ARTIFICIAL INTELLIGENCE 2

Embeddings and Representation Learning

Francesco Spinnato

* francesco.spinnato@unipi.it
University of Pisa

Where We Are

So far we have seen neural networks for several kinds of data:

- tabular data with MLPs
- sequential data with RNNs
- images and signals with CNNs
- compressed representations with autoencoders

These data types are continuous. Now we focus on a recurring problem:

How do we represent discrete objects as vectors?

Examples: words, medical codes, drugs, visits, patients

The Problem

Neural networks operate on numbers.

But many healthcare objects are **discrete tokens**:

Token type	Example
diagnosis code	ICD-10: E11.9
procedure code	CPT: 93000
medication	metformin
clinical word	fever
visit event	lab test, diagnosis (text), drug
visit	set of clinical events

We need a numerical representation of these tokens.

What Is a Token?

A **token** is a discrete unit that we decide to treat as an atomic symbol.

- In text, a token may be a word, a subword, or a character.
- In healthcare data, a token may be a diagnosis code, a procedure code, a medication code, or a clinical event.

The definition of token depends on the representation we choose.

For example, a patient history can be represented as a sequence of tokens:

$$[\text{E11.9}, \text{metformin}, 93000, \ldots]$$

Each element in the sequence is mapped to an integer ID and then to a vector.

$$[\text{E11.9}, \text{metformin}, 93000] \quad \rightarrow \quad [0, 1, 2]$$

These integer IDs are not numerical measurements. They are only indices.

Naive Solution: One-Hot Encoding

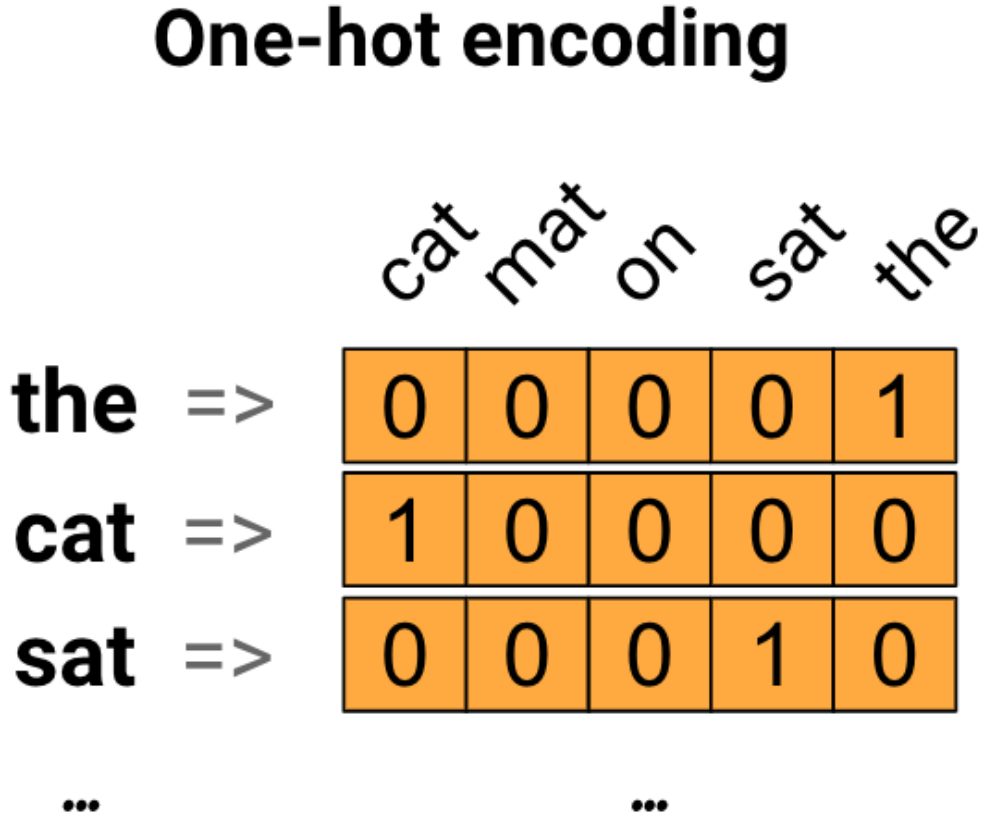
Suppose we have a vocabulary of V discrete tokens:

$$\mathcal{V} = \{w_1, w_2, \ldots, w_V\}$$

A token w_i can be represented as a one-hot vector:

$$\mathbf{x}_i = [0, \ldots, 0, 1, 0, \ldots, 0] \in \mathbb{R}^V$$

Only the i-th entry is equal to 1.



One-Hot Encoding Example

Let the vocabulary be:

$$\mathcal{V} = \{\text{diabetes}, \text{hypertension}, \text{asthma}, \text{metformin}\}$$

Then:

$$\text{diabetes} = [1, 0, 0, 0]$$

$$\text{hypertension} = [0, 1, 0, 0]$$

$$\text{metformin} = [0, 0, 0, 1]$$

Metformin is the most commonly prescribed medication to manage blood sugar in Type 2 diabetes

From Tokens to Documents: Bag-of-Words

One-hot encoding represents a **single token**, but often we want to represent a whole document, sentence, visit, or patient record.

A simple classical representation is **Bag-of-Words (BoW)**.

- BoW counts how many times each token appears.
- The order of tokens is ignored.
- BoW can be seen as the sum of the one-hot vectors of all words (tokens) appearing in that document.

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

Problems with Sparse Representations

One-hot and Bag-of-Words vectors are:

- **high-dimensional**: one dimension for each possible token
- **sparse**: most entries are zero
- **weakly semantic**: related tokens are not naturally close
- **order-insensitive** in BoW: token order is ignored
- **hard to generalize**: similar clinical concepts may look unrelated

For example, with one-hot encoding:

$$\text{distance}(\text{diabetes}, \text{metformin}) = \text{distance}(\text{diabetes}, \text{aspirin})$$

But clinically, these pairs are not equally related.

Dense Representations

An **embedding** maps a discrete object into a dense vector:

$$\text{object} \longrightarrow \mathbf{e} \in \mathbb{R}^d$$

where usually:

$$d \ll V$$

Instead of representing an object with thousands of sparse dimensions, we represent it with a compact dense vector.

This is the same broad idea seen in autoencoders, but the input objects are discrete symbols.

Embedding Intuition

The goal is not just compression.

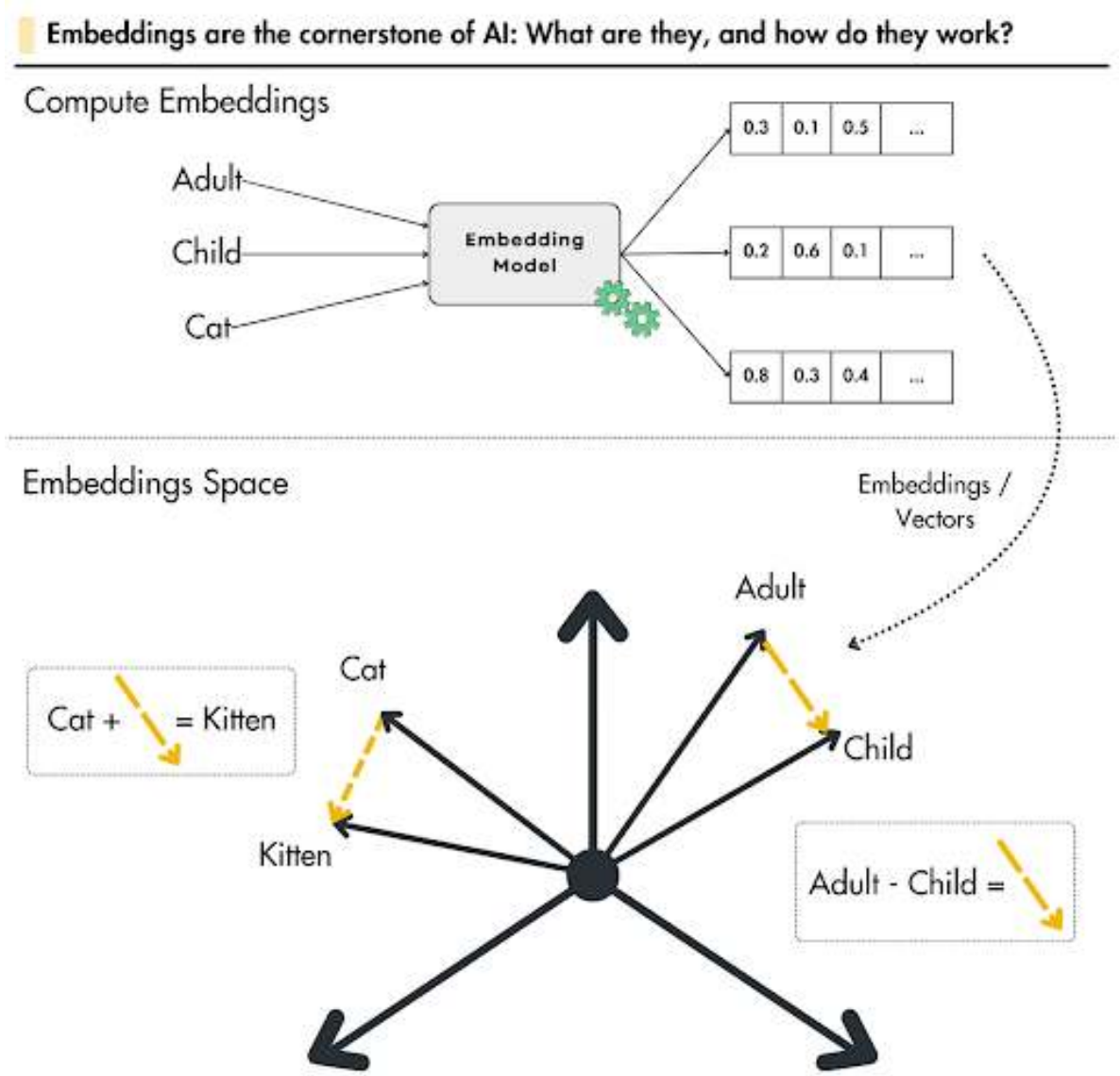
The goal is to learn a space where useful relationships become geometric relationships.

Similar objects should have nearby vectors.

$$\mathbf{e}_{\text{diabetes}} \approx \mathbf{e}_{\text{metformin}}$$

Different objects should have distant vectors.

$$\mathbf{e}_{\text{diabetes}} \not\approx \mathbf{e}_{\text{aspirin}}$$



Embedding Matrix

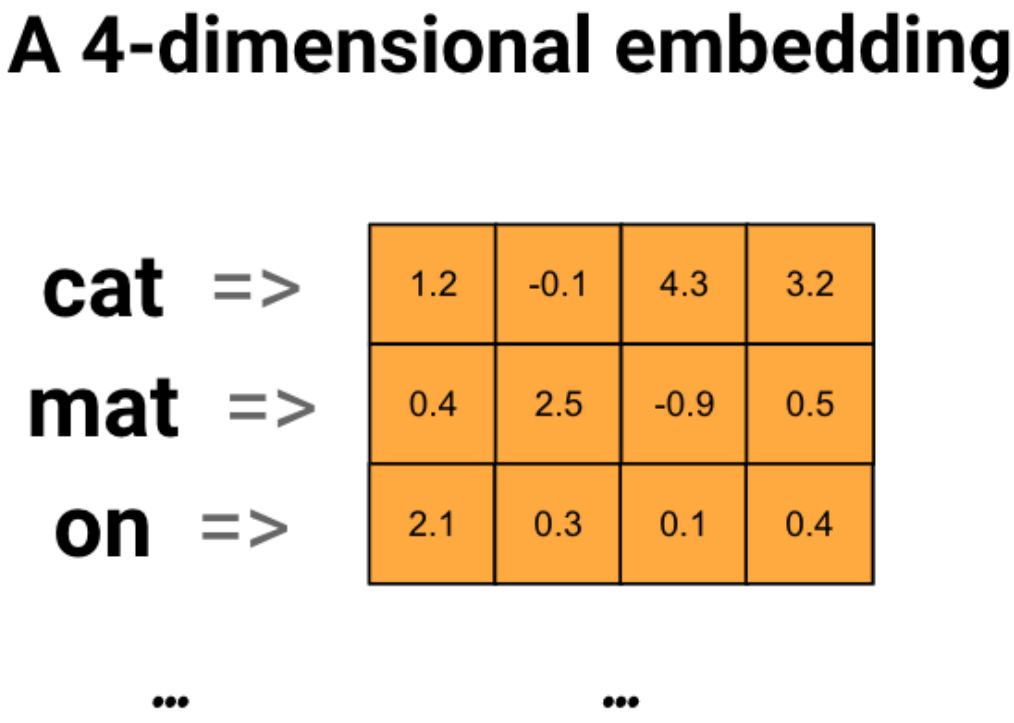
An embedding layer is just a matrix:

$$\mathbf{E} \in \mathbb{R}^{V \times d}$$

where:

- V is the vocabulary size
- d is the embedding dimension
- row i contains the embedding vector for object i

$$\mathbf{E}_{i,:} = \mathbf{e}_i$$



Embedding Lookup

Given an object index i, the embedding layer returns the i-th row of the embedding matrix:

$$\text{Embedding}(i) = \mathbf{E}_{i,:}$$

This operation is called **embedding lookup**.

It is equivalent to multiplying a one-hot vector by the embedding matrix:

$$\mathbf{x}_i^\top \mathbf{E} = \mathbf{E}_{i,:}$$

but it is implemented more efficiently.

Embedding Lookup Example

$$\text{Suppose metformin is encoded as } [0,0,0,1] \text{ and: } \quad \mathbf{E} = \begin{bmatrix} 0.2 & 1.1 \\ -0.4 & 0.8 \\ 1.5 & -0.3 \\ 0.7 & 0.1 \end{bmatrix}$$

Naive view: one-hot multiplication

$$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.2 & 1.1 \\ -0.4 & 0.8 \\ 1.5 & -0.3 \\ 0.7 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.7 & 0.1 \end{bmatrix}$$

Efficient view: row lookup

Instead of building the one-hot vector, we directly select row 4:

$$\text{Embedding}(4) = \mathbf{E}_{4,:}$$

$$= \begin{bmatrix} 0.7 & 0.1 \end{bmatrix}$$

Embeddings Are Parameters

The embedding matrix is learned from data.

