TEXAS
The University of Texas at Austin

I320D – Topics in Human Centered Data Science
Text Mining and NLP Essentials

Week 4: Representing Words, Sentences and Documents, Bag-of-words, N-grams, TF-IDF , Word Vectors, Document similarity and distance metrics

Dr. Abhijit Mishra

Week 3: Recap

- Lecture:
  - Regular Expressions and Finite State Automata
  - Text Pre-processing Techniques – cleaning text, normalization, stop word removal
    - Morphological Analysis – stemming , lemmatization
    - When to apply which operations
- Tutorial:
  - Text pre-processing using NLTK and SpaCy libraries

Week 4: Roadmap

- **Lecture:**
  - Representing Words, Sentences and Documents,
  - Bag-of-words, N-grams, TF-IDF , Word Vectors,
  - Document similarity and distance metrics
- **Lab:**
  - Document Representation and Similarity Measurement
  - Building Semantic Search systems

References

Readings:

[1] Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text
https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

[2] Bag of words and N-grams: https://en.wikipedia.org/wiki/Bag-of-words_model

[3] NLP: Word Embedding Techniques for Text Analysis

[4] Word embeddings, LSA, Word2Vec, Glove, ELMo
https://people.eng.unimelb.edu.au/mbouadjenek/papers/wordembed.pdf

https://medium.com/sfu-cspmp/nlp-word-embedding-techniques-for-text-analysis-ec4e91bb886f

Optional Readings:

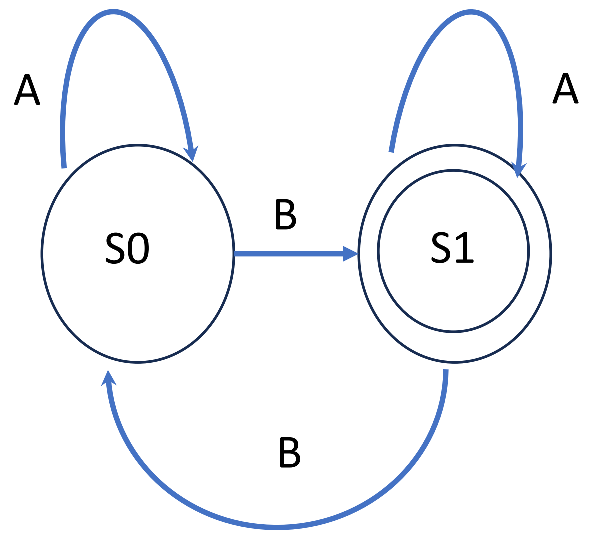
[1] Word Embeddings **https://people.eng.unimelb.edu.au/mbouadjenek/papers/wordembed.pdf**

Ongoing and upcoming assignment

- **Ongoing:** Shallow tweet search based on hashtags
  - Deadline: 02/11
- **Upcoming:** Semantic Search of Tweets (to be posted after Thursday's class)

Last week's exercise

- Design RegEx for any sequence of "A"s and "B"s with odd number of "B"s



A*B(A*BA*B)*A*

Lexical Analysis

Back to NLP Tasks

- **Classification**
- **Search**

Document Classification

Token Classification (sequence labeling)

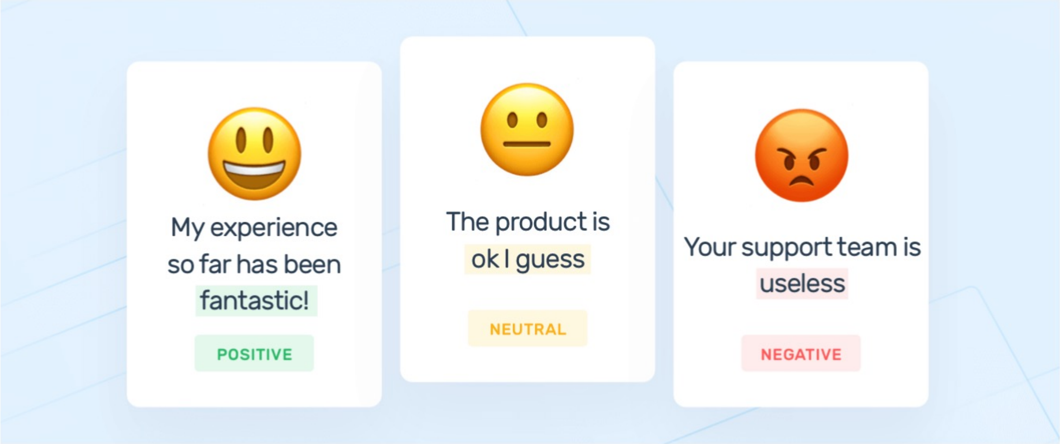
Text Generation (sequence to sequence)

