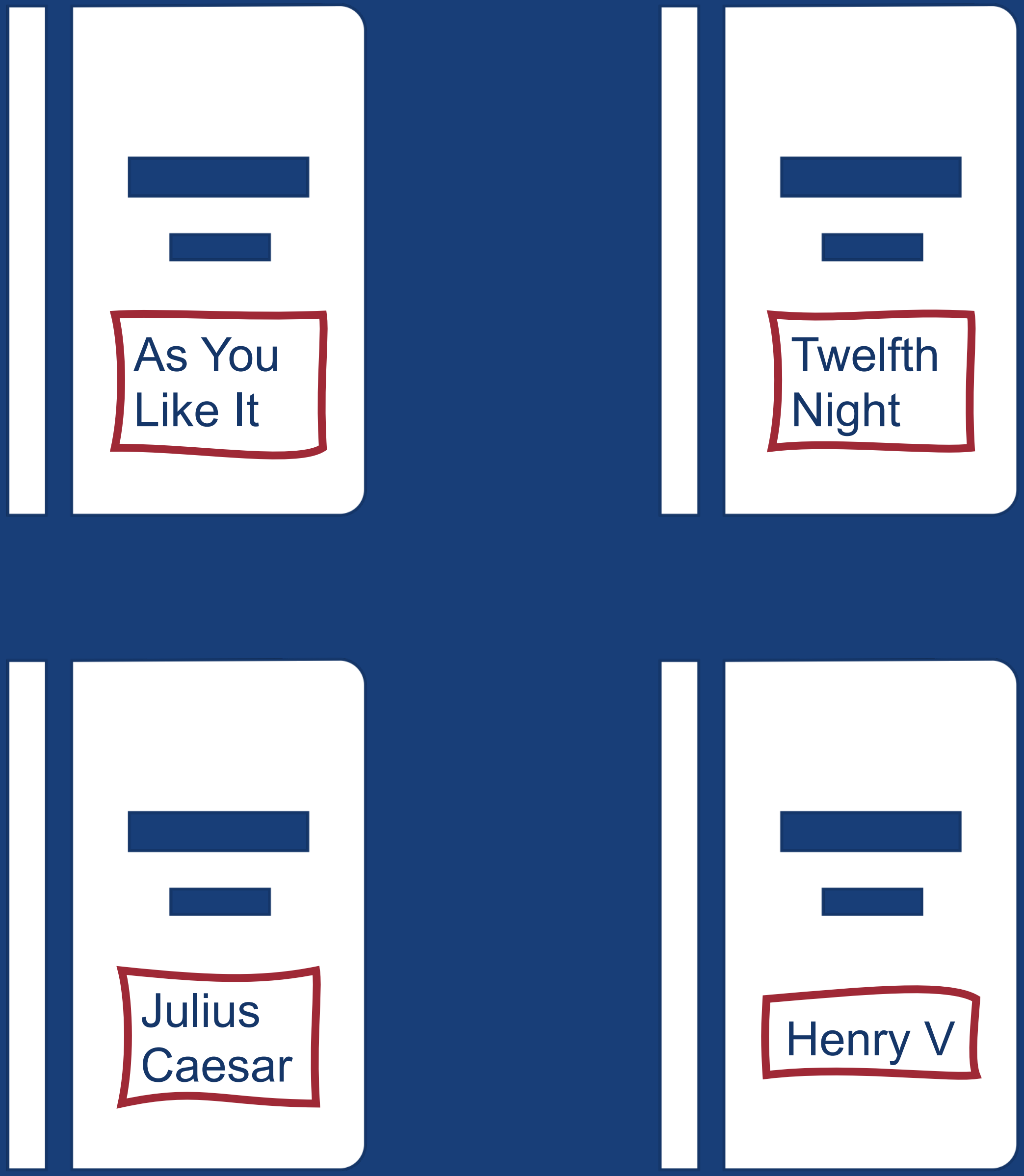


Word co-occurrence matrices originated from term-document matrices for information retrieval.

