Text Analytics & Business Application

Text Representation 1 – Basic Vectorization

Qinglai He

Department of Operations and Information Management

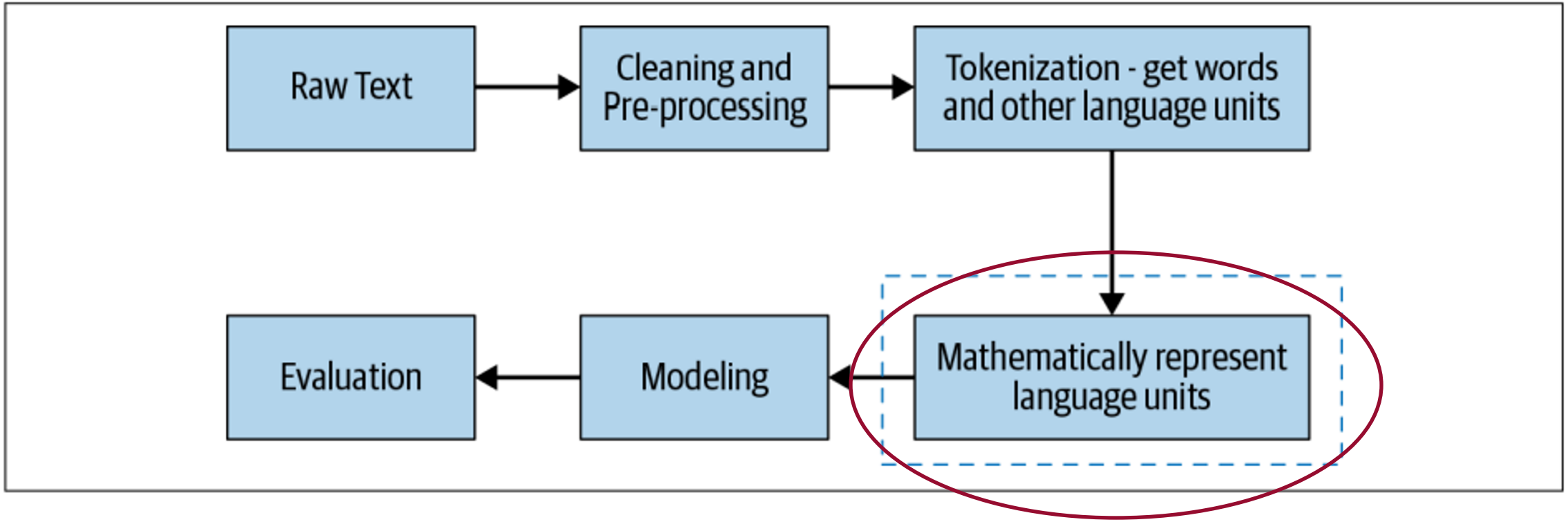
Wisconsin School of Business

WISCONSIN SCHOOL OF BUSINESS

Outline of Today's Class

- Vector Space Model
- Basic Vectorization Approaches
  - One-Hot Encoding
  - Bag of Words
  - Bag of N-Grams
  - TF-IDF

