CS 1671 / CS 2071 / ISSP 2071
Human Language Technologies

Session 6: Bag of words and n-grams

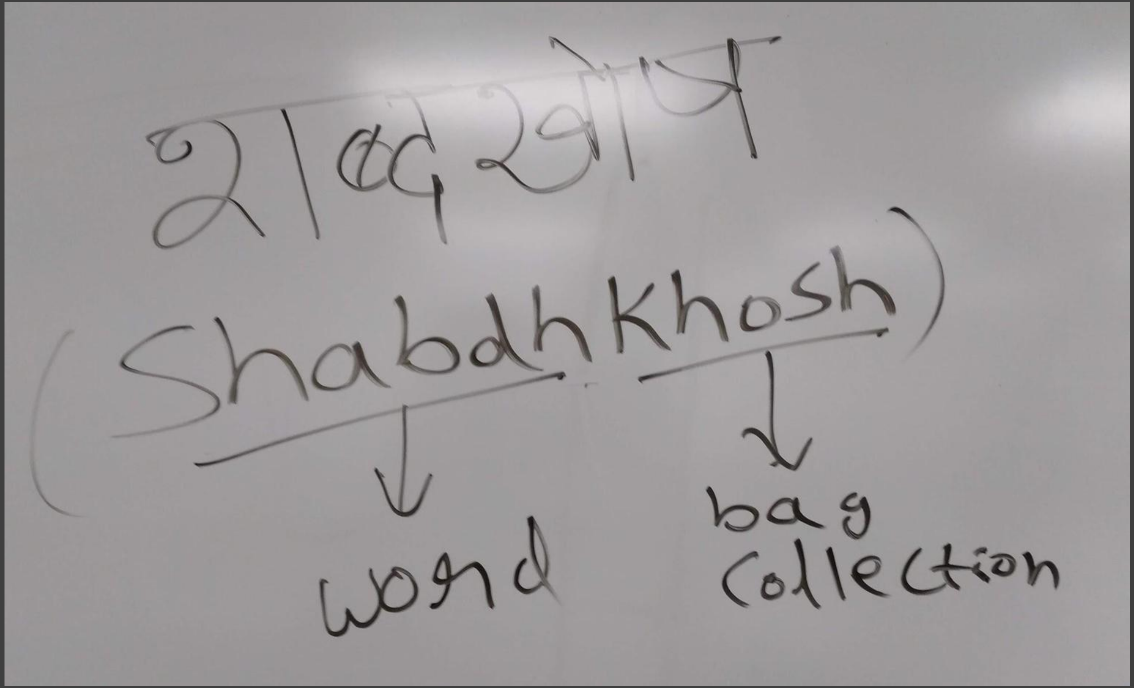
Michael Miller Yoder

February 2, 2026

University of Pittsburgh | School of Computing and Information

Quiz

- Go to **Quizzes > Quiz 02-02** on Canvas
- You have until **1:10pm** to complete it
- Allowed resources
    - Textbook
    - Your notes (on a computer or physical)
    - Course slides and website
- Resources not allowed
    - Generative AI
    - Internet searches


शब्दखोष
(Shabdhkhosh)
word
bag
collection

Course logistics: quiz

- Next in-class quiz is next class session, **this Wed Feb 4**
  - Looking over the reading is a great way to prepare
  - Session 6 (today): J+M 5.3-5.4
- Conceptual, not programming
- Lowest quiz score in the course will be dropped
- If you won't be in class, let me know and I can accommodate

Course logistics: project

- Project idea form to submit project ideas is **due this Thu Feb 5**
- Take a look at the example projects on the project website. You can submit one or more of those for the form, or submit your own idea!
- Have a potential project idea that involves deriving insight from a dataset of text, or building an NLP system that can do something with text? You can submit it!
    - Ideas do not need to be well-formed
    - Ideas that have data already available are more realistic
- You will later choose from an **anonymized** list of project ideas on Project Match Day, Feb 11

Course logistics: homework

- Homework 1 has been released. Is **due next Thu Feb 12 at 11:59pm**
- Homework assignments are programming-based
- Homework 1 covers text processing and regular expressions in Python

Review activity

What do you remember about what machine learning is or what you can do with it?

