Information Retrieval

Amin Nazari

Spring 2025

Chapter 5

Scoring and Space vector models

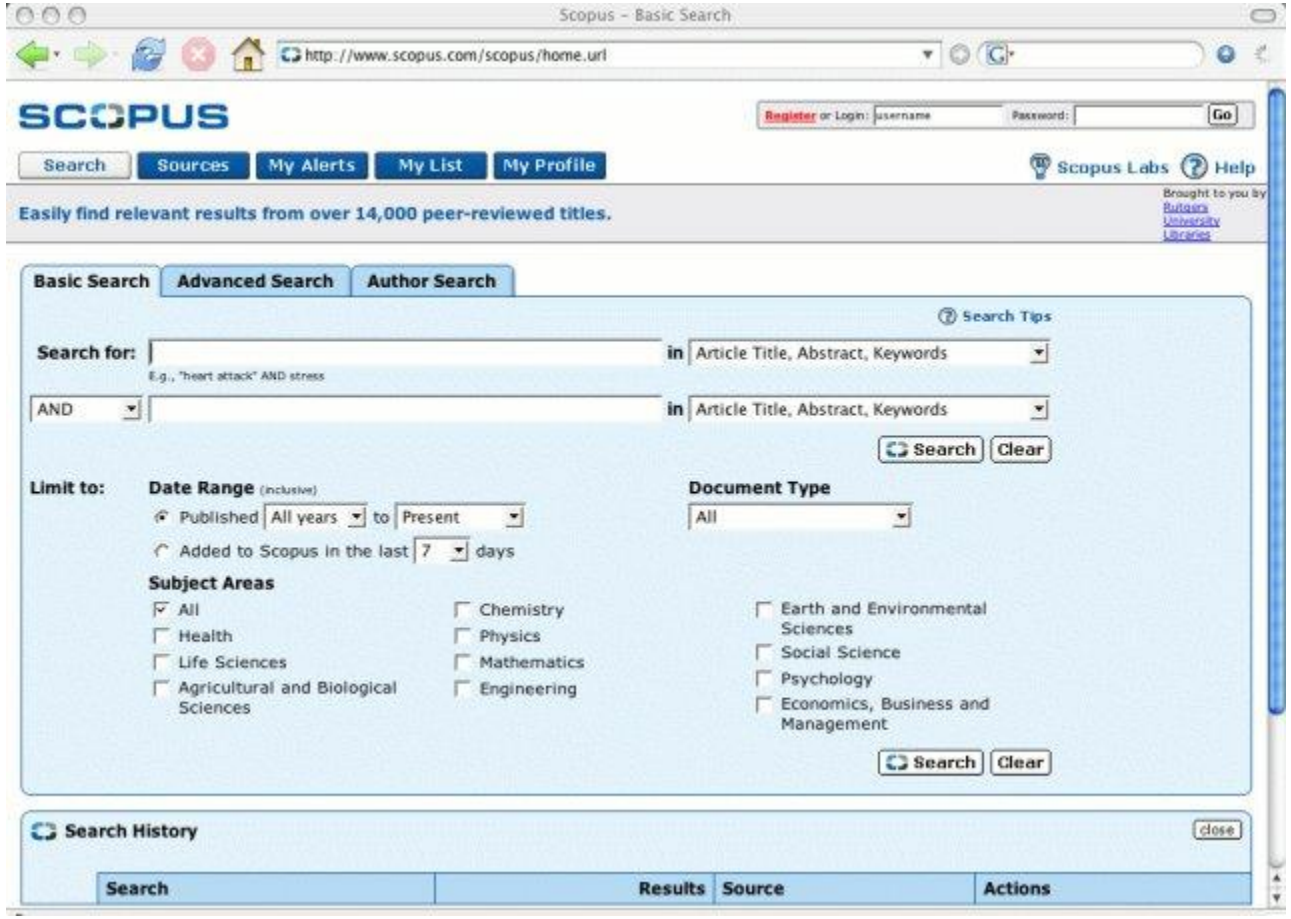
Outline

- Why ranked retrieval?
- Term frequency
- tf-idf weighting
- The vector space model
- Word2Vec

