Learning Word Meaning with Word2Vec

Presenter: Huynh Cong Minh
Student ID: 2470890
Course: Mathematical Foundation For Computer Science (CO5263)
Assignment Topic: Word Embedding – Word2Vec

In this presentation, I will introduce **the intuition** and **mechanics behind Word2Vec** — **a powerful technique for learning vector** representations of words based on their contexts.

Contents

1 Introduction and Motivation

Why Do We Need Word Embeddings?

Machines do not "**understand**" words the way humans do

To a computer, "**intelligence**" is just a string of characters

We must find a numeric representation for words that captures their meaning

This is the first step in **Natural Language Processing (NLP)**

Limitations of One Hot Encoding

- Each word is assigned a unique sparse vector

id	color
1	red
2	blue
3	green
4	blue

One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

- Cosine similarity is either 0 or 1 => No notion of similarity

The Key Idea: Context Matters

"You shall know a word by the company it keeps"
— *Linguistic insight from the 1950s*

- Words used in similar **contexts** often have **similar meanings**
- **Example:**
  - "cat" and "dog" appear in contexts like "The ___ runs fast"
  - "bank" has different meanings depending on surrounding words

2

What is Word2Vec ?

Word2Vec: Learning Meaningful Word Vectors

- Word2Vec is a predictive model, not just encoding
- Learns dense, low-dimensional vectors
- Words in similar contexts are placed close together in vector space
- This was a major breakthrough that paved the way for modern NLP models

What is Word2Vec?

- Word2Vec is a neural network-based model that learns vector representations of words.
- It is built on the idea that words appearing in similar contexts share similar meanings.
- It transforms the distributional hypothesis into a learnable objective.
- Trained to either:
  - Predict a word from its surrounding words (CBOW – Continuous Bag of Words), or
  - Predict context words from a given word (Skip-Gram)

Word2Vec Has Two Variants

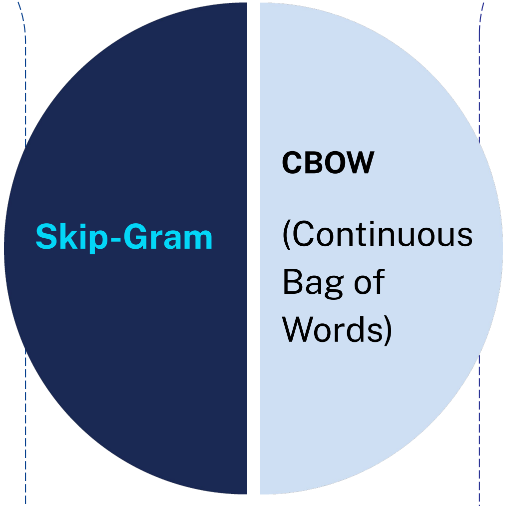
Word2Vec includes two model architectures:

Skip-Gram Model

Predicts context words from a center word.
Input: one center word
Output: multiple surrounding context words

"From one word, guess the neighbors."

Example:
Sentence: *"The quick brown fox jumps over the lazy dog"*
Center: "fox"
Prediction targets: "brown", "jumps"



CBOW (Continuous Bag of Words)

Predicts the center word from context words
Input: multiple context words
Output: one center word

"From the neighbors, guess the missing word."

Example:
Context: "brown", "jumps"
Prediction target: "fox"

3 Skip-gram Training

Notations Used in Skip-Gram Training

Symbol	Meaning
T	Total number of words (or center positions) in corpus
w_t	Center word at position t
w_{t+j}	Context word at distance j from center
m	Window size (number of context words on each side)
$\mathcal{V}$	Vocabulary set
$\boldsymbol{v}_w$	Input (center) embedding vector of word w
$\boldsymbol{u}_w$	Output (context) embedding vector of word w

Skip-Gram as a Probabilistic Problem

Problem Setup

Given a **center word**: w_t

Predict its **context words**:

$$w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m}$$

Skip-Gram as a Probability Task

We model the probability of observing **context words** given a **center word**:

$$P(\text{context} \mid w_t)$$

Assuming conditional independence between context words:

$$P(w_{t-m}, \ldots, w_{t+m} \mid w_t) = \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t)$$

Skip-Gram as a Probability Task

Sentence: The man loves his son

If we know the center word "loves", do context words like "man", "his", "son" affect each other?

Assumption in Skip-Gram:

They **do not**. Context words are **conditionally independent** given the center word.

So:

- If center word is "loves", then:
  - "his" does not affect "man"
  - "son" does not affect "his"

This leads to:

$$P(\text{man}, \text{his}, \text{son} \mid \text{loves}) = P(\text{man} \mid \text{loves}) \cdot P(\text{his} \mid \text{loves}) \cdot P(\text{son} \mid \text{loves})$$

Training Objective

The goal is to **maximize the likelihood** of context words given a center word.

Likelihood Objective

$$\prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t)$$

"We want to predict all surrounding context words for every center word in the corpus."

