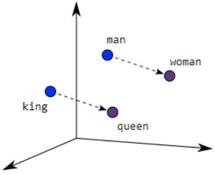
Vector Space Models & Word Embeddings

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

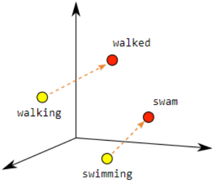
جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

LMH
Lady Margaret Hall

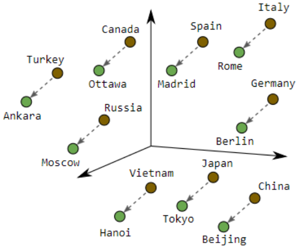
July 2, 2025



Male-Female



Verb Tense



Country-Capital

Table of Contents

Table of Contents (cont.)

Motivation

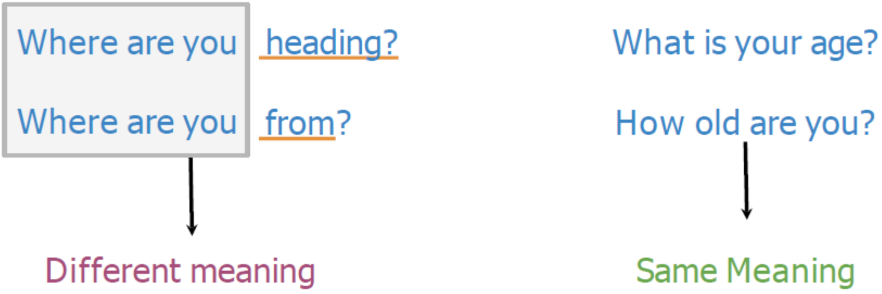
- Text is symbolic; machines require numerical representations to process it.
- Classical NLP struggled with sparse, high-dimensional vectors.
- Vector Space Models (VSMs) and Word Embeddings represent words in a continuous vector space.
- These models capture semantic meaning and relationships geometrically.
- Basis for modern NLP methods including Transformers.

Learning Outcomes

- Understand and implement basic Vector Space Models.
- Differentiate between word-by-word and word-by-document designs.
- Use similarity measures to compute semantic relatedness.
- Explain the rationale behind One-Hot vs. Dense word vectors.
- Understand and implement basic Word2Vec models (CBOW and Skip-gram).
- Appreciate the improvements of GloVe and fastText over earlier models.

Vector Space Model: **Introduction**

Why learn vector space models?



- Words and sentences can have different meanings depending on context.
- Vector space models help capture semantic similarity and differences.
- Useful for tasks like paraphrase detection, question answering, and information retrieval.

Fundamental

"You shall know a word by the company it keeps"
Firth, 1957

(Firth, J. R. 1957:11)

Vector Space Models (VSM)

- VSM represents words or documents as vectors in an n-dimensional space.
- Based on the **distributional hypothesis**: Words that occur in similar contexts have similar meanings.
- Forms the foundation for information retrieval, document classification, and word embeddings.

Vector Space Models – Formal Definition

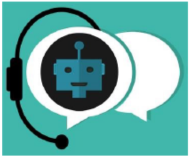
- A **term-document matrix**: Rows represent words (terms), columns represent documents (or vice versa).
- Each cell contains a value such as:
  - Term Frequency (TF)
  - TF-IDF (Term Frequency-Inverse Document Frequency)
  - Co-occurrence count
- The matrix is typically high-dimensional and sparse:
  - Size $= |\text{Vocabulary}| \times |\text{Documents}|$

Vector Space Model Applications

- You eat *cereal* from a *bowl* → Capturing semantic similarity (paraphrase understanding)
- You *buy* something and someone else *sells* it → Capturing relational meaning (analogies)

Information Extraction

Machine Translation



Chatbots

Word-by-Word vs Word-by-Document Designs

Idea

- Co-occurrence $\rightarrow$ Vector representation
- Relationships between words/documents