Zone Information Retrieval

- **Zone Information Retrieval** is a technique used in search engines and document retrieval systems where different parts (or "zones") of a document are assigned different weights or importance levels during indexing and retrieval.
  - **Zones** = Document sections with distinct semantic roles (e.g., title, abstract, body, headings, metadata).
  - **Weighting** = Some zones (like titles) are considered more important than others (e.g., footnotes).
  - **Scoring** = The search engine computes relevance scores by combining term matches across zones.

Zone


Scopus – Basic Search
http://www.scopus.com/scopus/home.url
SCOPUS
Register or Login: username Password: Go
Search Sources My Alerts My List My Profile
Scopus Labs Help
Easily find relevant results from over 14,000 peer-reviewed titles.
Brought to you by Rutgers University Libraries
Basic Search Advanced Search Author Search
Search Tips
Search for: in Article Title, Abstract, Keywords
E.g., "heart attack" AND stress
AND in Article Title, Abstract, Keywords
Search Clear
Limit to:
Date Range (inclusive)
Published All years to Present
Added to Scopus in the last 7 days
Document Type
All
Subject Areas
All
Health
Life Sciences
Agricultural and Biological Sciences
Chemistry
Physics
Mathematics
Engineering
Earth and Environmental Sciences
Social Science
Psychology
Economics, Business and Management
Search Clear
Search History
close
Search Results Source Actions

Weighting and Scoring

- Static weights
- Learning weights
- $S = \sum_{i=1}^{Z} w_i * z_i$

Scoring as the basis of ranked retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- Assign a score to each query-document pair, say in [0, 1].
- This score measures how well document and query “match”.

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Binary Retrieval Scoring(Jaccard coefficient)

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|} \qquad (A \neq \emptyset \text{ or } B \neq \emptyset)$$

- JACCARD (A, A) = 1
- JACCARD (A, B) = 0 if A ∩ B = 0
- A and B don’t have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- Query: “March ”
- Document “Caesar died in March”
- JACCARD(q, d) = 1/4

What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- We need a more sophisticated way of normalizing for the length of a document.

$$|A \cap B| / \sqrt{|A \cup B|}$$

Bag of words model

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

- Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Dot product scoring

- Score = Query(binary_vector) @ Doc(wordconts_vector)

What’s wrong with BoW?

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection
- A document containing this term is very likely to be relevant.
- We want high weights for rare terms.

TF

- **Term Frequency (TF):** Measures how often a word appears in a document. A higher frequency suggests greater importance. If a term appears frequently in a document, it is likely relevant to the document's content.

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

- **Limitations of TF Alone:** TF does not account for the global importance of a term across the entire corpus. Common words like **"the"** or **"and"** may have high TF scores but are not meaningful in distinguishing documents.

IDF

- **Inverse Document Frequency (IDF):** Reduces the weight of common words across multiple documents while increasing the weight of rare words. If a term appears in fewer documents, it is more likely to be meaningful and specific.

$$IDF(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

- The logarithm is used to dampen the effect of very large or very small values, ensuring the IDF score scales appropriately.
It also helps balance the impact of terms that appear in extremely few or extremely many documents.

Limitations of IDF Alone: IDF does not consider how often a term appears within a specific document.
A term might be rare across the corpus (high IDF) but irrelevant in a specific document (low TF).

Examples for idf

- Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Quiz

Document 1: "I love machine learning"
Document 2: "I love deep learning"

$$\text{TF}(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

$$\text{IDF}(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

Quiz

	deep	learning	love	machine
Doc1	?	?	?	?
Doc2	?	?	?	?

