The Map of Meaning

A Deep Dive into Vector Semantics and Word Embeddings

A Comprehensive and Intuitive Guide

Contents

1 The Quest for Meaning: Lexical Semantics **3**
- 1.1 The Problem with Meaning . . . 3
- 1.2 Lemmas and Senses . . . 3
- 1.3 Why Does It Matter? . . . 3

2 The Fabric of Language: Lexical Relations **3**
- 2.1 Synonymy: The Same Meaning? . . . 4
- 2.2 Similarity vs. Relatedness: A Crucial Distinction . . . 4
- 2.3 Other Key Relations . . . 4
- 2.4 Connotation: The Emotional Color . . . 4

3 The Cornerstone: The Distributional Hypothesis **5**
- 3.1 The Intuition: Learning a New Word . . . 5

4 The Geometric Revolution: Vector Semantics **5**
- 4.1 Meaning as Coordinates . . . 5
- 4.2 Why Vectors? . . . 6

5 Building the Vectors: Sparse Approaches **6**
- 5.1 The Co-occurrence Matrix . . . 6
  - 5.1.1 Term-Document Matrix . . . 6
  - 5.1.2 Term-Term Matrix (Word-Word Matrix) . . . 7
- 5.2 Characteristics of Sparse Vectors . . . 7

6 The Geometry of Meaning: Measuring Similarity **7**
- 6.1 The Dot Product and Its Flaw . . . 7
- 6.2 The Solution: Cosine Similarity . . . 7

7 Refining Sparse Vectors: TF-IDF **8**
- 7.1 The Paradox of Frequency . . . 8
- 7.2 Term Frequency (TF) . . . 8
- 7.3 Inverse Document Frequency (IDF) . . . 9
- 7.4 The Combined TF-IDF Weight . . . 9

8 The Shift to Dense Vectors: The Embedding Era **9**
- 8.1 Characteristics of Dense Vectors . . . 10
- 8.2 The Big Idea: Prediction Instead of Counting . . . 10
- 8.3 Word2Vec: The Prediction Task . . . 10

9 The Mechanics: How Embeddings Work **10**
9.1 The Baseline: One-Hot Encoding 11
9.2 The Embedding Matrix 11
9.3 The "Lookup Trick" 11

10 Conclusion: Language as Geometry **11**

1 The Quest for Meaning: Lexical Semantics

In Natural Language Processing (NLP), we often treat words as mere tokens or indices in a vocabulary. For tasks like basic text classification or N-gram modeling, this might suffice. But if we want machines to truly understand language—to answer questions, detect nuance, or engage in dialogue—we must grapple with **Lexical Semantics**: the linguistic study of word meaning.

1.1 The Problem with Meaning

What does a word mean? Traditional approaches often fall short.

- **The Index Approach:** "dog" is word #5342. This tells us nothing about what a dog is.
- **The Logical Approach:** The meaning of "dog" is the concept DOG. As Barbara Partee famously joked, "What's the meaning of life? LIFE." It's technically correct but entirely unhelpful computationally.

1.2 Lemmas and Senses

The relationship between words and concepts is complex.

- **Lemma:** The base or dictionary form of a word (e.g., 'run').
- **Sense:** The specific meaning associated with a lemma in a given context.

Words are often **polysemous** (having multiple senses). Consider the lemma "mouse":

1. A small rodent (Sense 1).
2. A hand-operated device that controls a cursor (Sense 2).

A robust model of meaning must be able to differentiate these senses and understand how they relate to other words (e.g., Sense 1 relates to "cheese"; Sense 2 relates to "click").

1.3 Why Does It Matter?

Understanding word meaning is crucial for practical applications.

In Practice: Question Answering and Plagiarism Detection

Question Answering:
Q: "How *tall* is Mt. Everest?"
A: "The official *height* of Mount Everest is 29029 feet."
To connect the question to the answer, the system must recognize that "tall" is semantically similar to "height".
Plagiarism Detection:
Systems must identify similarity beyond simple keyword matching, detecting paraphrased or reworded content by analyzing the underlying semantics.

