

Unsupervised and Unstructured Machine Learning

BA820 – Mohannad Elhamod

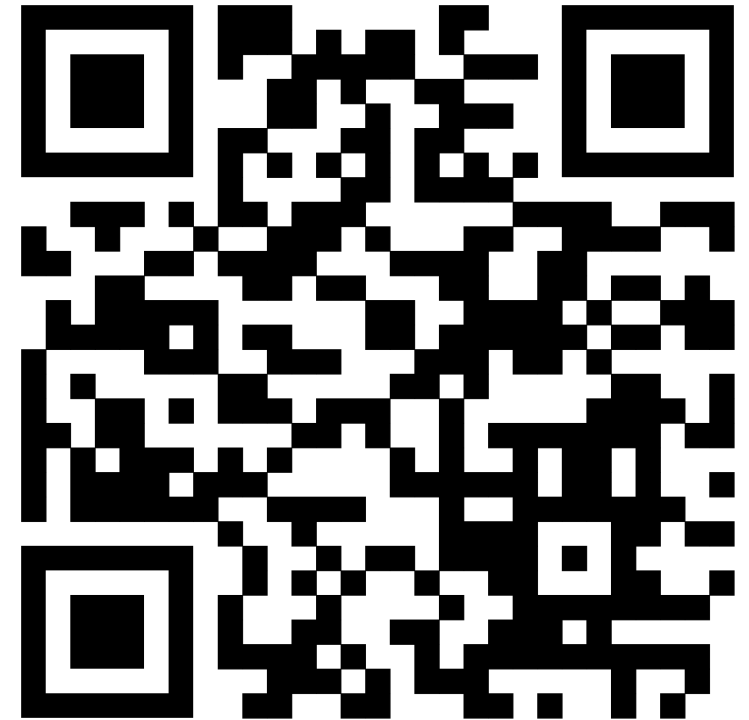
Midterm Review



Start Stop Continue

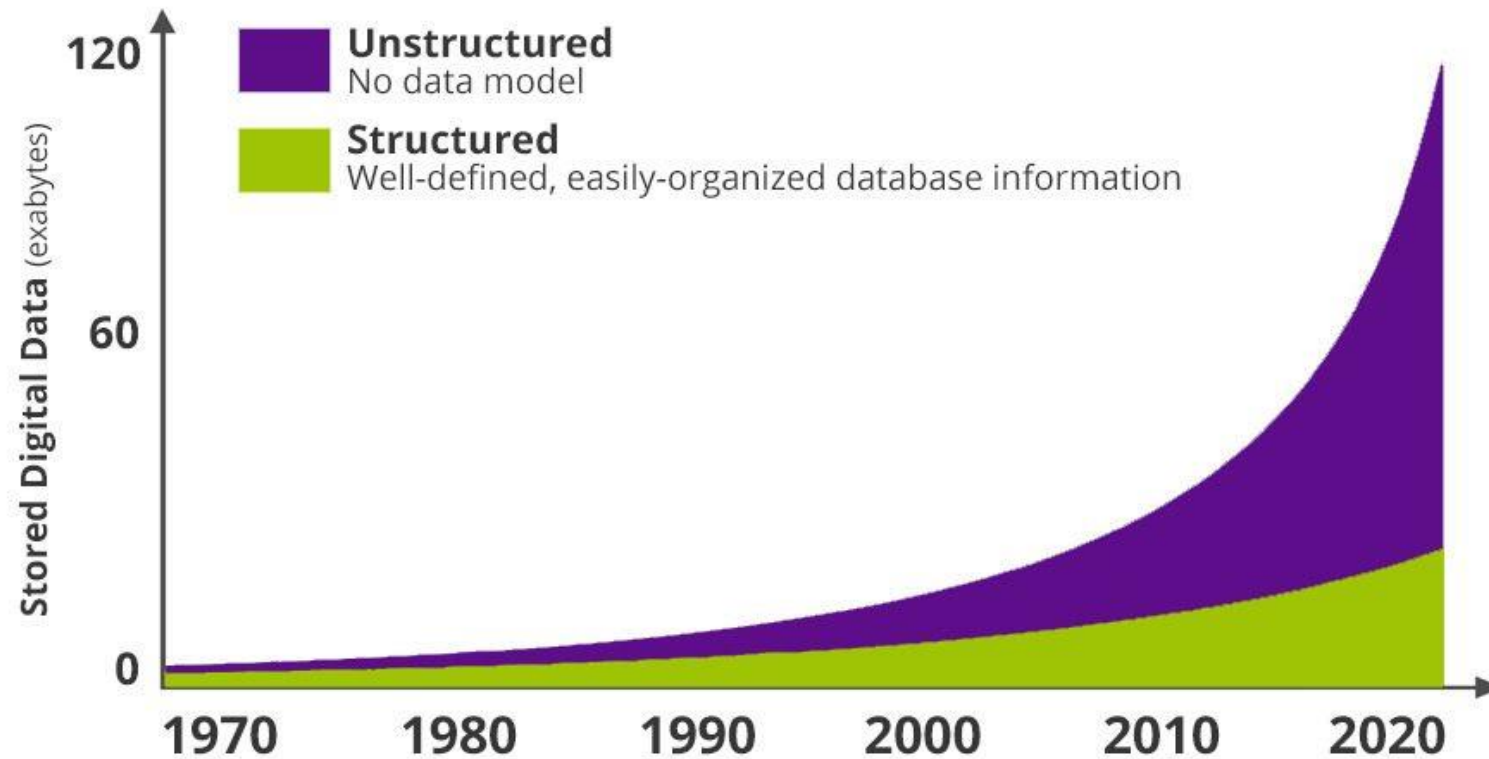
Please fill this out

- <https://forms.gle/VuHJUjtzLrrr1TNY8>



Text Mining