- It is part of the neural network parameters:

$$\theta = \{\mathbf{E}, \mathbf{W}_1, \mathbf{W}_2, \ldots\}$$

- During training, gradients update the rows of $\mathbf{E}$ that correspond to the objects observed in the batch.
- So an embedding is not manually designed. It is learned to be useful for the training objective. An embedding learns relationships from the task and the data.
- Examples:
  - words appearing in similar contexts get similar embedding vectors
  - medical codes appearing in similar patients get similar embedding vectors

Embeddings and Representation Learning

Word2Vec

Word2Vec: Main Idea

Word2Vec is a family of models for learning word embeddings.

The central assumption is distributional: **words that occur in similar contexts tend to have similar meanings.**

Example:

- "patient has fever and cough"
- "patient has fever and mucus"

The words **cough** and **mucus** may become close because they appear in similar clinical contexts.

Context Windows

Given a sequence of tokens $[w_1, w_2, \ldots, w_T]$ we define a context window of length m around each token. E.g., if $m = 2$, the model uses the two tokens before and after the center token w_t:

$$w_{t-2},\ w_{t-1},\ \mathbf{w_t},\ w_{t+1},\ w_{t+2}$$

The context tokens are: $\{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$

For example:

- patient has severe chest pain today

For the center word **chest**, the context is $\{\text{has}, \text{severe}, \text{pain}, \text{today}\}$:

- patient **has severe** *chest* **pain today**

Word2Vec learns from these local co-occurrence patterns.

Two Word2Vec Formulations

There are two classical Word2Vec formulations.

CBOW (continuous bag of words)

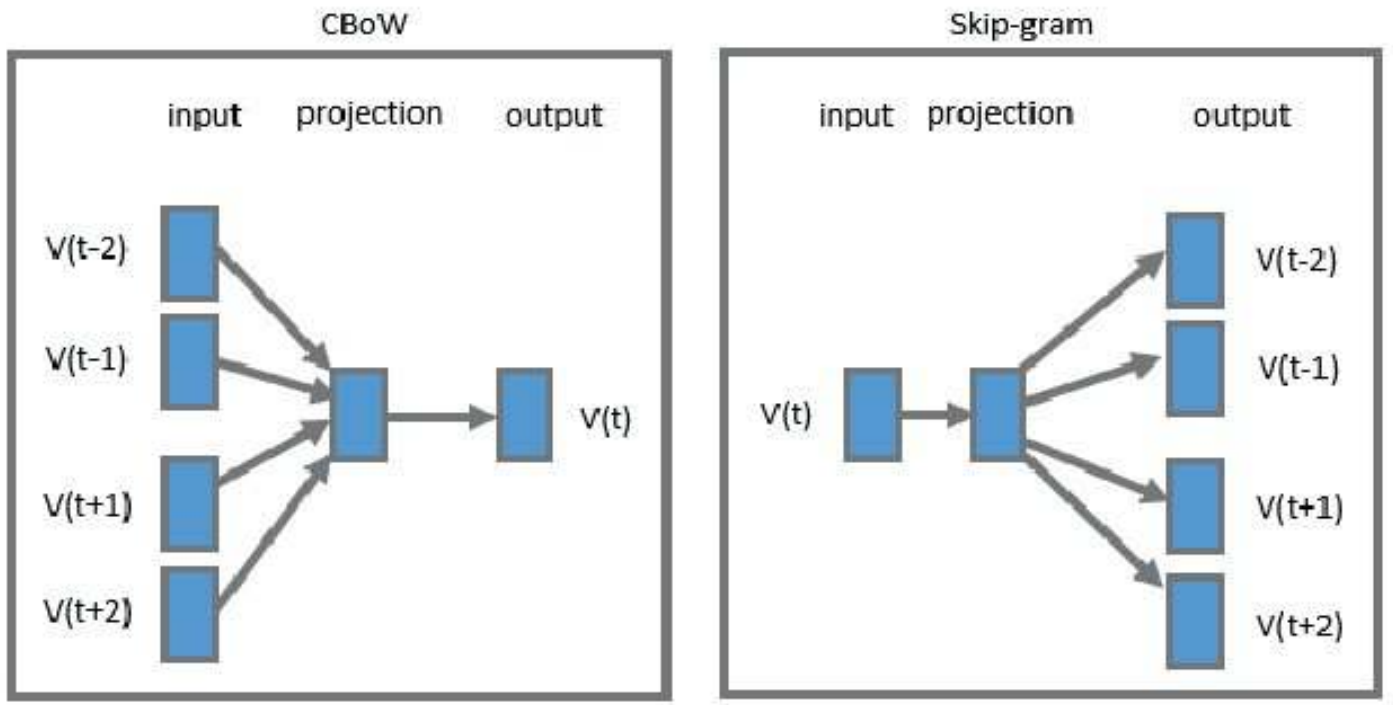
Predict the center word from its context.

$$\text{context} \rightarrow \text{center word}$$

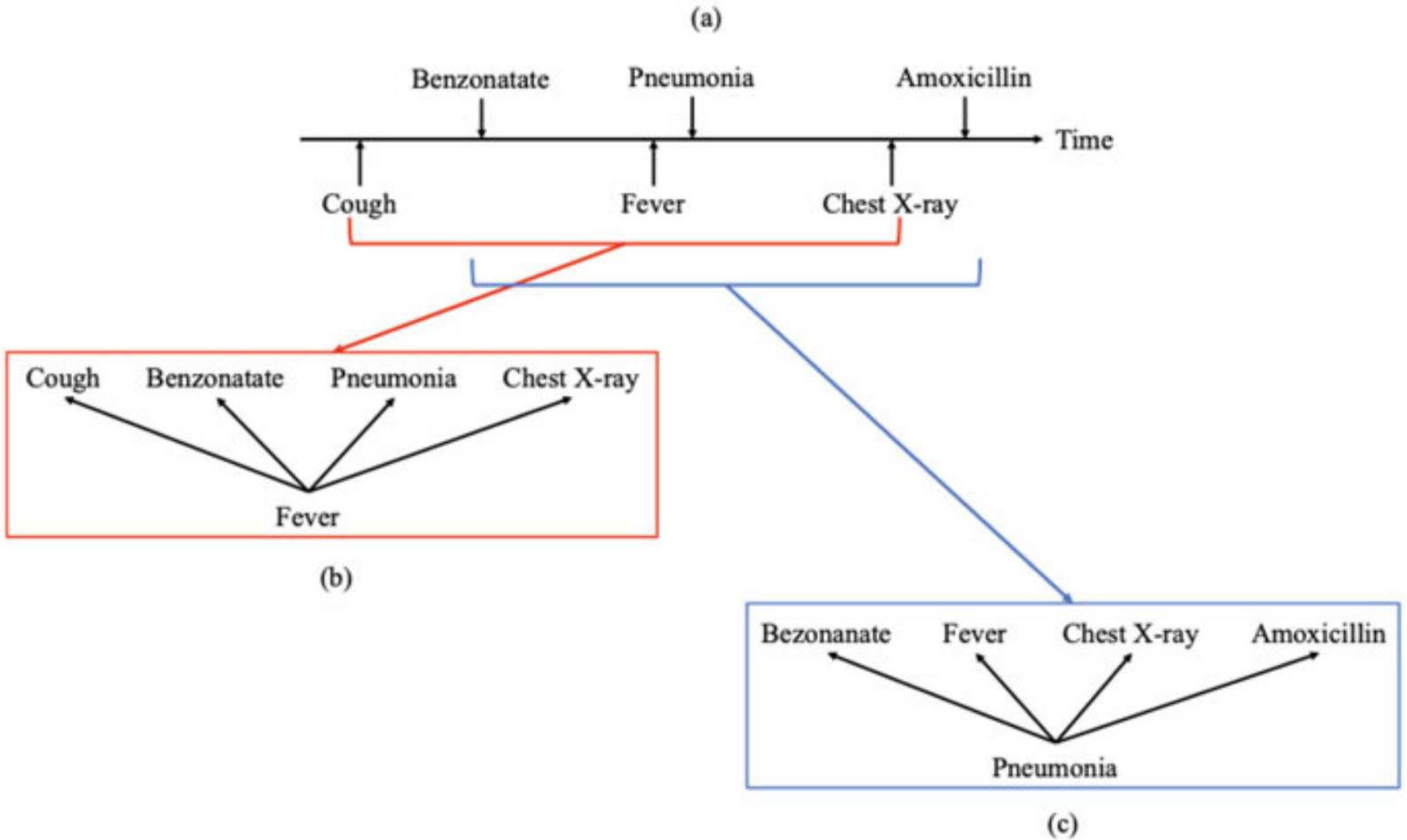
Skip-gram

Predict context words from the center word.

$$\text{center word} \rightarrow \text{context}$$



SkipGram Healthcare Example



Word2Vec (SkipGram)

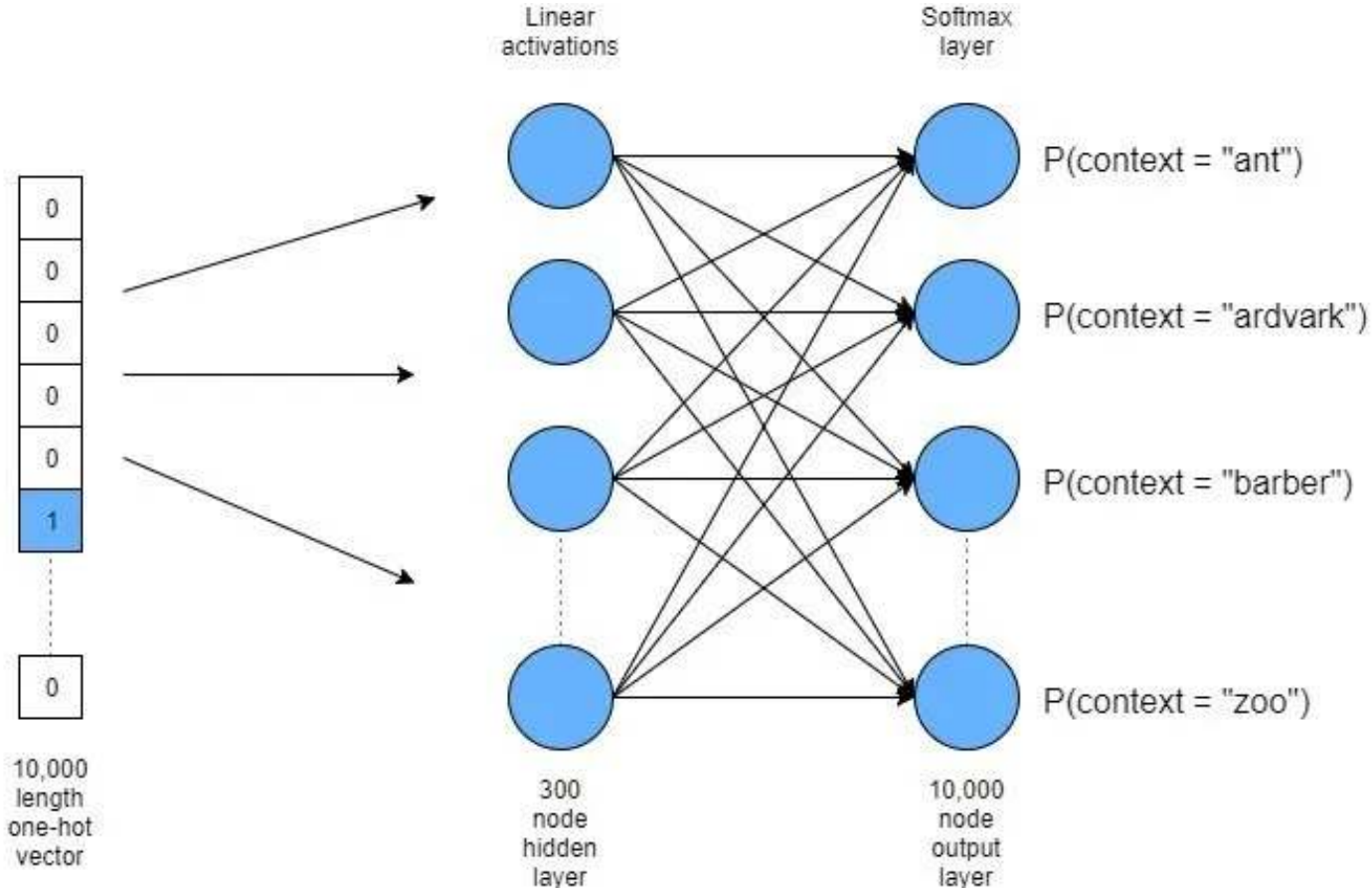
Word2Vec learns embeddings by solving a prediction problem. Given many observed token sequences, the model estimates parameters so that real center-context pairs become likely.

