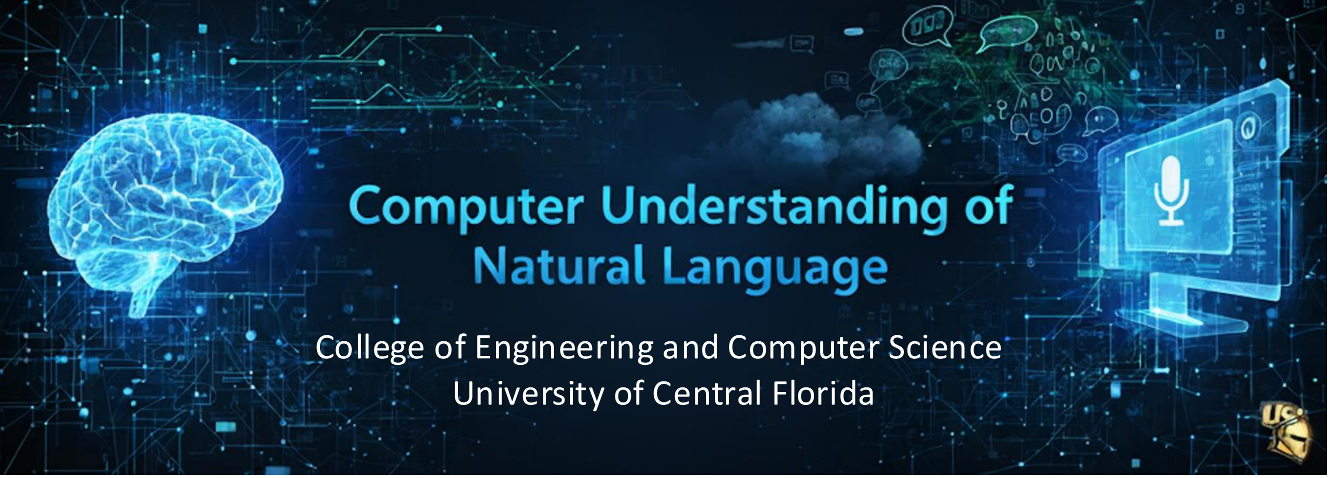


David Mohaisen, Professor
University of Central Florida
Department of Computer Science
Cybersecurity and Privacy Cluster

https://cs.ucf.edu/~mohaisen/

Advanced Word Embeddings and Evaluation

① Gradient descent, revisited
② Skip-gram with negative sample
③ Window-based representation
④ Count-based vs. direct prediction
⑤ Embedding meanings, word sense
⑥ How to evaluate word vectors

Gradient Descent

- $f(x) = x^2$, $f'(x) = 2x$, $f'(0) = 0$.
- Initial guess $x = 4$; $f'(x) = 2*4 = 8$
- Update: $x_{new} = x_{old} - a\, f'(x_{old})$
  $x_{new} = 4 – 0.1 * 8 – 3.2$
  $f(3.2) = 2*3.2 = 6.4$
  …
- Repeat: x_{new} = 3.2 – 0.1*6.4 = 2.56; f(x) = 2*2.56=5.12
  …

- Update equation (in matrix notation):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size* or *learning rate*

- Update equation (for single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

Gradient Descent

- Define variables:
  - Predictions (vector of predicted values):
  - Target (actual values):
  - The function is MSE loss; n = 3:

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \hat{y}_3]$$

$$\mathbf{y} = [y_1, y_2, y_3].$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Compute derivative w.r.t. th

$$\frac{\partial L}{\partial \hat{\mathbf{y}}} = \frac{2}{n}(\hat{\mathbf{y}} - \mathbf{y})$$