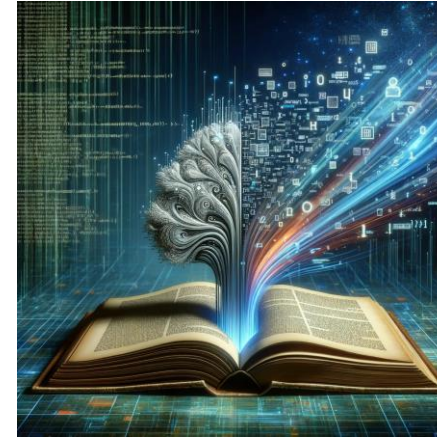
Most Data is No Longer Structured...



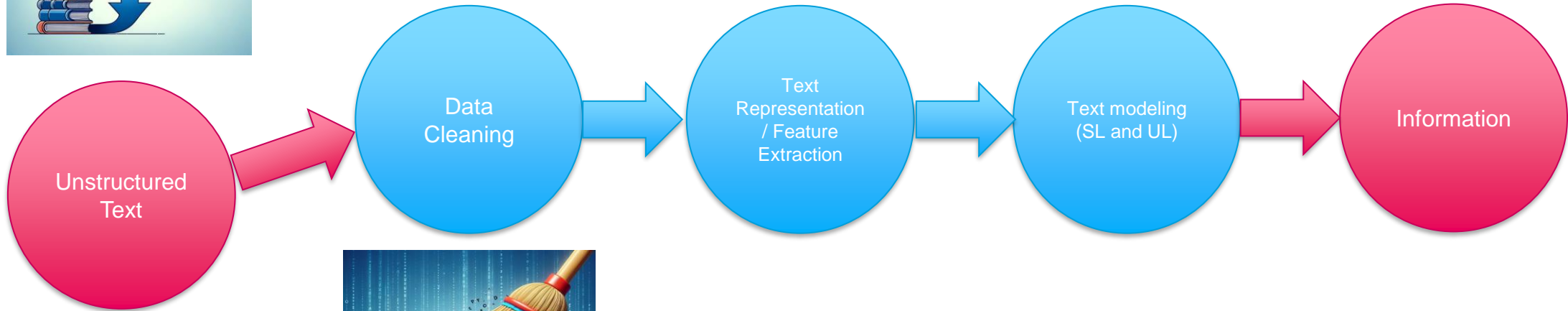
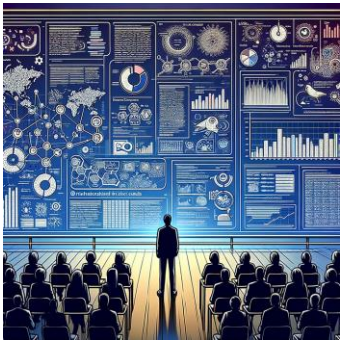
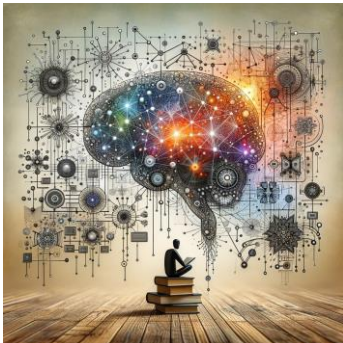
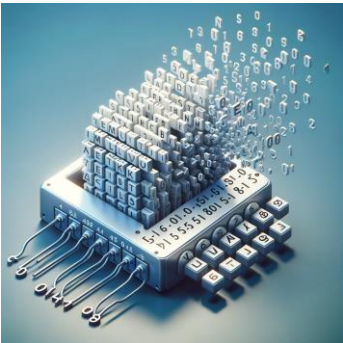
<https://www.komprise.com/>