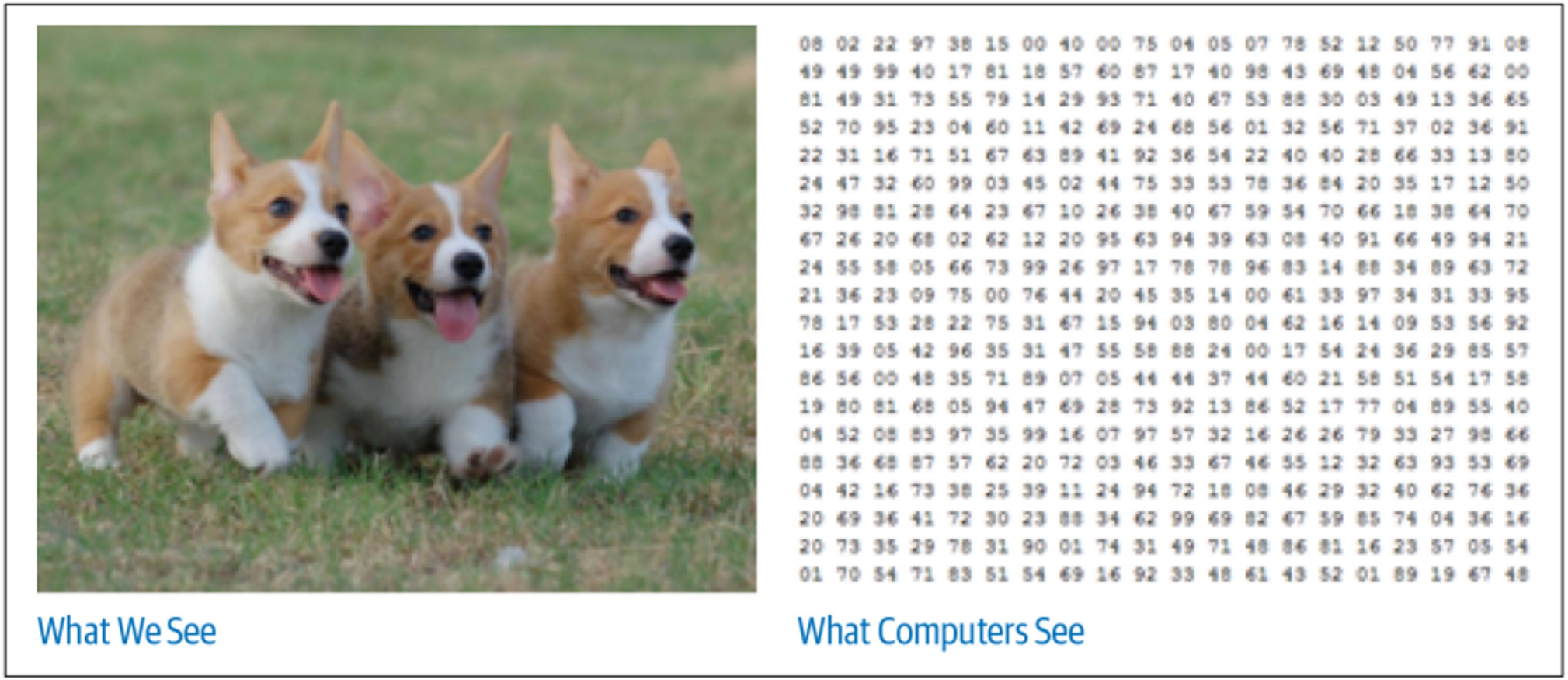
Scope of This Class Within the NLP pipeline



Feature Representation

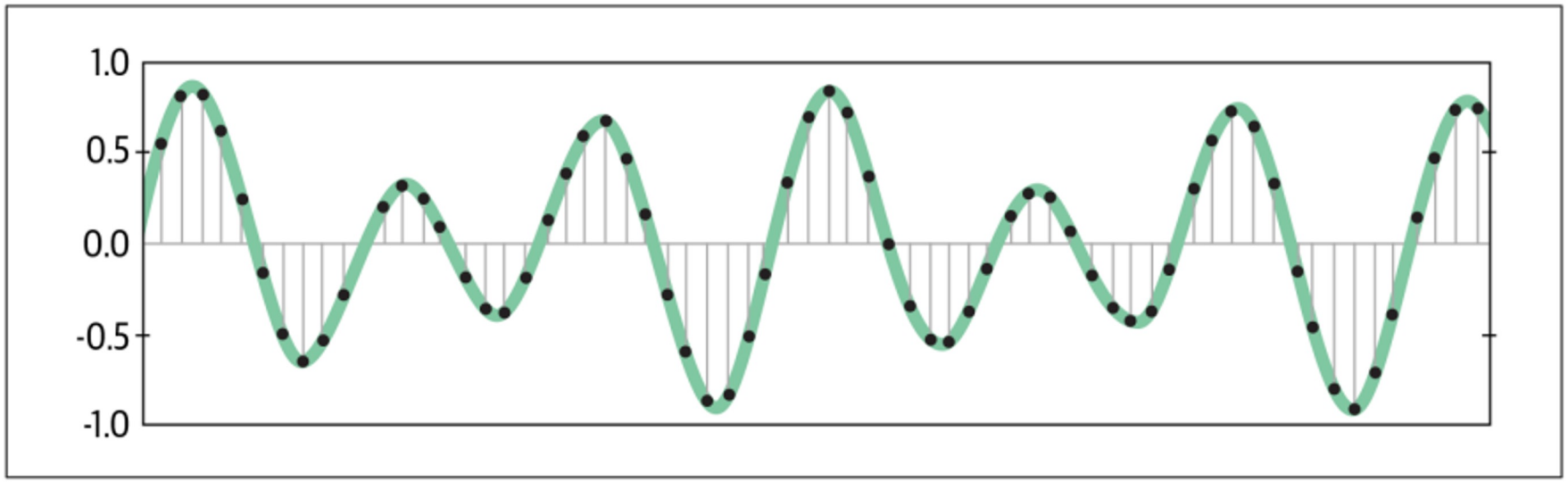
- A common step in any ML project
- Representation Object:
  - Text
  - Images
  - Videos
  - Speech