- Substitute values; let's say:
  - Compute difference:
  - Compute the gradient:

$$\hat{\mathbf{y}} = [2.0, 3.0, 4.0]$$

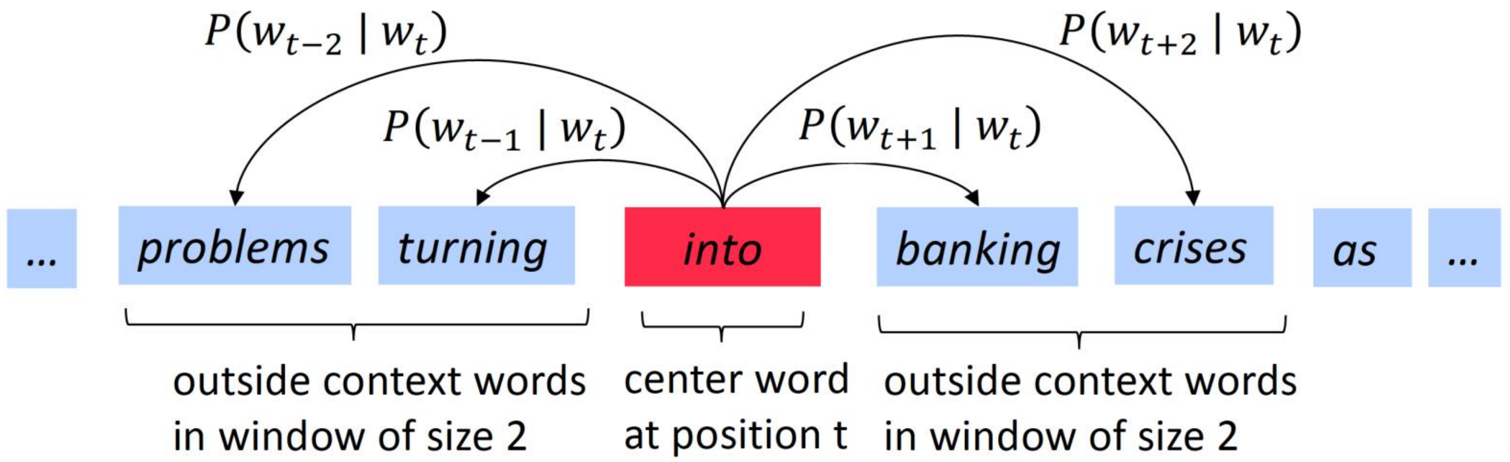
$$\mathbf{y} = [1.5, 3.5, 4.5]$$

$$\hat{\mathbf{y}} - \mathbf{y} = [2.0 - 1.5, 3.0 - 3.5, 4.0 - 4.5] = [0.5, -0.5, -0.5]$$

$$\frac{\partial L}{\partial \hat{\mathbf{y}}} = \frac{2}{3}[0.5, -0.5, -0.5] = [\frac{1}{3}, -\frac{1}{3}, -\frac{1}{3}]$$

Summary of Word2Vec

- Iterate over each word for the whole corpus
- Predict surrounding words using word vectors



$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- Update vectors so you can predict well

Stochastic Gradient Descent

- Problem: $J(\boldsymbol{\theta})$ is a function of all windows in the corpus (potentially billions)
  - So the update of $J(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$ is expensive
- You would wait a very long time before making a single update.
  - Very bad idea for pretty much all problems
- Solution: SGD: repeatedly sample windows, and update after each one.

Skip-gram with Negative Sample

- The normalization is too expensive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- Hence, we implement the skip-gram
- Idea: train a binary logistic regression for a true pair (center with actuals) versus several noise pairs (center with random words)

Skip-Gram with Negative Sampling

- Overall objective function to maximize

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J_t(\theta)$$

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j \sim P(w)} \left[\log \sigma\left(-u_j^T v_c\right)\right]$$

- Meaning: we maximize two words co-occurring in first log.

Skip-Gram: Procedure

- We take k negative samples (using the word probabilities)
- Maximize the probability that real outside word appears, minimize prob. that random words appear around the center word.
- Why not capture co-occurrence directly:
  - Window: similar to word2vec, use window around each word ⬄ capture both syntactic and semantic information.
  - Word-document co-occurrence matrix gives general topics (all sports terms will have similar entities) leading to ”latent semantic analysis”.