Document Similarity Computation and Ranking

* Text generation is a special case of text classification

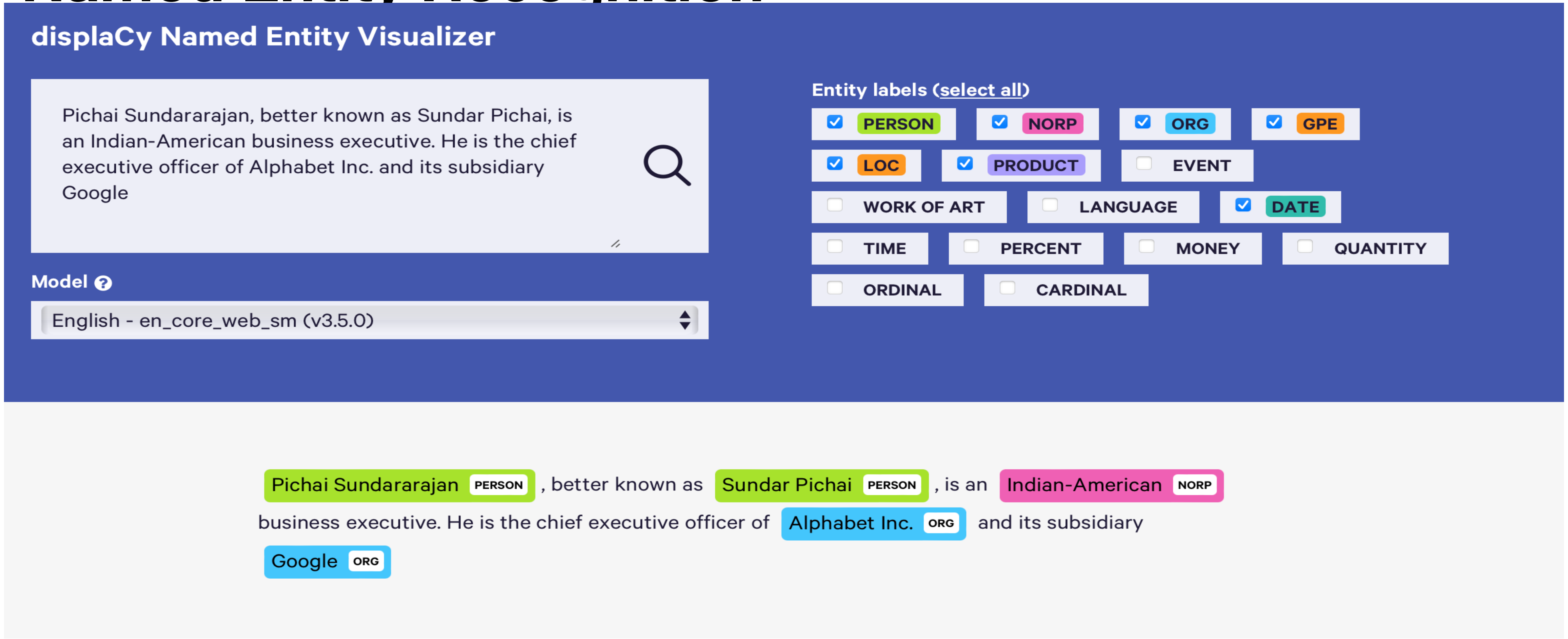
Document Classification

- Given a sentence / text snippet / document, categorize it into predefined categories based on its content / meaning
- Example:
  - Given a text, identify the sentiment expressed



Token Classification (Sequence Labeling)

- Task that involves assigning a label or tag to *each element* or *token* within a sequence of text
- Example: **Named Entity Recognition**



Sequence to Sequence (or Seq2Seq)

- Tasks involving the transformation or generation of one sequence into another (of varying length)
- In ML context: generative modeling or generative AI
- Example:
  - Machine Translation
  - **Input (French):** "Le temps est magnifique aujourd'hui.”
  - **Expected Output (English): “The weather is beautiful today”.**