- Rows: Words in a vocabulary
- Columns: Documents in a selection

	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

"wit" appears 3 times in Henry V

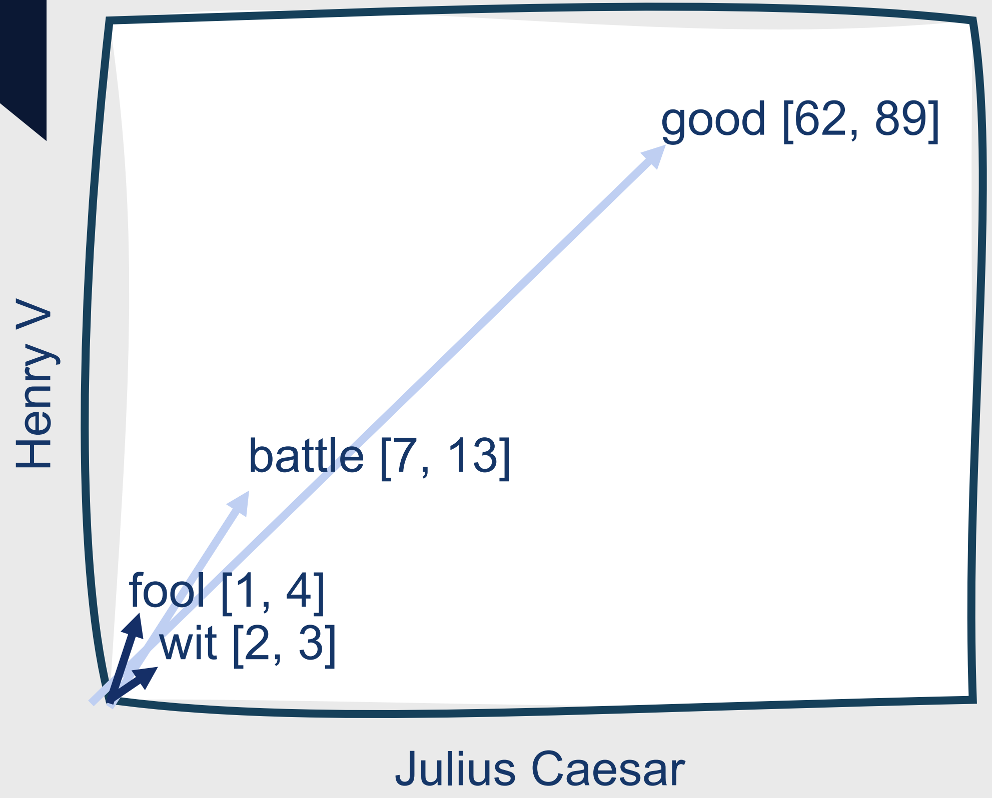


In a term-document matrix, rows could be viewed as word vectors.

- Each dimension corresponds to a document
- Words with **similar vectors** occur in **similar 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

In a term-document matrix, rows could be viewed as word vectors.



	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

Different Types of Context

- Documents aren't the most common type of context used to represent meaning in word vectors
- More common: **word context**
  - Referred to as a term-term matrix, word-word matrix, or term-context matrix
- In a **word-word matrix**, the columns are also labeled by words
  - Thus, dimensionality is $|V| \times |V|$
  - Each cell records the number of times the row (target) word and the column (context) word co-occur in some context in a training corpus

How can you decide if two words occur in the same context?

- Common **context windows**:
  - Entire document
    - Cell value = # times the words co-occur in the same document
  - Predetermined span surrounding the target
    - Cell value = # times the words co-occur in this span of words

Example Context Window (Size = 4)

- Take each occurrence of a word (e.g., strawberry)
- Count the context words in the four-word spans before and after it to get a word-word co-occurrence matrix

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

Example Context Window (Size = 4)

- A simplified subset of a word-word co-occurrence matrix could appear as follows, given a sufficient corpus

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	…

Vector for "strawberry"

So far, our co-occurrence matrices have contained raw frequency counts of word co-occurrences.