Skip-gram with Negative Sample: Details

- **Objective:** Learn dense word embeddings by predicting context words given a target word.
- **How It Works:**
  1. **Input:** Target word and its context window.
  2. **Positive Samples:** Word pairs from the corpus (w_{target}, $w_{context}$).
  3. **Negative Samples:** Random pairs (w_{target}, w_{noise}) generated from a noise distribution.

1. **Training:** Binary classification to distinguish real (positive) from random (negative) pairs.
2. **Output:** Embeddings that capture semantic similarity.

- **Advantages:**
  - Efficient for large vocabularies.
  - Captures semantic relationships (e.g., analogies, similarities).
  - Widely used in NLP tasks (e.g., translation, sentiment analysis).
- **Applications:** Pre-trained embeddings for downstream NLP tasks, semantic search, etc.

Example: Window-Based

- Window length = 1 (common 5-10)
- Symmetric (irrelevant whether left or right context).
- Example corpus:
  - I like deep learning
  - I like NLP
  - I enjoy flying

Example: Window-Based, cont.

- Example corpus:
  - I like deep learning
  - I like NLP
  - I enjoy flying

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Window-Based, cont.

- Solution: low dimensional vectors
  - Store most of the important info in a fixed small number of dimensions: dense vector
  - Usually 25-1000 dimensions, similar to word2vec
  - How to reduce the dimensionality?
    - Singular value decomposition (SVD) of co-occurrence matrix X; factor X in to USV^T such that U and V are orthogonal. Retain *k* singular values to generalize.
      - Expensive to compute for large matrices.