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in R^{|V|}$.
- So we have a |V|-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

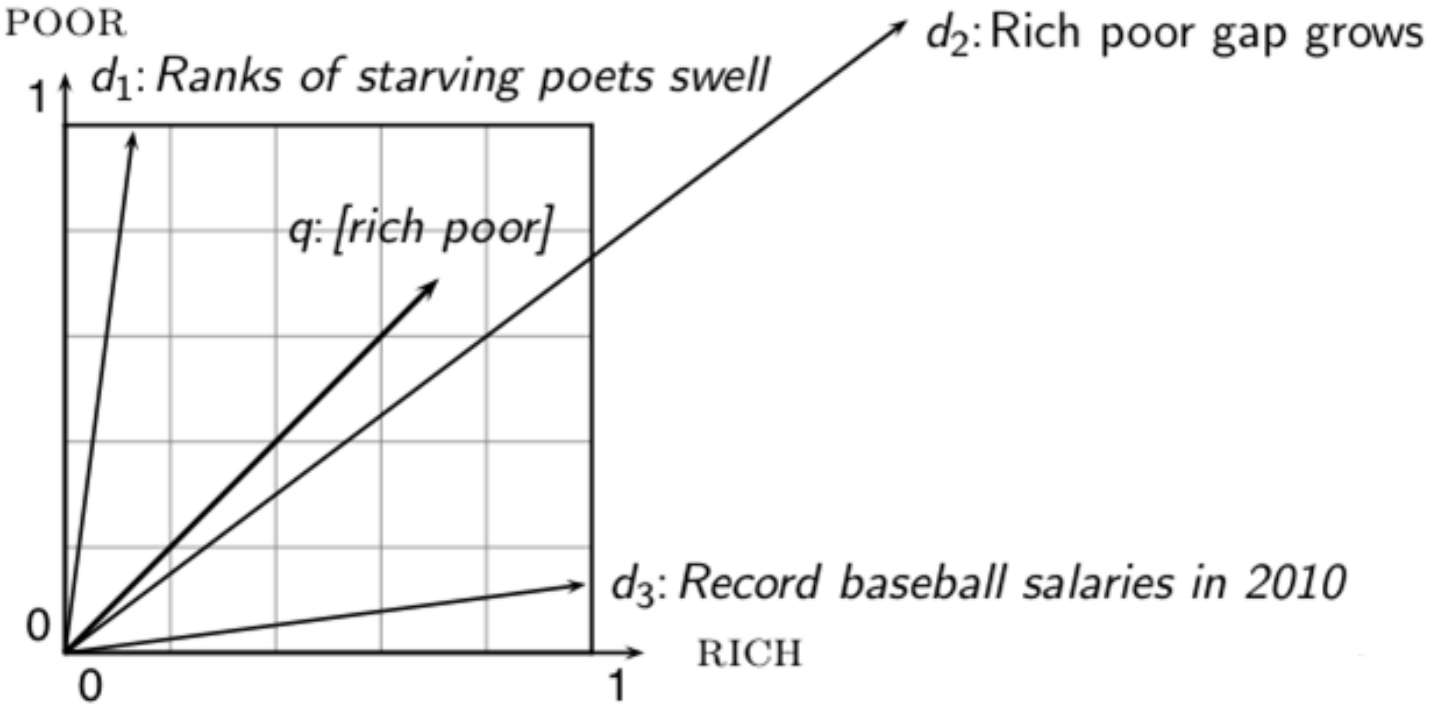
Queries as vectors

- Represent them as vectors in the high-dimensional space.
- Rank documents according to their proximity to the query
- Rank relevant documents higher than nonrelevant documents

Vector Space Similarity

- (-)Distance between two points.
- Euclidean distance is a bad idea. Because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea



Use angle instead of distance

- Rank documents according to angle with query.
- Thought experiment: take a document d and append it to itself. Call this document d'. d' is twice as long as d.
- "Semantically" d and d' have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity even though the Euclidean distance between the two documents can be quite large.