- However, this isn't the best measure of association between words
  - Some words co-occur frequently with many words, so won't be very informative
    - *the*, *it*, *they*
- We want to know about **words that co-occur frequently with one another, but less frequently across all texts**

This is where TF*IDF comes in handy!

- **TF*IDF**
  - Term Frequency * Inverse Document Frequency
- **Term Frequency:** The frequency of the word *t* in the document *d*
  - $tf_{t,d} = \text{count}(t, d)$
- **Document Frequency:** The number of documents in which the word *t* occurs
  - Different from collection frequency (the number of times the word occurs in the entire collection of documents)

Computing TF*IDF

- **Inverse Document Frequency:** The inverse of document frequency, where *N* is the total number of documents in the collection
  - $idf_t = \frac{N}{df_t}$
- IDF is higher when the term occurs in fewer documents
- What is a document?
  - Individual instance in your corpus (e.g., book, play, sentence, etc.)
- It is often useful to perform these computations in log space
  - TF: $\log_{10}(tf_{t,d}+1)$
  - IDF: $\log_{10} idf_t$

Computing TF*IDF

- TF*IDF is then simply the combination of TF and IDF
  - $tfidf_{t,d} = tf_{t,d} \times idf_t$

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?
- **TF(battle, d_1) = 1**

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?
- TF(battle, d_1) = 1
- **IDF(battle) = N/DF(battle) = 37/21 = 1.76**

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58		
wit	20	15		

word	df
battle	21
good	37
fool	3[illegible]
wit	34

Document frequencies from 37-document corpus

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?
- TF(battle, d_1) = 1
- IDF(battle) = N/DF(battle) = 37/21 = 1.76
- **TF*IDF(battle, d_1) = 1 * 1.76 = 1.76**

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?
- TF(battle, d_1) = 1
- IDF(battle) = N/DF(battle) = 37/21 = 1.76
- TF*IDF(battle, d_1) = 1 * 1.76 = 1.76
- **Alternately, TF*IDF(battle, d_1) = $log_{10}(1+1) * log_{10} 1.76 =$ 0.074**

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF*IDF

- TF*IDF(battle, d_1) = ?
- TF(battle, d_1) = 1
- IDF(battle) = N/DF(battle) = 37/21 = 1.76
- TF*IDF(battle, d_1) = 1 * 1.76 = 1.76
- Alternately, TF*IDF(battle, d_1) = $log_{10}(1+1) * \ log_{10} 1.76 = 0.074$

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

To convert our entire word co-occurrence matrix to a TF*IDF matrix, we need to repeat this calculation for each element.

	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

How does the TF*IDF matrix compare to the original term frequency matrix?

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

How does the TF*IDF matrix compare to the original term frequency matrix?

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

Occurs in every document …not important in the overall scheme of things!

How does the TF*IDF matrix compare to the original term frequency matrix?

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

Increases the importance of rarer words like “battle”

Note that the TF*IDF model produces a sparse vector.

- **Sparse:** Many (usually most) cells have values of 0

	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

Note that the TF*IDF model produces a sparse vector.

- **Sparse:** Many (usually most) cells have values of 0

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
battle	0.1	0.0	0.0	0.0	0.2	0.0	0.3
good	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fool	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wit	0.0	0.0	0.0	0.0	0.0	0.0	0.0

This can be problematic!

- However, TF*IDF remains a useful starting point for vector space models
- Generally combined with standard machine learning algorithms
  - Logistic Regression
  - Naïve Bayes

Now that we know how to create a vector space model, how can we use it to compute similarity between words?

- **Cosine similarity**
  - Based on the **dot product** (also called **inner product**) from linear algebra
    - $\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$
- Similar vectors (those with large values in the same dimensions) will have high values; dissimilar vectors (those with zeros in different dimensions) will have low values

Why don’t we just use the dot product?