2 The Fabric of Language: Lexical Relations

Words do not exist in isolation. Their meanings are defined by their relationships with other words. Understanding these relationships is key to building computational models of meaning.

2.1 Synonymy: The Same Meaning?

Synonyms are words with the same (or nearly the same) meaning. They should be substitutable in a sentence without changing the truth of the proposition.

- *Examples:* couch/sofa, car/automobile.

However, perfect synonymy is rare. Context and nuance often matter:

- We say "my big sister," but rarely "my large sister."
- A chemistry paper uses "H2O," while a surfing guide uses "water."

2.2 Similarity vs. Relatedness: A Crucial Distinction

It is vital to distinguish between similarity and relatedness.

Important Distinction: Similarity vs. Relatedness

Similarity: Words sharing core elements of meaning. They often belong to the same class.

- *Examples:* Cat and Dog (both are pets, mammals). Car and Bicycle (both are modes transport).

Relatedness (Association): Words frequently appearing together or belonging to the same **Semantic Field** or Frame, but not necessarily meaning the same thing.

- *Examples:* Coffee and Cup. Doctor and Scalpel.

"Coffee" is similar to "Tea", but "Coffee" is related to "Cup".

2.3 Other Key Relations

- **Antonymy:** Opposites concerning one specific feature (e.g., hot/cold, rise/fall).
- **Hypernymy:** Hierarchical "Is-A" relationships (e.g., Animal > Dog > Poodle).
- **Meronymy:** Part-Whole relationships (e.g., Wheel is part of a Car).

2.4 Connotation: The Emotional Color

Beyond the literal meaning (**denotation**), words carry emotional, cultural, or social baggage (**connotation**). This is the foundation of sentiment analysis.

Word	Denotation (Literal)	Connotation (Implied)
Home	A place where one lives	Warmth, safety, family
Cheap	Low in cost	Negative: Poor quality, stingy
Inexpensive	Low in cost	Positive: Good value
Slim	Having little fat	Positive: Fit, elegant
Skinny	Having little fat	Negative: Unhealthy, unattractive

Research (Osgood et al., 1957) suggests that affective meaning varies along three primary dimensions:

1. **Valence:** Pleasantness (Happy vs. Sad).
2. **Arousal:** Intensity of emotion (Excited vs. Calm).
3. **Dominance:** Degree of control (Powerful vs. Weak).

3 The Cornerstone: The Distributional Hypothesis

How can a computer learn these complex relationships—similarity, connotation, antonymy—from raw text alone? The answer lies in the **Distributional Hypothesis**, perhaps the most important idea in modern NLP.

The Big Idea: "You shall know a word by the company it keeps." (J.R. Firth, 1957)

Words that occur in similar contexts tend to have similar meanings.
The meaning of a word is defined by its environment—the distribution of words around it.

3.1 The Intuition: Learning a New Word

Imagine you encounter a word you don't know. How do you figure out what it means? You look at how it's used.

In Practice: The "Ongchoi" Example

Suppose we see:

- *Ongchoi* is delicious *sautéed* with *garlic*.
- *Ongchoi leaves* with salty sauces.

And we already know:

- *Spinach sautéed* with *garlic* over rice.
- *Collard greens* and other salty leafy greens.

By comparing the contexts ("sautéed", "garlic", "leaves"), we infer that Ongchoi is a type of leafy green vegetable, similar to spinach or collard greens.

This hypothesis gives us a concrete strategy for computation: measure the semantic similarity of words by measuring the similarity of the contexts in which they appear.

4 The Geometric Revolution: Vector Semantics

If we accept the Distributional Hypothesis, how do we implement it? We use **Vector Semantics**. We transform the abstract concept of "context" into a concrete mathematical object: a vector.

4.1 Meaning as Coordinates

We visualize meaning as a point in a multidimensional semantic space.

- Each word is represented by a vector of numbers.
- Words with similar contexts will have similar vectors.
- Therefore, similar words are geometrically close together in this space.

These vectors are called **Embeddings**, because the word's meaning is embedded into the vector space. This is the standard representation of meaning in all modern NLP.

