Words to Vectors

A Hands-On Introduction to Text Tokenisation & Embeddings

(Partially generated by Claude AI)

What this worksheet covers:

- Why computers need numbers, not words
- One-hot encoding
- The limitations of one-hot vectors
- Dense word embeddings
- Similarity between word vectors (dot product, cosine similarity)
- How embeddings capture meaning

Motivation: Why Vectors?

Computers cannot process raw text directly. Before any learning can happen, each word must be converted into a list of numbers — a **vector**. This process is called *representation learning* or *embedding*, and it is the very first stage of every modern Large Language Model (LLM).

Definition: A **vector** is simply an ordered list of numbers, e.g. $\mathbf{v} = [0.2,\ -1.4,\ 0.8]$. In our context, each word maps to one such list.

Stage 1: Building a Vocabulary

The first step is to break up the given text into words, or sometimes we also break up words into sub-words. Each unique item that comes from this process is called a token, and we call the process "tokenization".

Once we have all of our tokens from the text, assign each one a unique integer index.

Example 1: A Tiny Vocabulary

Consider the sentence: *"the cat sat on the mat"*

After collecting unique words and sorting alphabetically:

Word	Index
cat	0
mat	1
on	2
sat	3
the	4

We have 5 tokens, so if we name our vocabulary V, then the magnitude of V is 5.

Stage 2: One-Hot Encoding

Definition: We'll be using a vector that has zeros everywhere except a single "1" in position i. This vector is called "the i^{th} standard basis vector. Using notation, we would write this vector:

$$\mathbf{e}_i = [0, 0, 0, 0, \ldots, 0, 1, 0, 0, \ldots, 0]$$

Getting back to our words and embedding, the simplest way to convert a word to a vector (length of the vector is the magnitude of the vocabulary) is to put a "1" in its corresponding spot in the vocabulary, and a zero everywhere else. This is called **one-hot encoding**, meaning the new vector has only one "1" and the remaining elements are zero. Here are examples from continuing our previous vocabulary.

Example 2: One-Hot Vectors (magnitude of V is 5)

Using the vocabulary from Example 1:

$$\begin{aligned}
\mathbf{v}_{\text{cat}} &= [1,\ 0,\ 0,\ 0,\ 0] \\
\mathbf{v}_{\text{mat}} &= [0,\ 1,\ 0,\ 0,\ 0] \\
\mathbf{v}_{\text{on}} &= [0,\ 0,\ 1,\ 0,\ 0] \\
\mathbf{v}_{\text{sat}} &= [0,\ 0,\ 0,\ 1,\ 0] \\
\mathbf{v}_{\text{the}} &= [0,\ 0,\ 0,\ 0,\ 1]
\end{aligned}$$

Was this a "good" embedding? When we're converting words to a numerical representation, we would like to be able to measure how close two words are to each other. This means we'll need a way of measuring how similar two vectors are to each other.

To do this, we'll need to define some arithmetical operations on two vectors.

Definition: Let vectors $\mathbf{a}$ and $\mathbf{b}$ each have the same number of elements (let's say they both have k numbers). Then we can define **the dot product** between the two vectors:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 + \cdots a_k b_k$$

For example, if $\mathbf{a} = [1, 2, 4, 2]$ and $\mathbf{b} = [3, -1, 0, 1]$, then

$$\mathbf{a} \cdot \mathbf{b} = 1 \cdot 3 + 2 \cdot (-1) + 4 \cdot (0) + 2 \cdot (1) = 3 - 2 + 2 = 3$$

Example for you to try: If $\mathbf{a} = [1, 2, 3]$ and $\mathbf{b} = [3, 0, -1]$, find $\mathbf{a} \cdot \mathbf{b}$.

