**Word Embeddings and Word2Vec Models**

**1. Why Word Embeddings?**

Traditional text representation methods use **one-hot encoding**, where each word is a vector with one ‘1’ and rest ‘0’s. For example, in vocabulary [cat,dog,apple][cat, dog, apple][cat,dog,apple]:

* cat → [1, 0, 0]
* dog → [0, 1, 0]
* apple → [0, 0, 1]

**Problem:**

* These vectors are **sparse** (mostly zeros).
* They treat words as **completely independent**, ignoring meaning or similarity.
* No notion of semantic closeness. For example, "cat" and "dog" vectors are orthogonal, although both are animals.

**2. Word Embeddings: Dense Vector Representations**

**Goal:** Represent words as **dense vectors** in a lower-dimensional space where semantically similar words have vectors close to each other.

* Example: "king" and "queen" vectors should be close.
* Enables models to **generalize** better in NLP tasks.

**3. Word2Vec: Learning Word Embeddings from Context**

Word2Vec is a shallow neural network model that learns word embeddings by predicting words from their contexts or vice versa.

**4. CBOW Model (Continuous Bag of Words)**

**Intuition:**

* Given surrounding words (context), predict the middle word.
* Example sentence:  
  "The cat sits on the mat"
* To predict "sits," use context words like ["The", "cat", "on", "the"]

**How does CBOW work mathematically?**

* Input: One-hot vectors of context words.
* Embedding: Each word maps to a dense vector.
* Average the context word embeddings.
* Pass through output layer to predict target word probability distribution.
* Train to maximize probability of correct target word.

**Example:**

Sentence: "The dog barked loudly"

Suppose window size = 1, target word = "dog"

Context: ["The", "barked"]

* Vectorize "The" and "barked"
* Compute average of their embeddings
* Predict probability of target word "dog"

**Why averaging?**

* The order of context words is ignored (bag of words).
* Averaging simplifies and speeds training.

**5. Skip-gram Model**

**Intuition:**

* Given the center word, predict surrounding context words.
* Example sentence:  
  "The cat sat on the mat"
* Given "sat," predict ["The", "cat", "on", "the"]

**How does Skip-gram work mathematically?**

* Input: One-hot vector of the center word.
* Pass through embedding layer to get dense vector.
* Use this vector to predict each context word independently.
* Train by maximizing the likelihood of the actual context words.

**Example:**

Sentence: "Birds fly in the sky"

Target word: "fly"

Context words: ["Birds", "in"]

* Given "fly," predict "Birds" and "in"

**Sequence Models**

**Background: What is a Sequence Model?**

* Sequence models process input data arranged in a sequence (ordered elements).
* They model dependencies over time or position.
* Common in language (words in sentences), audio (speech frames), or other time series data.
* Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Transformers are popular architectures.

**1. 1-to-1 Sequence Model**

**Definition**

* Single input element → Single output element.
* No actual "sequence" transformation, but sometimes considered the simplest form of sequence modeling.

**Use Cases**

* Traditional classification (e.g., image or word classification).
* Predicting the label of a single word.

**Example**

* Classify whether a single word is positive or negative sentiment.

**Intuition**

* No temporal dependencies or sequence relationships considered.
* Treats input as independent instances.

**2. 1-to-Many Sequence Model**

**Definition**

* Single input vector → Output sequence.
* The model generates a sequence conditioned on one input.

**Use Cases**

* Image captioning: input is image embedding, output is a sentence.
* Music generation from a seed vector.

**Intuition**

* Input acts as a context or “seed” that conditions the entire output sequence.
* Model learns to decode or generate the sequence step-by-step, often with autoregressive dependencies.

**3. Many-to-1 Sequence Model**

**Definition**

* Input sequence → Single output.
* Model summarizes the entire sequence into one representation, then predicts output.

**Use Cases**

* Sentiment classification of a sentence.
* Predicting stock market trend from historical price sequences.

**Intuition**

* Model aggregates information from the entire input sequence.
* Often uses final hidden state of RNN/LSTM as summary representation.
* Ignores detailed output per time step.

**4. Many-to-Many Sequence Model**

**Definition**

* Input sequence → Output sequence.
* Can be **same length** or **different length** sequences.
* Examples:
  + Part-of-Speech (POS) tagging.
  + Named Entity Recognition (NER).
  + Time series forecasting at each time step.

**b) Many-to-Many (Different Length)**

* Examples:
  + Machine translation (input sentence → translated sentence).
  + Speech recognition (audio frames → text transcript).
* Typically modeled with encoder-decoder architectures:
  + Encoder processes input sequence into a fixed or variable representation.
  + Decoder generates output sequence step-by-step.

**Intuition**

* Model learns complex mapping between sequences.
* Decoding step generates output autoregressively, conditioning on previous outputs and input representation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Type** | **Input Size** | **Output Size** | **Key Characteristic** | **Common Applications** |
| 1-to-1 | Single vector | Single vector | No sequence modeling | Word classification |
| 1-to-Many | Single vector | Sequence | Generates sequence from one input | Image captioning, music gen |
| Many-to-1 | Sequence | Single vector | Summarizes input sequence | Sentiment analysis |
| Many-to-Many | Sequence | Sequence | Maps sequences → sequences | Translation, POS tagging |