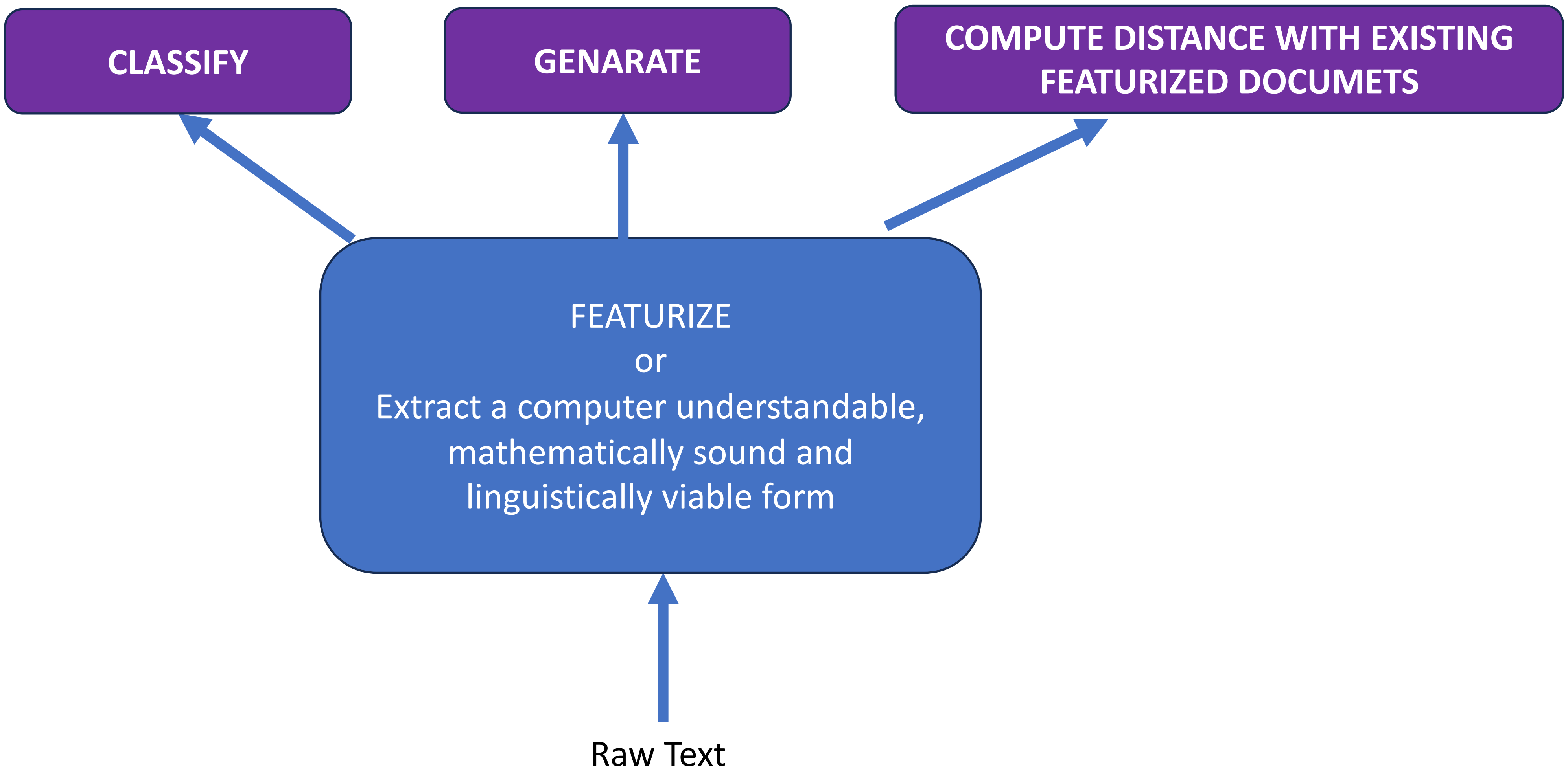
Distance Computation or Distance Based Document Ranking

- Given two document, quantify the how meaning-wise similar they are (a.k.a. semantic similarity)
- **Semantic Search (*a.k.a* information retrieval):** Given a query and N documents, measure the pairwise similarity between the query and each document, and rank documents based on similarity
  - Example: Semantic web-search, book search in library

Exercise: Identify the task type

- **Scenario 4:** Summarize a lengthy document into a concise paragraph Answer: Seq2Seq
- **Scenario 5:** Identify duplicate documents in a folder and delete them Answer: Document Similarity
- **Scenario 6:** Match job descriptions with the most relevant resumes from a pool of candidates. Answer: Document Similarity / Search

ML Centric Solutions for the 4-tasks



Featurization : Representing Words

- Extract a computer understandable, mathematically and linguistically acceptable format