Definition: Given a vector $\mathbf{a}$, the **length** of the vector is defined to be:

$$\|\mathbf{a}\| = \sqrt{\mathbf{a} \cdot \mathbf{a}} = \sqrt{a_1^2 + a_2^2 + \cdots a_k^2}$$

(this is reminiscent of the Pythagorean Theorem) For example, let $\mathbf{a} = [1, 2, 4, 2]$. Then

$$\|\mathbf{a}\| = \sqrt{1^2 + 2^2 + 4^2 + 2^2} = \sqrt{1 + 4 + 16 + 4} = \sqrt{25} = 5$$

Examples to try: Let $\mathbf{a} = 0, 1, 2, 1]$ and $\mathbf{b} = [1, 2, 0, 1]$. Compute the following expressions:

1. $\mathbf{a} \cdot \mathbf{b}$
2. $\|\mathbf{a}\|$
3. $\mathbf{b} \cdot \mathbf{b}$

(Solutions: $3, \sqrt{6}, 6$)

The (Cosine) Similarity Between Two Vectors It can be show that the quantity computed by the following is always a number between -1 and 1:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

This would mean that if this number is close to the number 1, then the two vectors are **very similar**. A value of -1 means that they are negatively similar, or **opposite**. A value of 0 means that the two vectors are unrelated to each other.

Example: What is the cosine similarity of $\mathbf{a} = [0, 1, 2, 1]$ and $\mathbf{b} = [0, 2, 4, 2]$?

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{12}{\sqrt{6}\sqrt{24}} = \frac{12}{\sqrt{144}} = \frac{12}{12} = 1$$

These vectors are perfectly similar (notice that $\mathbf{b}$ was found by multiplying every element of $\mathbf{a}$ by 2).

Example: What is the cosine similarity of $\mathbf{a} = [0, 1, 2, 1]$ and $\mathbf{b} = [1, 2, 0, 1]$?

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{3}{\sqrt{6}\sqrt{6}} = \frac{3}{6} = \frac{1}{2}$$

These vectors are slighly related to each other.

Example: What is the cosine similarity between $\mathbf{a} = [0, 1, 2]$ and $\mathbf{b} = [1, -2, 1]$?
Since the dot product is zero, the cosine similarity is 0.

Now let's go back to Example 2 (you might check our computation there again).

Properties of One-Hot Vectors

- Any two distinct words have dot product 0; the model sees them as completely unrelated.
- Almost all entries are zero. For a realistic vocabulary of 50,000 words, each vector has 49,999 zeros. (This is also called "sparse").
- No semantic information: "cat" and "kitten" are just as different as "cat" and "democracy".

So one-hot encodings of our words is not coding contextual meaning. We'll need to try a different kind of embedding.

Stage 3: Dense Embeddings

To overcome the limitations of one-hot encoding, LLMs learn **dense embedding vectors**: short, real-valued vectors (typically 64–4096 entries in each vector) where **similar** words end up close together.

Suppose we have k words in our vocabulary, and we want each embedded vector to have n numbers. Then we can write the embedding array of numbers (also called a matrix) using k rows and n columns. We would say that the matrix is $k \times n$ (number of rows by number of columns).

Here's an example of what might be a better embedding matrix than the one we had before:

Example 3.1 A Learned Embedding Table

Suppose our model has learned the following 5×3 embedding matrix A.

$$\mathbf{E} = \begin{array}{c|ccc} & \text{col}_1 & \text{col}_2 & \text{col}_3 \\ \hline \text{cat} & 0.9 & 0.1 & -0.2 \\ \text{mat} & 0.1 & 0.8 & 0.3 \\ \text{on} & -0.4 & 0.2 & 0.9 \\ \text{sat} & 0.2 & 0.6 & 0.5 \\ \text{the} & -0.1 & 0.0 & 0.1 \end{array}$$

So the embedding for "cat" is simply the first row: $\mathbf{a}_{\text{cat}} = [0.9,\ 0.1,\ -0.2]$.

(In practice the number of columns is $\approx$ 768–4096*; we use* 3 *for legibility.)*

Stage 4 — Measuring Similarity

Once words are vectors, we can quantify how "similar" two words are.