Structure of this course

MODULE 1	Prerequisite skills for NLP	text normalization, linear alg., prob., machine learning

	Approaches	How text is represented	NLP tasks
MODULE 2	statistical machine learning	n-grams	language modeling text classification
MODULE 3			
MODULE 4			language modeling text classification sequence labeling

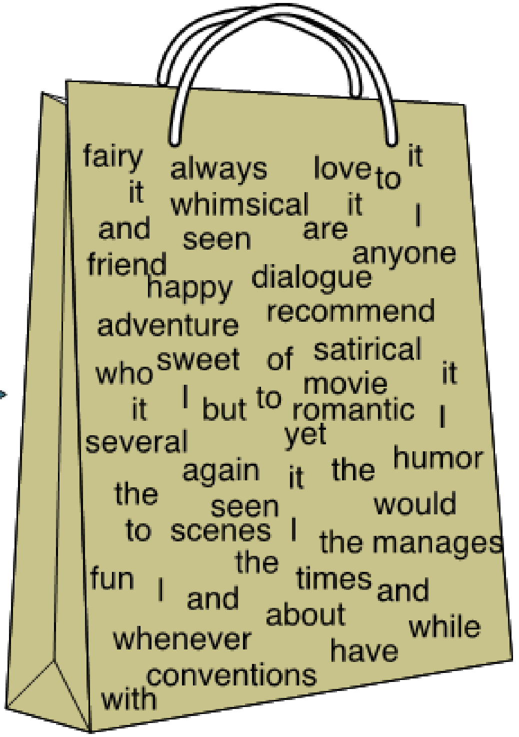
MODULE 5	NLP applications and ethics	

Overview: Bag of words, n-grams

- Term-document and term-term matrices
- Cosine similarity
- N-grams
- Coding activity

Bag of words document representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Slide from Jurafsky & Martin

Documents Can Be Represented as Bags of Words

A BAG OF WORDS is a BAG or MULTISET of the words in a document.

- Like a set, except that identical elements can appear multiple times
- You could also think of it like a `Counter` object in Python

```
bow = {'and': 23502,
       'or': 12342',
       'the': 54939508,
       ...
       'hippopotamus': 1}
```

- If you take out sequencing information, any document can be viewed as a bag of words.

Slide credit: David Mortensen

Bags of Words Can Be Represented as Sparse Vectors

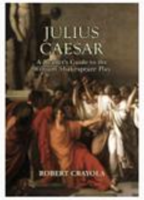
- So far, we've represented bags of words as dense MAPS, but sometimes it is useful to represent them as sparse vectors
- Led V be the sent of all words in the document collection D. Then our BoW vectors for documents in D will have $|V|$ dimensions.
- Each dimension corresponds to a word `type` and the value at this dimensions corresponds to the number of TOKENS of that type in the document that the vector is representing

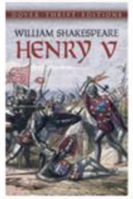
Slide credit: David Mortensen

Term-document and term-term matrices

Term-document matrix

- Each cell is the count of term *t* in a document *d* ($tf_{t,d}$).
- Each document is a count vector in $\mathbb{N}^V$, a column below.



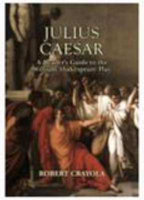


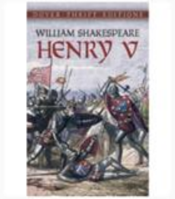
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Slide credit: David Mortensen

Term-document matrix

- Two documents are similar of their vectors are similar.