Word by Word Design

Number of times they ***occur together within a certain distance*** *k*

I like simple data

I prefer simple raw data

k=2

	simple	raw	like	I
data	2	1	1	0

n

Word by Document Design

Number of times a word *occurs within a certain category*

Corpus

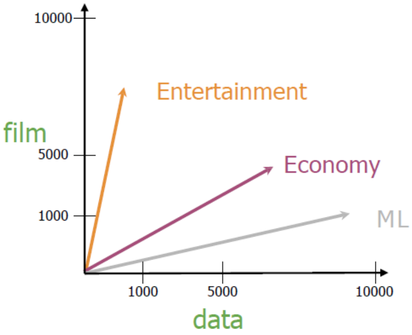
	Entertainment	Economy	Machine Learning
data	500	6620	9320
film	7000	4000	1000

- **Word-by-Word:**
  - Matrix built from co-occurrence counts between words.
  - Captures semantic similarity directly.
  - Better for word similarity tasks.
- **Word-by-Document:**
  - Matrix built from word frequencies in documents.
  - Useful for document classification and search.
  - Better for document-level tasks.

Tradeoffs: Choice depends on the task: word similarity vs. document analysis.

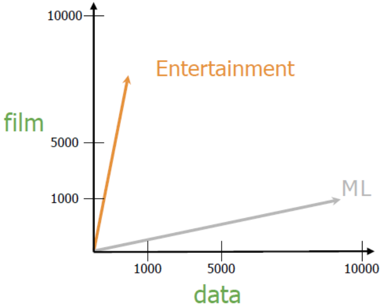
Similarity Measures

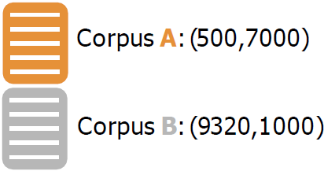
Vector Space



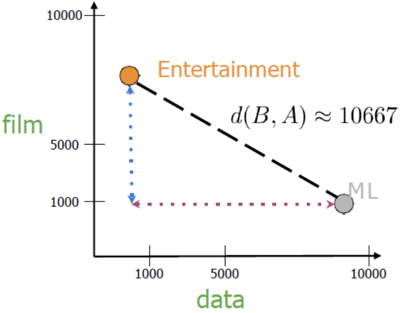
	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

Measures of "similarity:"
Angle
Distance





- Measures the straight-line distance between two points in space.
- Formula: $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- Sensitive to the scale (magnitude) of the vectors.



Corpus A: (500,7000)

Corpus B: (9320,1000)

$$d(B, A) = \sqrt{\underbrace{(B_1 - A_1)^2}_{} + \underbrace{(B_2 - A_2)^2}_{}}$$

$$c^2 = a^2 + b^2$$

$$d(B, A) = \sqrt{(-8820)^2 + (6000)^2}$$

Euclidean Distance for n-dimensional vectors

	data	boba ($\vec{w}$)	ice-cream ($\vec{v}$)
AI	6	0	1
drinks	0	4	6
food	0	6	8

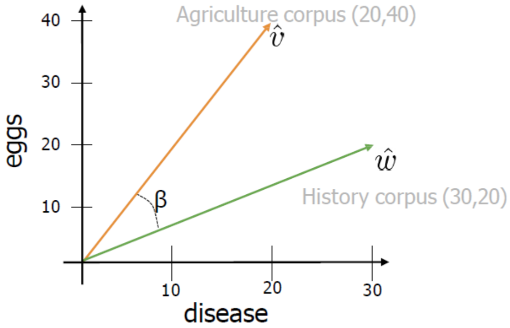
$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

$$d(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} \longrightarrow \text{Norm of } (\vec{v} \; \vec{w})$$

```python
# Create numpy vectors v and w
v = np.array([1, 6, 8])
w = np.array([0, 4, 6])
# Calculate the Euclidean distance d
d = np.linalg.norm(v-w)
# Print the result
print("The Euclidean distance between v and w is: ", d)
```

```
The Euclidean distance between v and w is: 3
```



$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

$$\boxed{\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}}$$

$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$

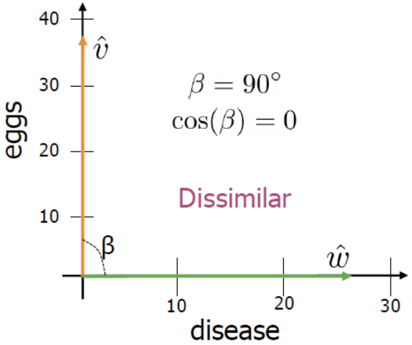
$$= 0.87$$

- Measures the cosine of the angle between two vectors.
- Formula: $\text{cosine}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$
- Ranges from -1 (opposite) to 1 (same direction).