Image Representation



How we see an image versus how computers see it

Speech Representation



Sampling a speech wave

```
[-1274, -1252, -1160, -986, -792, -692, -614, -429, -286, -134, -57, -41,
-169, -456, -450, -541, -761, -1067, -1231, -1047, -952, -645, -489, -448,
-397, -212, 193, 114, -17, -110, 128, 261, 198, 390, 461, 772, 948, 1451,
1974, 2624, 3793, 4968, 5939, 6057, 6581, 7302, 7640, 7223, 6119, 5461,
4820, 4353, 3611, 2740, 2004, 1349, 1178, 1085, 901, 301, -262, -499,
-488, -707, -1406, -1997, -2377, -2494, -2605, -2675, -2627, -2500, -2148,
-1648, -970, -364, 13, 260, 494, 788, 1011, 938, 717, 507, 323, 324, 325,
350, 103, -113, 64, 176, 93, -249, -461, -606, -909, -1159, -1307, -1544]
```

Speech signal represented by a numerical vector

An Example of Text Representation

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Text Representation

- We’re given a piece of text, and we’re asked to find a scheme to represent it mathematically.
- Text representation approaches are classified into four categories:
  - Basic vectorization approaches (commonly used)
  - Distributed representations/embedding (commonly used)
  - Universal language representation
  - Handcrafted features

Some Prior Knowledge

Before we start learning about text representation approaches, we need to understand the following definitions