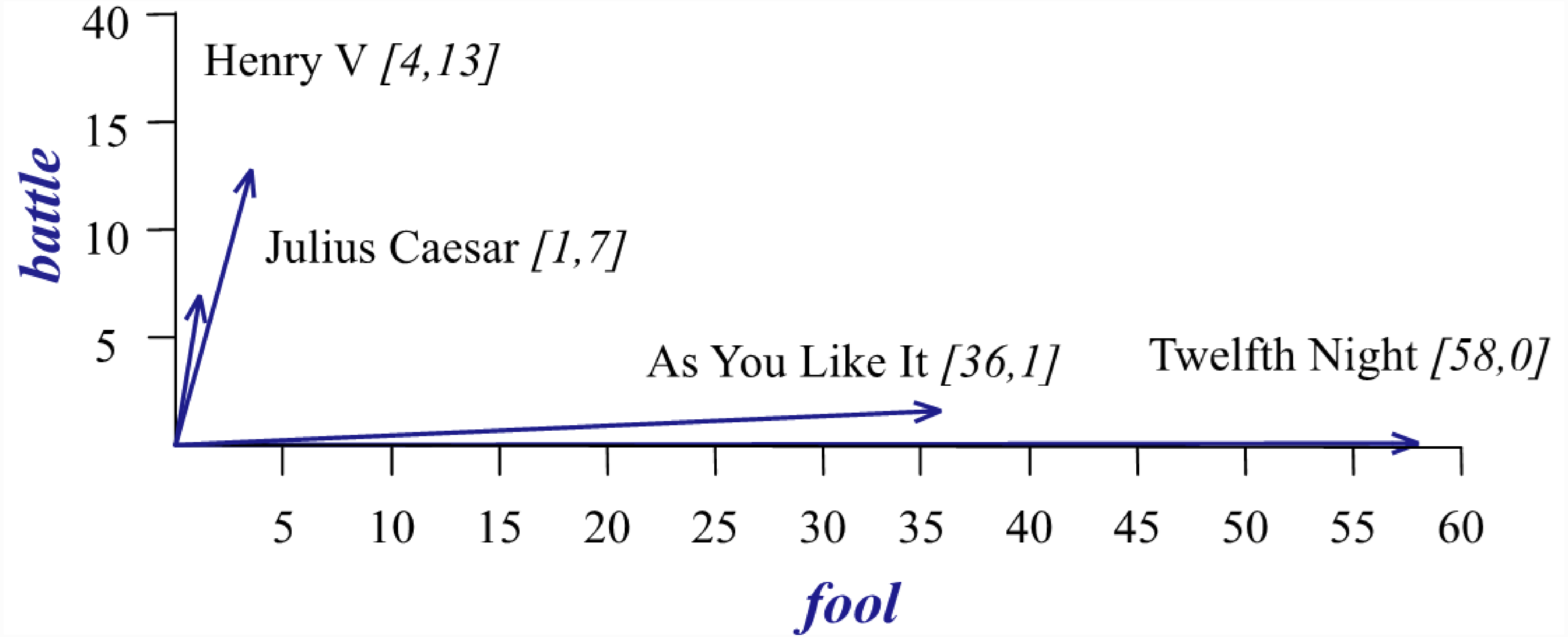




	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Slide credit: David Mortensen

Visualizing document vectors



Slide from Jurafsky & Martin

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 for comedies are different from tragedies
- Comedies have more *fools* and fewer *battles*

Slide adapted from Jurafsky & Martin

Term-document matrix: word vectors

Two words are similar if their vectors are similar.

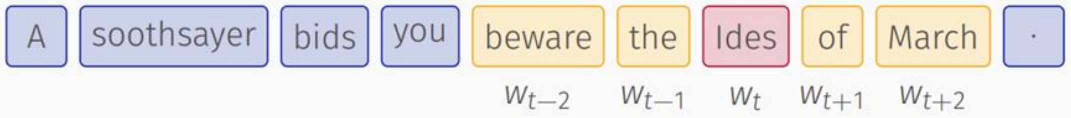
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

- *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"

Slide adapted from David Mortensen, Jurafsky & Martin

Term-term matrix (or word-word or word-context matrix)

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of a few words (e.g. 3, 5, 7):



- A word is now defined by a vector over counts of words in context.
  - If a word w_j occurs in the context of w_i, increase $count_{ij}$.
- Assuming we have V words,
  - Each vector is now of length V.
  - The word-word matrix is $V \times V$.