Example 4.1: Cosine Similarity

Using $\mathbf{e}_{\text{cat}} = [0.9, 0.1, -0.2]$ and $\mathbf{e}_{\text{sat}} = [0.2, 0.6, 0.5]$:

Step 1 — Dot product:

$$\mathbf{e}_{\text{cat}} \cdot \mathbf{e}_{\text{sat}} = (0.9)(0.2) + (0.1)(0.6) + (-0.2)(0.5) = 0.18 + 0.06 - 0.10 = 0.14$$

Step 2 — Magnitudes:

$$\|\mathbf{e}_{\text{cat}}\| = \sqrt{0.9^2 + 0.1^2 + (-0.2)^2} = \sqrt{0.81 + 0.01 + 0.04} = \sqrt{0.86} \approx 0.927$$

$$\|\mathbf{e}_{\text{sat}}\| = \sqrt{0.04 + 0.36 + 0.25} = \sqrt{0.65} \approx 0.806$$

Step 3 — Cosine similarity:

$$\cos(\mathbf{e}_{\text{cat}}, \mathbf{e}_{\text{sat}}) = \frac{0.14}{0.927 \times 0.806} \approx \frac{0.14}{0.747} \approx 0.187$$

A modest positive similarity — the words co-occur in the same short sentence, but are not semantically close.

Worksheet

Words to Vectors — Practice Problems

Use the following corpus and embedding table for **all questions** unless stated otherwise.

Corpus sentence: *"dog runs fast and cat runs slow"*

Vocabulary (alphabetical order):

Word	Index
and	0
cat	1
dog	2
fast	3
runs	4
slow	5

Embedding matrix E ($d = 3$):

$$\mathbf{E} = \begin{pmatrix} 0.1 & -0.3 & 0.0 \\ 0.8 & 0.5 & -0.1 \\ 0.9 & 0.4 & 0.0 \\ 0.0 & 0.2 & 0.7 \\ 0.3 & 0.6 & 0.4 \\ -0.1 & 0.1 & -0.8 \end{pmatrix} \begin{matrix} \leftarrow \text{and} \\ \leftarrow \text{cat} \\ \leftarrow \text{dog} \\ \leftarrow \text{fast} \\ \leftarrow \text{runs} \\ \leftarrow \text{slow} \end{matrix}$$

Question 1 — Vocabulary & One-Hot Encoding

(a) How many unique tokens does this vocabulary contain?

(b) Write the one-hot vector for the word **"dog"**.

(c) Write the one-hot vector for the word **"slow"**.

(d) What is the dot product of the one-hot vectors for "dog" and "slow"? What does this tell you?

Question 2 — Looking Up Embeddings

(a) Using the embedding matrix $\mathbf{E}$, write down the embedding vector $\mathbf{e}_{\text{dog}}$ for the word **"dog"**.

(b) What are the dimensions of the full embedding matrix $\mathbf{E}$? State them as *rows* $\times$ *columns*.

Question 3 — Dot Product & Cosine Similarity

(a) Compute the **dot product** of $\mathbf{e}_{\text{cat}}$ and $\mathbf{e}_{\text{dog}}$.

(b) Compute the **magnitude** $\|\mathbf{e}_{\text{cat}}\|$ and $\|\mathbf{e}_{\text{dog}}\|$.

(c) Compute the **cosine similarity** between "cat" and "dog". Round to three decimal places.

(d) Now compute the cosine similarity between "cat" and "slow". Which pair is more similar according to the embedding? Does this match your intuition?

Question 4 — Conceptual Questions

(a) A vocabulary has $V = 30{,}000$ words. How many numbers are stored in a single one-hot vector? How many of them are non-zero?

(b) If the embedding dimension n is 512, how many numbers are stored in the complete embedding matrix $\mathbf{E}$ for the same 30,000-word vocabulary?

(c) In one or two sentences, explain why one-hot encoding is a poor representation for semantic tasks (e.g. sentiment analysis).

(d) Suppose a well-trained embedding satisfies

$$\mathbf{e}_{\text{king}} - \mathbf{e}_{\text{man}} + \mathbf{e}_{\text{woman}} \approx \mathbf{e}_{\text{queen}}.$$

What does this tell us about the *geometry* of a good embedding space? Hint: The equation shows that semantic relationships are coded as directions in the embedding space. What does subtracting "man" from "king" leave you with?