The Intuition: The Map of Language

Imagine creating a map where every word is a city. If "dog" and "cat" often appear in similar sentences (near "pet", "feed", "walk"), they are placed close together on the map. "Computer" would be far away, near "data" and "process". The distance between the cities represents the similarity of the words.

4.2 Why Vectors?

Why move away from simple word counts to vectors?

1. **Generalization:** In traditional methods (like Bag-of-Words), "car" and "automobile" are completely distinct features. If a model learns that "car" is positive in a review, it learns nothing about "automobile". With vectors, v_{car} is very similar to $v_{automobile}$. The model can generalize its learning from one to the other.
2. **Dimensionality Reduction:** Vectors allow us to compress vast amounts of contextual information into a manageable format.
3. **Mathematical Operations:** Vectors allow us to use the tools of linear algebra (like distance and angles) to measure and manipulate meaning.

5 Building the Vectors: Sparse Approaches

There are two main families of word embeddings: Sparse (Frequency-based) and Dense (Prediction-based). We begin with the more intuitive, count-based **Sparse Vectors**.

5.1 The Co-occurrence Matrix

The first step is to capture the context distribution explicitly. We do this by building a Co-occurrence Matrix. This matrix counts how often words appear together.

5.1.1 Term-Document Matrix

Here, the dimensions represent documents. A word's meaning is defined by the documents it appears in. This is foundational to Information Retrieval.

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4

- The vector for "battle" (the row) is $[1, 0, 7, 13]$.
- The vector for "As You Like It" (the column) is $[1, 114, 36, ...]$.

We can see that "Fool" is the kind of word that occurs in comedies; "Battle" occurs in historical plays/tragedies.

5.1.2 Term-Term Matrix (Word-Word Matrix)

More commonly for lexical semantics, the dimensions represent other words in the vocabulary. A word's meaning is defined by the words that appear nearby (within a fixed window, e.g., ±5 words).

	computer	data	result	pie	sugar	...
cherry	2	8	9	442	25	...
strawberry	0	0	1	60	19	...
digital	1670	1683	85	5	4	...
information	3325	3982	378	5	13	...

- "Cherry" and "Strawberry" have similar vectors (high counts near "pie" and "sugar").
- "Digital" and "Information" have similar vectors (high counts near "computer" and "data").

5.2 Characteristics of Sparse Vectors

These count-based vectors are:

- **High-Dimensional:** The length of the vector is the size of the vocabulary (e.g., 50,000).
- **Sparse:** Most entries are zero, as most words do not co-occur with most other words.

6 The Geometry of Meaning: Measuring Similarity

Now that we have vectors, how do we quantify the similarity between them? We need a metric that aligns with the intuition that similar vectors should be "close".

6.1 The Dot Product and Its Flaw

The simplest measure of similarity is the dot product.

$$\text{similarity}(v, w) = v \cdot w = \sum_{i=1}^{N} v_i w_i$$

The dot product is high when two vectors have large values in the same dimensions.

The Flaw: The dot product favors long vectors. The length of a vector is $|v| = \sqrt{\sum_{i=1}^{N} v_i^2}$. In count-based matrices, frequent words (like "the", "of") have long vectors because they occur frequently. The dot product conflates *frequency* with *similarity*.

6.2 The Solution: Cosine Similarity

We need a metric that measures the alignment (the angle) between the vectors, independent of their length. This is achieved by normalizing the dot product by the lengths of the two vectors. This metric is the **Cosine Similarity**.

The Math: Cosine Similarity

The cosine similarity is derived from the geometric definition of the dot product:

$$v \cdot w = |v||w| \cos\theta$$

Rearranging to solve for the cosine of the angle θ between the vectors:

$$\cos(v, w) = \frac{v \cdot w}{|v||w|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

The Intuition: The Angle Matters, Not the Distance

Imagine two vectors starting at the origin.

- If they point in the exact same direction, the angle is $0°$, and $\cos(0°) = 1$ (Maximum Similarity).
- If they are perpendicular (orthogonal), the angle is $90°$, and $\cos(90°) = 0$ (No Similarity).
- If they point in opposite directions, the angle is $180°$, and $\cos(180°) = -1$.