Slide credit: David Mortensen

Sample Contexts of ± 7 Words

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or 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

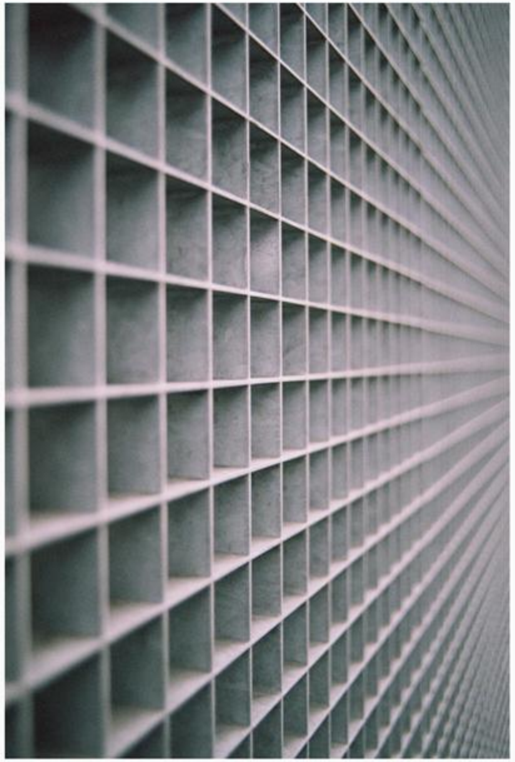
	aardvark	digital	data	pinch	result	sugar ...
⋮						
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
computer	0	2	1	0	1	0
information	0	1	6	0	4	0
⋮						

Slide credit: David Mortensen

The Word–Word Matrix

We showed only a 4×6 matrix, but the real matrix is $50,000 \times 50,000$.

- So it is very sparse: Most values are 0.
- That's OK, since there are lots of efficient algorithms for sparse matrices.



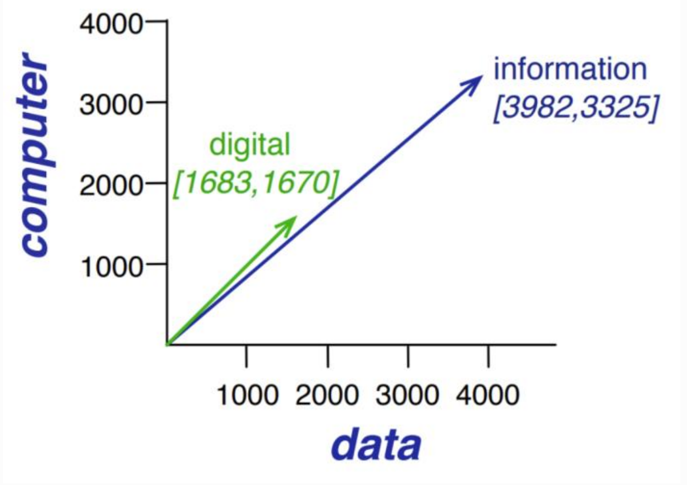
Slide credit: David Mortensen

Cosine similarity

Measuring similarity between word or document vectors

	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

Do we care about magnitude/word frequencies? (No)



Slide adapted from Jurafsky & Martin

Cosine is Used to Measure the Similarity between Word Vectors

- Given two target words represented with vectors ***v*** and ***w***.
- The **dot product** or **inner product** is usually used as the basis for similarity.

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$$

- $\mathbf{v} \cdot \mathbf{w}$ is high when two vectors have large values in the same dimensions.
- $\mathbf{v} \cdot \mathbf{w}$ is low (in fact 0) with zeros in complementary distribution.
- We also do not want the similarity to be sensitive to word-frequency.
- So normalize by vector length and use the cosine as the similarity

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

$$\frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \cos(\mathbf{v}, \mathbf{w})$$