Example in Python

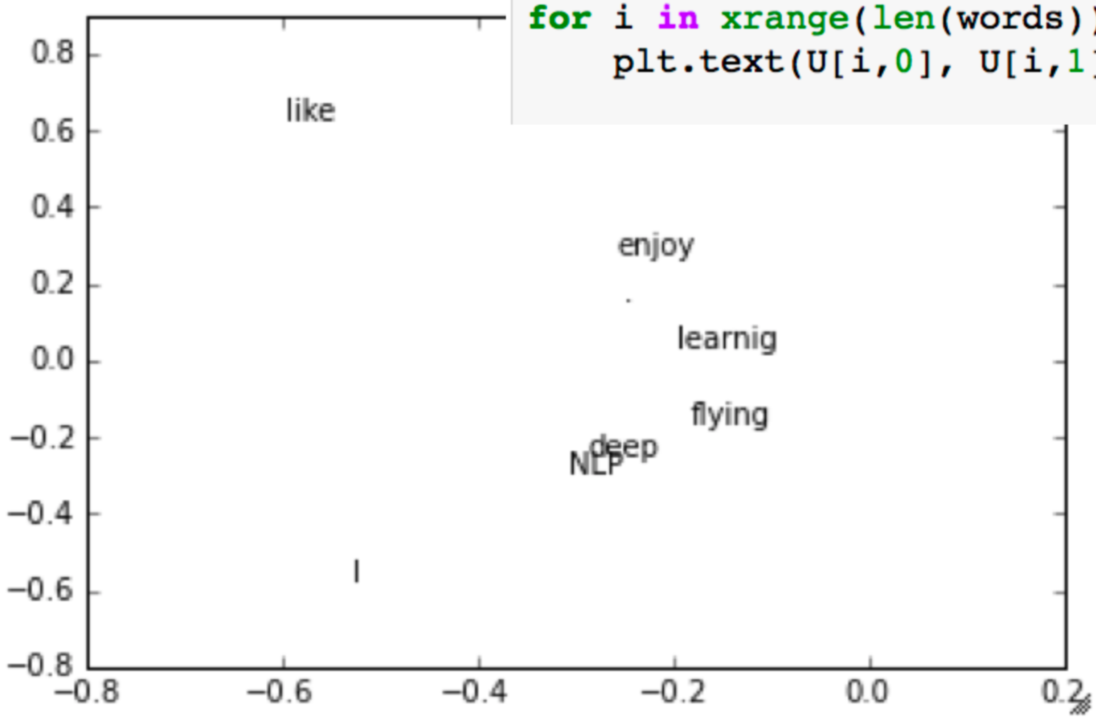
```python
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

Example: Visualization

- Printing first two columns of U corresponding to the biggest singular values

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



Hacks to X

- Scaling the count helps a lot:
  - Problem: function words (the, he, has, a, etc.) are too frequent ⇔ syntax has too much impact. Some fixes:
    - min(X, *t*), with *t*=100
    - Ignore them all
  - Ramped windows that count closer words more
  - Use Pearson correlations instead of counts, then set negative values to 0
  - Other approaches?

Count-Based vs. Direct Prediction

Count-based

- Fast training
- Efficient use of stats
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

Direct-Prediction

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Encoding Meaning

- Rations of co-occurrence probabilities can encode meaning components

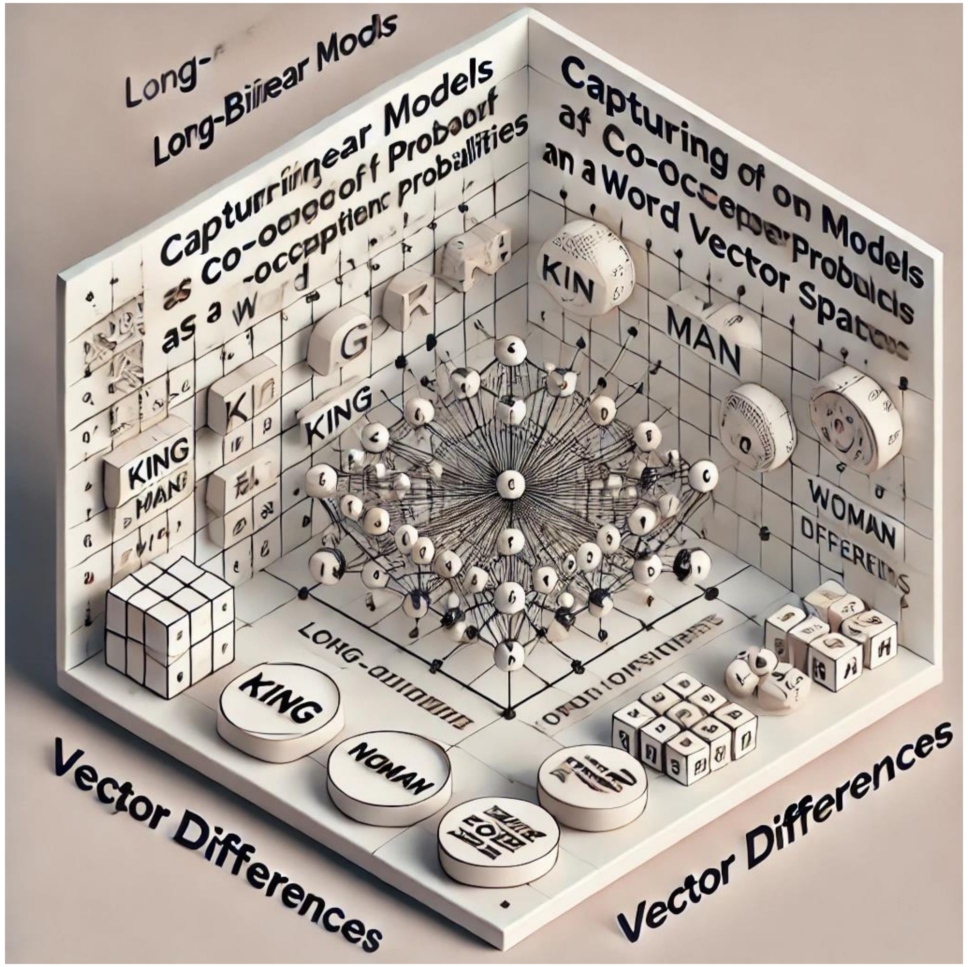
	x = solid	x = gas	x = water	x = random
$P(x \mid \text{ice})$	large	small	large	small
$P(x \mid \text{steam})$	small	large	large	small
$\frac{P(x \mid \text{ice})}{P(x \mid \text{steam})}$	large	small	~1	~1

Encoding Meanings

- How can we capture ratios of co-occurrence probabilities as a linear meaning components in a word vector space?

A: Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

with vector differences $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$



How to Evaluate Word Vectors?

- Intrinsic:
  - Specific subtask
  - Fast to compute
  - Help understand the system
  - Not clear if real helpful unless correlation to real task is established
- Extrinsic
  - Evaluation on a real task
  - Can take long time to compute