Convert to Log-Likelihood

$$\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t)$$

"Taking the log transforms product into sum — this simplifies optimization."

Modeling $P(w_o \mid w_c)$ with Softmax

To model the probability of a context word given a center word, we use the **softmax function**

$$P(w_o \mid w_c) = \frac{\exp(\boldsymbol{u}_{w_o}^\top \boldsymbol{v}_{w_c})}{\sum_{i \in \mathcal{V}} \exp(\boldsymbol{u}_i^\top \boldsymbol{v}_{w_c})}$$

- We want to assign **higher probability** to more relevant context words
- We compute **similarity** via dot product
- Softmax converts similarity scores into a **valid probability distribution**

Modeling $P(w_o \mid w_c)$ with Softmax

Raw scores (dot product)
man: 2.1
his: 3.5
son: 1.8
car: 0.5

Probabilities after softmax
man: 0.23
his: 0.58
son: 0.15
car: 0.04

The word "his" is **most similar** to the center word, so it gets the **highest probability**.

Skip-Gram Loss Function (Using Softmax)

We combine the log-likelihood and softmax to form the final loss:

$$\mathcal{L} = -\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log \left(\frac{\exp(\boldsymbol{u}_{w_{t+j}}^\top \boldsymbol{v}_{w_t})}{\sum_{i \in \mathcal{V}} \exp(\boldsymbol{u}_i^\top \boldsymbol{v}_{w_t})} \right)$$

Simplified Loss: One Center–Context Pair

To understand training in detail, we focus on one pair: **center word**, **context word**

Loss for one pair:

$$\mathcal{L} = -\log P(w_o \mid w_c)$$

With softmax:

$$P(w_o \mid w_c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{j \in \mathcal{V}} \exp(\boldsymbol{u}_j^\top \boldsymbol{v}_c)}$$

Gradient Derivation for Center Vector

To compute the gradient of the loss with respect to the center word vector, we go through the following steps:

Step 1: Loss for a single (center, context) pair

$$\mathcal{L} = -\log P(w_o \mid w_c) = -\log \left(\frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{j \in \mathcal{V}} \exp(\boldsymbol{u}_j^\top \boldsymbol{v}_c)} \right)$$

Gradient Derivation for Center Vector

Step 2: Expand the log expression

$$\mathcal{L} = -\boldsymbol{u}_o^\top \boldsymbol{v}_c + \log\left(\sum_{j\in\mathcal{V}} \exp(\boldsymbol{u}_j^\top \boldsymbol{v}_c)\right)$$

Gradient Derivation for Center Vector

Step 3: Compute gradient term-by-term

- First term

$$\frac{\partial}{\partial \boldsymbol{v}_c}\left(-\boldsymbol{u}_o^\top \boldsymbol{v}_c\right) = -\boldsymbol{u}_o$$

Gradient Derivation for Center Vector

Step 4: Compute gradient term-by-term

- Second term (using chain rule and softmax)

$$\frac{\partial}{\partial \boldsymbol{v}_c} \log\left(\sum_{j\in\mathcal{V}} \exp(\boldsymbol{u}_j^\top \boldsymbol{v}_c)\right) = \sum_{j\in\mathcal{V}} P(w_j \mid w_c)\boldsymbol{u}_j$$

Gradient Derivation for Center Vector

Final gradient expression

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_c} = \underbrace{\sum_j P(w_j \mid w_c)\boldsymbol{u}_j}_{\text{Expected (model's guess)}} - \underbrace{\boldsymbol{u}_o}_{\text{Ground truth}}$$

What Does the Gradient Mean?

Real-Life Analogy: *Guessing the Dish*

You're training a waiter to guess the customer's favorite dish based on a hint.

- The customer says: *"I want something hot, soupy, and perfect for a cold day."* *(*This is the **center word)**.
- The waiter guesses:

    - 🍲 *Chicken Soup* – 50%
    - 🍜 *Beef Pho* – 30%
    - 🍛 *Curry Rice* – 20%
- But the **correct dish** (ground truth) is: *Beef Pho*

What Does the Gradient Mean?

So What's the Problem?

- The model doesn't put **100% confidence** on the correct dish.
- It's **splitting the score** between many possible dishes.

Gradient Meaning

Expected output = Weighted average of all guesses

Ground truth = The actual dish vector

Gradient = Expected – Ground Truth

This shows **how far off** the model's average guess is from the truth.

> ➢ The gradient tells us how to **move the center vector .** So that next time, it points closer to the correct context vector (i.e. *Beef Pho*).

What Does the Gradient Do?