What can we do with this data?

- Text classification
- Text generation
- Text summarization
- Music recommendation
- Image categorization
-



Text Mining



Collecting Unstructured Data

- How can we extract this data?
 - Datasets found publicly (UCI, Kaggle, [Common Crawl](#), [Wikimedia Downloads](#), etc.).
 - Using APIs (e.g., [twitter API](#), [News API](#)).
 - Web scrapping (e.g., [BeautifulSoup](#), [Selenium](#))
 - Own private data.



Data Cleaning with *Regex*

- Just like structured/tabular data, we generally need to clean up the text to make it more useful.
 - Examples: case-sensitivity, punctuation, etc.
- *Regex* is used for finding string matches and formatting text:
 - Playground:
https://www.w3schools.com/python/python_regex.asp
 - Cheat sheet:
<https://www.debuggex.com/cheatsheet/regex/python>

How to Represent Text?

- Computers do not understand text...
- We need to represent text in a language they understand... numbers!
- Simple proposals (we are just brainstorming here):
 - Each sentence is represented in terms of the words it contains...
 - This is called Tokenization.
 - Each word is represented by a number...
 - This is called Vectorization.

Tokenization: Some Terminology

- **Document:** A body of text (e.g., a tweet, a pdf, an article, etc.).
- **A token:** The building block of a document.
 - Examples: character-level, word-level, ...
- **A separator:** Special tokens that split a document into tokens.
 - Examples: punctuation, spaces,
- Demo

Need for Advanced Text Pre-Processing

- Some tokens may occur too frequently in any text without contributing much to its meaning. These are called stop words and are generally removed.
- Other issues:
 - **Stemming:** big, bigger, biggest
 - **Lemmatization:** drive drove driven
 - **Homonyms:** bank (river or money?)
 - **Synonym:** Yes, sure.
- We will come back to all of this later...

Syntax vs. Semantics

- Notice that:
 - Different tokens might have the same meaning (e.g., like, enjoy)
 - The same token might have different meanings (is *like* something, to *like* something)
- **So, while tokens represent syntax, we really care about meaning/semantics.**
- In many cases, you can only get the meaning through **context** (i.e., the token's place with respect to other tokens.)
- We will come back to all of this later...