- More frequent words tend to co-occur with more words and have higher co-occurrence values with each of them
- Thus, the **raw dot product will be higher for frequent words**
- This isn’t good! ☹️
  - We want our similarity metric to tell us how similar two words are regardless of frequency
- The simplest way to fix this problem is to **normalize for the vector length** (divide the dot product by the lengths of the two vectors)

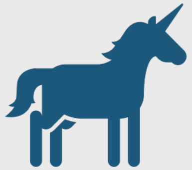
Normalized Dot Product = Cosine of the angle between two vectors

- The cosine similarity metrics between two vectors **v** and **w** can thus be computed as:
  - $\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$
- This value ranges between 0 (dissimilar) and 1 (similar)

Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

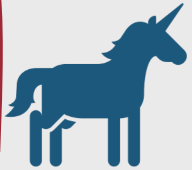
cos(unicorn, information) = ?



Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

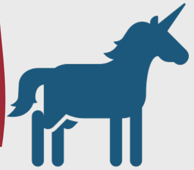
$$\text{cos(unicorn, information)} = \frac{[442,8,2]\cdot[5,3982,3325]}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}}$$



Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

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



Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\text{cos(unicorn, information)} = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = 0.017$$

Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

cos(unicorn, information) = $\frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = 0.017$

cos(digital, information) = $\frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = 0.996$

Example: Computing Cosine Similarity

	glitter	data	computer
unicorn	442	8	2
digital	5	1683	1670
information	5	3982	3325

cos(unicorn, information) = $\frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = 0.017$

cos(digital, information) = $\frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = 0.996$

0101

Result: *information* is way closer to *digital* than it is to *unicorn*!

So, we can compute word vectors and we can compute the similarity between them.

- All good?
  - Kind of….🤔

Limitations of Classic Word Representation Strategies

- No capacity to infer deeper semantic content
- Can’t encode the following using a bag-of-words vector:
  - Synonyms
  - Antonyms
  - Positive/negative connotations
  - Related contexts

Additionally, remember that bag of words representations are sparse.

- Very **high-dimensional**
- Lots of **empty** (zero-valued) cells

- **Lower-dimensional** (~ 50-1000 cells)
- Most cells with **non-zero** values

- We'd also prefer to be able to encode other dimensions of meaning than word type alone
  - *Good* should be:
    - Far from *bad*
    - Close to *great*

We'd prefer to have dense vectors.

It turns out that dense vectors are preferable for NLP tasks for many reasons!

- Easier to include as **features** in machine learning systems
  - Classifiers have to learn ~100 weights instead of ~50,000
- Fewer **parameters** → lower chance of overfitting
  - May generalize better to new data
- Better at capturing **synonymy**
  - Words are not distinct dimensions; instead, dimensions correspond to meaning components

What is the best way to generate dense word vectors?

- The answer changes quite frequently:
  - https://gluebenchmark.com/leaderboard/
  - https://rajpurkar.github.io/SQuAD-explorer/
- Current state-of-the-art models are **bidirectional** (trained to represent words using both their left and right context), **contextual** (produce different vectors for different word senses) models built using **Transformers** (a type of neural network)

We'll cover state-of-the-art embedding models later this semester, when we're discussing research papers.

- Next class period, we'll cover two basic, essential models:
  - **Word2Vec:**
    - https://code.google.com/archive/p/word2vec/
  - **GloVe:**
    - https://nlp.stanford.edu/projects/glove/

Summary: Vector Semantics

- **Word embeddings** are vector representations of meaning
- A vector for a word is computed based on the **contexts** in which the word occurs
  - Context = Documents or windows of words
- Word embeddings can be **sparse** or **dense**
  - **Sparse:** Bag of words representations
  - **Dense:** Word2Vec, GloVe
- Dense embeddings are generally **better for NLP tasks**
- **TF*IDF** vectors are bag of words representations that encode meaning based on a combination of term frequency and inverse document frequency
- **Cosine similarity** can be used to determine the similarity between two word vectors