# Week 3 (Cont.)
# Word Embeddings: Count-based Approach

Nhung Nguyen
slides courtesy of Phong Le

# Recap

- A brief introduction to distributional semantics
- A very simple method using co-occurrence vectors
- Pros and cons of using vectors to represent word meanings

# Count-based approach: Term-document matrix

- If word *u* appears in document *d*, *d* is a context of *u*

1. Collect a lot of documents (from, e.g. Wikipedia)
2. Count how many time a word *u* appearing with a document *d*
3. The meaning of word *u* is vector [count($u$,$d_1$), count($u$,$d_2$),...]

Shakespeare plays

| | 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 |

# Term-document matrix (cont.): Document vectors

Two ways to extract information from the matrix

1. *Column-wise*: a *document* is represented by a |V|-dim vector (V: vocabulary)

| | 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 |

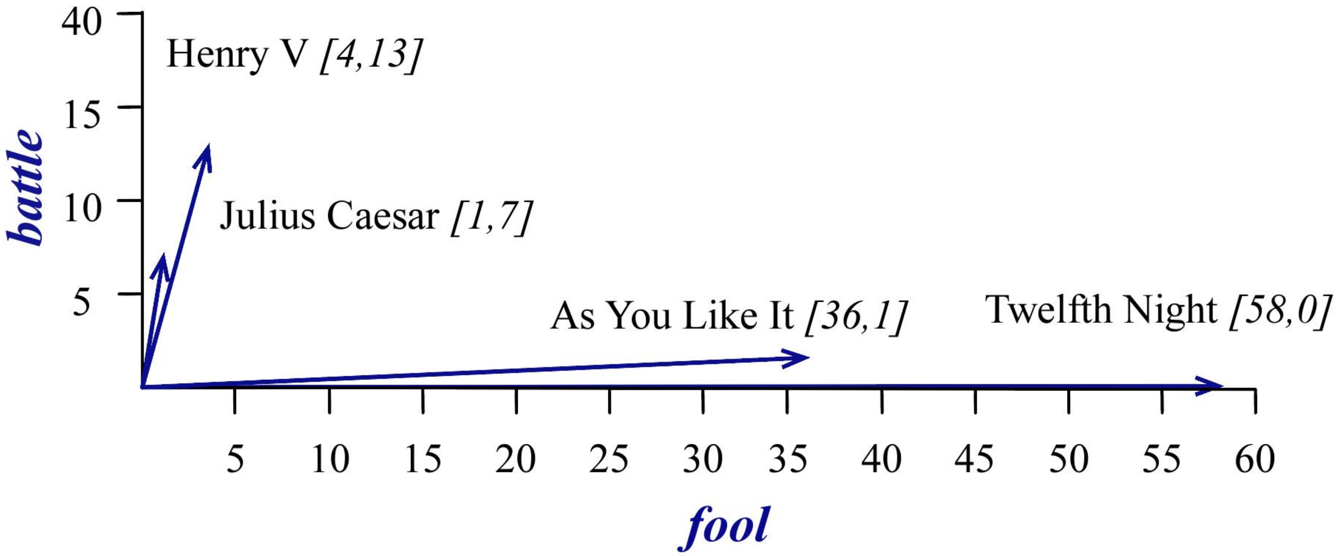
Widely used in information retrieval:

- find similar documents
- find documents close to a query

# Term-document matrix (cont.): Document vectors

- Find similar documents:

*Two documents that are similar will tend to have similar words*

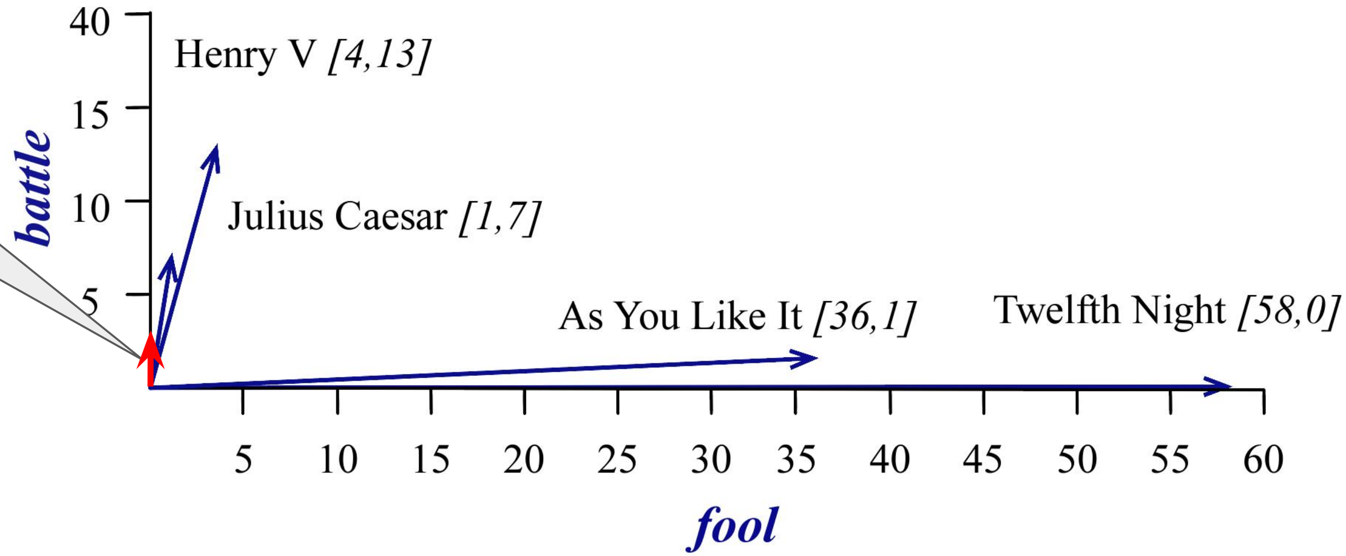


# Term-document matrix (cont.): Document vectors

- Find documents close to a query

*Consider a query as a document*

Query:
"Horrible battle"



# Term-document matrix (cont.): Word vectors

2. *Row-wise*: a *word* is represented by a |D|-dim vector (D: document set)

| | 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 |

# Count-based approach: Term-term matrix

- we have seen it before (co-occurrence vectors): Count how many times a word *u* appearing with a word *v*

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | jam, a pinch each of, |
| their enjoyment. Cautiously she sampled her first | **pineapple** | and another fruit whose taste she likened |
| well suited to programming on the digital | **computer**. | In finding the optimal R-stage policy from |
| for the purpose of gathering data and | **information** | necessary for the study authorized in the |

| | aardvark | computer | data | pinch | result | sugar | ... |
|---|---|---|---|---|---|---|---|
| apricot | 0 | 0 | 0 | 1 | 0 | 1 | |
| pineapple | 0 | 0 | 0 | 1 | 0 | 1 | |
| digital | 0 | 2 | 1 | 0 | 1 | 0 | |
| information | 0 | 1 | 6 | 0 | 4 | 0 | |

# Count-based approach: raw frequency is bad

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | jam, a pinch each of, |
| their enjoyment. Cautiously she sampled her first | **pineapple** | and another fruit whose taste she likened |
| well suited to programming on the digital | **computer**. | In finding the optimal R-stage policy from |
| for the purpose of gathering data and | **information** | necessary for the study authorized in the |

- Not all contextual words are equally important: of, a, … vs. sugar, jam, fruit…
- Which words are important, which ones are not?
  - infrequent words are more important than frequent ones (examples?)
  - correlated words are more important than uncorrelated ones (examples?)
  - ...

→ weighing schemes (TF-IDF, PMI,...)

# Weighing terms: TF-IDF (for term-document matrix)

- tf (term frequency): frequency count

Frequency of term *t* in document *d*

$$tf(t,d) = \log_{10}(1 + count(t,d))$$

- idf (inverse document frequency): popular terms (terms that appear in many documents) are down weighed

Total number of documents in the collection

$$idf(t) = \log_{10}\frac{N}{df(t)}$$

Number of documents in which term *t* occurs

- TF-IDF: $tf\text{-}idf(t,d) = tf(t,d) \times idf(t)$

# Weighing terms: TF-IDF (example)

Shakespeare plays

| 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 |

| | 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 |

raw frequency

tf-idf

| | 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 |

# Count-based approach: so many 0s

| | aardvark | computer | data | pinch | result | sugar | ... |
|---|---|---|---|---|---|---|---|
| apricot | 0 | 0 | 0 | 1 | 0 | 1 | |
| pineapple | 0 | 0 | 0 | 1 | 0 | 1 | |
| digital | 0 | 2 | 1 | 0 | 1 | 0 | |
| information | 0 | 1 | 6 | 0 | 4 | 0 | |

- Many word pairs should have > 0 counts, but their corresponding matrix entries are 0s because of lacking data (data sparsity)

→ Laplace smoothing: adding 1 to every entry (pseudocount)

# Pros

- Simple and intuitive
- Dimensions are meaningful (e.g. each dim is a document / a contextual word)

  → easy to debug and interpret *(Think about Explainable AI)*

# Cons

- Word/document vectors are sparse (dims are |V|, vocabulary size, or |D|, number of documents, often from 2k to 10k) → difficult for machine learning algorithms
- How to represent word meaning in a specific context?

From sparse vectors to dense vectors

- Employ dimensionality reduction (e.g. latent semantic analysis - LSA)
- Use a different approach: prediction (coming up next)