Cosine similarity

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Quiz: Cosine similarity

Document 1: "I love machine learning"
Document 2: "I like deep learning“

Query = “I love cats”

Quiz: Cosine similarity

Document 1: "I love machine learning"
Document 2: "I like deep learning“

Query = “I love cats”

Ranked retrieval in the vector space model

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top K (e.g., $K = 10$) to the user

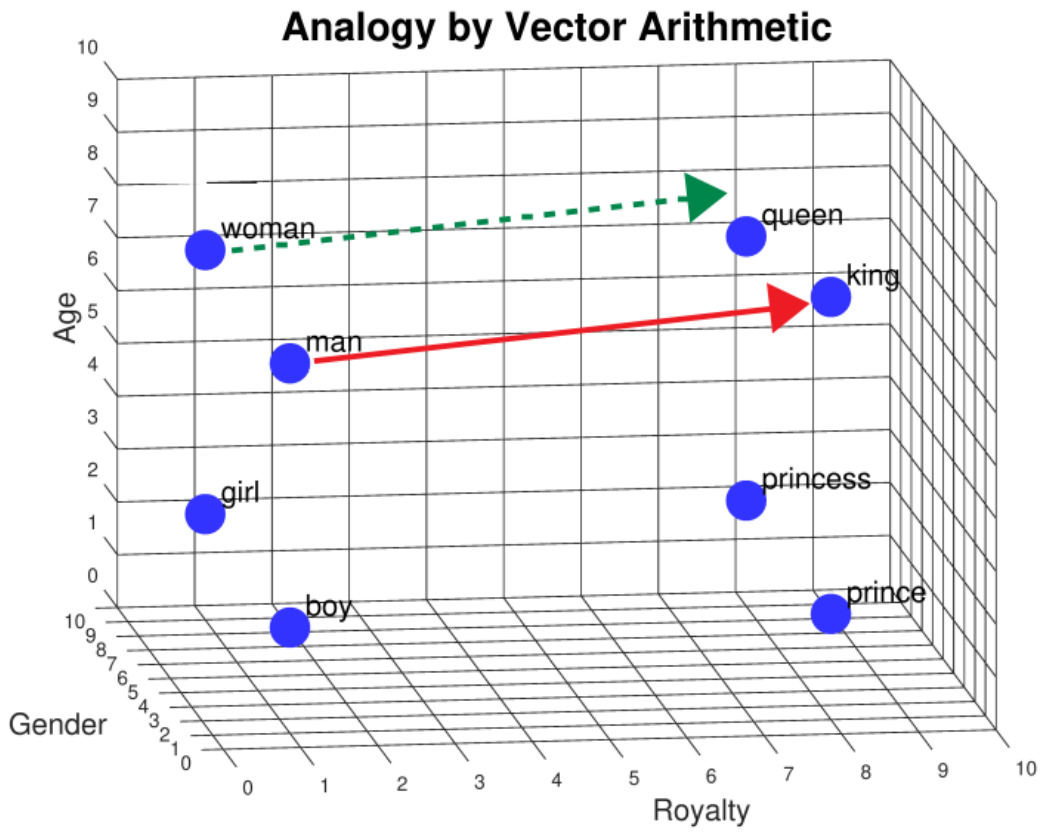
Which Formula Should You Use?

Library	TF Formula	IDF Formula	Normalization	Best For
Scikit-Learn	Raw count	`log(N/(1+n_t)) + 1`	L2 (default)	ML models
Gensim	Raw count	`log(N/n_t)`	None	Topic modeling
spaCy	Raw count	`log(N/n_t)`	Optional	NLP pipelines

What’s wrong with TFIDF?