Machine Readable Formats

Integers

Floats

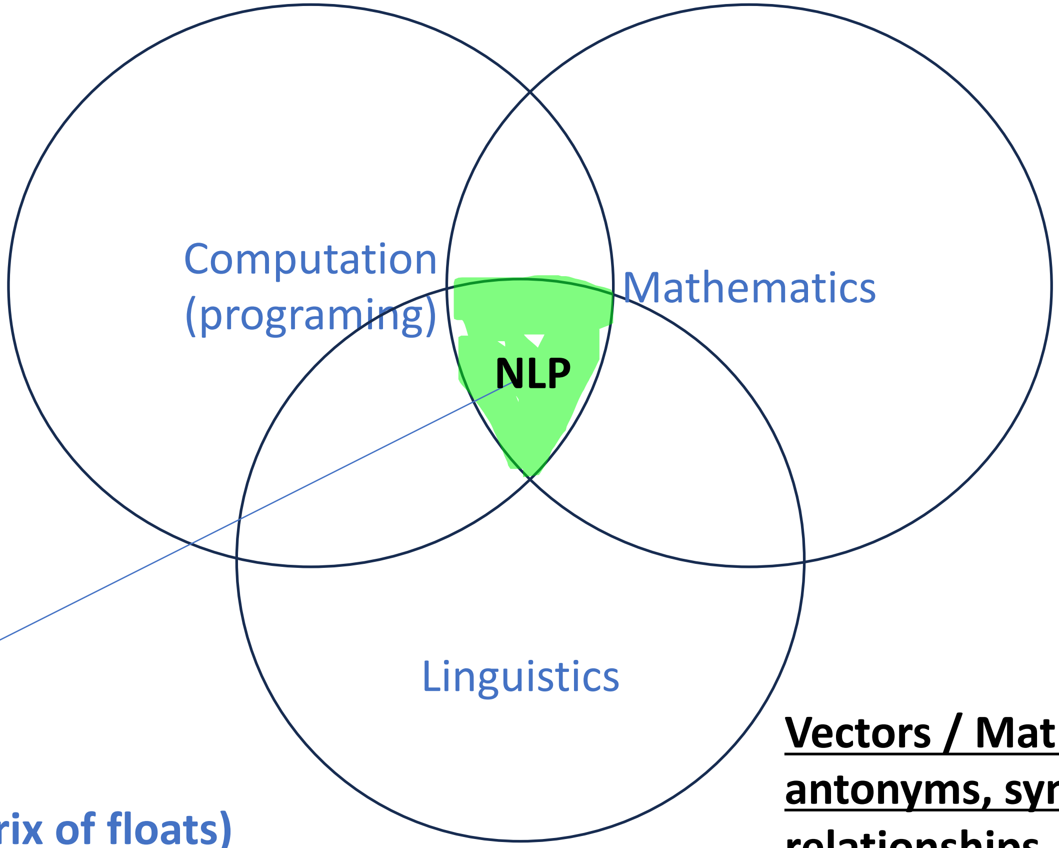
Strings

Pixels (tuples)

Waves

Lists (Matrix)

Set



Mathematical Data Formats

Integers

Rational numbers (Floats)

Real numbers

Complex numbers

Notations

Matrix

Set

Feature Vectors (matrix of floats)

Vectors / Matrices capturing synonyms, antonyms, syntagmatic and paradigmatic relationships

How to Represent Words in Documents

- 1-hot vectors
- Term-frequency – Inverse Document Frequency (TF-IDF)
- Word vectors learned using unlabeled corpora
  - Matrix Factorization based (e.g., Latent Semantic Analysis)
  - Neural Network based (Word2Vec, Glove)

1-hot Vectorization

- Words are categorical in nature – can represent in 1-hot format
- Consider this example

Let us learn machine learning.

- Extract words (or more formally tokens in the sentence)

[“Let”,“us”, “learn”, “machine learning”, “.”]

- Treat each unique word as a categorical representation
- Say, convert words to 1-hot vector

[“000001”,“000010”, “000100”, “001000”, “010000”, “100000”]

Example Corpus

- Consider we have three example sentences

1. *let us learn machine learning*
2. *machine learning emphasizes on learning programs from data*
3. *machine learning is a branch of AI*

Example

- Consider we have three example sentences

1. *let us learn machine learning*
2. *machine learning emphasizes on learning programs from data*
3. *machine learning is a branch of AI*

Unique words (a.k.a. vocabulary):

[“let”, “us”, “learn”, “machine”, “learning”, “emphasizes”, “on”, “programs”, “from”, “data”, “is”, “a”, “branch”, “of”, “AI”]

Example

- **Represent words in one hot form (recall encoder.fit( ) ?)**

“let”: [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]

“us”: [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]

“learn”: [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]

“machine”: [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

“learning”: [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]

“emphasizes”: [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

“on”: [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]

“programs”: [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]

“from”: [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]

“data”: [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]

“is”: [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]

“a”: [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]

“branch”: [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]

“of”: [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]

“AI”: [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

Problem with 1-hot vectorization

- Presence / absence based featurization does not specify the strength of the word
- Sometimes we maintain a count of occurrence of the words

Example (again)

- Consider we have three example sentences

1. *let us learn machine learning*
2. *machine learning emphasizes on learning programs from data*
3. *machine learning is a branch of AI*

Unique words (a.k.a. vocabulary):

[“let”, “us”, “learn”, “machine”, “learning”, “emphasizes”, “on”, “programs”, “from”, “data”, “is”, “a”, “branch”, “of”, “AI”]

Count Vectorization example

“let”: [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]

“us”: [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]

“learn”: [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]

“machine”: [0,0,0,0,0,0,0,0,0,0,0,3,0,0,0]

“learning”: [0,0,0,0,0,0,0,0,0,0,4,0,0,0,0]

“emphasizes”: [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

“on”: [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]

“programs”: [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]

“from”: [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]

“data”: [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]

“is”: [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]

“a”: [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]

“branch”: [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]

“of”: [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]

“AI”: [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

But

- We don’t have 1-sentence to deal with
- We have a corpus of millions of sentence
- Each unique word should be represented in the same way irrespective of wherever and how many times it appears