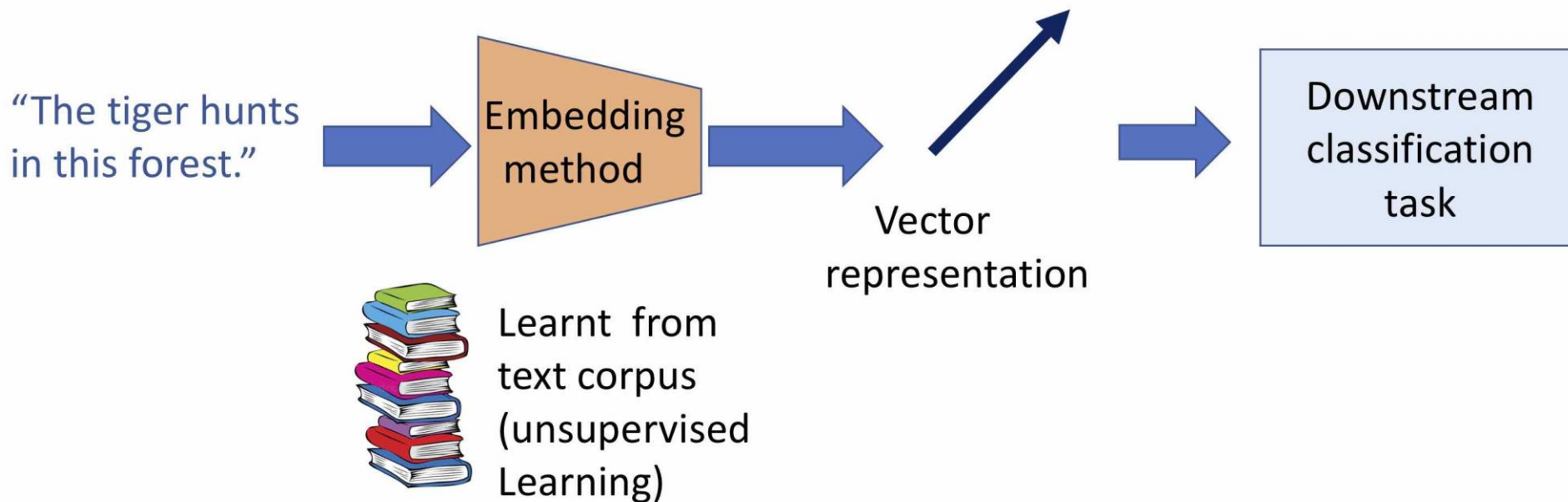
Text Modeling

- Once text data is in the proper representation (i.e., tokenized and vectorized), we can apply the methods we have learned so far:
 - Unsupervised ML (e.g., dimensionality reduction, clustering, etc.).
 - Supervised ML (e.g., classification, translation, etc.).

Text Vectorization

Vectorization: Text as Numbers

Typical pipeline for unsupervised text embedding



[Offconvex.org](https://offconvex.org)

How Would You Represent a Document?

- Let's start simple. We could represent a document simply as a collection of tokens.
 - Vectorization by document (not token).
- This approach is called Bag of Words (BoW).
- Demo**

Sentence	hockey	fun	i	like	golf
I like golf!			1	1	1
I like hockey.	1		1	1	
Hockey and golf are fun!	1	1			1

Bag of Words (BoW)

- **Cons:**

- Disregards word order!
- Number of features can be exhaustive.
- **Frequency bias.**
 - (e.g., If the word “space” appears in a children’s book, it carries more significance than when it appears in an article about galaxies.

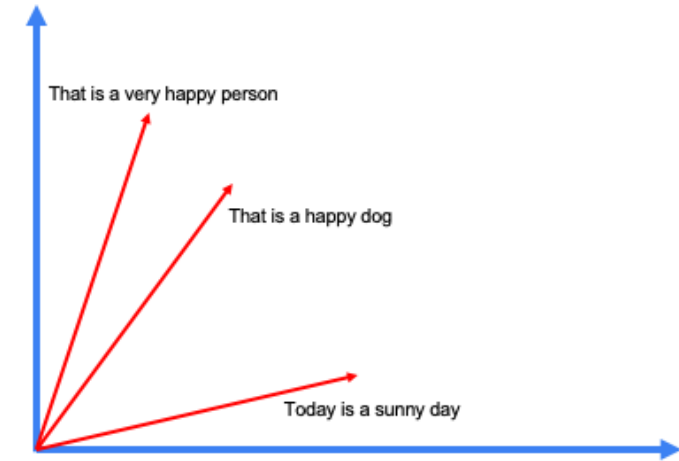
- **Pros:**

- Simple to implement

	I	hate	love	golf	soccer
I hate golf and love soccer	1	1	1	1	1
I hate soccer and love golf	1	1	1	1	1

Document Similarity

- How do we measure if two documents are similar?
 - We need a metric like Euclidean distance.
 - But... What if documents have different lengths?
- We need a metric that is robust to differences in document size...
 - Enter Cosine Similarity.