For **skip-gram**, the model learns probabilities of the form $p(w_o \mid w_c)$

where:

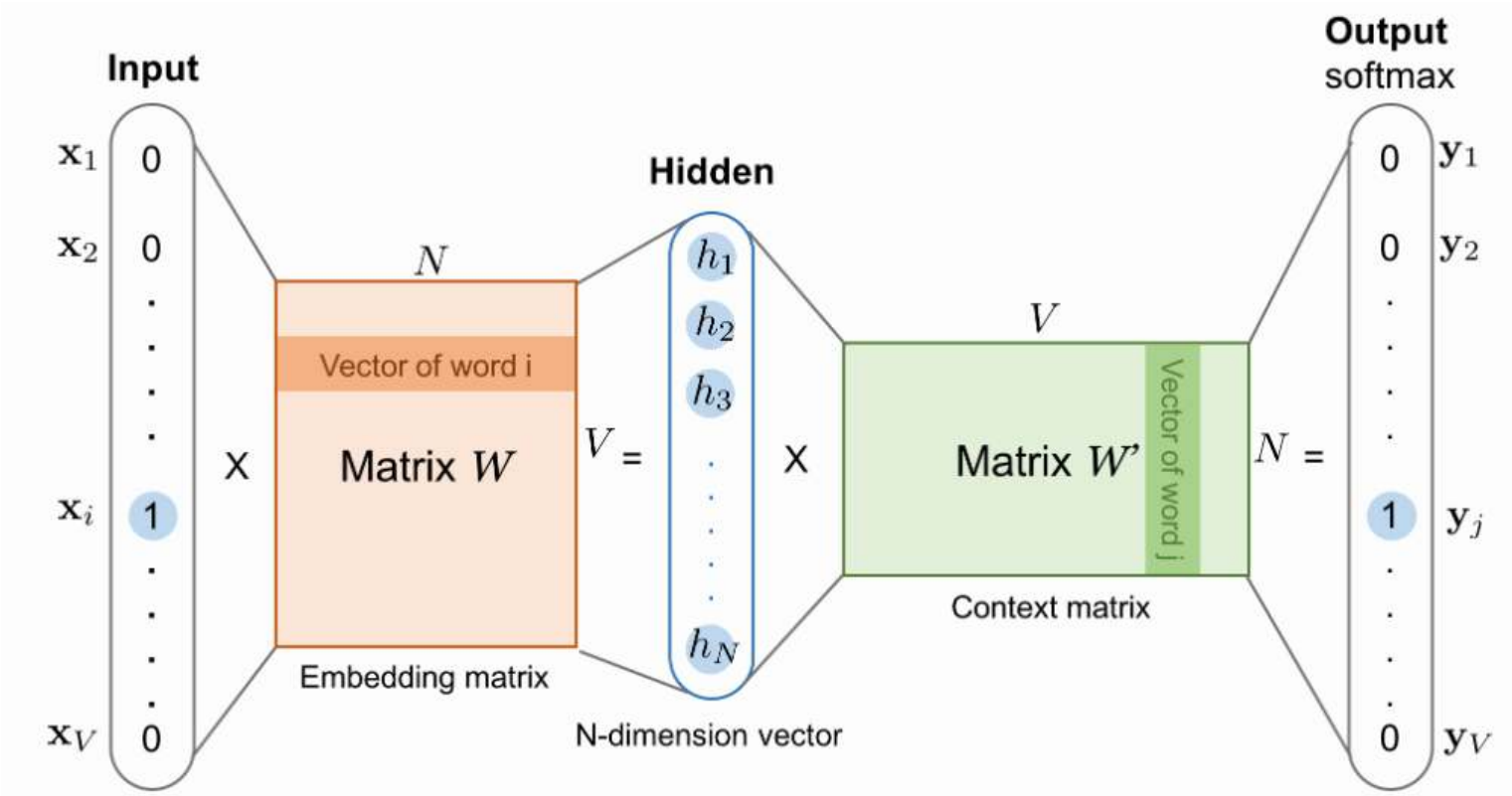
- w_c : center token
- w_o : observed context token

The learned parameters are the embedding vectors.



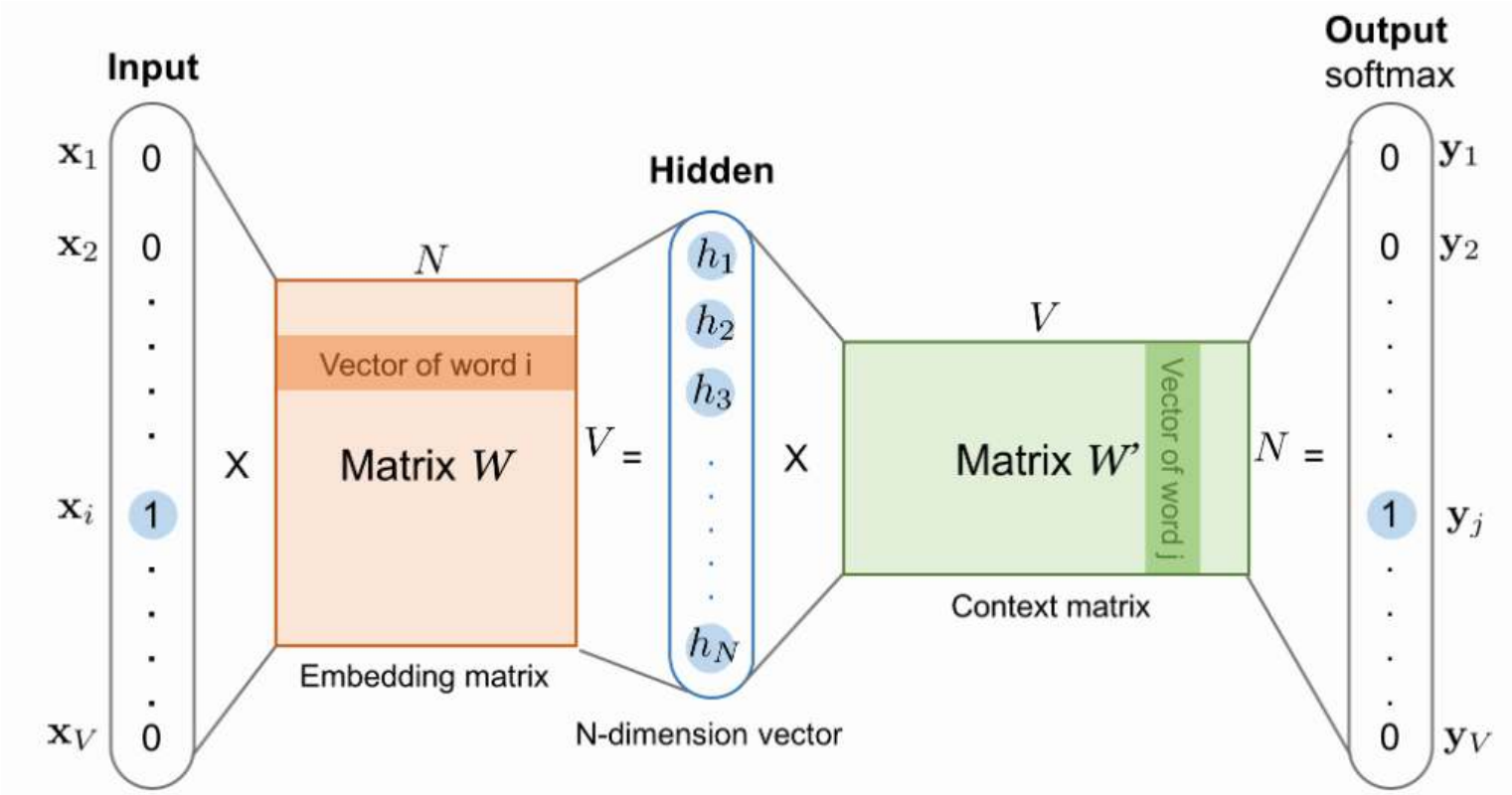
Word2Vec (SkipGram)

The first step is to multiply the one-hot encoded center token by an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$ ($\mathbf{W}$ in the picture). This operation simply selects the i-th row of $\mathbf{E}$. The selected row $\mathbf{h}$ is the embedding of the center token.



Word2Vec (SkipGram)

Then the hidden vector $\mathbf{h}$ is multiplied by a context matrix $\mathbf{W}' \in \mathbb{R}^{d \times V}$. This produces one score for each token in the vocabulary. A softmax converts these scores into probabilities $p(w_o|w_c)$ (for each possible w_o).



Skip-Gram Objective

For a center token w_t, the model predicts nearby context tokens w_{t+j}. The goal is to maximize the likelihood of the observed center-context pairs:

$$\prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} p(w_{t+j} \mid w_t)$$

That is, we learn those weights, or embeddings, so that when the center token w_t is given as input to the model, the tokens that actually appear around it in the corpus receive high probability.
Equivalently, **minimize the negative log-likelihood**:

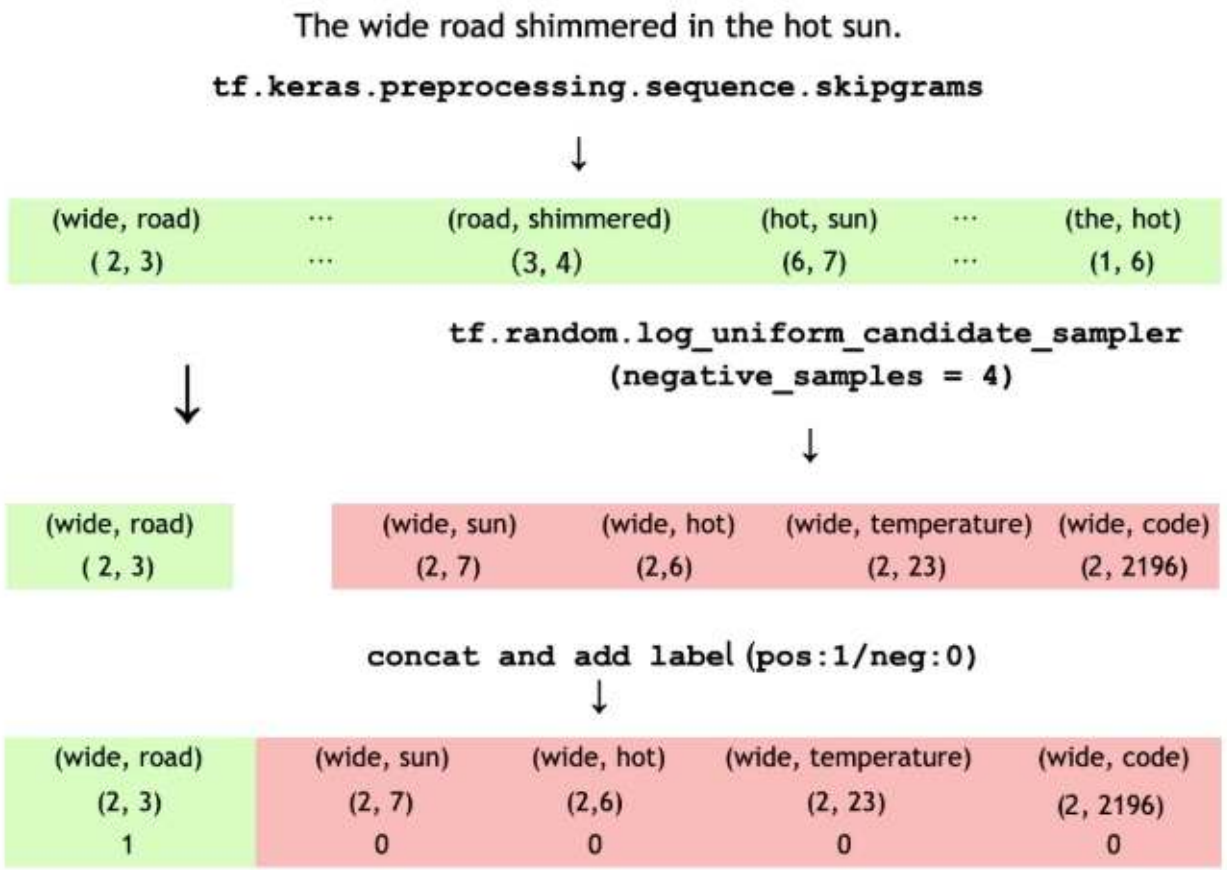
$$-\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j} \mid w_t)$$

In practice, the optimization is performed with SGD, using center-context pairs sampled from the corpus.

Negative Sampling

Predicting the correct context token with a softmax over the whole vocabulary is expensive. Negative sampling replaces this with a binary classification problem:

- **Positive pair**: a real center-context pair observed in the corpus:
  - $(\text{wide}, \text{road}) \rightarrow 1$
  - The model should assign it a high score.
- **Negative pair**: a randomly sampled fake pair:
  - $(\text{wide}, \text{code}) \rightarrow 0$
  - The model should assign it a low score.



For each positive pair, we sample a few negative pairs.

Negative Sampling

For each pair, the model computes a dot product between two learned vectors. It acts as a compatibility score, similar to a similarity measure: larger values mean that the two tokens are more likely to appear in the same context.

- Suppose a 2D embedding ($d = 2$) and the learned embeddings are:

$$\text{chest} \to [0.2, 0.7] \qquad \text{pain} \to [0.8, 0.6] \qquad \text{banana} \to [-0.6, 0.1]$$

Positive pair

$$(\text{chest}, \text{pain}) \to 1$$

$$[0.2, 0.7] \cdot [0.8, 0.6]$$

$$= 0.2 \cdot 0.8 + 0.7 \cdot 0.6 = 0.58$$

$$\sigma(0.58) \approx 0.64$$

Negative pair

$$(\text{chest}, \text{banana}) \to 0$$

$$[0.2, 0.7] \cdot [-0.6, 0.1]$$

$$= 0.2 \cdot (-0.6) + 0.7 \cdot 0.1 = -0.05$$

$$\sigma(-0.05) \approx 0.49$$

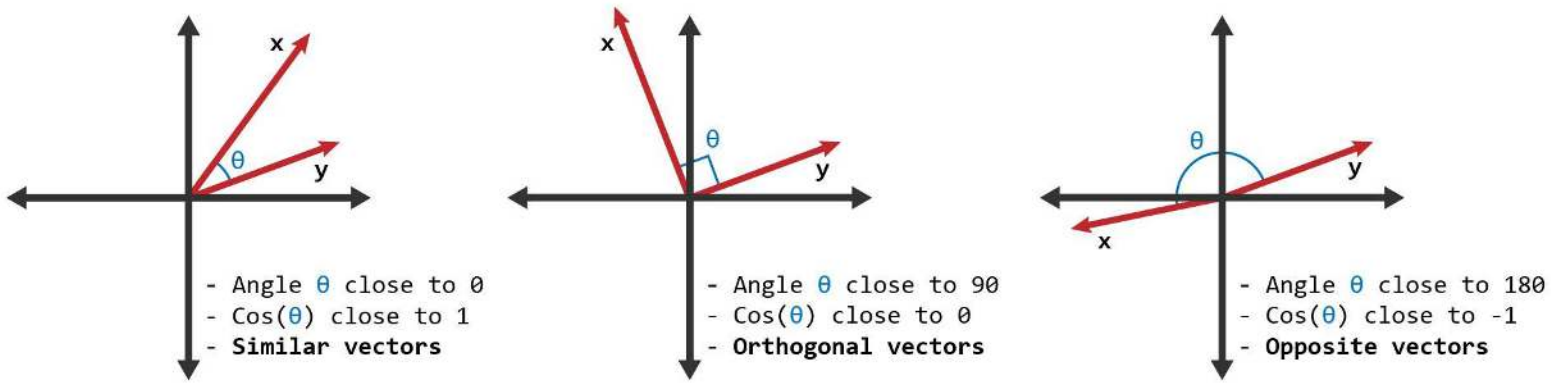
Cosine Similarity

Word2Vec training uses dot products as compatibility scores.
After training, when we want to compare embeddings, we often use cosine similarity, which is a normalized dot product.