Representing Documents

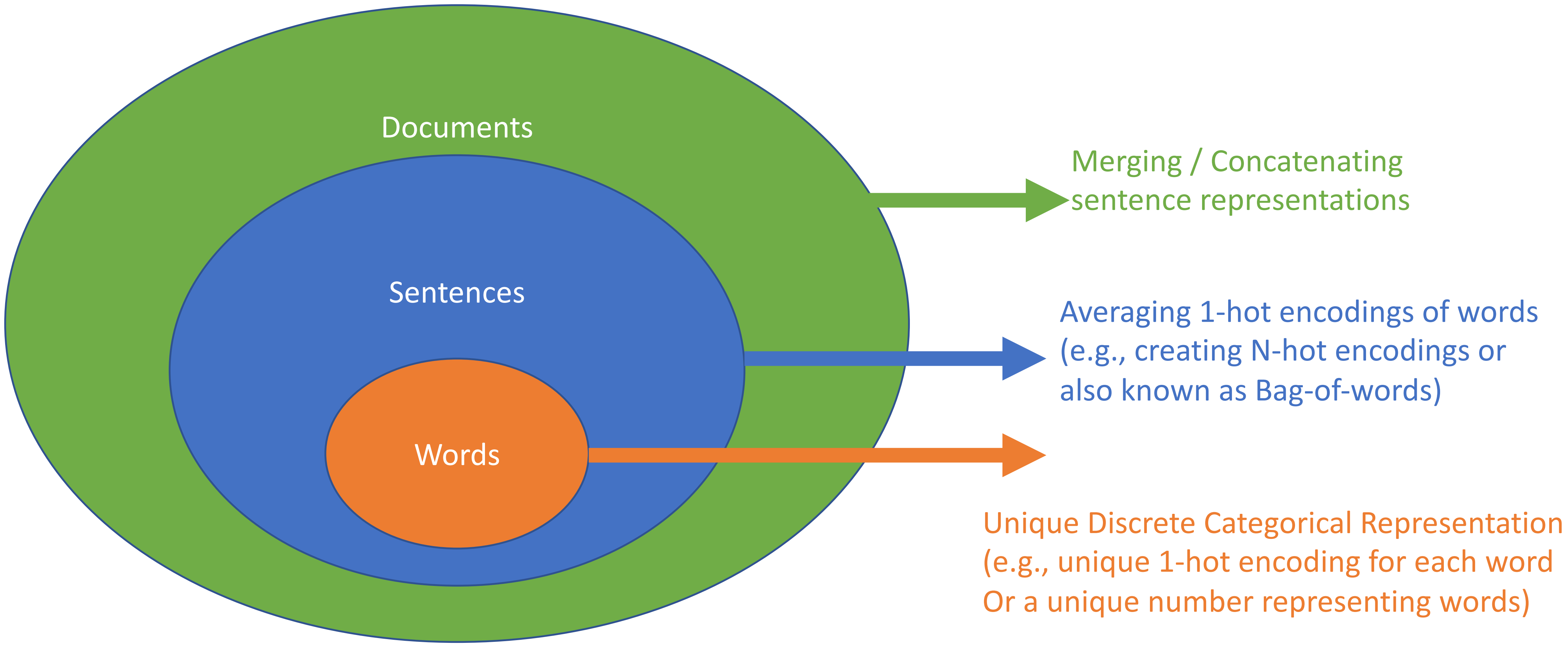
- **Objective**: Get fixed length vectors from variable length input
- Concatenating 1-hot vectors for words is not a good idea

1. *let us learn machine learning* (Five 1-hot vectors)
2. *machine learning emphasizes on learning programs from data*

(Seven 1-hot vectors)

3. *machine learning is a branch of AI* (Seven 1-hot vectors)

Representing text in computer understandable form



Representing Documents (BoW)

- **Objective**: Get fixed length vectors from variable length input
- Concatenating 1-hot vectors for words is not a good idea
- **Solution:** Form N-hot vectors of vocabulary size (a.k.a Bag-of-words)

E.g.

“machine learning emphasizes on learning programs from data”

`[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]`

“let us learn machine learning”

`[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

Feature Vector Length = number of unique words = vocab length

Representing Documents (BoW - count)

- **Objective**: Maintain Frequency instead of presence / absence

E.g.

"machine learning emphasizes on learning programs from data"

[0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

"let us learn machine learning"

[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Feature Vector Length = number of unique words = vocab length

Problems with BoW

- Does not respect sequential aspects of text (i.e., sequential context capturing)
- Treats all words equally irrespective of how common / uncommon they are
- Does not capture polysemy (i.e., contextual meaning variation)
  - Bank (river) = Bank (financial institution)

Representing Documents (TF-IDF)

- We can set the indices with normalized counts (also known as Term-frequency-Inverse- document-frequency or **Tf-Idf**)
- Determines how important is a word
  - $\mathrm{TF(w)} = \text{how many times a word}(w)\text{ appeared in the sentence}$
  - $\mathrm{IDF(w)} = \dfrac{\text{total number of sentences}}{\text{number of sentences word }(w)\text{ appears in}}$
- We can take $\log\big(\mathrm{IDF(w)}\big)$ to scale it better
- E.g.,
  - TF-IDF(“learning”) = TF(“learning”) * IDF(“learning”)

Representing Documents (TF-IDF)

Tf-Idf example:

1. *let us learn machine learning*
2. *machine learning emphasizes on learning programs from data*
3. *machine learning is a branch of AI*

TF-IDF(“learning”, 1) $=1 * \log\left(\frac{3}{3}\right) = 0$

TF-IDF(“AI”, 3) $=1 * \log\left(\frac{3}{1}\right) = \log 3 = 0.47$

Representing Documents (TFIDF)

Tf-Idf example:

1. *machine learning is a branch of AI*

TF-IDF(“machine”) $=1 * \log\left(\frac{3}{3}\right) = 0$

TF-IDF(“AI”) $=1 * \log\left(\frac{3}{1}\right) = \log 3 = 0.47$

TF-IDF(“branch”) $= 1 * \log\left(\frac{3}{1}\right) = \log 3 = 0.47$

…

Representation = `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.47, 0.47, 0.47, 0.47, 0.47]`