Once we compute the gradient of the loss with respect to the **center word vector**, we use it to update the vector using **Stochastic Gradient Descent (SGD)**.

$$\mathbf{v}_c \leftarrow \mathbf{v}_c - \eta \cdot \left(\sum_j P(w_j \mid w_c) \cdot \mathbf{u}_j - \mathbf{u}_o \right)$$

➢ **The center vector gets pulled toward the true context word.**

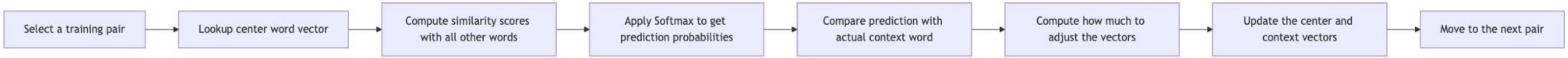
Note: We Also Update Output Vectors During Training

Even though we mainly focus on updating the center word vector, the output word vectors are also updated at each training step.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_j} = \begin{cases} (\hat{y}_j - 1) \cdot \mathbf{v}_c & \text{if } j = o \\ \hat{y}_j \cdot \mathbf{v}_c & \text{if } j \neq o \end{cases}$$

- **If the context vector (true context word), the output vector is pulled closer to the center vector**
- **Otherwise, the output vector is pushed away from the center vector**

Skip-Gram Training Summary (for 1 center-context pair)



4 | Demonstration

Demo Setup – Word2Vec on Toy Corpus

Model Configuration

- Model type: Skip-Gram
- Training method: Softmax
- Embedding dimension: 100
- Context window: Max size = 2
- Batch size: 128
- Epochs: 100
- Learning rate: 0.01

Toy Corpus Dataset Overview

- ~1,000 words
- Contains clear semantic clusters:

Theme	Examples
Family	father, mother, brother, sister
Animals	dog, cat, lion, tiger
Professions	teacher, student, doctor, nurse
Colors	red, blue, green, yellow
Vehicles	car, bus, train, airplane
Royalty	king, queen, prince, princess

To demonstrate how similar words (within the same theme) are embedded closer together in vector space — purely from training!

5 | Real-world Applications

Real-World Application: Word2Vec in Business

REAL - TIME

Word embeddings aren't just theory — they power **real-time** NLP systems in production.

LARGE - SCALE

YouNet Media, a leading social listening company in Vietnam, uses embeddings to analyze **large-scale** social media data.

VARIETY

Applications span from **crisis detection to sentiment analysis** in Vietnamese.

Crisis Detection with Word Embeddings

- Traditional keyword-based filters struggle with slang, misspellings, sarcasm.
- **Example user complaints:**
    - “Pepsi thiu vl”
    - “nặng mùi như toilet”
    - “vl thật sự, kinh”

➢ Word2Vec groups these semantically similar phrases by vector proximity — even if words don’t overlap.

➢ Enables real-time clustering and early warning alerts for brand crises

Sentiment Analysis Enhancement

- Vietnamese sentiment is expressed via:
  - Slang: “dở ẹc”, “rác vãi”
  - Emoji: 😤, 😭
  - Sarcasm: “không như kỳ vọng”

➢ Embeddings generalize sentiment meaning through learned context, not fixed rules.

➢ **Result:** improved accuracy and robustness in sentiment classification, even for unseen expressions.

Business Impact

Embedding-powered systems give brands a competitive edge:

- ➢ Faster crisis response (hours of lead time)
- ➢ Deeper understanding of customer voice
- ➢ More scalable and adaptive NLP pipelines

Word2Vec plays a critical role in unlocking insights from noisy, fast-changing user content.

6 Limitations

Limitations: Context Awareness & Vocabulary

- Lacks context awareness: Each word has one fixed vector, regardless of meaning in different contexts
    - ***Example*:** "bank" (river vs financial institution) → same vector
- Cannot handle OOV (Out-of-Vocabulary) words: Words not seen during training get no vector
    - ***Example*:** "covid" if it wasn’t in training data → no embedding

Limitation – Word Order

- Ignores word order beyond the context window
    - ***Example*****:** "man bites dog" vs "dog bites man" → same representation because context words are the same

Limitation – Computational Expense

- For each prediction, softmax must sum over entire vocabulary
  - Vocabulary can be 100k+ words → extremely slow training

$$\mathcal{O}(|V| \cdot d)$$

- This leads to high resource usage and long training time

Limitation – Word Structure & Larger Units

- Ignores word structure: Morphological variants treated as unrelated
    - ***Example*:** "run", "runs", "running" → separate embeddings
- No direct support for phrases or sentences: Must average word vectors, losing structure and meaning
    - ***Example*:** "United Nations" = average("United", "Nations")

7 Conclusion and QA

Conclusion

Word2Vec bridges language and math

- Converts words into dense, low-dimensional vectors
- Captures semantic similarity based on context

Two model architectures:

- Skip-Gram: Predicts context words from a center word
- CBOW: Predicts the center word from surrounding context

Training Objective:

- Maximize probability of context given center (or vice versa)
- Learn embeddings via softmax, cross-entropy loss, and SGD.

Limitations to consider:

- Ignores polysemy (e.g., “bank”)
  Fixed vector per word
- Cannot handle unseen words
- Computationally expensive

Word2Vec was a turning point in NLP — enabling machines to learn from **raw text** and opening the door for powerful **language understanding** models like **BERT** and **GPT.**

Q&A

THANK YOU!