- Semantic Ignorance: Example: "car" and "automobile" are considered unrelated.
- Out-of-Vocabulary (OOV) Problem: Cannot Handle New Words
- No Positional or Word Order Awareness: Ignores Word Sequence
- Sparse and High-Dimensional Vectors: Computationally Inefficient

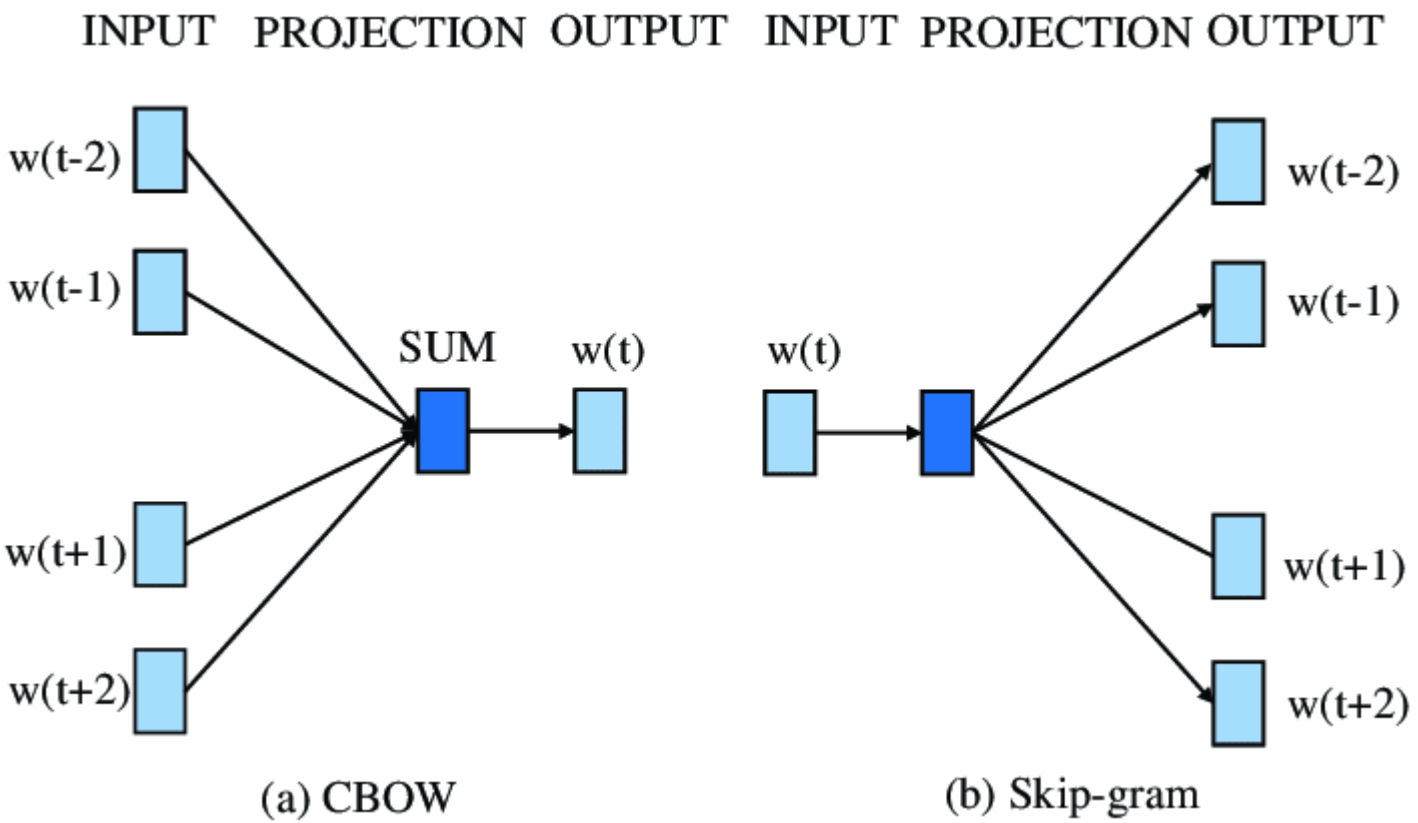
Word2Vec


Analogy by Vector Arithmetic
woman
queen
king
man
girl
princess
boy
prince
Age
Gender
Royalty

What is Word2Vec?