Cosine Similarity (cont.)

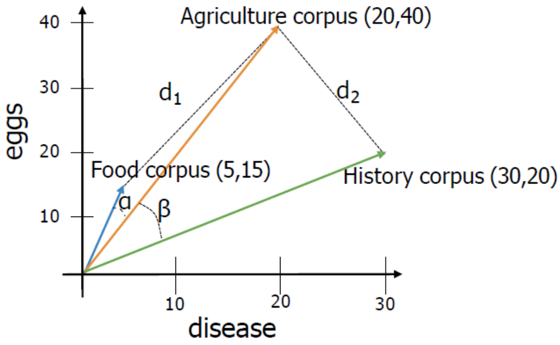
- Less sensitive to magnitude, focuses on orientation.

Cosine Similarity (cont.)



Euclidean distance vs Cosine similarity

- Cosine similarity when corpora are different sizes

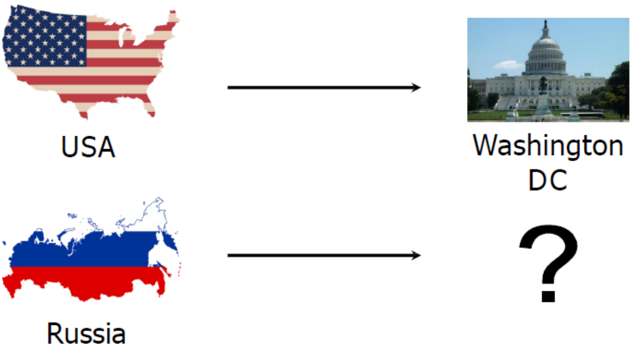


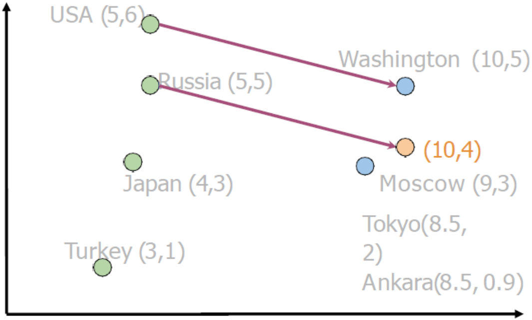
Euclidean distance: $d_2 < d_1$

Angles comparison: $\beta > \alpha$

The cosine of the angle between the vectors

Manipulating word vectors





Washington - USA = $\begin{bmatrix} 5 & -1 \end{bmatrix}$

Russia + $\begin{bmatrix} 5 & -1 \end{bmatrix}$ = $\begin{bmatrix} 10 & 4 \end{bmatrix}$

Moscow

[Mikolov et al, 2013, Distributed Representations of Words and Phrases and their Compositionality]

	$\overbrace{\qquad\qquad\qquad\qquad}^{d>2}$		
oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3

How can you visualize if your representation captures these relationships?

oil & gas

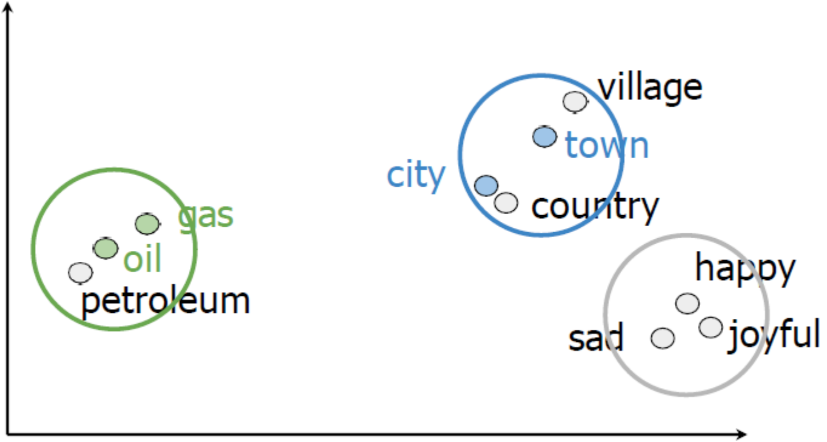
town & city

Visualization of word vectors (cont.)