- Basic terminology in text mining
- Vector space models

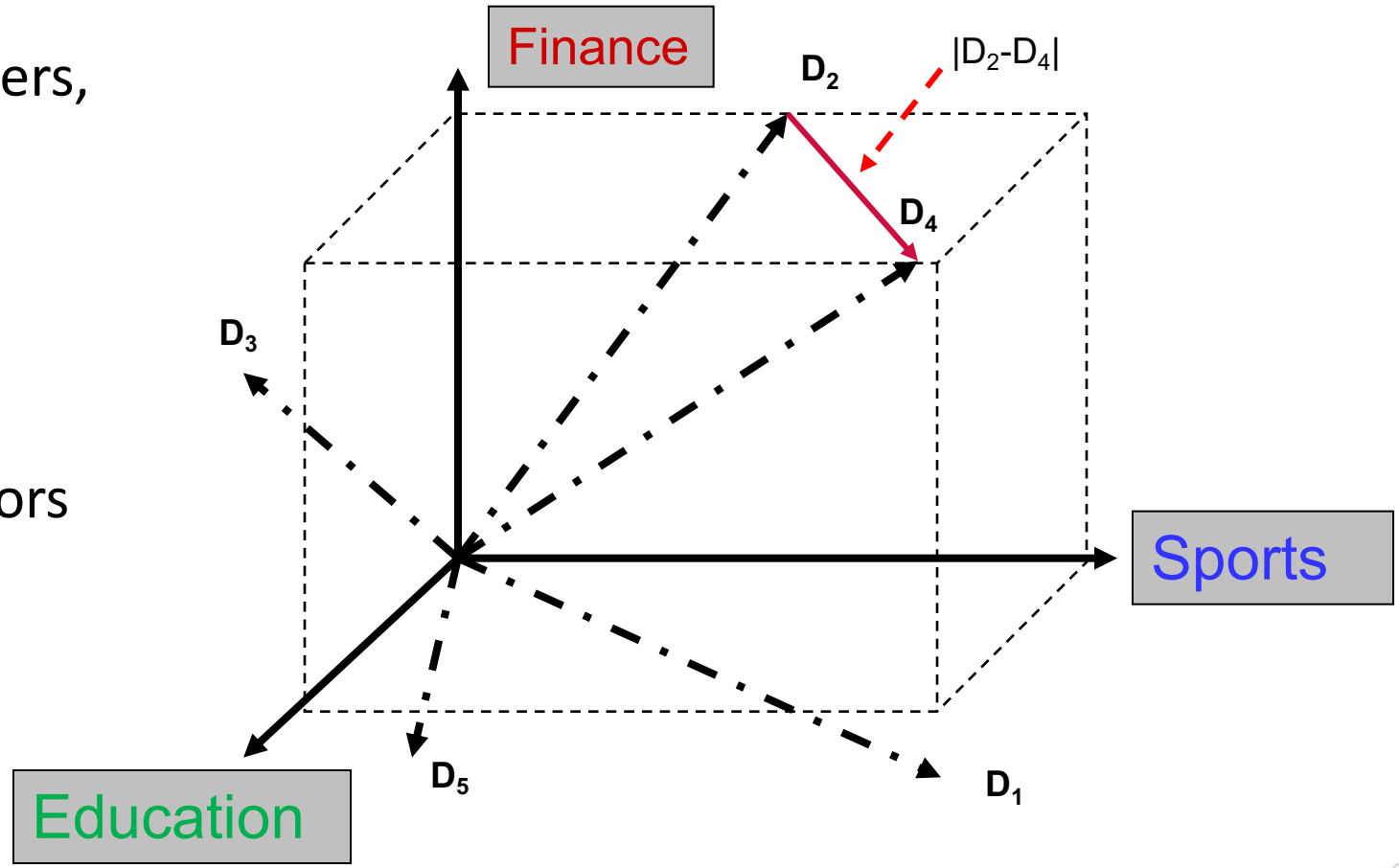
1. Basic Terminology in Text Mining

- **Corpus:** a collection of documents (database).
  - For example, a corpus contains 16 documents (16 txt files)
- **Document:** a basic unit in text analysis. It has one or multiple sentence (e.g., 1 txt file).
  - For example, "Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.“
- **Document Term Matrix:** a matrix consisting of documents in a row and terms in columns
  - Example of document term matrix :

		Terms			
Documents		data	result	statistics	analysis
	Document1	0	1	0	1
	Document2	1	0	1	0
	Document3	0	0	1	0
	Document4	1	1	0	0

2. Vector Space Model

- **Vector Space Model**: We'll represent text units (characters, phonemes, words, phrases, sentences, paragraphs, and documents) with vectors of numbers.
- A mathematical model that represents text units as vectors (feature vectors).



Vector Space Model

- Represent documents by concept vectors
  - Each concept defines one dimension
  - *k* concepts define a high-dimensional space
  - Element of vector corresponds to concept weight
    - E.g., $d=(x_1,\ldots,x_k)$, x_i is "importance" of concept i in d
- Distance between the vectors in this concept space
  - Relationship/similarity among documents

Things to Consider to Build a VS Model

- How to define/select the “basic concept”
  - Concepts are assumed to be orthogonal
- How to assign weights
  - Weights indicate how well the concept characterizes the document
- How to define the distance metric

Some Solutions

- Concept are assumed to be orthogonal
  - Ideally, linearly independent basis vectors
    - “Non-overlapping” in meaning
  - Ideally, no ambiguity
- Weights can be assigned automatically and accurately
  - Existing solutions
    - Terms or N-grams, a.k.a., Bag-of-Words
    - Topics
- Various distance measures
  - Cosine similarity
  - Euclidean distance
  - …

Basic Vectorization Approaches

Basic Vectorization Approaches

- Let's start with a toy corpus:

D1	Dog bites man.
D2	Man bites dog.
D3	Dog eats meat.
D4	Man eats food.

- Lowercasing text and ignoring punctuation, the vocabulary of this corpus is comprised of six words: [dog, bites, man, eats, meat, food]

Basic Vectorization Approaches

- **One-Hot Encoding**
- Bag of Words (BoW)
- Bag of N-Grams
- TF-IDF

1. One-Hot Encoding Representation

- Each word w in the corpus vocabulary is given a unique integer ID W_{id} that is between 1 and |V|, where V is the set of the corpus vocabulary.
- Each word is then represented by a V-dimensional binary vector of 0s and 1s.
- This is done via a |V| dimension vector filled with all 0s except for the index, where index = $\mathbf{W}_{id}$.

