7.2 Lesson: Word Embeddings

Word Embeddings

While BOW models offer a simple way to represent text as vectors, they suffer from several key limitations:

- They produce high-dimensional sparse vectors (most entries are 0), with 1 dimension per word in the vocabulary.
- They ignore word order entirely, so phrases like "dog bites man" and "man bites dog" look identical.
- They treat each word as completely independent, so “cat” and “feline” share no representation.
- They cannot generalize to phrases or words not seen during training, since vocabulary is fixed.

To move beyond these limitations, we need dense, low-dimensional vectors that encode semantic meaning and contextual usage of words — this is the motivation for word embeddings.

The key insight behind word embeddings is that we no longer represent words by their position in a vocabulary list, as in BOW models. Instead, each word is mapped to a point in a continuous, multi-dimensional vector space.

In this space, the axes do not correspond to specific words but to abstract semantic features — dimensions of meaning — that are not predefined or directly interpretable. Words that appear in similar contexts end up near each other in this space, and the directions between points often reflect meaningful relationships.

Figure 1's 3-D plots show how word embeddings capture analogies. For example, the Male-Female plot links

"king"→"queen" and "man"→"woman" to illustrate gender vectors, and so on.

The nearly parallel arrows in each plot highlight the consistent geometric offsets that encode these semantic relationships.

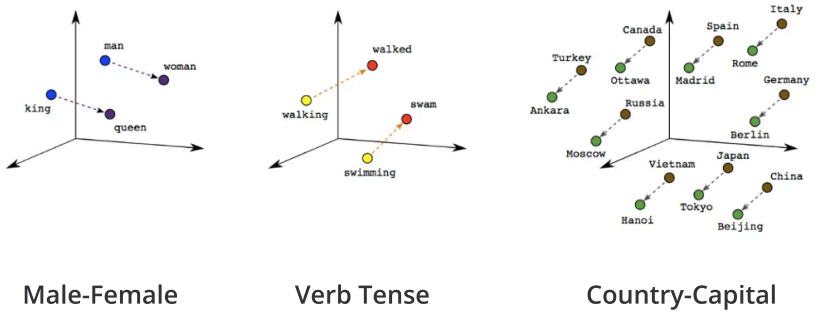

Figure 1: Linear Relationships between Words

Source: Image from developers.google.com in Anala, S. (2020, October 26). A guide to word embedding. (https://medium.com/data-science/a-guide-to-word-embeddings-8a23817ab60f) *Medium*.

There are 2 general classes of word embeddings:

Static embeddings inspired by deep learning language models

- **Word2vec** (skip-gram, continuous bag of words): **GloVe, fastText**
- Each word has a unique embedding computed from co-occurrence statistics
- Problems: Can not deal with polysemy (different meanings for same token, e.g., "lead the way" versus "lead bullets")

Contextual Embeddings

- **ELMo, BERT**
- Compute dynamic embeddings based on a word occurrence in its sentence
- Created by LLMs using transformers
- Can deal with polysemy

Word2Vec uses skip-grams, which are like N-grams with left and right context around a word. Figure 2 shows a skip-gram with a window size of 5.

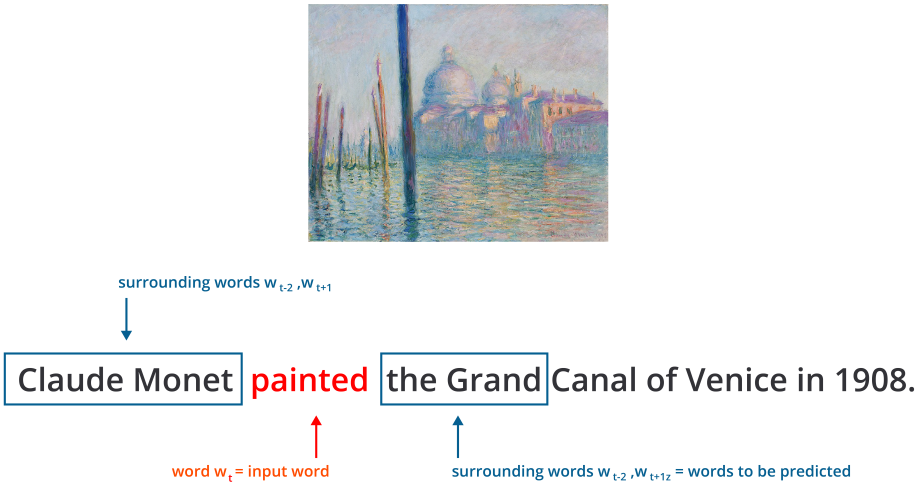

Figure 2: Skip-Grams

Image source: Monet, C. (1908). *Le Grand Canal* [Painting]. Museum of Fine Arts, Boston, MA, United States.

Two shallow networks are trained to predict the left and right context of the target word. Figure 3 shows what it would see during training.