	$\overbrace{\qquad\qquad}^{d>2}$		
oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3

$\xrightarrow{\text{PCA}}$

	$\overbrace{\qquad\qquad}^{d=2}$	
oil	2.30	21.2
gas	1.56	19.3
city	13.4	34.1
town	15.6	29.8


village
town
city
country
gas
oil
petroleum
happy
sad
joyful

Word Embedding: **One-Hot vs. Dense Vectors**

Integer Representation of Words

Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000

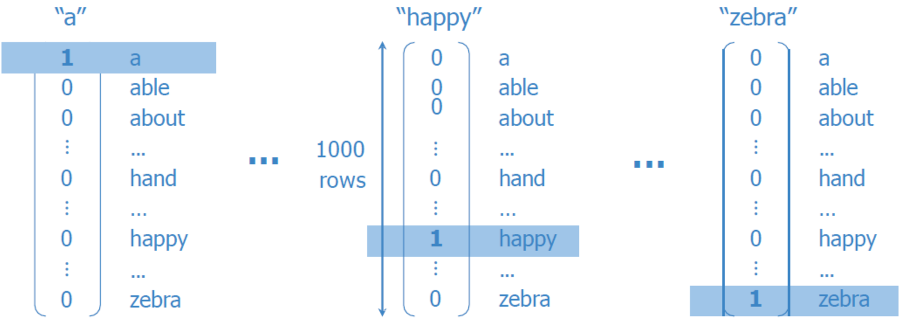
Integer Representation of Words (cont.)

\+ Simple

\- Ordering: little semantic sense

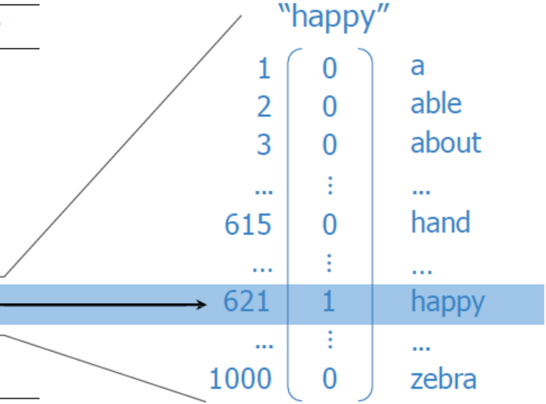
hand < happy < zebra
615 ?! 621 ?! 1000

One-hot vectors



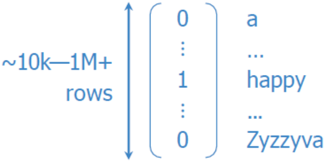
One-hot vectors (cont.)

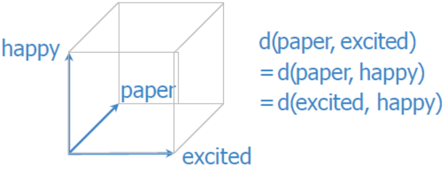
Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000



One-hot vectors (cont.)

- \+ Simple
- \+ No implied ordering
- \- Huge vectors
- \- No embedded meaning





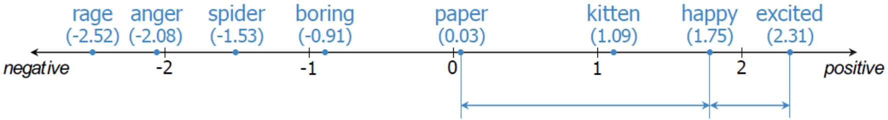
One-Hot Encoding

- Each word is represented as a unique vector.
- Vector has 1 at one position, 0 elsewhere.
- **Problems:**
  - High-dimensional and sparse.
  - No semantic meaning or similarity captured.