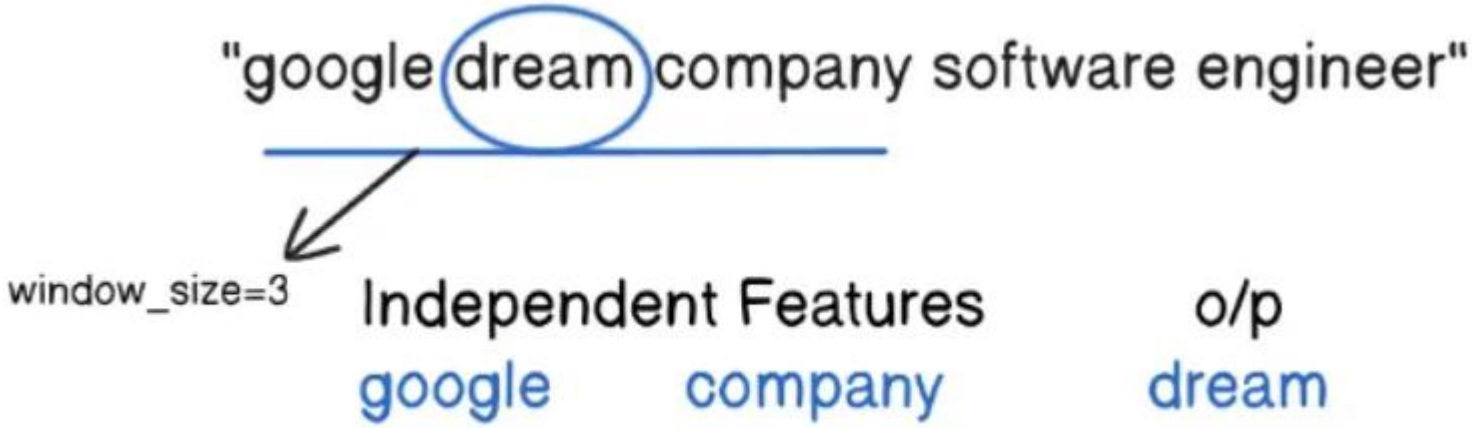
- Definition: A neural network-based algorithm for generating word embeddings.
- Purpose: Converts words into numerical vectors to capture meaning and relationships.
- Key Idea: "Words appearing in similar contexts have similar meanings."

Word2Vec


INPUT
PROJECTION
OUTPUT
INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
w(t+1)
w(t+2)
SUM
w(t)
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
(a) CBOW
(b) Skip-gram