Exercise

- Compute TF * IDF for "*machine learning emphasizes on learning programs from data*"
- **Consider this corpus**

1. *let us learn machine learning*
2. *machine learning emphasizes on learning programs from data*
3. *machine learning is a branch of AI*

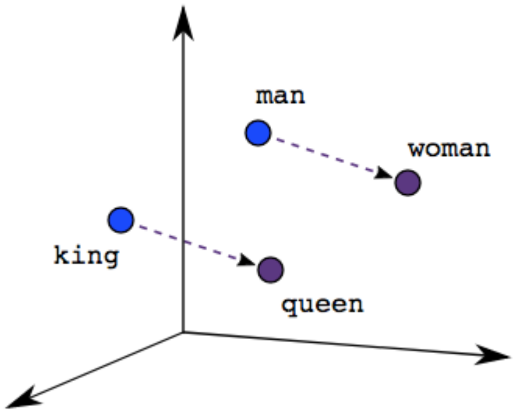
- **And this vocabulary**

["let", "us", "learn", "machine", "learning", "emphasizes", "on", "programs", "from", "data", "is", "a", "branch", "of", "AI"]

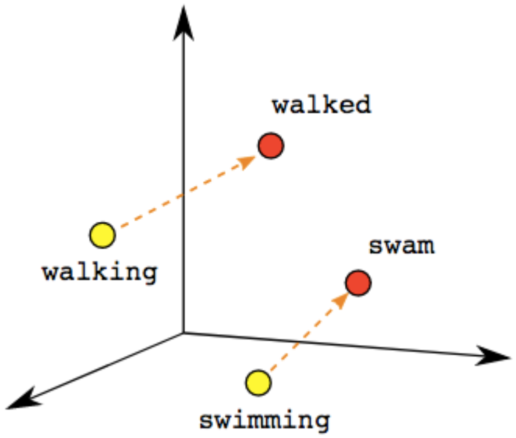
Getting richer representations

- One-hot / n-hot vectors are sparse, shallow, are not linguistically well motivated
- We rather want dense representations that capture semantic relationships between words
- **“Word vectors”** learned from large text corpora offer such dense representations
- Can generalize across tasks and reduce sparsity

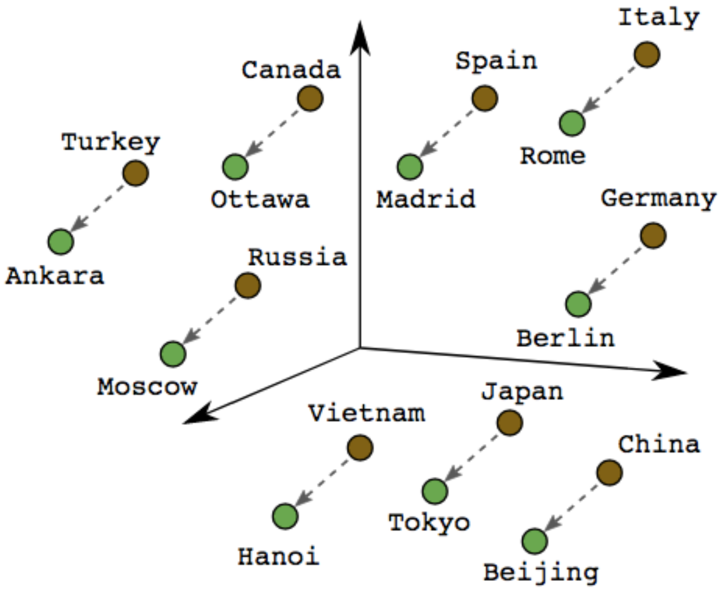
Word vector examples



Male-Female



Verb Tense



Country-Capital

How to use document representations for capturing semantic similarity?

Vector Based Semantic Similarity

- Intuition: Farther points are “dissimilar” in nature
- Distance between two N-dimenstional points explains (dis)similarity
- Two popular distance metrics