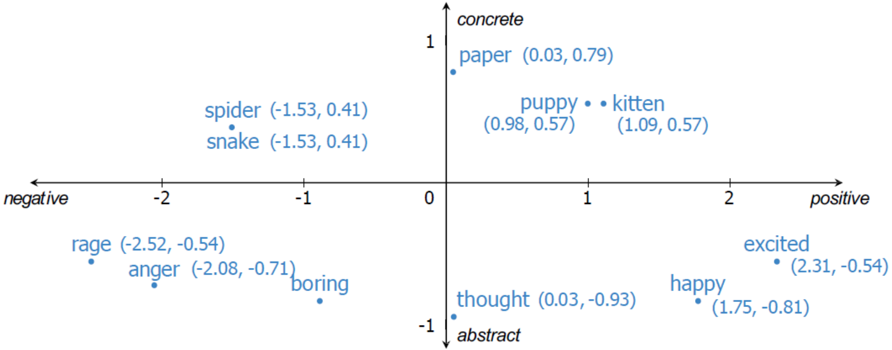
Dense Embeddings

- Vectors are learned from data.
- Low-dimensional (e.g., 100–300).
- Capture semantic relationships.
- Example: $\texttt{"king"} - \texttt{"man"} + \texttt{"woman"} \approx \texttt{"queen"}$

Meaning as vectors



Meaning as vectors (cont.)



Word embedding vectors

+ Low dimension
+ Embed meaning
  - e.g. semantic distance

    forest $\approx$ tree   forest $\not\approx$ ticket

  - e.g. analogies

    Paris:France :: Rome:?

"happy"

~100—~1000 rows

$$\begin{pmatrix} 0.123 \\ \vdots \\ -4.059 \\ \vdots \\ 1.891 \end{pmatrix}$$

Terminology

integers

word vectors

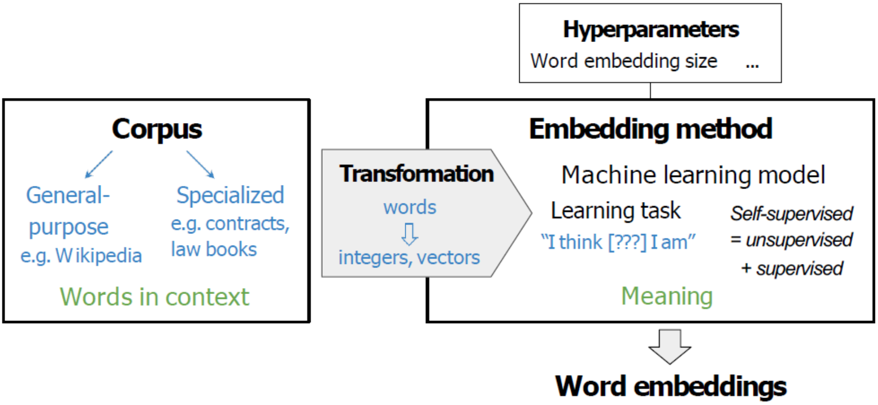
one-hot vectors

word embedding vectors

"word vectors"

word embeddings

Word embedding process



Embedding Method: **Word2Vec**

Word Embeddings – Intuition

- Dense vector representations of words learned from context.
- Encode syntactic and semantic meaning.
- Vectors reflect relationships like synonymy, analogy, etc.

Basic Word Embedding Methods

- **word2vec** (Google, 2013)
  - Continuous bag-of-words (CBOW)
  - Continuous skip-gram / Skip-gram with negative sampling (SGNS)
- **Global Vectors (GloVe)** (Stanford, 2014)
- **fastText** (Facebook, 2016)
  - Supports out-of-vocabulary (OOV) words

- Deep learning-based, contextual embeddings:
  - **BERT** (Google, 2018)
  - **ELMo** (Allen Institute for AI, 2018)
  - **GPT-2** (OpenAI, 2018)
- Tunable pre-trained models available

Word2Vec Overview

- Word2Vec learns word embeddings from large text corpora.
- Two main architectures:
  - Continuous Bag-of-Words (CBOW)
  - Skip-gram
- Both models use neural networks to predict words based on context.