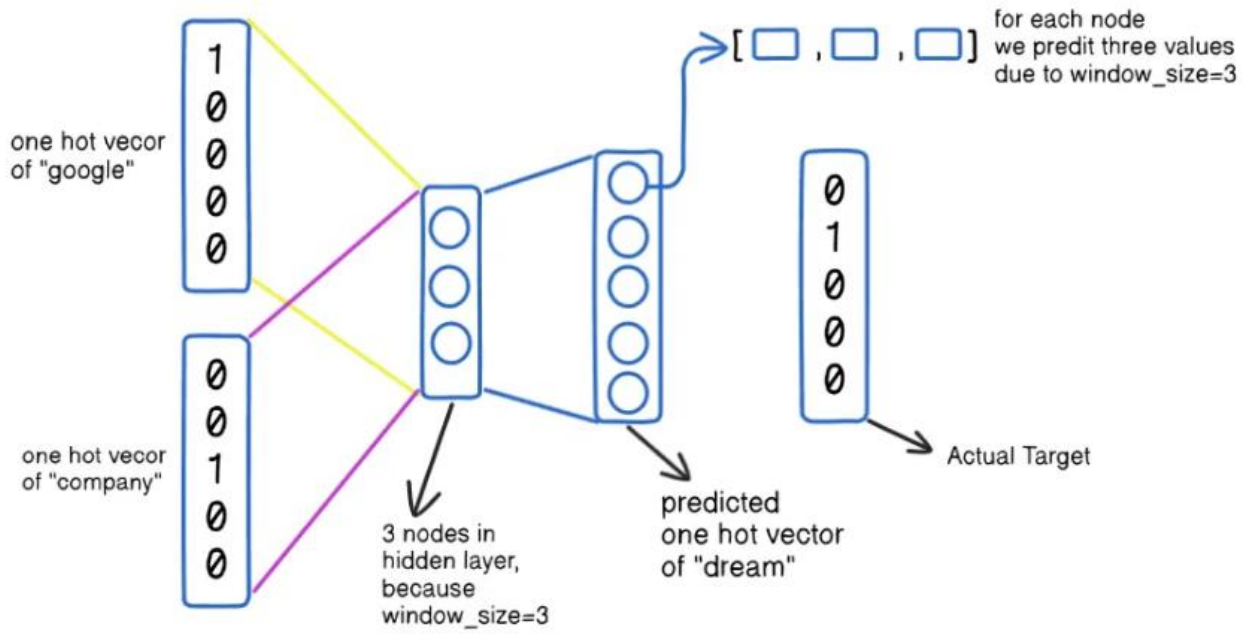
Word2Vec

- Continuous Bag-of-Words (CBOW): Predict the target word (center word) from the surrounding context words.