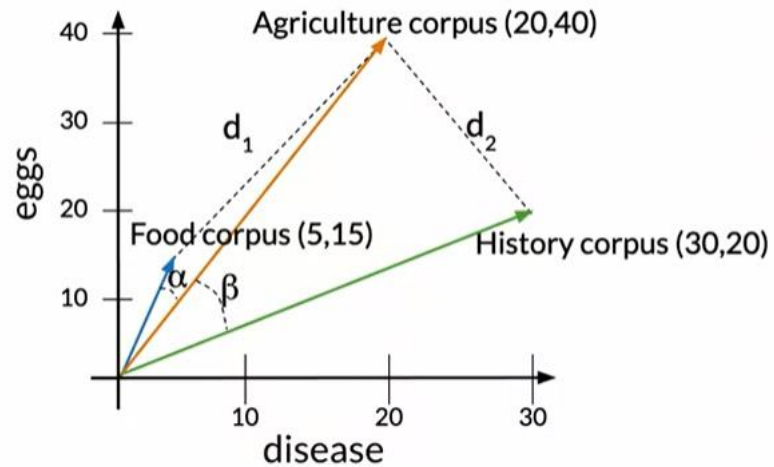


<https://mlops.community>

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Document Similarity

Euclidean distance vs Cosine similarity



The cosine of the angle
between the vectors

[medium.com](https://medium.com/@davidfriesen/euclidean-distance-vs-cosine-similarity-400000000000)

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Solving for Frequency Bias: TF-IDF

- Term Frequency – Inverse Document Frequency
- Instead of simply counting the absolute number of occurrences, how about we adjust in terms of token presence across documents?

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

$$w_{x,y} = tf_{x,y}$$

BoW frequency

→ Portion of documents the token appears in

Solving for Frequency Bias: TF-IDF

- The more commonly a word exists across documents the less important it is!
- What happens when a term (i.e., token) appears in all documents?
- **Demo**

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

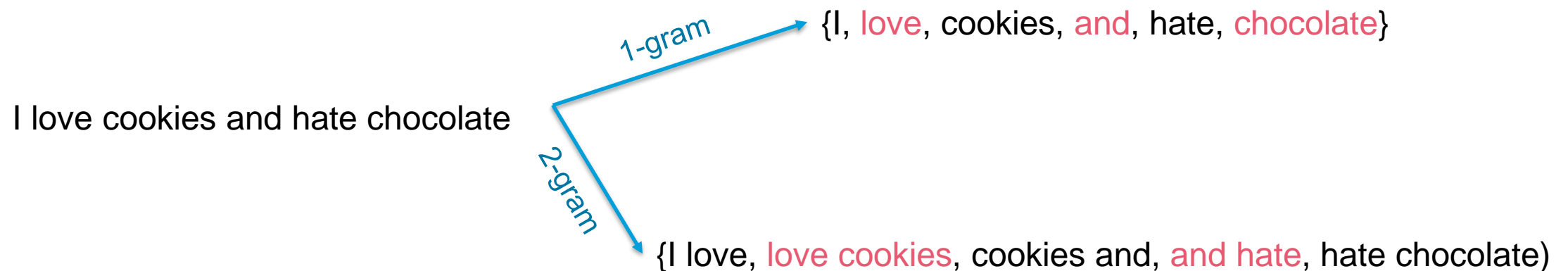
Context Matters...

So far...

- We calculated a score for each individual token (e.g., the frequency of the word in BoW).
- Have our methods captured context so far?
 - Context has so far been captured as *“the collection of words that appear together in a document”*, regardless of order.

Can We Do Better?

- Tokens that are immediately next to each other are generally more related to each other.
- Can we apply this concept at the tokenization level?



n-grams

- An n -gram is a sequence of n items (i.e., tokens).
- n refers to the context size.
- The larger n is, the more context a token captures.