- **Goal:** Predict target word from context.
- **Architecture:**
  - **Input:** Surrounding words (context)
  - **Output:** Target word
- Fast and accurate for frequent words.
- **Example:**
  - Context: "the ___ sat on the mat" $\rightarrow$ Predict "cat"

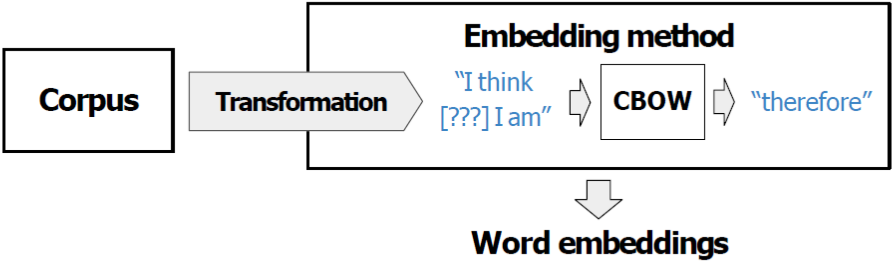


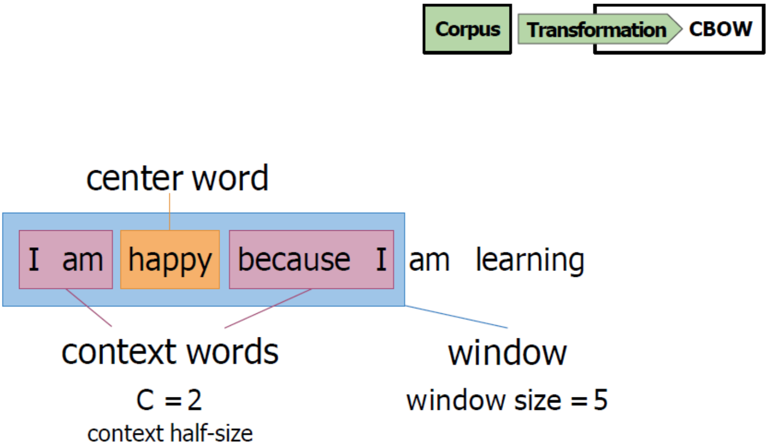
Figure 2: CBOW Architecture

Center word prediction: rationale

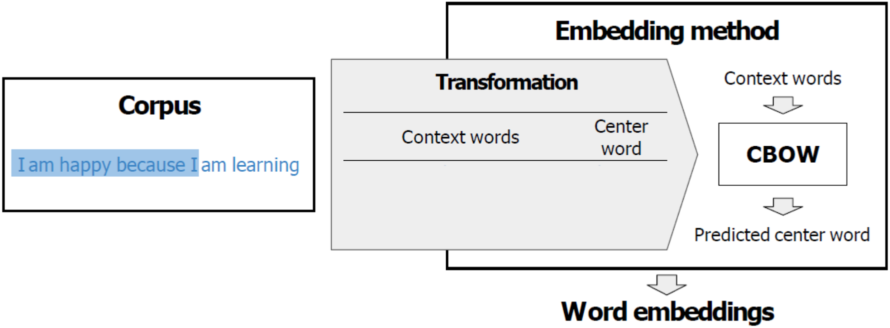
Corpus | Transformation | CBOW

The little _____?_____ is barking

dog
puppy
hound
terrier
...

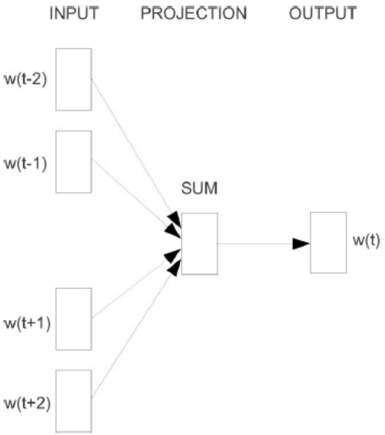

Corpus
Transformation
CBOW
center word
I am happy because I am learning
context words
C = 2
context half-size
window
window size = 5