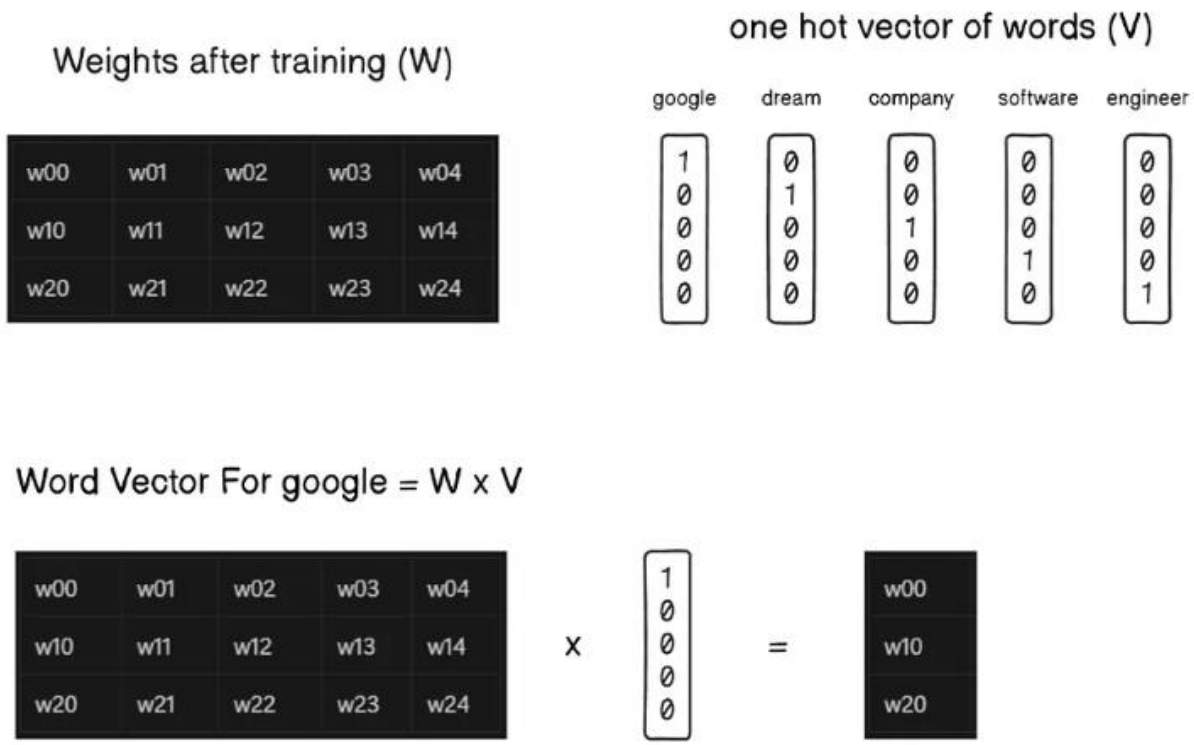
Word2Vec

1. Convert it into one hot encoding.
2. Create Dataset
3. Next step, we pass it to the Neural Network



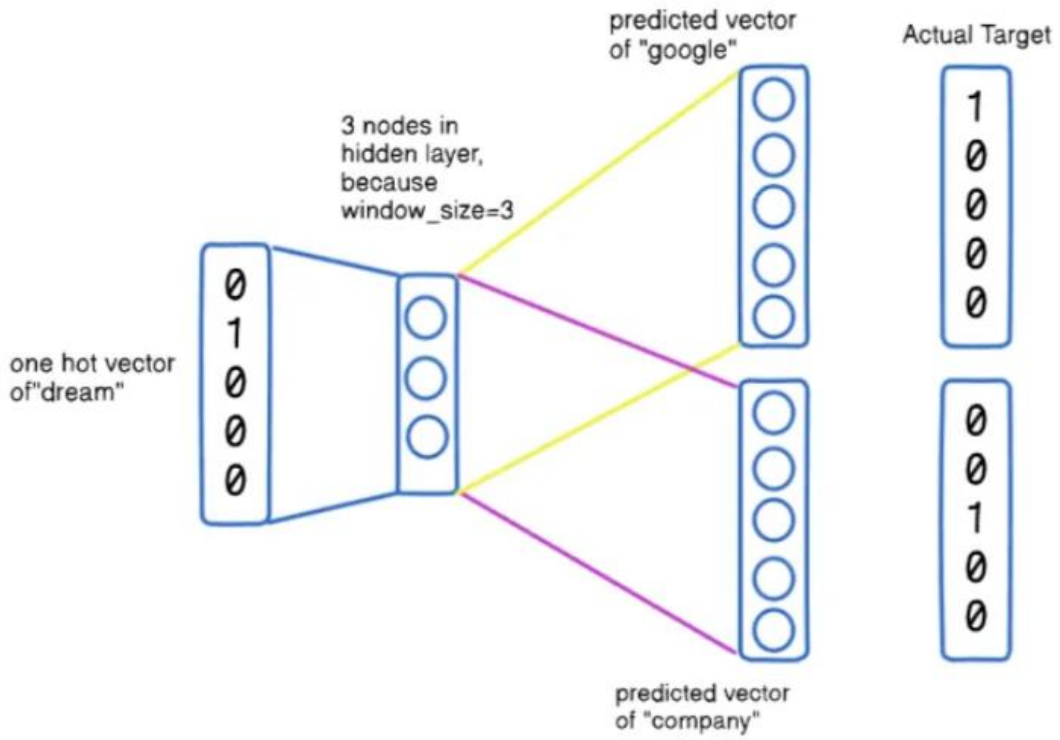
Word2Vec

1. Word Embeddings



Word2Vec

- Skip-Gram: Predict the surrounding context words given a target word.



Word2Vec (Apps)

- Semantic Similarity: Measuring similarity between words.
- Text Classification: Improving feature representation for classification tasks.
- Machine Translation: Enhancing translation quality by providing better word embeddings.
- Information Retrieval: Improving search results by understanding word semantics.
- Recommendation Systems: Enhancing recommendations by understanding user preferences and item descriptions.

Word2Vec (Limitations)

- Context Independence: It doesn’t consider the order of words or their syntactic roles.
- Out-of-Vocabulary Words: It cannot handle words that were not present in the training corpus.
- Fixed Embedding Size: All words are represented by fixed-size vectors, regardless of their frequency or importance.

Word2Vec (Limitations)

- Context Independence: It doesn’t consider the order of words or their syntactic roles.
- Out-of-Vocabulary Words: It cannot handle words that were not present in the training corpus.
- Fixed Embedding Size: All words are represented by fixed-size vectors, regardless of their frequency or importance.

Python

- BoW
- TF-IDF

HW

- Apply the BoW and TF-IDF and Word2Vec to your dataset.
- Save these matrix for the future works.