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

It measures the angle between two vectors.

- close to 1: similar direction
- close to 0: unrelated direction
- close to -1: opposite direction

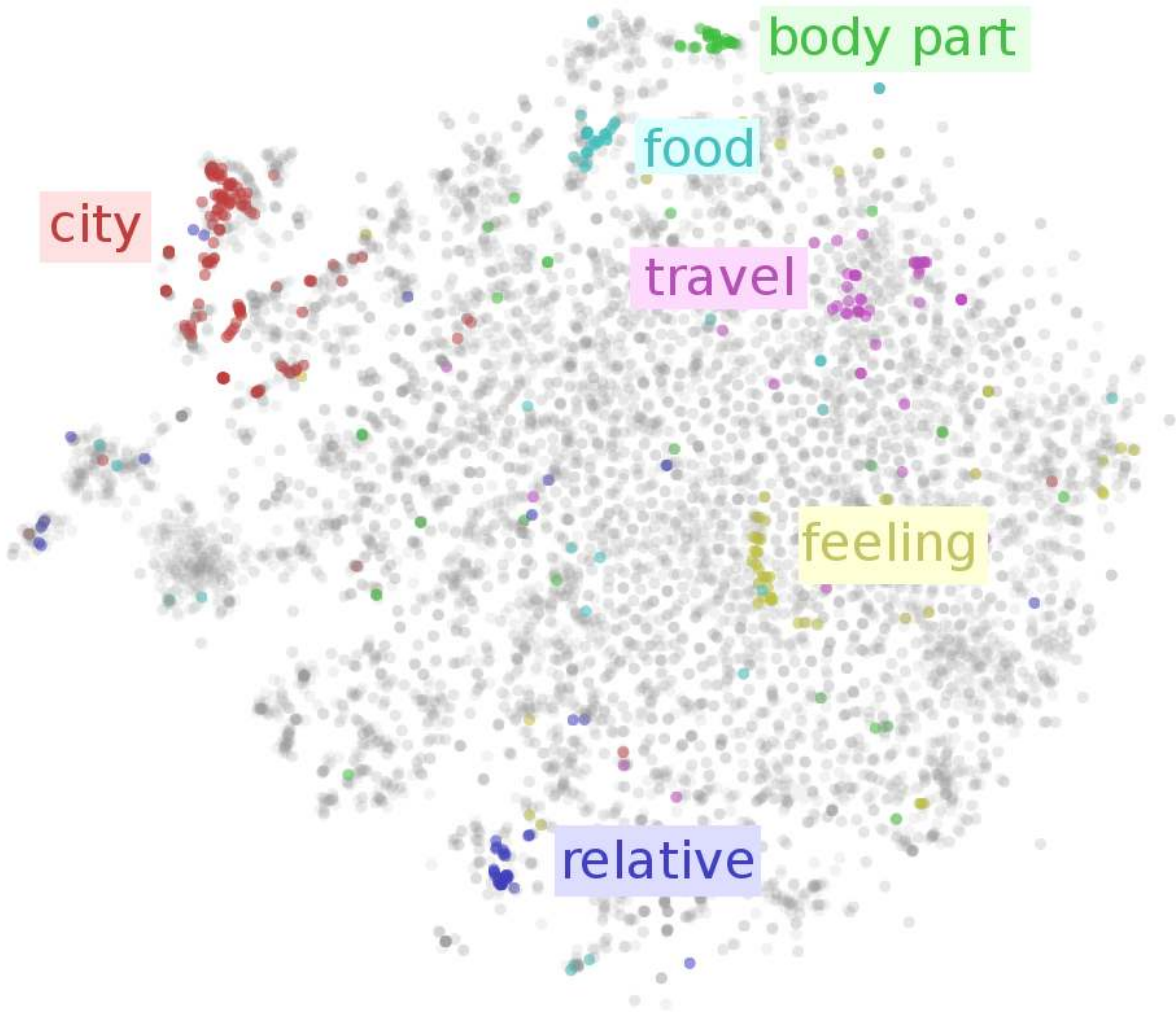


What Word2Vec Gives Us

After training, each word has a vector representation.

These vectors can be used for:

- similarity search
- visualization
- clustering
- input to neural networks
- initialization of downstream models



Embeddings in Modern NLP

Besides Word2Vec, classical static embedding methods include:

- **GloVe**: learns embeddings from global word co-occurrence statistics
- **FastText**: represents words using subword information, useful for rare words

These approaches learn one **static** vector per word. Modern language models produce **contextual embeddings**. Example:

- in "cold weather", **cold** refers to temperature
- in "cold symptoms", **cold** refers to illness

Contextual models can produce different vectors for the same token depending on the surrounding words.

A typical model for discrete sequences starts with an embedding layer. Then the embedded sequence is processed by RNN, CNN, attention layer, or transformer

Embeddings and Representation Learning

Visualization and dimensionality reduction

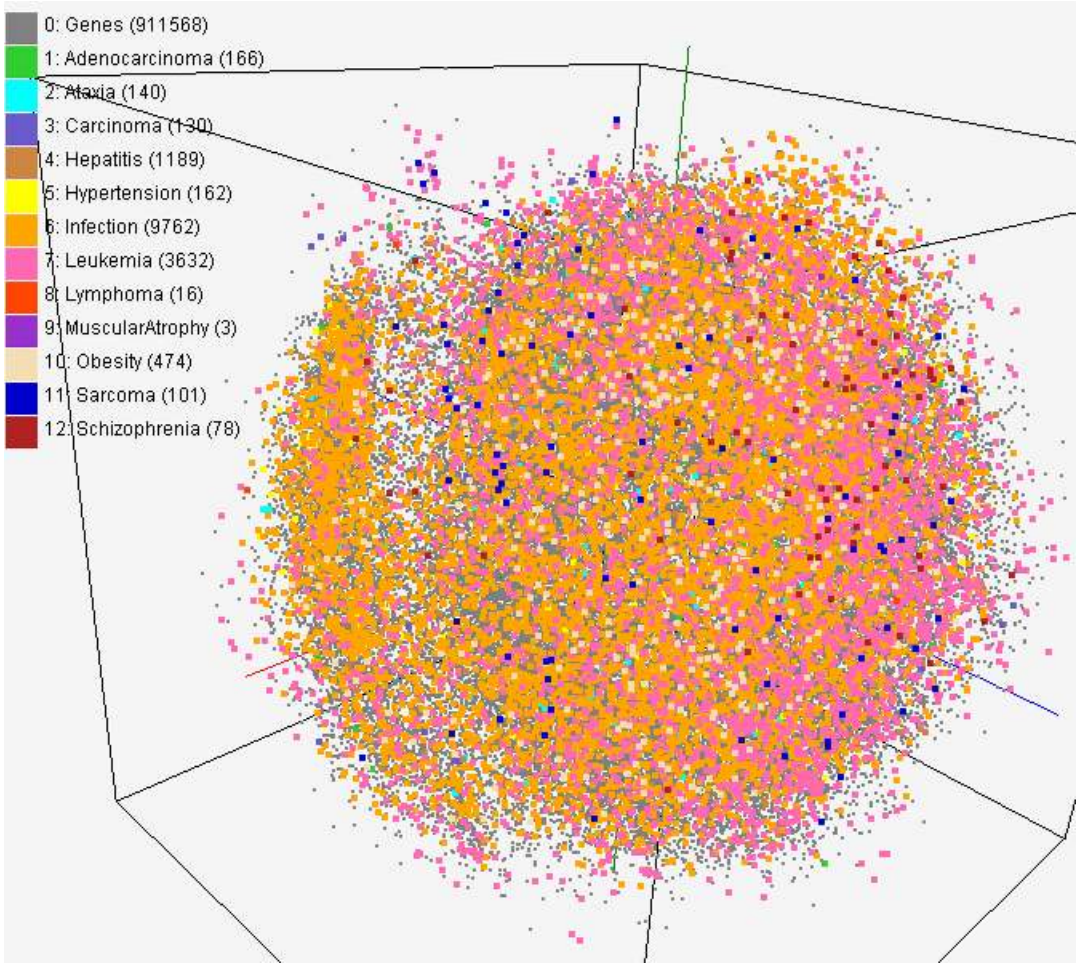
Embedding Visualization

Why visualize embeddings?

- Embeddings often have tens, hundreds, or thousands of dimensions.
- Humans cannot directly inspect high-dimensional geometry.
- So we project embeddings into 2D or 3D to see whether meaningful structure appears.

Typical questions:

- do similar concepts cluster together?
- are there outliers?
- are there subgroups?
- are representations dominated by frequency or artifacts?



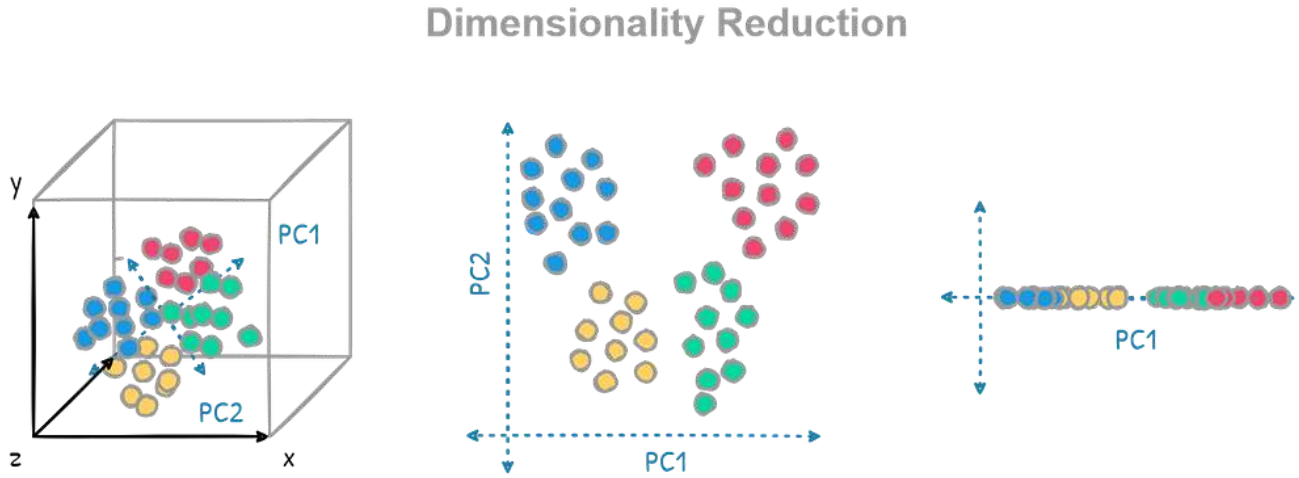
Dimensionality Reduction

Dimensionality reduction maps high-dimensional points into a lower-dimensional space. Let $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ be the original high-dimensional points.

We want:

$$\mathbf{z}_1, \ldots, \mathbf{z}_N \in \mathbb{R}^2$$

such that important relationships between points are preserved.



PCA vs t-SNE

Two common visualization methods:

PCA

- Linear projection.
- Preserves directions of maximum variance.
- Fast and deterministic.

t-SNE

- Nonlinear projection.
- Preserves local neighborhoods.
- Useful for visual clusters.

Principal Component Analysis

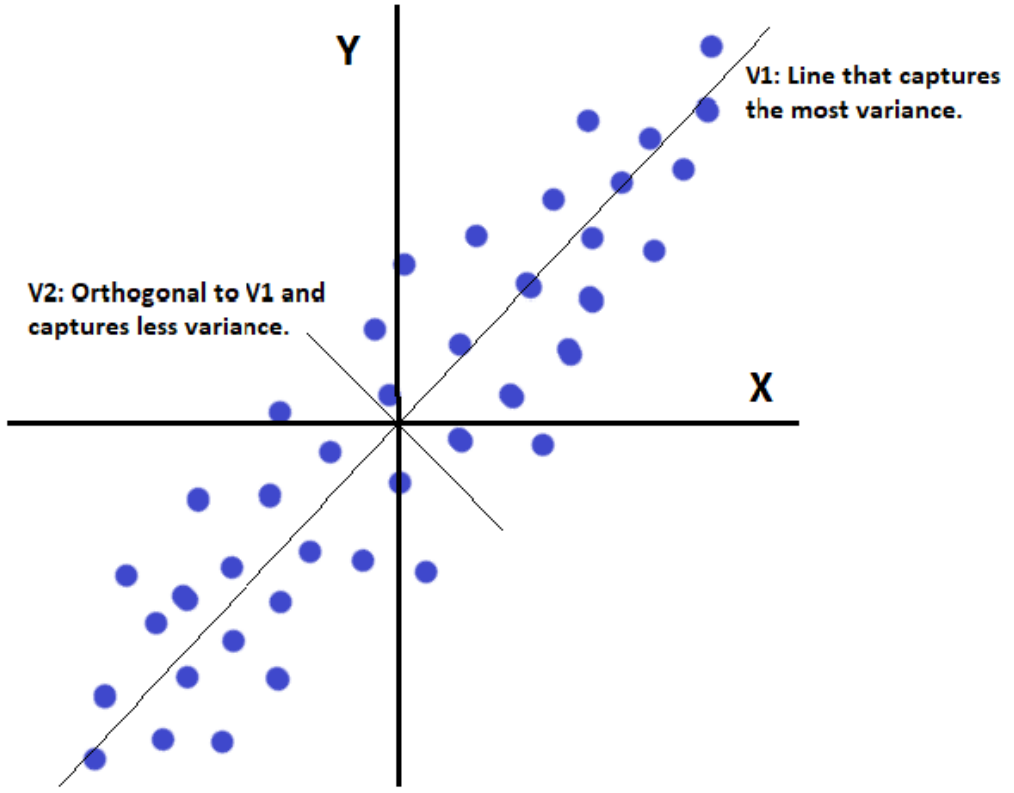
Principal Component Analysis (PCA) is a linear dimensionality reduction method. Given high-dimensional points:

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$$

PCA finds new axes, called **principal components**, such that:

- the first component captures the largest possible variance
- the second component captures the largest remaining variance... and so on...
- the components are orthogonal to each other

For visualization, we keep only the first two components: $\mathbb{R}^d \to \mathbb{R}^2$