$$D(X, Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

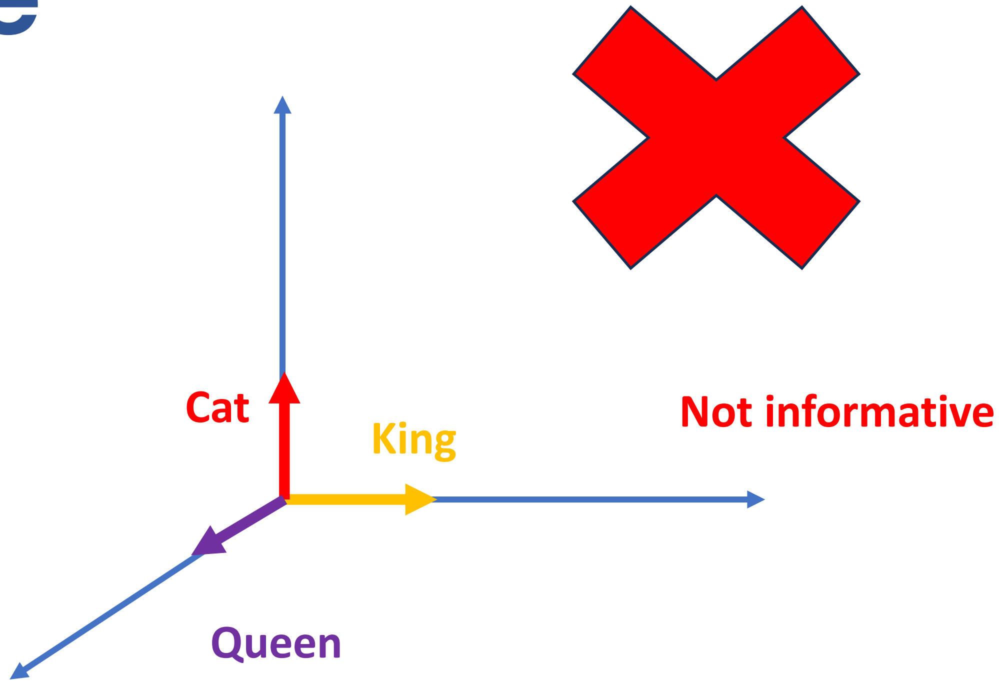
Euclidean Distance

$$D_{cosine}(X, Y) = 1 - \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2} \cdot \sqrt{\sum_{i=1}^{N} y_i^2}}$$

Cosine Distance

One-hot vector example

Word	1-hot vector
Queen	[1, 0, 0]
King	[0, 1, 0]
Cat	[0, 0, 1]

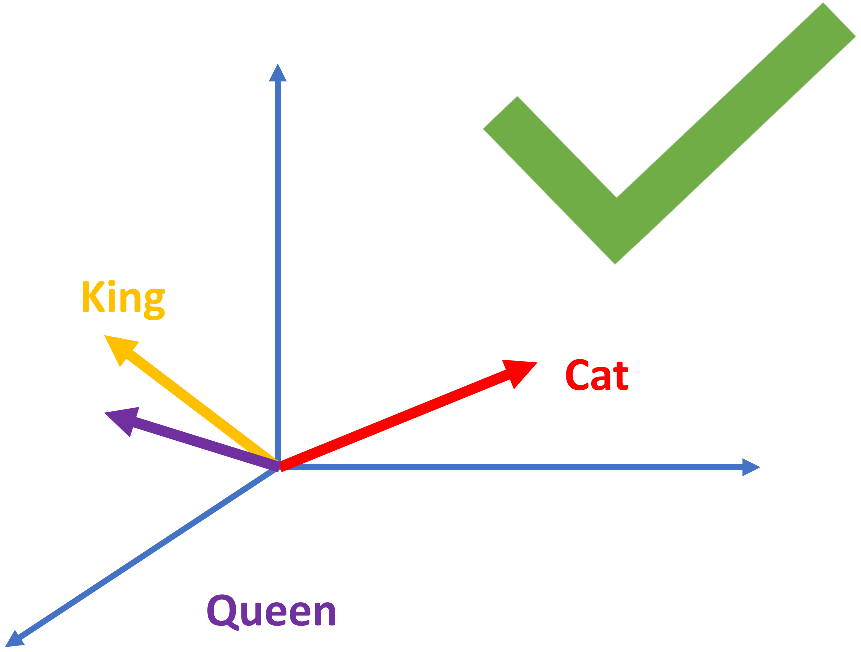


$$D_{cosine}(\text{“}Cat\text{”}, \text{“}King\text{”}) = D_{cosine}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = D_{cosine}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 1$$

$$D_{Euclid}(\text{“}Cat\text{”}, \text{“}King\text{”}) = D_{Euclid}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = D_{Euclid}(\text{“}King\text{”}, \text{“}Queen\text{”}) = \sqrt{2}$$

Instead, we need

Word	1-hot vector
Queen	[1.5, -1.3, -0.9]
King	[2.1, -0.7, 0.2]
Cat	[0.3, 1.9, -0.4]