(Note: For raw frequency counts, vectors are non-negative, so the cosine ranges from 0 to 1).

7 Refining Sparse Vectors: TF-IDF

We have established that co-occurrence counts are the basis for sparse vectors and cosine similarity is the metric. However, raw frequency counts are problematic.

7.1 The Paradox of Frequency

- **Constraint 1 (Information):** High frequency is often informative. If "sugar" appears near "apricot" often, that's a strong signal.
- **Constraint 2 (Noise):** Extremely high frequency can be noise. Words like "the", "it", or "they" appear everywhere and tell us very little about the specific meaning of the words they neighbor.

How do we balance this? We need a weighting scheme that emphasizes informative words and downplays common words. The most famous scheme in Information Retrieval (especially for Term-Document matrices) is **TF-IDF** (Term Frequency-Inverse Document Frequency).

7.2 Term Frequency (TF)

Question: How important is this word within this specific document?

Instead of using the raw count $count(t, d)$ (term t in document d), we often apply a logarithmic transformation (squashing). This dampens the effect of very high frequencies.

The Math: Log Term Frequency

$$tf_{t,d} = \log_{10}(count(t, d) + 1)$$

Example:

- "the" (1000 occurrences): $\log_{10}(1001) \approx 3.0$
- "data" (10 occurrences): $\log_{10}(11) \approx 1.04$

The difference is now much smaller, reducing the bias towards "the".

7.3 Inverse Document Frequency (IDF)

Question: How surprising (rare) is this word across the entire collection?

IDF measures how much information a word provides. Words that occur in very few documents are more distinctive and thus more valuable.

- N: Total number of documents in the collection.
- df_t: Document Frequency of term t (the number of documents containing t).

The Math: Inverse Document Frequency

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

Example (N=100 documents):

- Word A appears in 5 documents ($df = 5$): $idf = \log_{10}(100/5) = \log_{10}(20) \approx 1.30$ (High weight).
- Word B appears in 100 documents ($df = 100$): $idf = \log_{10}(100/100) = \log_{10}(1) = 0$ (Zero weight).

Words like "the" or "good" that appear in every document have an IDF of 0.

7.4 The Combined TF-IDF Weight

The final weight combines these two factors:

$$w_{t,d} = tf_{t,d} \times idf_t$$

TF-IDF successfully highlights words that are frequent locally (high TF) but rare globally (high IDF). These are the words that truly define the topic of a document.

Important Distinction: Applying Weighting to Word-Word Matrices

While TF-IDF is traditionally used for Term-Document matrices, the principle of emphasizing informative co-occurrences is also used for Word-Word matrices, often using a related metric called **Pointwise Mutual Information (PMI)**.

8 The Shift to Dense Vectors: The Embedding Era

TF-IDF weighted sparse vectors were the standard for many years. However, they have limitations:

1. **Sparsity and Size:** They are very large and mostly empty, making them computationally cumbersome for machine learning models (too many weights to tune).
2. **Failure to Generalize Well:** As discussed earlier, they don't capture synonymy effectively because each dimension is distinct (e.g., "car" dimension vs. "automobile" dimension).

This led to the development of **Dense Vectors** (the modern concept of "Embeddings").

8.1 Characteristics of Dense Vectors

- **Low-Dimensional:** Typically 50-300 dimensions (instead of 50,000).
- **Dense:** Most elements are non-zero real numbers.
- **Learned Representation:** The values are learned through a training process, rather than counted explicitly.

Type	Example Vector (Conceptual)
Sparse (TF-IDF)	[0, 0, 5.2, 0, ..., 0, 1.1, 0, ...] (Length 50,000)
Dense (Word2Vec)	[0.25, -0.11, 0.67, ..., 0.03] (Length 300)

8.2 The Big Idea: Prediction Instead of Counting

How do we generate these dense vectors? The revolutionary approach, popularized by models like Word2Vec (Mikolov et al., 2013), is to shift from counting co-occurrences to *predicting* them.