PCA as a Projection

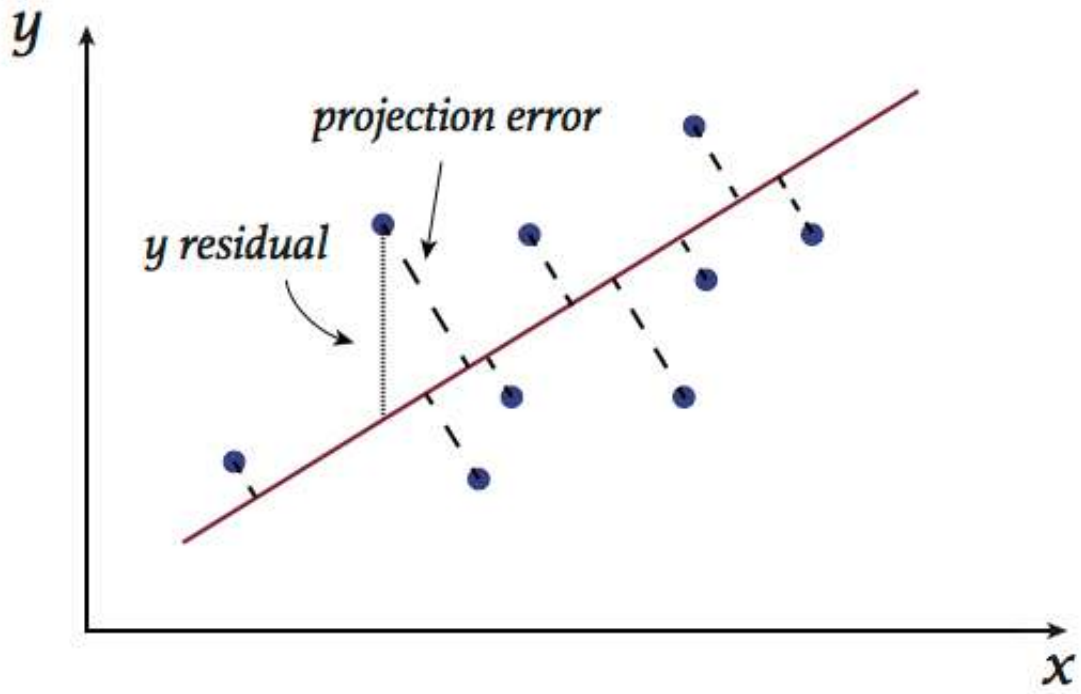
If we keep two principal components, we project each vector $\mathbf{x}_i$ onto two directions:

$$\mathbf{z}_i = \begin{bmatrix} \mathbf{x}_i^\top \mathbf{u}_1 & \mathbf{x}_i^\top \mathbf{u}_2 \end{bmatrix}$$

where:

- $\mathbf{u}_1$ is the first principal direction
- $\mathbf{u}_2$ is the second principal direction
- $\mathbf{z}_i \in \mathbb{R}^2$ is the projected point

PCA therefore creates a 2D view by looking at the data from the directions of maximum variance.



Limitations of PCA

PCA is useful, but it has important limitations.

- It is linear.
- It may miss nonlinear structure.
- It focuses on directions with high variance.
- High variance does not always mean semantic importance.
- Nearby points in the original space may not remain nearby after projection (it does not preserve local neighborhoods).

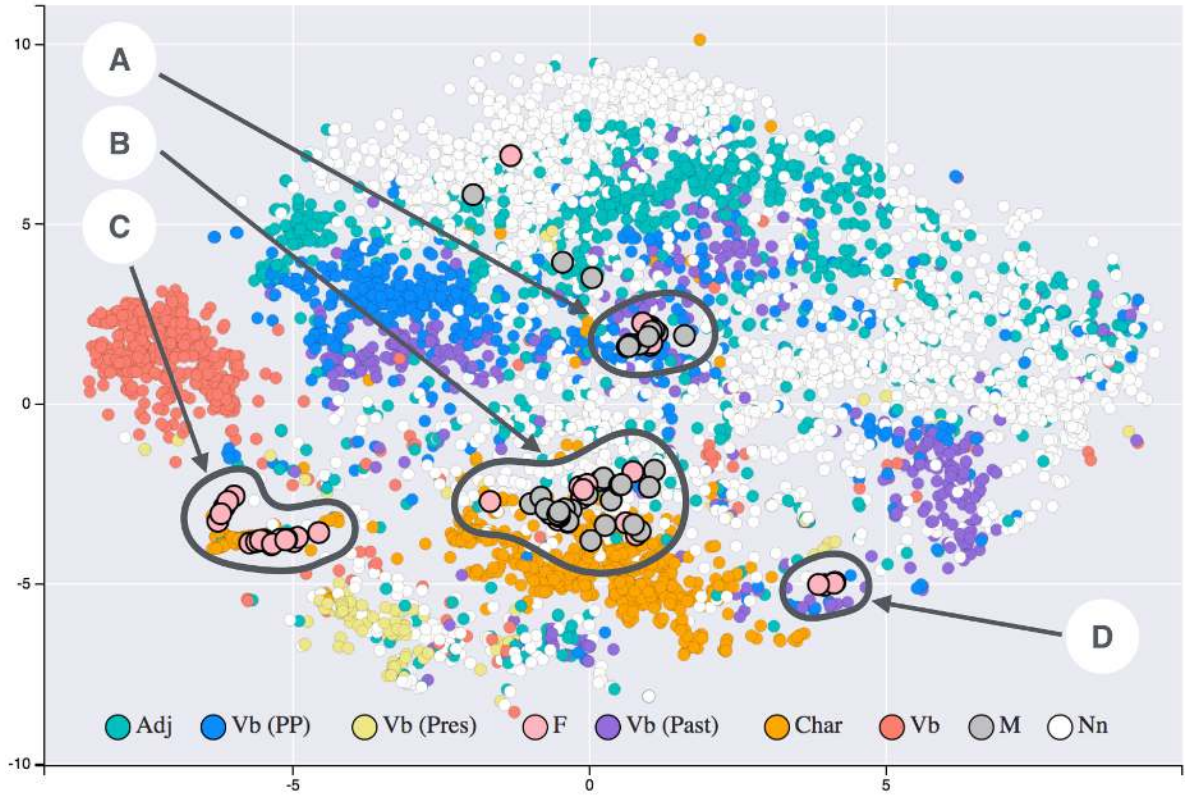
So PCA is often a good first visualization method, but not always enough for complex embedding spaces.

t-SNE

The t-distributed stochastic neighbor embedding (t-SNE) algorithm tries to preserve local neighborhoods.

- In high dimension, the fact that $\mathbf{e}_i \approx \mathbf{e}_j$,
- should imply that after projection: $\mathbf{z}_i \approx \mathbf{z}_j$

where $\mathbf{z}_i \in \mathbb{R}^2$



t-SNE does not try to preserve raw distances directly. Instead, it converts distances into probabilities.

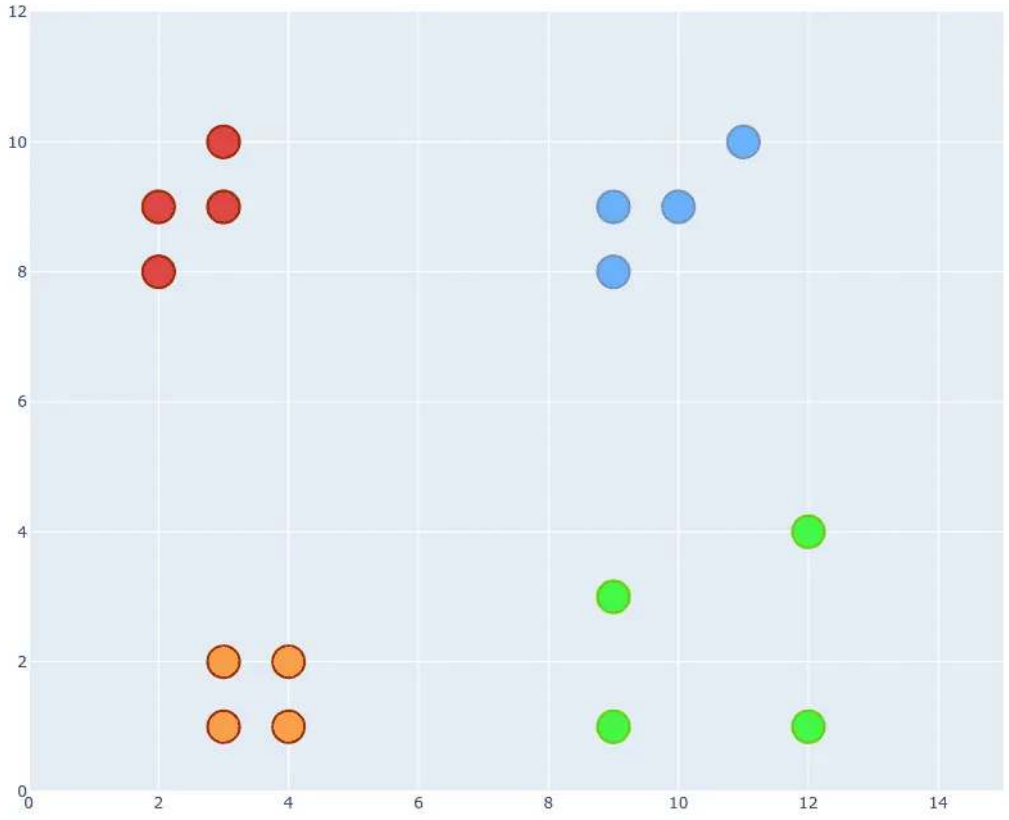
Step 1: Similarities in the Original Space

For each point $\mathbf{x}_i$, t-SNE asks:

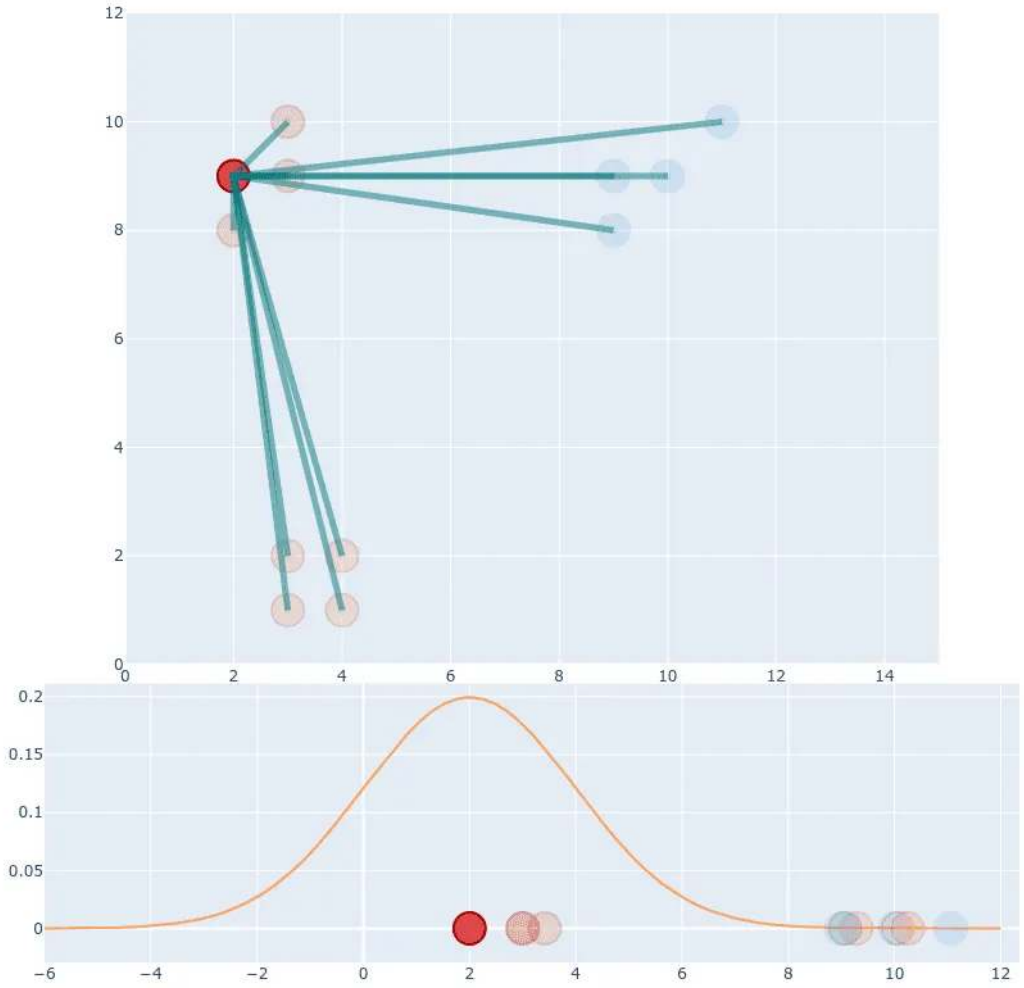
Which other points are likely to be its neighbors?

The similarity of $\mathbf{x}_j$ to $\mathbf{x}_i$ is defined as a conditional probability $p_{j|i}$, which is the probability that $\mathbf{x}_i$ picks $\mathbf{x}_j$ as a neighbor.

- Nearby points should have large probability.
- Far-away points should have small probability.



Gaussian Neighborhoods



For each point $\mathbf{x}_i$, t-SNE places a Gaussian distribution centered on $\mathbf{x}_i$.

The closer $\mathbf{x}_j$ is to $\mathbf{x}_i$, the larger the probability.

$$p_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}\right)}$$

Also $p_{i|i} = 0$, i.e., a point is not considered a neighbor of itself.

Gaussian Neighborhoods

The numerator measures how close $\mathbf{x}_j$ is to $\mathbf{x}_i$, it's a sort of similarity measure:

$$\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)$$

- If $\mathbf{x}_j$ is close to $\mathbf{x}_i$ then $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is small, so the value is large.
- If $\mathbf{x}_j$ is far from $\mathbf{x}_i$ then $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is large so the value is close to zero.
- The denominator normalizes all values so that: $\sum_{j \neq i} p_{j|i} = 1$

The Role of sigma

The parameter σ_i controls the width of the Gaussian around $\mathbf{x}_i$.