These diagrams illustrate a skip-gram model for the word “painted:” the upper network feeds a one-hot “painted” vector into N hidden units and outputs a two-hot vector that lights up “Claude” and “Monet” as left-context words, while the lower network repeats the process to predict

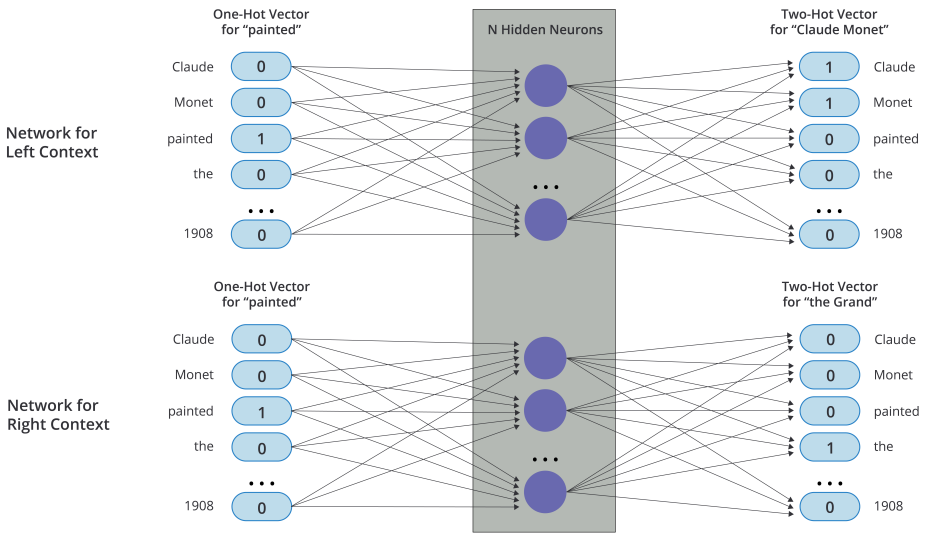

Figure 3: Training a Word2Vec Network

the right-context words “the” and “Grand.”

The networks are trained by sliding the context window through a large corpus of text. After training, when you input the target word, the network will predict possible contexts.

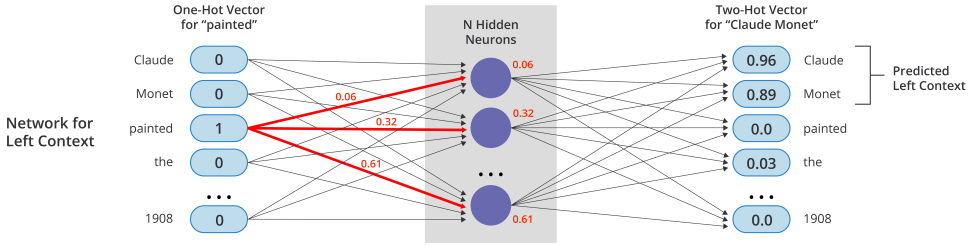

Figure 4: Predicting Left Context

Figure 4 diagrams a word-embedding network: a one-hot vector for “painted” feeds into a hidden layer of N nodes, and the softmax output gives high probabilities to “Claude” and “Monet” (0.96, 0.89) while all other vocabulary items are near 0, showing how the model predicts the left-context words preceding “painted.”

The embedding — the representation of the meaning of the target as defined by the contexts in which it appears — **is the vector of activations produced by the hidden layer when the word is input as a one-hot vector.**

In Figure 5, the diagram shows a one-hot word vector multiplied by an

embedding weight matrix: the active row (third in a 6-word vocabulary) is selected, yielding the 3 numbers 0.06, 0.32, 0.61 as the resulting word embedding.

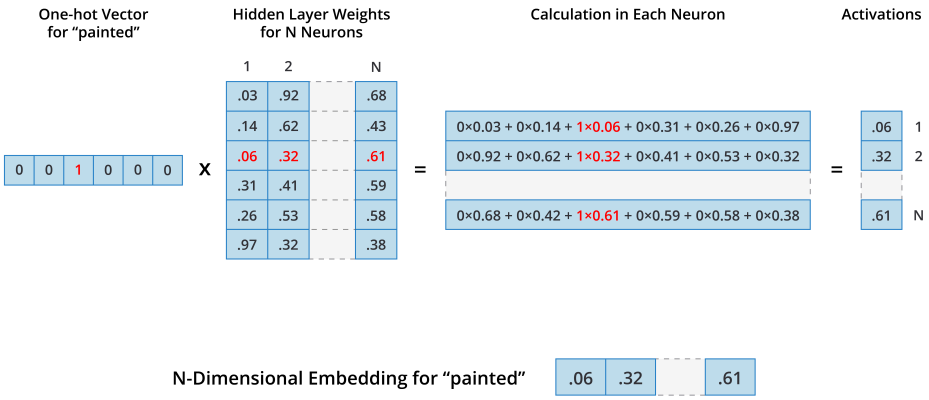

Figure 5: Extracting the Word Embedding

A key point is that **the number N of hidden neurons** (and hence **the number of dimensions** in the vector space) **is arbitrary**; the larger the embedding, the richer the semantic content of the embedding is. Also note that there is no obvious meaning to assign to the individual dimensions — a common problem in deep learning!

The result of this framework is that similar words are near each other in the vector space and dissimilar words are far apart.