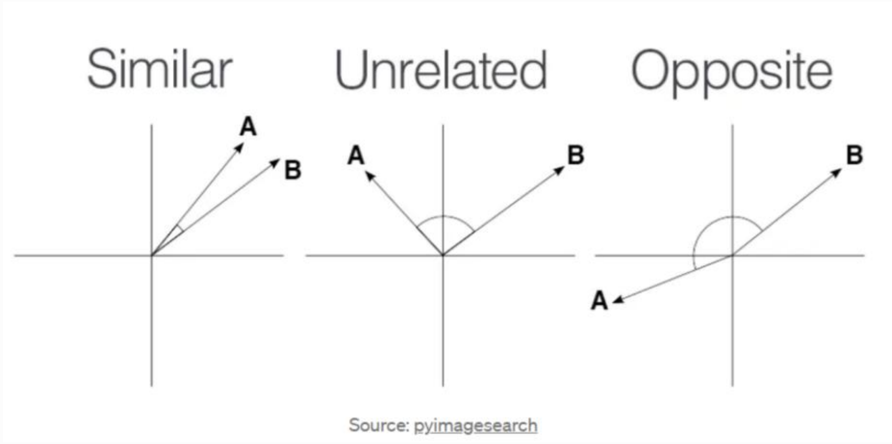
Slide credit: David Mortensen

Cosine as a similarity metric for vectors

-1: vectors point in opposite directions

+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

Slide adapted from Jurafsky & Martin, David Mortensen

N-grams

N-grams

- A sequence of *n* words
- “Pittsburgh is cold in the winter.”
- Representations of documents:
  - Unigram: counts of all individual words
    - {Pittsburgh, is, cold, in, the, winter}
  - Bigram: counts of all sequences of 2 words
    - {Pittsburgh is, is cold, cold in, in the, the winter}
  - Trigram: counts of all sequences of 3 words
    - {Pittsburgh is cold, is cold in, cold in the, in the winter}
  - 4gram, etc
- Term-document matrix becomes even sparser!

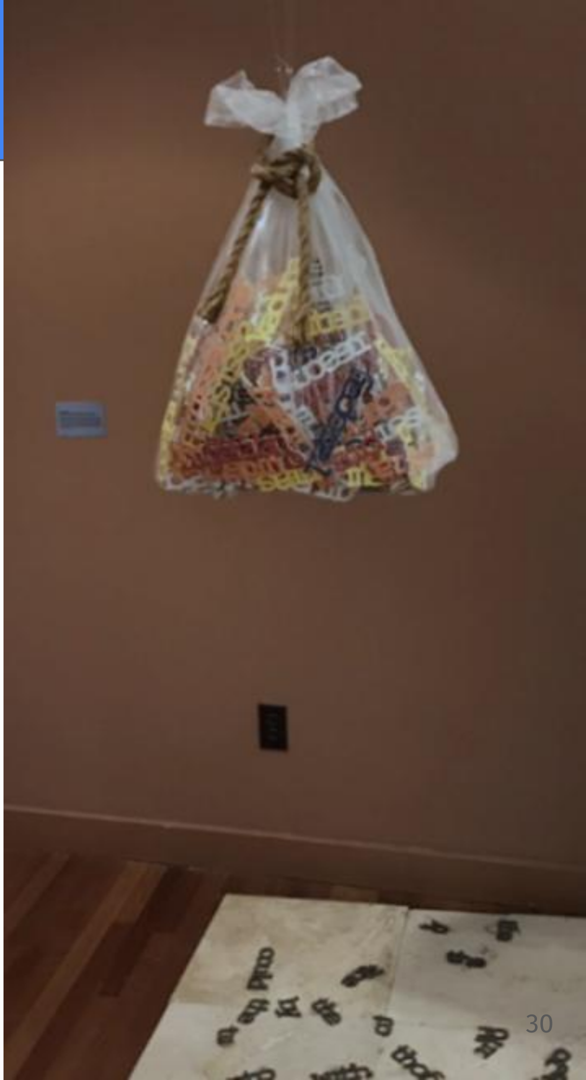
Coding activity:
clickbait classification revisited

N-gram document representations on JupyterHub

1. Go to this nbgitpuller link (also available on course website)
2. Log in with your Pitt username if necessary
3. Start a server with **TEACH – 6 CPUs, 48 GB**
4. Load custom environment at **/ix1/cs1671-2026s/class_env**
    1. If you have multiple accounts on the CRCD, put in **cs1671-2026s** for Account
5. This should pull the cs1671_spring2026_jupyterhub folder into your JupyterLab
6. Open **session6_clickbait_ngrams.ipynb**

Conclusion

- **Bag of words** representations of documents
    - Counts of terms in documents (no order info)
    - Can be represented in vector form as...
- **Term-document matrices**
- **Term-term (word-word) matrices**
    - How many times words are used in contexts of other words
- **Cosine** to measure similarity between vectors for documents or words
- **N-grams** are sequences of *n* words (tokens)



Questions?