Small σ_i:

- only very close points get high probability
- the neighborhood is narrow
- the model focuses on very local structure

Large σ_i:

- more points get non-negligible probability
- the neighborhood is wider
- the model considers a broader region

t-SNE does not use the same σ for all points, each point gets its own σ_i. This allows t-SNE to define neighborhoods in a locally adaptive way.

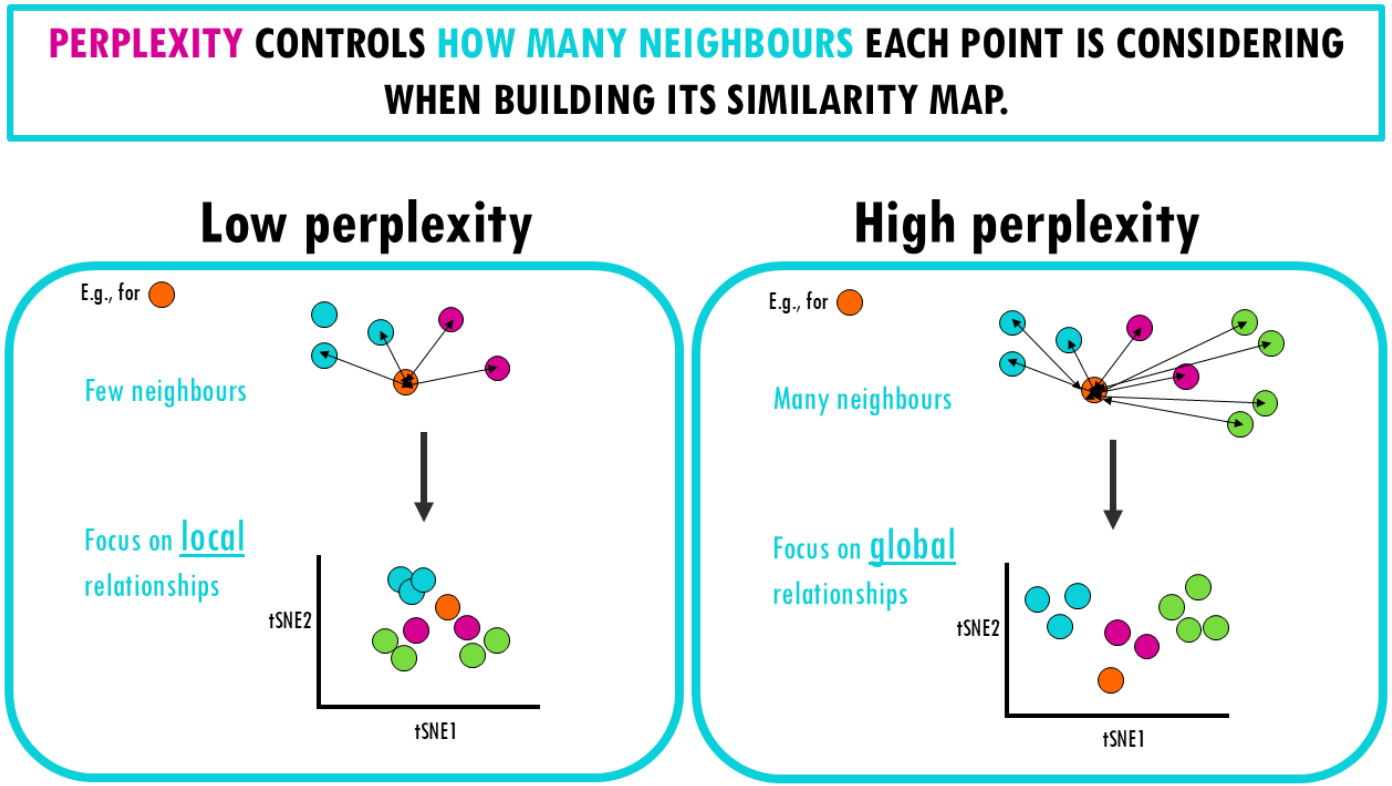
Perplexity

The user does not directly choose σ_i. Instead, the user chooses **perplexity**.

Perplexity is a soft target for the number of neighbors considered around each point.

- Low perplexity: smaller effective neighborhoods, more emphasis on very local structure
- High perplexity: larger effective neighborhoods, more smoothing over nearby points

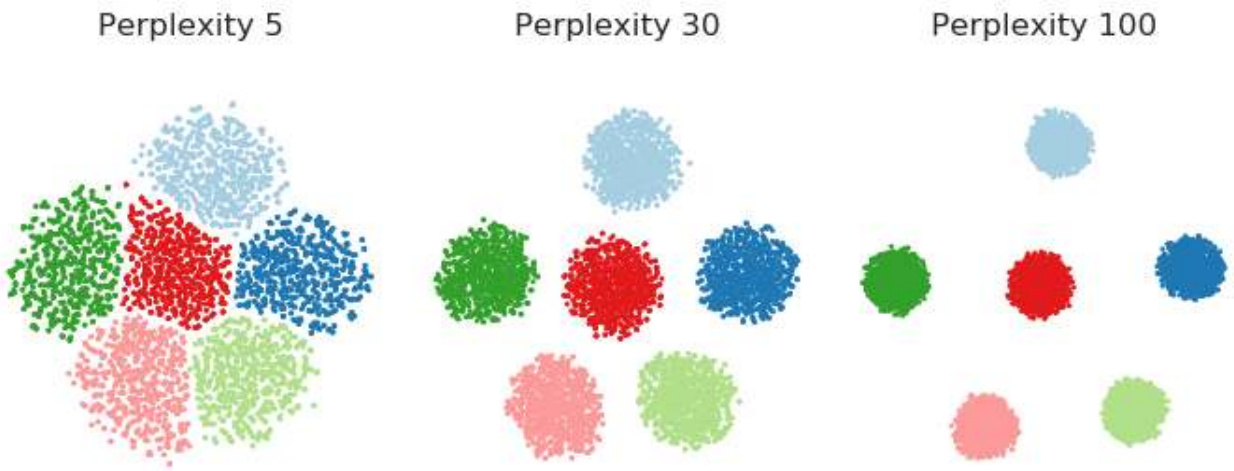
Typical values are often between 5 and 50.



Perplexity Intuition

Suppose we set $\text{perplexity} = 30$. Then t-SNE tries to choose each σ_i so that point $\mathbf{x}_i$ has roughly 30 relevant neighbors.

- This does not mean exactly 30 neighbors.
- It means that the probability distribution around $\mathbf{x}_i$ has an effective neighborhood size of about 30.
- So perplexity controls the scale at which similarities are measured.



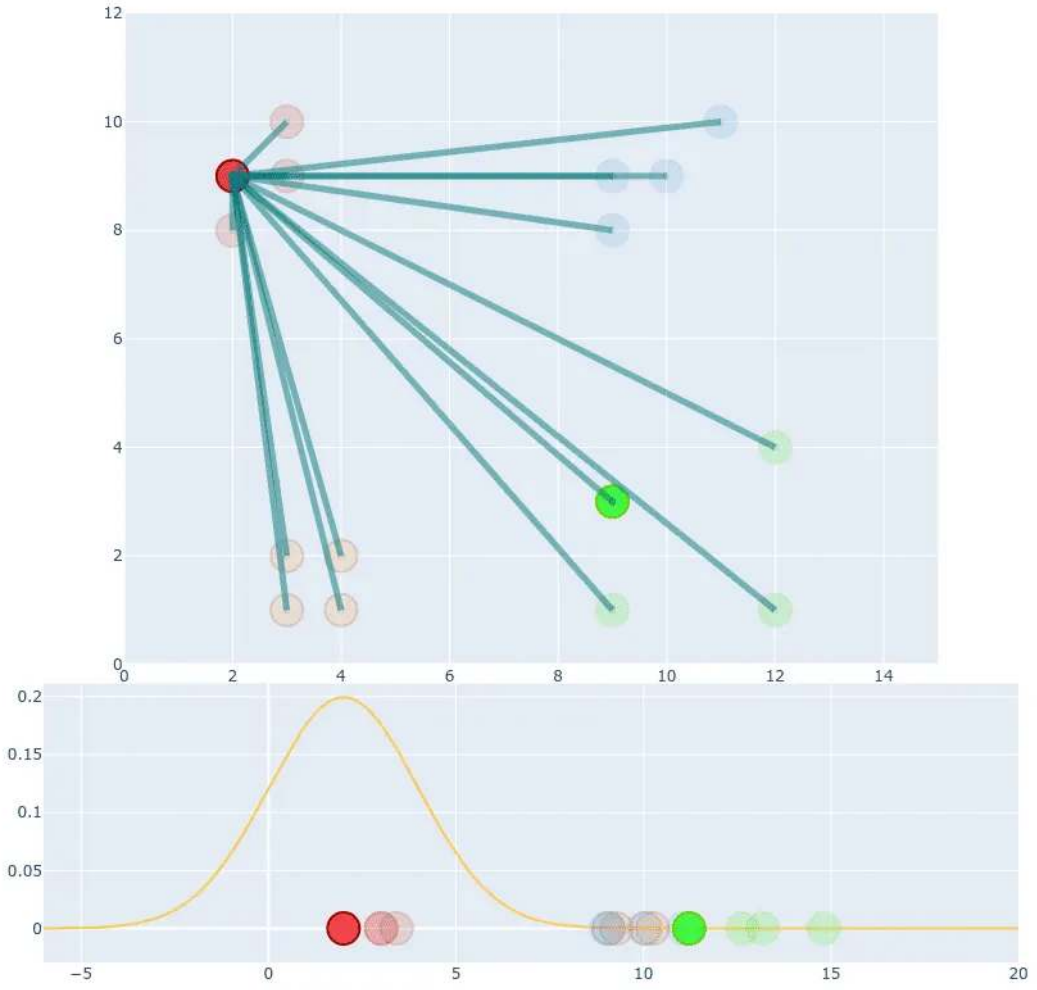
From Conditional to Symmetric Probabilities

The conditional probabilities are not symmetric, i.e., $p_{j|i} \neq p_{i|j}$ because the Gaussian around $\mathbf{x}_i$ may have a different width from the Gaussian around $\mathbf{x}_j$.

t-SNE converts them into symmetric pairwise similarities:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

p_{ij} represents the similarity between points i and j in the original space.



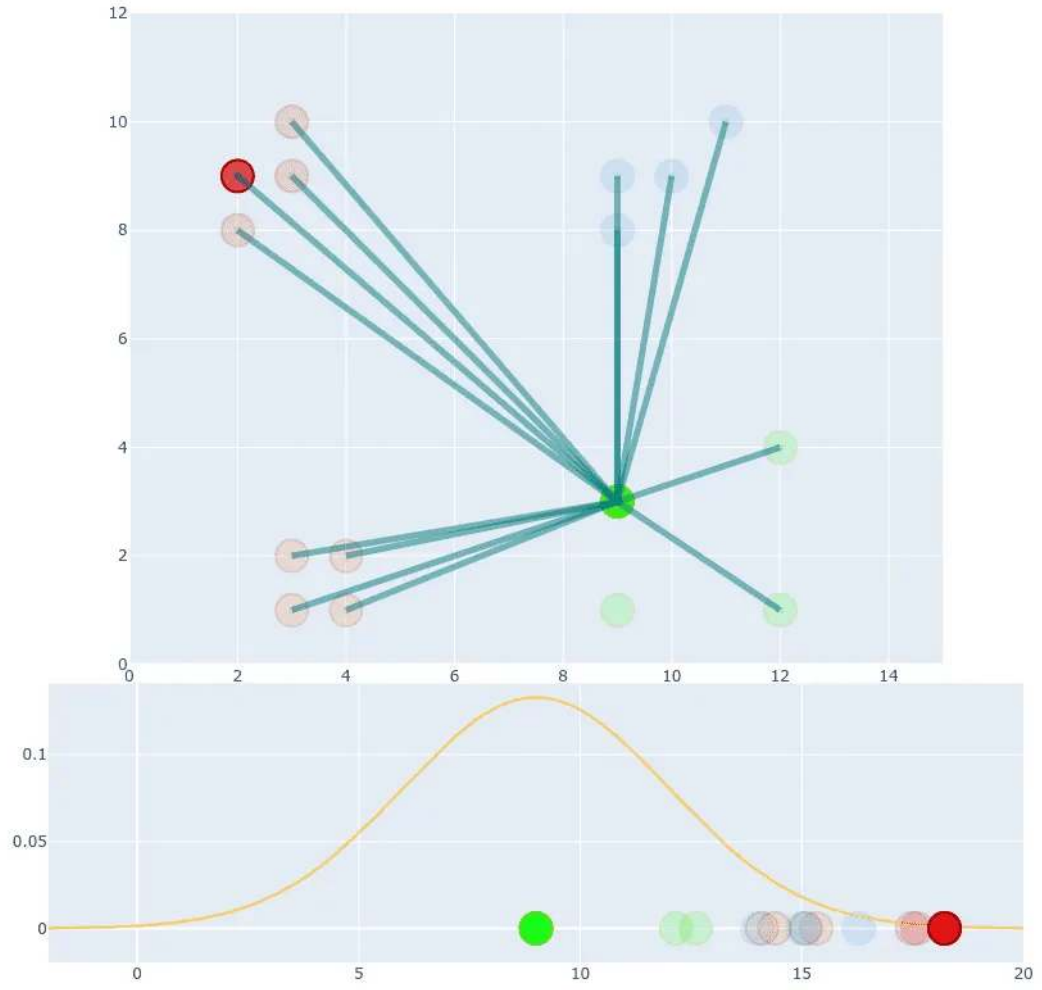
From Conditional to Symmetric Probabilities

The conditional probabilities are not symmetric, i.e., $p_{j|i} \neq p_{i|j}$ because the Gaussian around $\mathbf{x}_i$ may have a different width from the Gaussian around $\mathbf{x}_j$.

t-SNE converts them into symmetric pairwise similarities:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

p_{ij} represents the similarity between points i and j in the original space.



Step 2: Create a Low-Dimensional Map

Now t-SNE creates one low-dimensional point for each original point:

$$\mathbf{x}_i \in \mathbb{R}^d \quad \longrightarrow \quad \mathbf{z}_i \in \mathbb{R}^2$$

Initially, the $\mathbf{z}_i$ points are usually placed randomly.

Then t-SNE asks:

Do the neighborhoods in the low-dimensional map match the neighborhoods in the original space?