  - Unclear if subsystem is the problem or its iterations
  - If replacing exactly one subsystem with another improves accuracy ⬄ winning

Intrinsic Evaluation

- **Focus**: Measures the quality of embeddings on tasks that directly test word similarity or relatedness.
- **Methods**:
  - **Word Similarity Tasks**: (1) Use human-annotated datasets like WordSim-353 or SimLex-999. (2) Evaluate cosine similarity b/w vectors and compare against human judgments.
  - **Analogy Tasks**: (2) Solve analogies like "king - man + woman = queen." (2) Popular dataset: Google Analogy Dataset.
  - **Clustering**: Group similar words together and measure coherence.
  - **Qualitative Analysis**: Visualize embeddings using techniques like t-SNE or PCA.
- **Advantages**: (1) Quick and interpretable. (2) Independent of specific NLP applications.
- **Disadvantages**: May not reflect performance on downstream tasks.

Extrinsic Evaluation

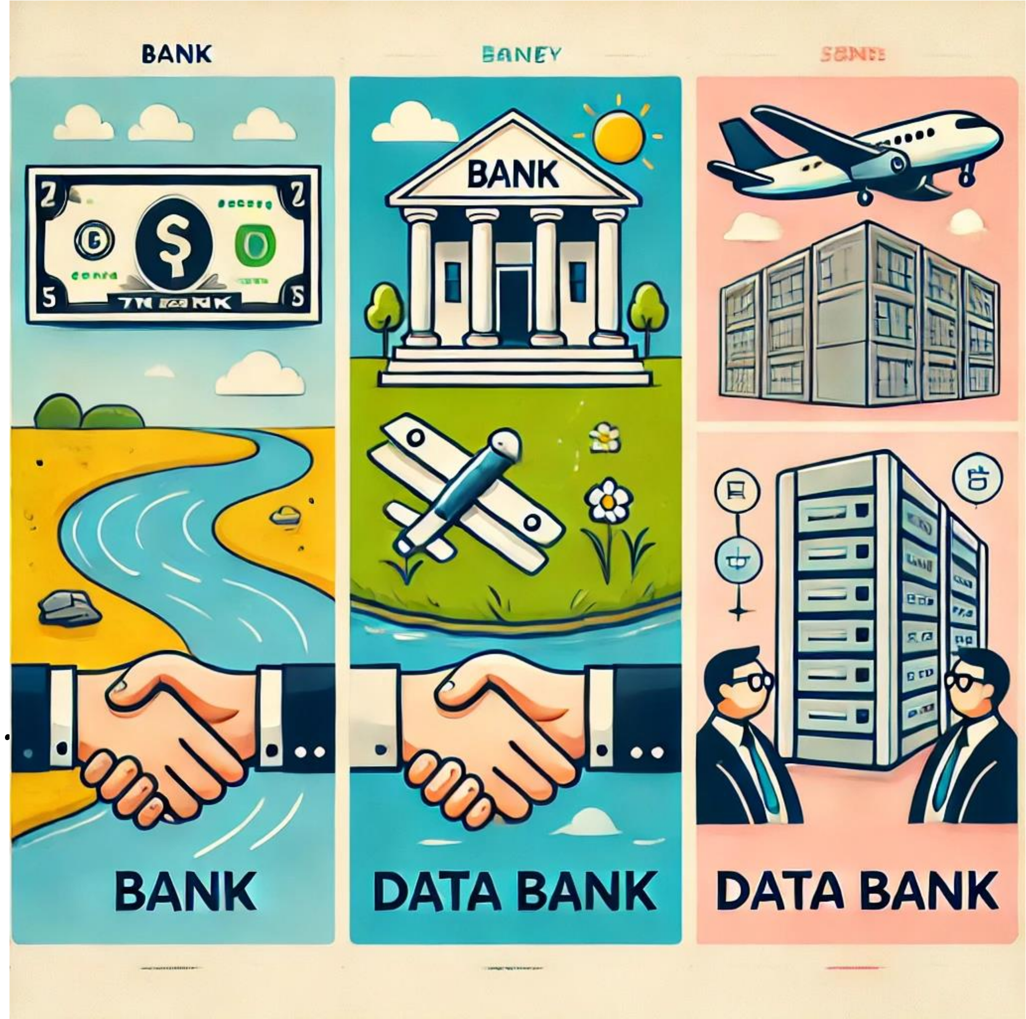
- **Focus**: Measures the performance of word embeddings in real-world NLP applications.
- **Methods**:
  - **Text Classification**: Use embeddings as input features for tasks like sentiment analysis or spam detection.
  - **Named Entity Recognition (NER)**: Assess embeddings in identifying entities (e.g., names, dates).
  - **Machine Translation:** Evaluate their role in generating accurate translations.
  - **Question Answering**: Test embeddings in retrieving correct A to Q.
- **Adv.**: Measures practical utility in real-world tasks.
- **Disadvantages**: Computationally expensive and task-specific.

Word Sense

- **Words Have Multiple Meanings**
  - Ambiguity is common, especially for frequent or historical words.
  - Example: "Pike" (fish, weapon, road).
- **Challenge with Word Embeddings**