Figure 6's scatterplot of word embeddings shows distinct clusters: green points for positive words (*amazing, wonderful, good*), red points for negative words (*dislike, bad, worse*), yellow points for synonyms *car* and *automobile*, and blue points for vegetables (*spinach, ongchoi, chard*). Their positions reveal how

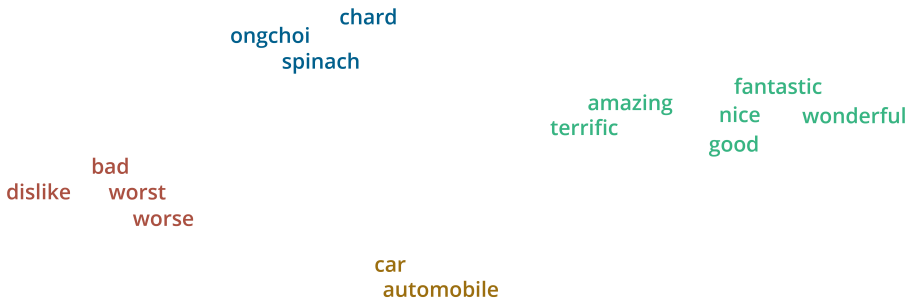

Figure 6: Words in Embedding Space

the model groups semantically similar words.

They also allow simple forms of analogical reasoning.

The small parallelogram, in Figure 7, shows the 2 relation vectors (apple→tree and grape→vine) as roughly parallel, highlighting how analogies emerge from vector geometry.

To solve: "Apple is to tree as grape is to _____?"

- **Use vector arithmetic**

  X = tree - apple + grape

- **Search for the closest word (using cosine sim):**

  argmin_x (cosine_sim(tree -apple + grape, X) → vine

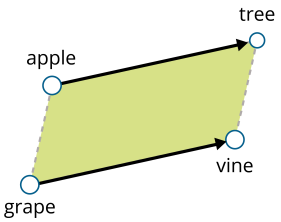

Figure 7: Word Analogies in Embedding Space

The overall result is that meaningful relationships between words are embedded in the vector space:

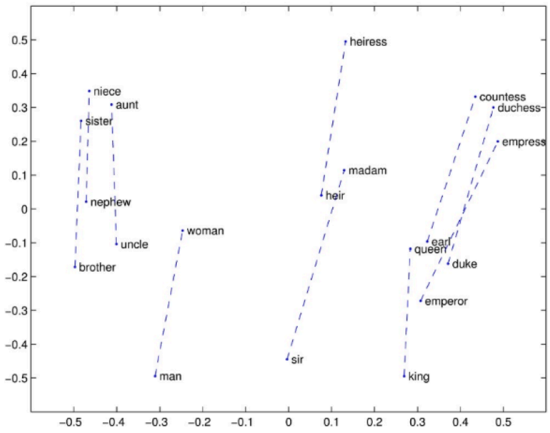

Figure 8: Relations in Embedding Space

Source: Jurafsky, D., &Martin, J. H. (2025). Speech and language processing (https://web.stanford.edu/~jurafsky/slp3/6.pdf), p.127.

Embeddings have broad applications in linguistics — for example, providing

useful insights into historical trends in word use:

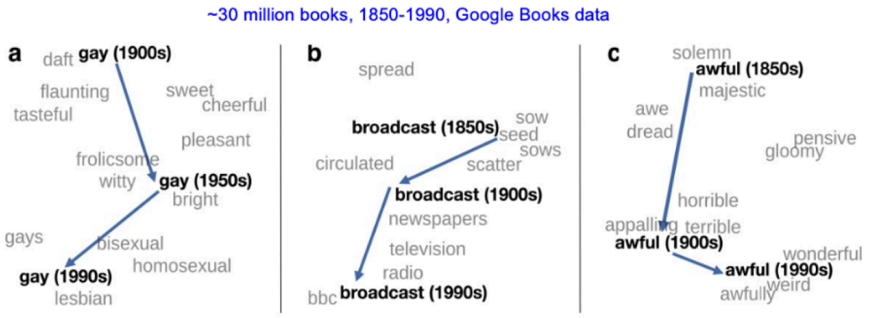

Figure 9: Semantic Change in Embedding Space

Source: Jurafsky, D., & Martin, J. H. (2025). Speech and language processing (https://web.stanford.edu/~jurafsky/slp3/6.pdf): An introduction to natural language processing, computational linguistics, and speech recognition (3rd ed. draft) (https://web.stanford.edu/~jurafsky/slp3/6.pdf), p.128.

Alternative Embedding Strategies

Word2Vec and **GloVe** are classic methods for learning word embeddings—while Word2Vec predicts nearby words, GloVe captures global co-occurrence statistics. **FastText** extends this by using subword units, improving performance on rare or misspelled words. **ELMo** introduced context-sensitive embeddings using deep RNNs.

Since word embeddings are created from shallow networks, they are essentially pre-trained models (as per Week 5) and can themselves be embedded in a deep network and be retrained as they process text for a particular application.

Finally, modern approaches based on the transformer architecture generate rich, contextualized embeddings for words, sentences, or entire documents, producing the results we see in modern LLMs.