To answer this, it computes another probability distribution, q_{ij}

Similarities in the Low-Dimensional Space

In the low-dimensional space, t-SNE defines similarities as a Student-t distribution with one degree of freedom:

$$q_{ij} = \frac{\left(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{z}_k - \mathbf{z}_l\|^2\right)^{-1}}$$

where:

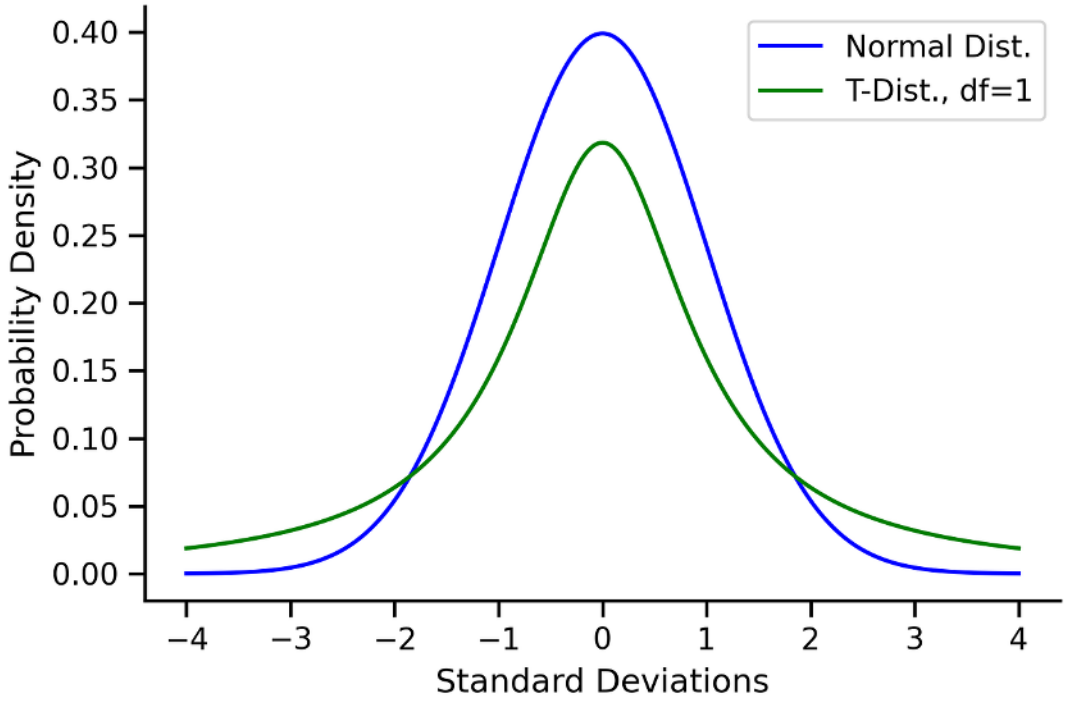
- $\mathbf{z}_i$ and $\mathbf{z}_j$ are points in the 2D map
- q_{ij} is their similarity in the 2D map
- nearby points get high q_{ij}
- far-away points get low q_{ij}
- also $q_{ii} = 0$

Why Student-t Instead of Gaussian?

- The Student-t distribution has heavier tails than a Gaussian.
- A heavy-tailed distribution allows moderately distant points in 2D to still have non-negligible probability.

This helps reduce the **crowding problem**:

- many high-dimensional points cannot all fit near each other in 2D
- a Gaussian in 2D would push too many points into a crowded center
- the Student-t allows dissimilar points to spread farther apart



Step 3: Match the Two Distributions

At this point we have:

- Original-space similarities, p_{ij}
- Low-dimensional similarities, q_{ij}

The goal is:

$$q_{ij} \approx p_{ij}$$

- If two points are neighbors in the original space, they should also be neighbors in the 2D map.
- If two points are not neighbors in the original space, they do not need to be close in the 2D map.

KL Divergence Objective

t-SNE measures the mismatch between p_{ij} and q_{ij} using Kullback-Leibler divergence:

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- The cost is small when $q_{ij} \approx p_{ij}$
- The cost is large when:
  - p_{ij} is large in the original space
  - but q_{ij} is small in the 2D map

So t-SNE penalizes separating points that were neighbors in the original space*.

The 2D coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_N$ are the parameters that t-SNE optimizes by gradient descent to minimize C.

*t-SNE is more tolerant of placing some originally distant points close together.

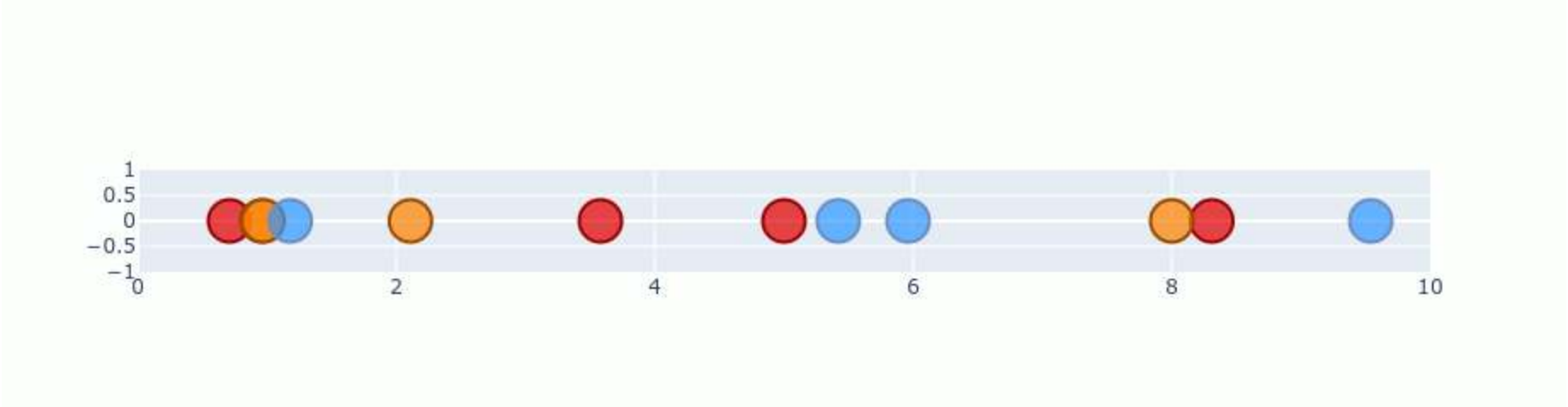
Attraction and Repulsion

If $p_{ij} > q_{ij}$:

- points i and j are more similar in the original space than in the map
- they should be pulled closer

If $p_{ij} < q_{ij}$:

- points i and j are too close in the map compared to the original space
- they should be pushed apart



t-SNE Summary

Given high-dimensional points:

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$$

t-SNE proceeds as follows:

1. Compute pairwise similarities p_{ij} in the original space
2. Initialize low-dimensional points $\mathbf{z}_i \in \mathbb{R}^2$
3. Compute pairwise similarities q_{ij} in the low-dimensional space
4. Minimize the KL divergence through gradient descent
5. Use the final $\mathbf{z}_i$ as the 2D visualization

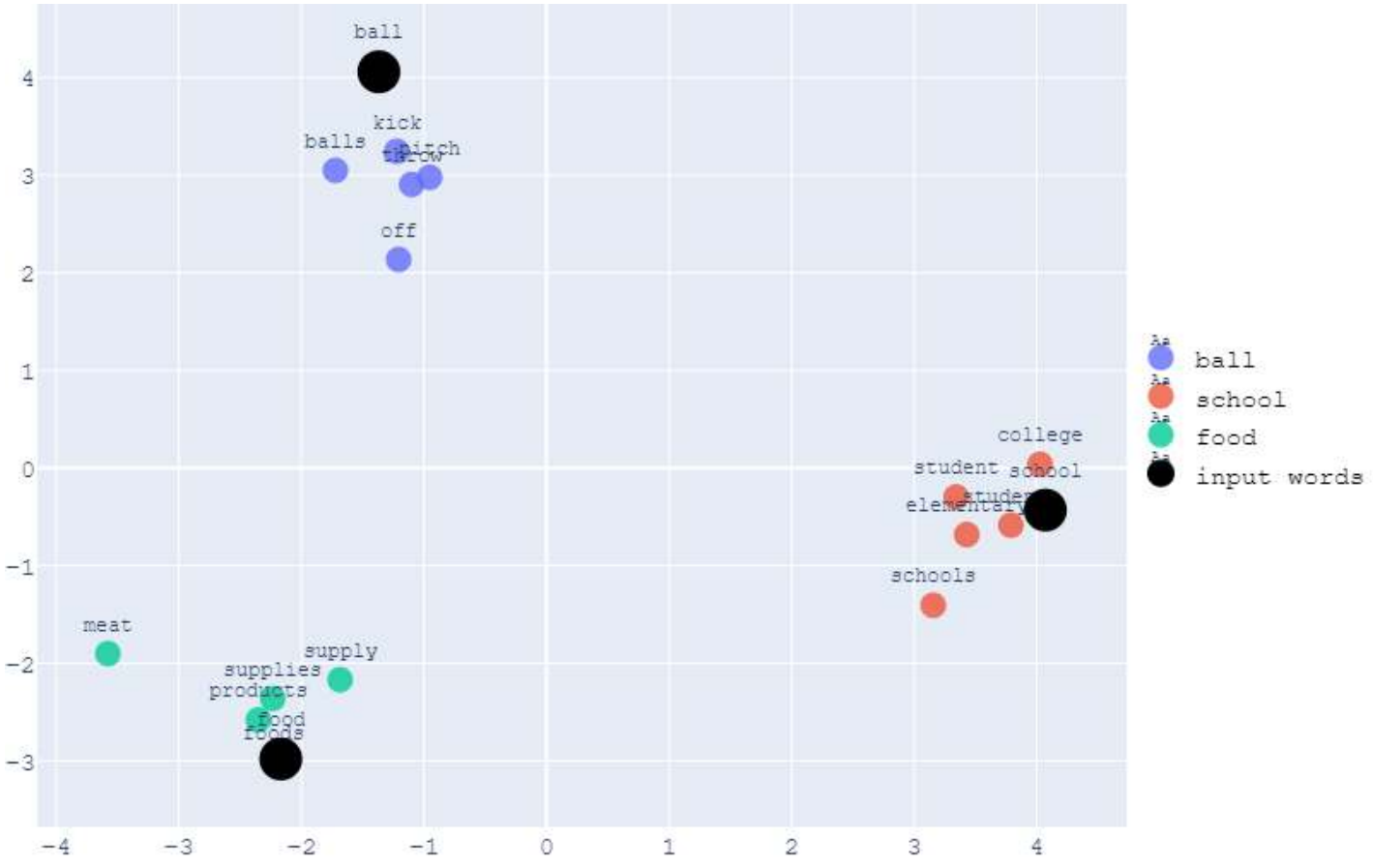
What t-SNE Preserves

t-SNE mainly preserves **local neighborhoods**.

- words with similar contexts may form groups
- medical codes with similar usage may appear close
- patients with similar representations may cluster

But

- t-SNE does not necessarily preserve global geometry.
- Distances between far-away clusters should be interpreted carefully.



Important Practical Choices

When using t-SNE, the result depends on several choices:

- perplexity
- learning rate
- number of iterations
- initialization
- random seed
- input preprocessing
- distance metric

A common workflow is:

1. standardize or normalize features if needed
2. optionally reduce first with PCA
3. run t-SNE with several perplexities
4. check whether visible patterns are stable
5. avoid overinterpreting one plot

Perplexity in Practice

A reasonable strategy for determining perplexity is to try multiple values, for example:

$$5, \quad 10, \quad 30, \quad 50$$

- Small perplexity values emphasize very local patterns.
- Large perplexity values emphasize broader neighborhoods.
- Stable structures across multiple perplexities are more trustworthy.
- Structures that appear only for one hyperparameter setting should be treated cautiously.

t-SNE is exploratory: always inspect sensitivity to hyperparameters.

Embeddings and Representation Learning

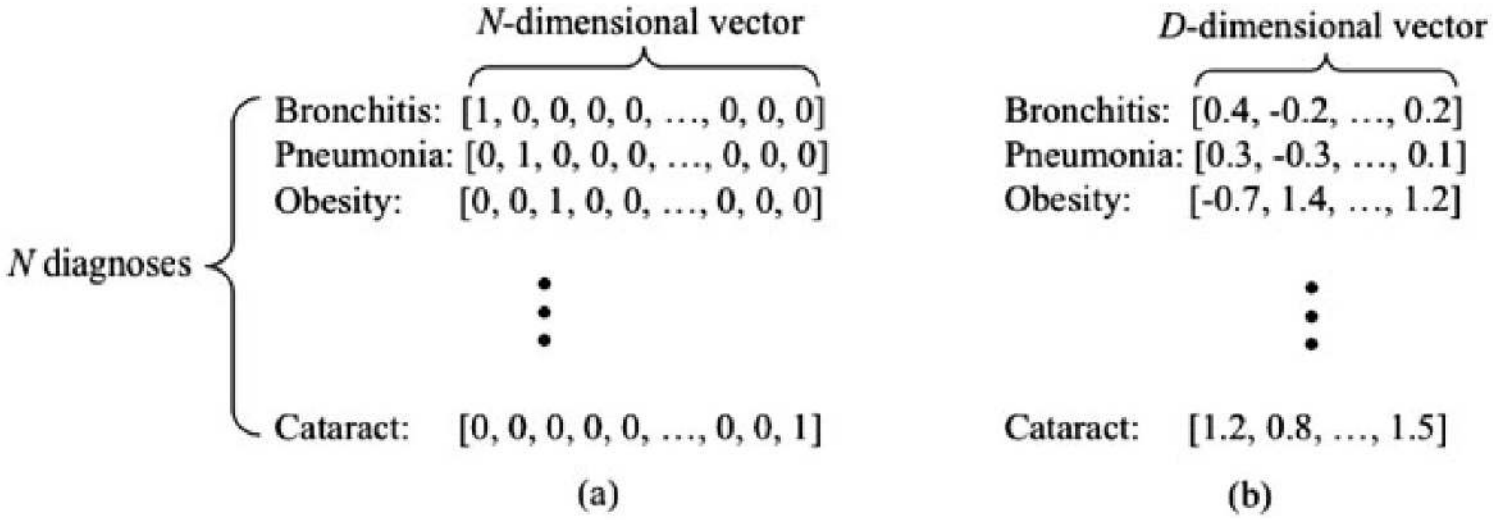
Healthcare applications

From EHR Events to Medical Embeddings

Word2Vec can be applied to longitudinal EHR data by treating medical codes like words in a sequence. In the Sutter-PAMF heart failure prediction study, the EHR data came from primary care patients between 2000 and 2013.