  - Static embeddings (e.g., Word2Vec) assign one vector per word.
  - Do they capture all meanings, or create a mess?
- **Solution: Contextual Embeddings**
  - Models like BERT assign different vectors based on context.
  - Helps resolve ambiguity dynamically.

Word Sense, example

- **Word: "Bank" – 5 Different Meanings**
  1. **Financial Institution**
     1. *"I deposited money in the bank."*
     2. Meaning: A place to store and manage money.
  2. **Riverbank**
     1. *"He sat on the bank of the river."*
     2. Meaning: The land beside a river.
  3. **Action of Banking (Tilting)**
     1. *"The airplane banked sharply to the left."*
     2. Meaning: To tilt or turn while in motion.
  4. **Storage (e.g., Data Bank)**
     1. *"The company maintains a bank of customer information.*
     2. Meaning: A collection or repository.
  5. **Verb: Trust or Rely**
     1. *"You can bank on her to deliver the project."*
     2. Meaning: To trust or depend on someone.



Word Sense, Another Example (Pike)

- A sharp point or staff
- A type of elongated fish
- A railroad line or system
- A type of road
- The future (coming down the pike)
- A type of body position (as in diving)
- To kill or pierce with a pike
- To make one's way (pike along)
- In Australian English, pike means to pull out from doing
- something: I reckon he could have climbed that cliff, but he
- piked!

Global Context Solutions

- **Attention Mechanisms**:
  - Focus on relevant words across the text.
- **Contextual Embeddings**:
  - Models like BERT/GPT use full text for word meaning.
- **Topic Models**:
  - Capture document themes for better representation.

Improving Word with Global Context

- Idea: cluster word windows around words, retain with each word assigned multiple different clusters.