One-Hot Encoding Representation

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

One-Hot Encoding Examples

D1	Dog bites man.
D2	Man bites dog.
D3	Dog eats meat.
D4	Man eats food.

- We first map each of the six words to unique IDs:
  - Indexes: dog = 1, bites = 2, man = 3, meat = 4 , food = 5, eats = 6.
- Approach (1): Let's consider the document D1: "dog bites man".
  - Dog is represented as [1 0 0 0 0 0], as the word "dog" is mapped to ID 1
  - Bites is represented as [0 1 0 0 0 0]
  - Thus, D1 is represented as [[1 0 0 0 0 0] [0 1 0 0 0 0] [0 0 1 0 0 0]].
- **(More common)** Approach (2): Combing the word-level vectors together from approach (1), we get another type of one-hot encoding of D1: "dog bites man": [1 1 1 0 0 0]

One-Hot Encoding: Pros

- One-hot encoding is intuitive to understand and **straightforward** to implement

One-Hot Encoding: Cons

- The size of a one-hot vector is directly proportional to size of the vocabulary
  - most real-world corpora have large vocabularies
- Does not give a fixed-length representation for text
  - i.e., if a text has 10 words, you get a longer representation for it as compared to a text with 5 words.
- It treats words as atomic units and has no notion of (dis)similarity between words.
- Out of vocabulary (OOV) problem
  - At runtime, we get a sentence: “man eats fruits.” The training data didn’t include “fruit” and there’s no way to represent it in our model.
  - The only way is to retrain the model: start by expanding the vocabulary, give an ID to the new word, etc.

Basic Vectorization Approaches

- One-Hot Encoding
- **Bag of Words (BoW)**
- Bag of N-Grams
- TF-IDF

2. Bag of Words

- A classical text representation technique that has been used commonly in NLP
- Key idea:
  - represent the text under consideration as a bag (collection) of words while ignoring the order and context.
- BoW maps words to unique integer IDs between 1 and |V|
- Each document in the corpus is then converted into a vector of |V| dimensions where in the i_{th} component of the vector, i = $\mathbf{W}_{id}$, is simply the number of times the word w occurs in the document
  - i.e., we simply score each word in V by their occurrence count in the document.

Bag-of-Words Examples

For our toy corpus:

- Indexes: dog = 1, bites = 2, man = 3, meat = 4 , food = 5, eats = 6.
- D1 becomes [1 1 1 0 0 0]
- D4 becomes [0 0 1 0 1 1]
- How about we have a new sentence ‘Dog bites dog’?
  - Its BoW representation is [2 1 0 0 0 0]

D1	Dog bites man.
D2	Man bites dog.
D3	Dog eats meat.
D4	Man eats food.

Bag-of-Words: Pros

- **Simple** to understand and implement
- BoW **captures more semantic similarity** of documents compared to the one-hot encoding
  - For example, with the frequency information, the distance between D1 and D2 is 0 as compared to the distance between D1 and D4, which is 2.

Bag-of-Words: Cons

- The size of the vector increases with the size of the vocabulary
- It does not capture the similarity between different words that mean the same thing
- Out of vocabulary (OOV) problem
  - i.e., new words that were not seen in the corpus that was used to build the vectorizer
- As the name indicates, it is a “bag” of words—word order information is lost in this representation.
  - i.e., Both D1 and D2 will have the same representation in this scheme.

Basic Vectorization Approaches

- One-Hot Encoding
- Bag of Words (BoW)
- **Bag of N-Grams**
- TF-IDF

3. Bag of N-grams

- All the representation schemes we've seen so far treat words as independent units.
- There is no notion of phrases or word ordering. The bag-of-n-grams (BoN) approach tries to remedy this.
  - It does so by breaking text into chunks of n contiguous words
  - Each chunk is called an **n-gram**
  - Each document in the corpus is represented by a vector of length |V|.