The Big Idea: Self-Supervised Learning

We train a classifier on a "fake" prediction task. We don't actually care about the task itself, but we use the internal weights learned by the classifier as the word embeddings. The "supervision" comes from the data itself (hence, "self-supervision"). The fact that two words appear together in the corpus provides the training signal. No human labeling is required.

8.3 Word2Vec: The Prediction Task

Let's consider the Word2Vec Skip-Gram model. The task is: Given a target word, predict its neighbors.

Corpus: "...the quick brown fox jumps over..."

Training Strategy (Simplified):

1. **Positive Examples:** Treat actual neighboring words as correct predictions. (e.g., Target: "fox", Context: "brown" → Positive).
2. **Negative Examples:** Randomly sample words from the vocabulary that are NOT neighbors. (e.g., Target: "fox", Context: "keyboard" → Negative).
3. **Training:** Train a classifier (e.g., logistic regression) to distinguish between positive and negative pairs.

During this training process, the model adjusts its internal weight matrix (the embeddings) so that words appearing in similar contexts (like "fox" and "cat") end up with similar weights, enabling the classifier to perform the task effectively.

9 The Mechanics: How Embeddings Work

How are these dense vectors physically stored and accessed in a model?

9.1 The Baseline: One-Hot Encoding

Before embeddings, words were represented using One-Hot Encoding. If the vocabulary size $V = 10,000$:

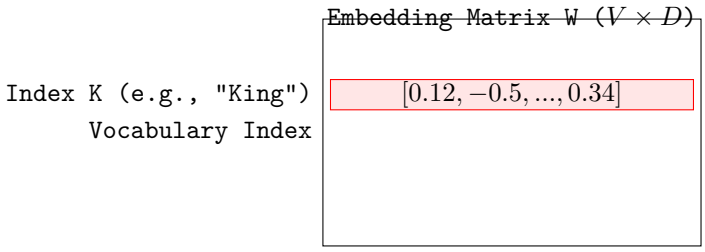
- Each word is a vector of length 10,000.
- It contains a '1' at the index corresponding to the word, and '0's everywhere else.

$$v_{King} = [0, 0, ..., 1, ..., 0, 0]$$

9.2 The Embedding Matrix

The core of a dense embedding model is the **Embedding Matrix** (W).

- **Rows:** Equal to the vocabulary size (V).
- **Columns:** Equal to the desired embedding dimension (D, e.g., 300).
- **Values:** The learned weights (initialized randomly, then adjusted during training).



9.3 The "Lookup Trick"

To get the embedding for a word, we multiply its One-Hot vector by the Embedding Matrix W.

$$\text{Embedding}_K = \text{OneHot}_K \times W$$

Because the One-Hot vector is all zeros except for a single '1' at index K, this matrix multiplication effectively just "selects" the K-th row of the matrix W.

The Math: The Lookup Multiplication

$$\underbrace{[0, 1, 0]}_{\text{Input } (1\times3)} \times \underbrace{\begin{bmatrix} w_{00} & w_{01} \\ \mathbf{w_{10}} & \mathbf{w_{11}} \\ w_{20} & w_{21} \end{bmatrix}}_{\text{Weight Matrix } W} = \underbrace{[\mathbf{w_{10}}, \mathbf{w_{11}}]}_{\text{Embedding Output}}$$

This means the rows of the weight matrix *are* the word embeddings. Training the model (adjusting the weights) is the process of learning the embeddings.

10 Conclusion: Language as Geometry

The journey into Vector Semantics transforms how we view language computationally. We move away from treating words as isolated symbols and embrace the Distributional Hypothesis: meaning is derived from context.

By representing words as vectors—whether sparse counts weighted by TF-IDF or dense, learned embeddings from models like Word2Vec—we convert the abstract concept of meaning into concrete geometry. Similarity becomes the angle between vectors, and semantic relationships emerge as patterns and directions within this high-dimensional space. This geometric map of meaning is the foundation upon which nearly all modern advancements in Natural Language Processing are built.