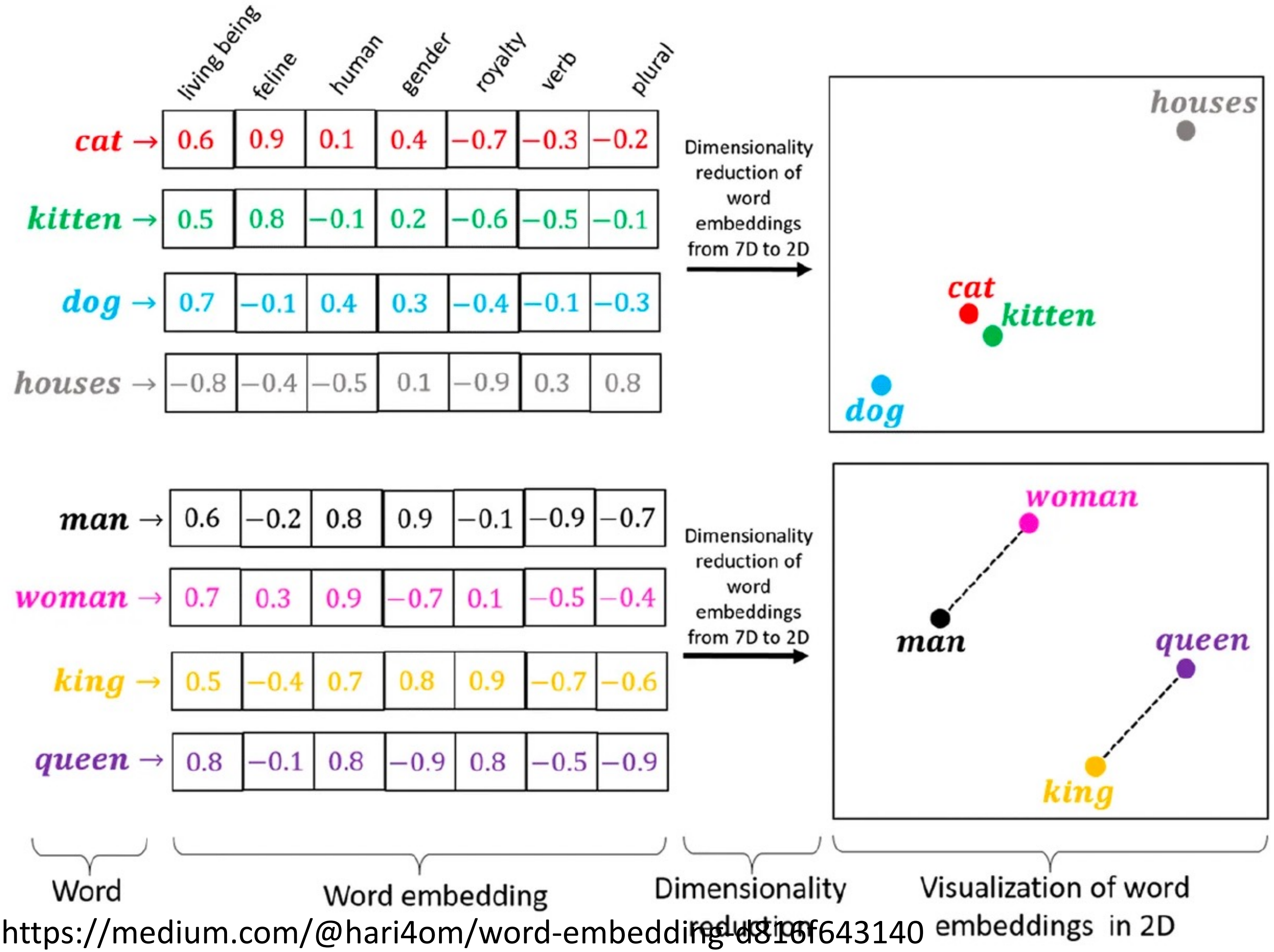


$$D_{cosine}(\text{“}Cat\text{”}, \text{“}King\text{”}) = 1.17, D_{cosine}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = 1.38,$$
$$D_{cosine}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 0.19$$

$$D_{Euclid}(\text{“}Cat\text{”}, \text{“}King\text{”}) = 3.21, D_{Euclid}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = 3.45$$
$$D_{Euclid}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 1.38$$

Representing words as “real” vectors

- Requirements:
  - Vectors should be dense i.e., N-dimentional, where N << vocabulary size
  - Vectors should be semantic “representations” of words
    - I.e., not random dense vectors
  - Vectors distances (similarities) should be interpretable and should capture semantic relationships across different dimensions



https://medium.com/@hari4om/word-embedding-d816f643140

Some pre-trained word vectors

- **Word2Vec:**
  - Developed by Google, Word2Vec is one of the earliest and most well-known word vector models.
  - It learns word embeddings by predicting the context words given a target word or vice versa.
  - Pre-trained Word2Vec models are available in various sizes and trained on large text corpora like Google News.

Some pre-trained word vectors (1)

- **GloVe (Global Vectors for Word Representation):**
  - GloVe is another popular word vector model that focuses on capturing global word co-occurrence statistics.
  - It leverages both local and global context to create word embeddings.
  - Pre-trained GloVe models are available in different dimensions and trained on diverse text sources.

How to get sentence level features from word vectors?

- Given a tokenized input sentence of N tokens , $s = [w_1, w_2 \ldots, w_N]$
- Download and initialize a pre-trained word vector (such as GloVE)
- for each token w_i
  - Find a vector for w_i by “looking u”p in GloVE
  - If token not found, assign a default zero vector to the token
- Average all token vectors to get a sentence level representation

Problems with word vectors

- Still do not respect sequential aspects of text (i.e., sequential context capturing)
- Do not capture polysemy (i.e., contextual meaning variation)
  - Bank (river) = Bank (financial institution)
- Computational complexity
  - Not as light-weight as BoW

Solution?

- Direct context vector extraction from sentences / paragraphs
  - E.g., Bidirectional Encoder Representations from Transformers (BERT)
  - Recently developed Large Language Models e.g., GPT series

Summary

- In this lecture:
  - We discussed various techniques to represent words and documents
  - Adequate multidimensional representations of documents crucial for classification and similarity measurement
  - We did not go through technical details of how word vectors are built (planned under WEEK 11. Deep learning for NLP - II (Mar 25 - Mar 29)
- **Next class:**
  - **Lab: document representation and semantic similarity measurement**