Bag of N-grams

- N-grams: a contiguous sequence of N tokens from a given piece of text
- E.g., ‘Text mining is to identify useful information.’
  - Unigrams: ‘text’, ‘mining’, ‘is’, ‘to’, ‘identify’, ‘useful’, ‘information’, ‘.’
  - Bigrams: ‘text mining’, ‘mining is’, ‘is to’, ‘to identify’, ‘identify useful’, ‘useful information’, ‘information .’
  - Trigrams: ‘text mining is’, ‘mining is to’, ‘is to identify’, ‘to identify useful’, ‘identify useful information’, ‘useful information .’.

Bag of N-grams Examples

For our toy corpus, Let's construct a 2-gram (a.k.a. bigram) model for it:

- Indexes: {dog bites, bites man, man bites, bites dog, dog eats, eats meat, man eats, eats food}
- D1: [1,1,0,0,0,0,0,0]
- D2: [0,0,1,1,0,0,0,0]

D1	Dog bites man.
D2	Man bites dog.
D3	Dog eats meat.
D4	Man eats food.

Note: the BoW scheme is a special case of the BoN scheme, with n=1.

Bag of N-grams: Pros

- It captures some context and word-order information in the form of n-gram.
- Resulting vector space is able to capture some semantic similarity.

Bag of N-grams: Cons

- As n increases, dimensionality (and therefore sparsity) only increases rapidly.
- It still provides no way to address the OOV problem.


QUIZ

Basic Vectorization Approaches

- One-Hot Encoding
- Bag of Words (BoW)
- Bag of N-Grams
- **TF-IDF**

4. TF-IDF

- In all the three approaches we've seen so far, all the words in the text are treated as equally important—there's no notion of some words in the document being more important than others.
- TF-IDF, or term frequency–inverse document frequency, addresses this issue.
- It aims to quantify the **importance** of a given word relative to other words in the document and in the corpus.

Zipf's law tells us

- Head words take large portion of occurrences, but they are semantically meaningless
  - E.g., the, a, an, we, do, to
- Tail words take major portion of vocabulary, but they rarely occur in documents
  - E.g., *sesquipedalianism*
- The rest is most representative
  - To be included in the controlled vocabulary