The authors used diagnosis codes, medication codes, procedure codes. If several codes occurred during the same visit, they were randomly ordered within the same timestamp.

The final vocabulary contained 38,599 unique medical concepts. Each concept was represented using a 100-dimensional embedding vector.



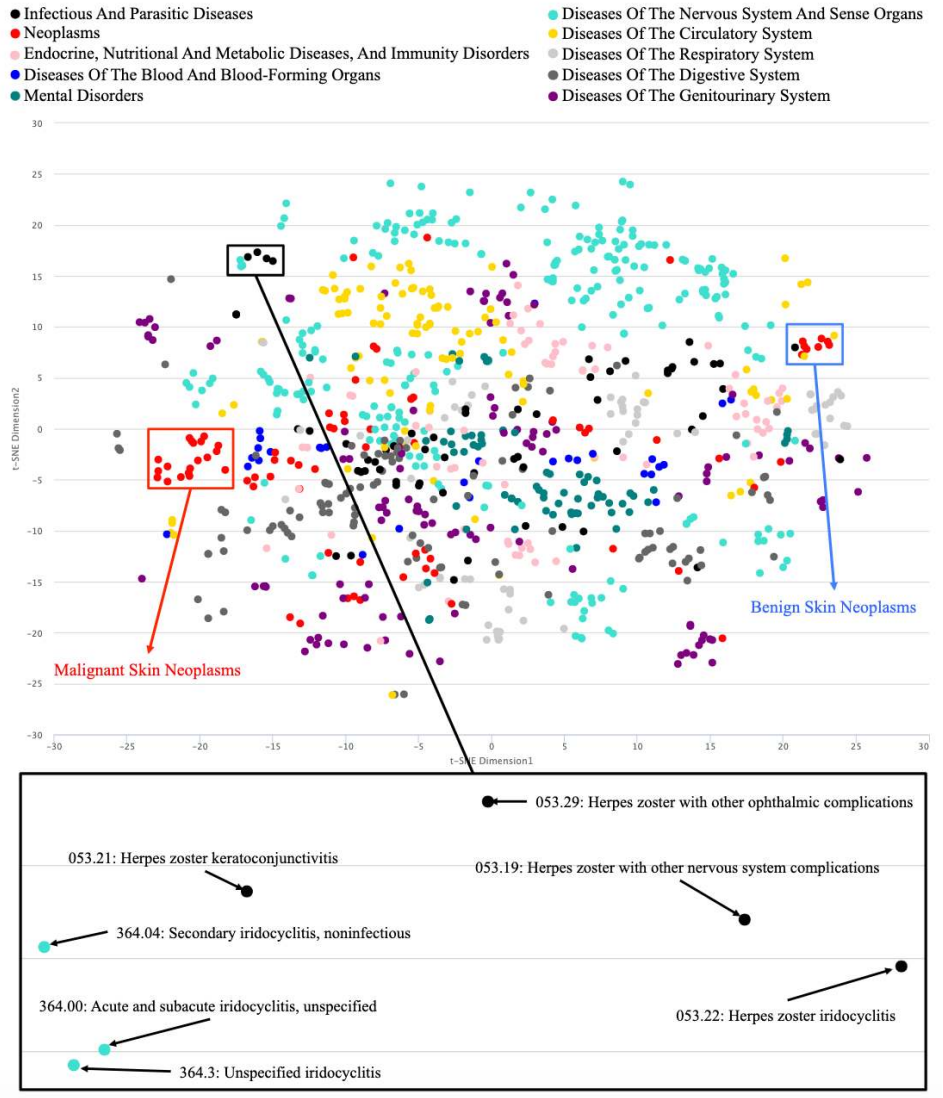
What Word2Vec Learns from EHR

Word2Vec learns medical concept vectors from co-occurrence patterns in patient histories.

Concepts that appear in similar clinical contexts are placed close together in the embedding space, e.g.:

- diagnosis codes from similar disease categories tend to cluster together
- clinically different diagnoses from the same broad category can still be separated
- diagnoses from different categories can be close if they often co-occur

This means the embedding space reflects clinical usage patterns.



Cross-Type Medical Similarity

	Diagnoses	Medications	Procedures
Acute upper respiratory infections (465.9)	-Bronchitis, not specified as acute or chronic (490) -Cough (786.2) -Acute sinusitis, unspecified (461.9) -Acute bronchitis (466.0) -Acute pharyngitis (462)	-Azithromycin 250 mg po tabs -Promethazine-Codeine 6.25-10 mg/5ml po syrp -Amoxicillin 500 mg po caps -Fluticasone Propionate 50 mcg/act na susp -Flonase 50 mcg/act na susp	-Pulse oximetry single -Serv prov during reg sched eve/wkend/hol hrs -Chest PA & lateral -Gyn cytology (pap) pa -Influenza vac (flu clinic only) 3+yo pa
Diabetes mellitus (250.02)	-Diabetes mellitus (250.00) -Mixed hyperlipidemia (272.2) -Other abnormal glucose (790.29) -Obesity, unspecified (278.00) -Pure hypercholesterolemia (272.0)	-Metformin hcl 500 mg po tabs -Metformin hcl 1000 mg po tabs -Glucose blood vi strp -Lisinopril 10 mg po tabs -Lisinopril 20 mg po tabs	-Diabetic eye exam (no bill) -Diabetes education, int -Ophthalmology, int -Diabetic foot exam (no bill) -Influenza vac 3+yr (v04.81) im
Edema (782.3)	-Anemia, unspecified (285.9) -Congestive heart failure, unspecified (428.0) -Unspecified essential hypertension (401.9) -Atrial fibrillation (427.31) -Chronic kidney disease, Stage III (moderate) (585.3)	-Furosemide 20 mg po tabs -Hydrochlorothiazide 25 mg po tabs -Hydrocodone-Acetaminophen 5-500 mg po tabs -Cephalexin 500 mg po caps -Furosemide 40 mg po tabs	-Debridement of nails, 6 or more -OV est pt min serv -EKG -ECG and interpretation -Chest PA & lateral
Tear film insufficiency, unspecified (375.15)	-Blepharitis, unspecified (373.00) -Senile cataract, unspecified (366.10) -Presbyopia (367.4) -Preglaucoma, unspecified (365.00) -Other chronic allergic conjunctivitis (372.14)	-Glasses -Erythromycin 5 mg/gm op oint -Patanol 0.1 % op soln	-Refraction -Visual field exam extended -Visual field exam limited -Referral to ophthalmology, int -Ophthalmology, int
Benign essential hypertension (401.1)	-Hyperlipidemia (272.4) -Essential hypertension (401.9) -Pure hypercholesterolemia (272.0) -Mixed hyperlipidemia (272.2) -Diabetes mellitus (250.00)	-Hydrochlorothiazide 25 mg po tabs -Atenonol 50 mg po tabs -Lisinopril 10 mg po tabs -Lisinopril 40 mg po tabs -Lisinopril 20 mg po tabs	-ECG and interpretation -Influenza vac 3+yr im -Immun admin im/sq/id/perc 1st vac only -GI, int -OV est pt lev 3

A key advantage of Word2Vec is that diagnoses, medications, and procedures are mapped into the same vector space.

This allows the model to find clinically meaningful neighbors across different code types.

Heart Failure Prediction with Word2Vec

The learned medical concept embeddings were used to represent patients.

- A patient vector was obtained by summing the Word2Vec embeddings of the medical codes in the patient history.
- These patient representations were then used to predict heart failure.
- Compared with one-hot encoding, Word2Vec-based representations improved prediction performance across several models.
- The improvement was especially strong for KNN, because KNN depends directly on distances between patient representations.

However standard Word2Vec assumes that tokens have a meaningful order but:

- A patient history is ordered over time, but inside a single visit, several medical codes are often recorded together without a meaningful internal order.
- This motivates models that explicitly represent the hierarchical structure of data.

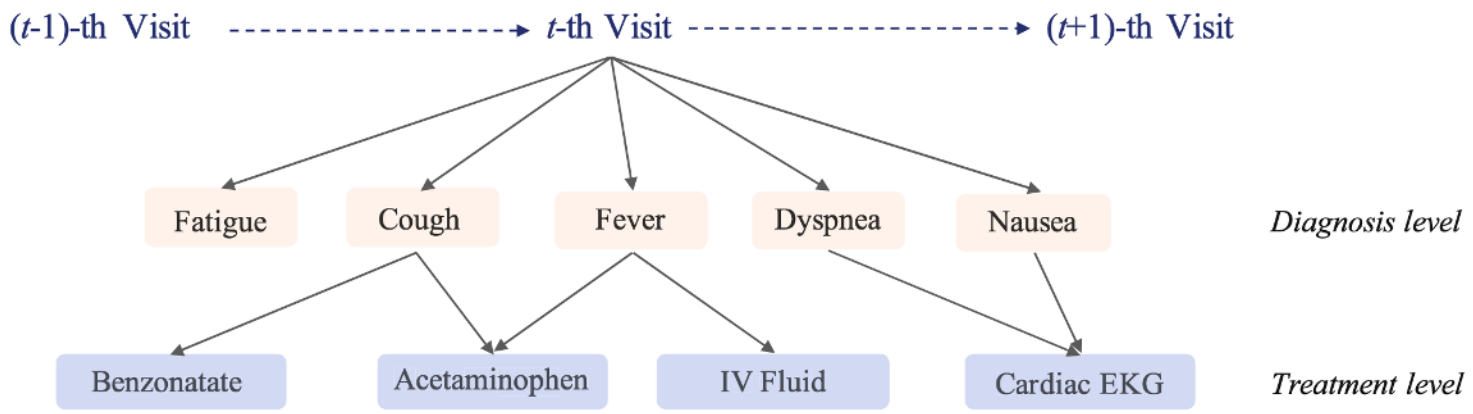
Med2Vec: Two-Level EHR Embedding

Med2Vec extends Word2Vec by modeling two levels of EHR structure:

1. the visit level
2. the code level

A patient record is represented as a sequence of visits. The goal is given a visit, predict what medical codes appeared in nearby visits.

- Each visit contains a set of medical codes, such as diagnoses, procedures, and medications.
- Unlike Word2Vec, Med2Vec does not assume a strict order between codes inside the same visit.

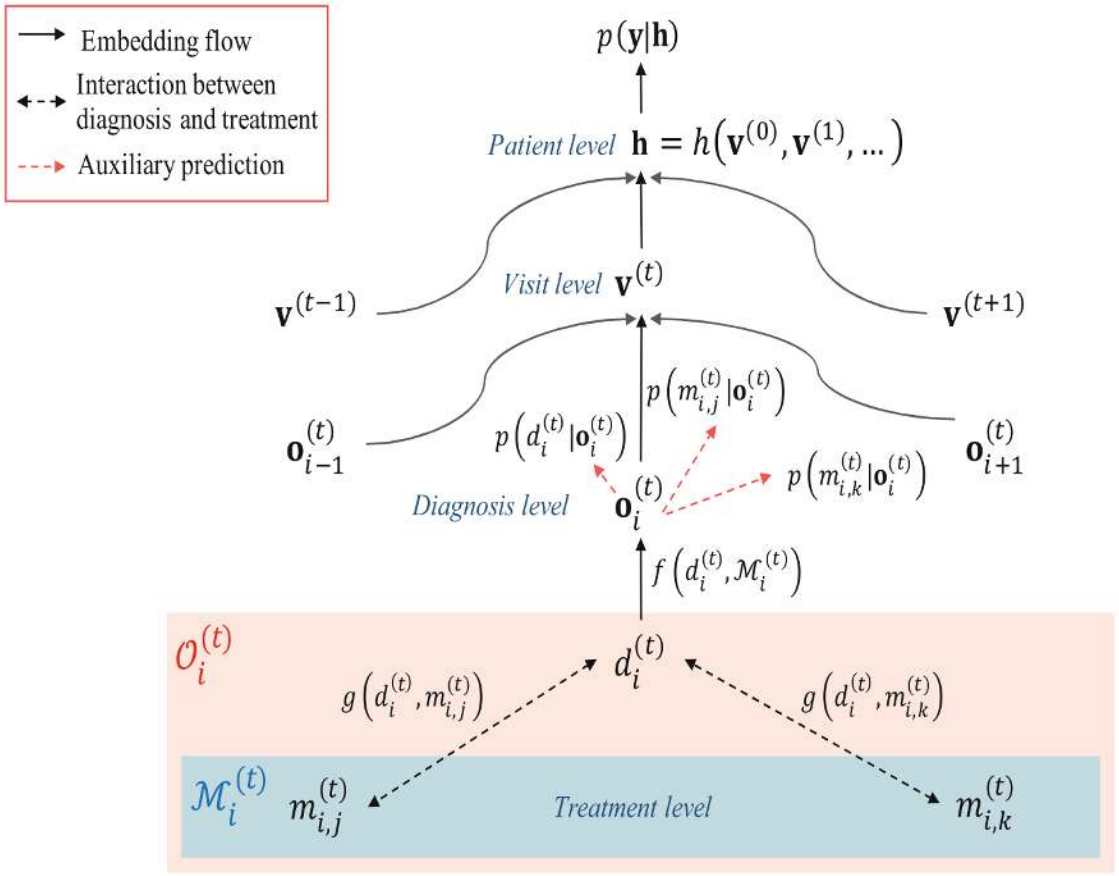


MIME: Modeling Internal EHR Structure

MIME adds an even more detailed clinical structure. It models the relationships between diagnoses and their associated treatments inside each visit. The sequence of visit embeddings is then used to form a patient representation $\mathbf{h}$.

- At the treatment level, MIME models the interaction between a diagnosis and a treatment.
- At the diagnosis level, these interactions are combined into a diagnosis-object embedding:
- At the visit level, diagnosis-object embeddings are aggregated:

In heart failure prediction, MIME outperformed baseline models.



References

- Introduction to Deep Learning for Healthcare, Chapter 5
- https://medium.com/data-science/t-sne-clearly-explained-d84c537f53a