How Much Context Is Needed?

- How do small and large n values compare?
 - A large n captures more context, but it's more restrictive in terms of finding matches.
 - A small n captures less context. but a token is more likely to be matched in other documents.
- Maybe we can include both? Any draw backs?
- **Demo**



Meaning: The Illusive Goal...

Where Does Meaning Come From?

I went to the **branch** and deposited some money.

I went to the **bank** and deposited some money.

I went to the **ATM** and deposited some money.

Words which frequently appear in **similar contexts** have **similar meaning**.

[Lena-votta](#)

Representing Meaning as a Vector

- So, why don't we construct vector representations such that words that appear in the same context have similar vectors.
 - embeddings = vector representation = features = latent space.
 - v_1 and v_2 are similar $\Rightarrow \cos_similarity(v_1, v_2) \approx 1$
- Example from images.

Word Embeddings

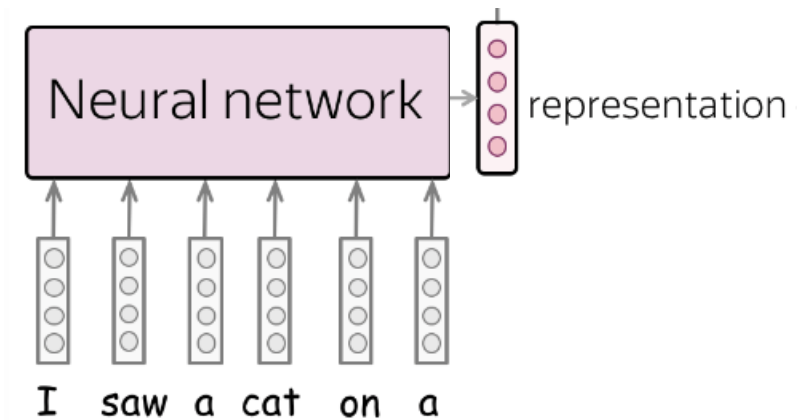
- Represent each word as a vector of some pre-defined size/dimensionality.
- You can visualize these embeddings using PCA or t-SNE.
- Some interesting properties will appear:
 - [Demo 1 \(directionality\)](#)
 - [Demo 2 \(semantics\)](#)

How are such embeddings constructed?

- Collect and tokenize some training corpus.
- For each token, start with a randomly initialized vector.
- Construct a table showing which tokens co-occur and which don't.
- For words that co-occur, push the vectors towards each other.
- For words that don't, push them away from each other.
- Examples of word embeddings:
 - [Word2Vec](#)
 - [GloVe](#)

Something is still missing...

- Well, now we have word embeddings...
 - But, how do we get sentence embeddings?!
- We need to somehow aggregate the word embeddings.
 - There are many ways to do this...
 - The most basic one is taking the average embedding.
 - Another way is to take the TF-IDF weighted averaging.
 - Or.... Using a *neural network*. Welcome to Deep Learning!



So, what's the big deal?

Demo (Predict the next word)

Demo

Recap

Model	Captures Token Position/Order	Captures Semantics/Meaning?
BOW	No	No
N-gram	Yes	No
Word2Vec (with averaging)	No	Yes

Is there any hope of capturing both?

Application 1: Sentiment Analysis

Approaches

- Supervised:
 - We have an annotated dataset (positive vs. negative sentences).
- Unsupervised:
 - Unsupervised ML is mostly about similarity...
 - So, how about we use compare to positive and negative sentences as an indication of sentiment.

Unsupervised Sentiment Score

- S_p : Similarity to a highly positive embedding
- S_n : Similarity to a highly negative embedding
- A sentiment score would be:
 - $S = S_p - S_n$
- **Demo**

What About Clustering?

- Clustering groups datapoints that are “similar”.
- This similarity may or may not align with sentiments.

Application 2: Topic Modeling

Non-negative Matrix Decomposition

- Works by approximating the text representation as the product of two matrices:
 - W : text in terms of topics
 - H : topics in terms of tokens.
- **Demo**