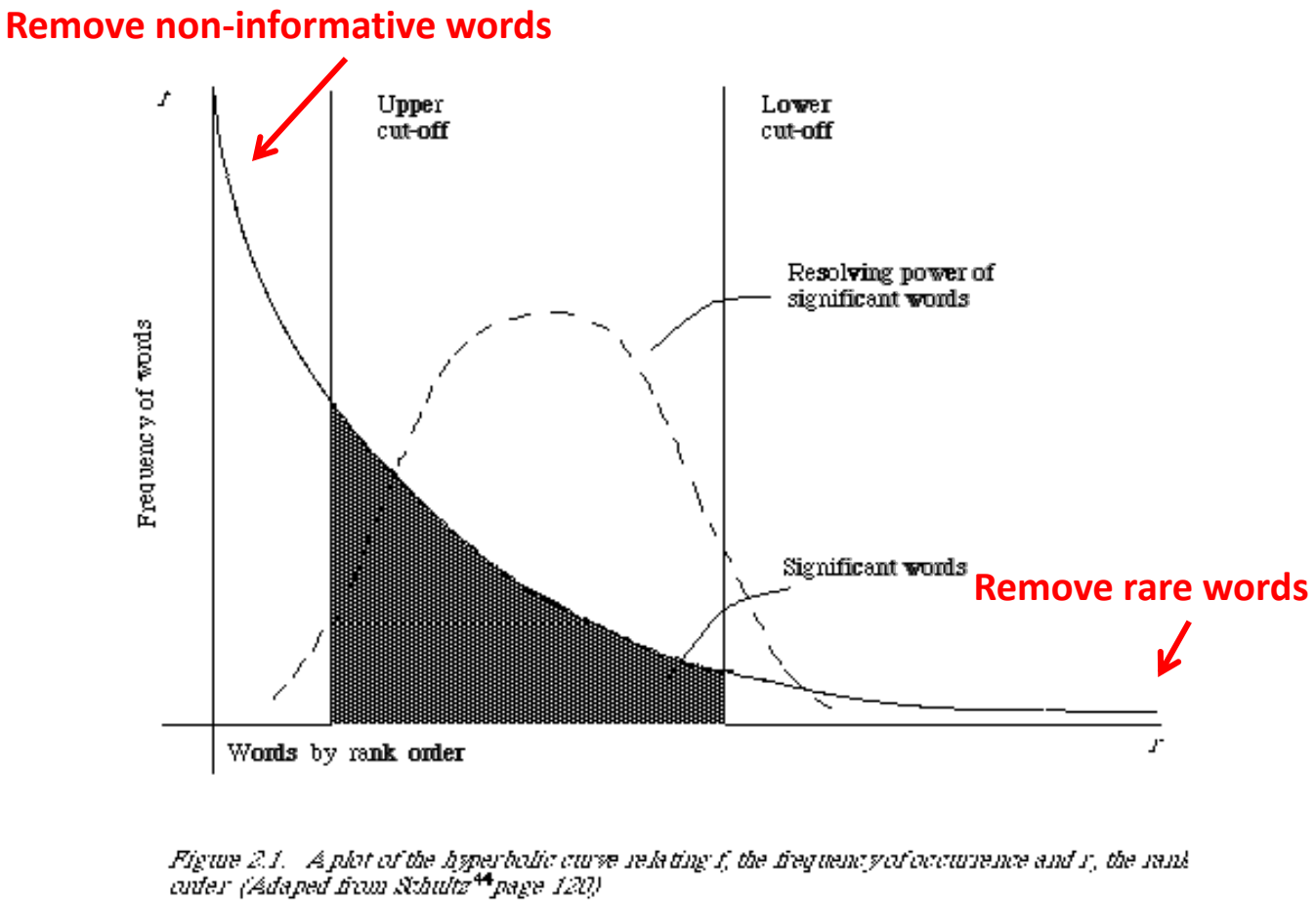


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[44] page 120)

Intuition behind TF-IDF

- If a word *w* appears many times in a document d_i but does not occur much in the rest of the documents d_j in the corpus, then the word w must be of great importance to the document d_i.
- The importance of *w* should increase in proportion to its frequency in d_i, but at the same time, its importance should decrease in proportion to the word's frequency in other documents d_j in the corpus.
- To sum up: A more discriminative word in the corpus is more important.

TF (term frequency)

- TF (term frequency) measures how often a term or word occurs in a given document.
- TF of a term t in a document d is defined as:

$$\mathrm{TF}(t, d) = \frac{(\text{Number of occurrences of term } t \text{ in document } d)}{(\text{Total number of terms in the document } d)}$$

IDF (inverse document frequency)

- IDF (inverse document frequency) measures the importance of the term across a corpus.
- IDF of a term t is calculated as follows:

$$\mathrm{IDF}\left(t\right) = \log_e \frac{(\text{Total number of documents in the corpus})}{(\text{Number of documents with term } t \text{ in them })}$$

Example – Calculate TF-IDF using the following equations

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Term	Term count in Review 1	Term count in Review 2	Term count in Review 3
this	1	1	1
movie	1	1	1
is	1	2	1
very	1	0	0
scary	1	1	0
and	1	1	1
long	1	0	0
not	0	1	0
slow	0	0	0
spooky	0	0	1
good	0	1	1

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Term	Term count in Review 1	Term count in Review 2	Term count in Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
this	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	2/8	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	0	0	0	0	0
spooky	0	0	1	0	0	1/6
good	0	1	1	0	1/8	1/6

Caution: Understand the Concepts

- There are many variations of how to calculate TF, IDF and TF-IDF scores.
- Vector space model != bag-of-words
- Bag-of-words != TF-IDF
- Cosine similarity is superior to Euclidean distance (wrong)

Basic Vectorization Approaches: Summary

- They're discrete representations
  - i.e., they treat language units (words, n-grams, etc.) as atomic units. This discreteness hampers their ability to capture relationships between words.
- Pros: Empirically effective; Intuitive; easy to implement; well-studied.
- Cons: It assumes term independence. The feature vectors are sparse and high-dimensional representations. The dimensionality increases with the size of the vocabulary, with most values being zero for any vector (i.e., sparse vectors).
  - This hampers learning capability. Further, high-dimensionality representation makes them computationally inefficient.
- Cons: They cannot handle OOV (out of vocabulary) words. Directly applying these representations in the ML models will involve lots of parameter tuning.


QUIZ

Summary of Today's Class

- Vector Space Model
- Basic Vectorization Approaches
  - One-Hot Encoding
  - Bag of Words
  - Bag of N-Grams
  - TF-IDF

Exercises using Google Colab