From corpus to training



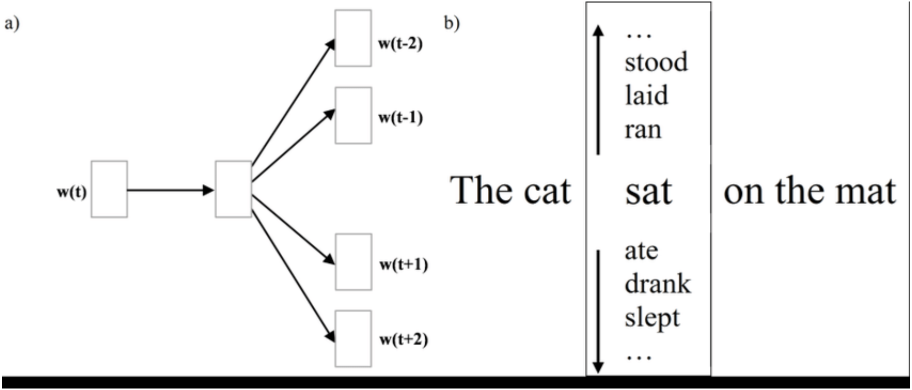
Word2Vec – Skip-Gram

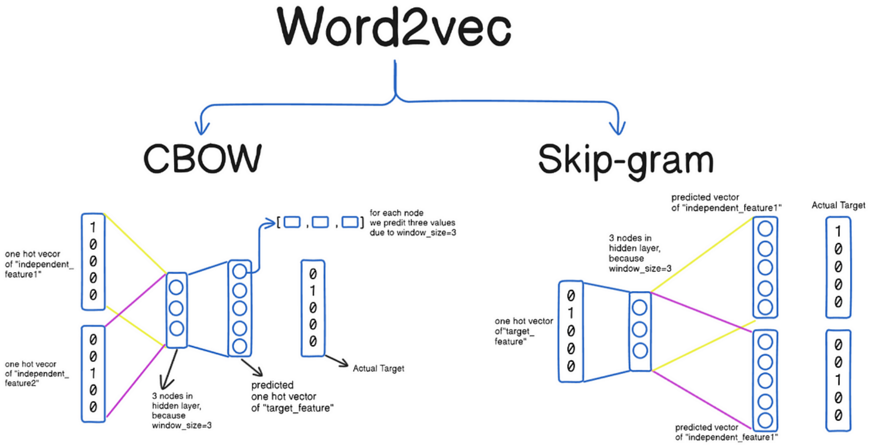
- **Goal:** Predict context words from a target word.
- **Architecture:**
  - **Input:** Target word
  - **Output:** Surrounding words (context)
- Better for rare words.
- **Example:**
  - Input: "cat" → Output: "the", "sat", "on", "the", "mat"



Source: Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). [Efficient Estimation of Word Representations in Vector Space]

Figure 3: Skip-Gram Architecture


a)
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
b)
...
stood
laid
ran
The cat
sat
on the mat
ate
drank
slept
...


Word2vec
CBOW
Skip-gram
one hot vecor of "independent_feature1"
one hot vecor of "independent_feature2"
1 0 0 0 0
0 0 1 0 0
3 nodes in hidden layer, because window_size=3
predicted one hot vector of "target_feature"
for each node we predit three values due to window_size=3
0 1 0 0 0
Actual Target
one hot vector of"target_feature"
0 1 0 0 0
3 nodes in hidden layer, because window_size=3
predicted vector of "independent_feature1"
predicted vector of "independent_feature1"
Actual Target
1 0 0 0 0
0 0 1 0 0

Limitations of Word2Vec

- Does not consider sub-word information.
- Same vector for all senses of a word (polysemy issue).
- Ignores word order within the context window.

GloVe – Global Vectors

GloVe – Global Vectors

Proposed by: Pennington et al., 2014

- Combines global matrix factorization with local context windows.
- Captures co-occurrence statistics of words across entire corpus.
- Embeddings reflect ratios of co-occurrence probabilities.

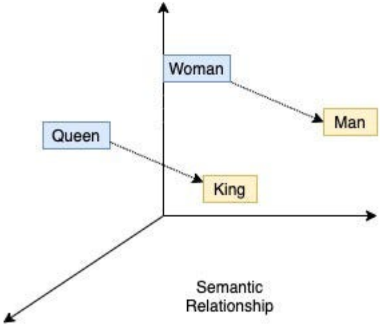
Loss function:

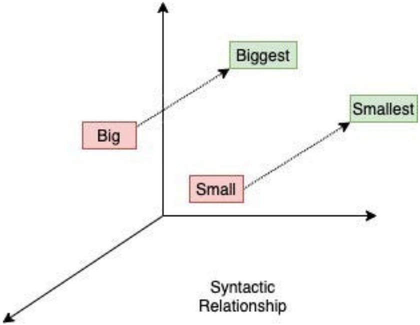
$$J = \sum_{i,j=1}^{V} f(P_{ij}) \left(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2$$

where:

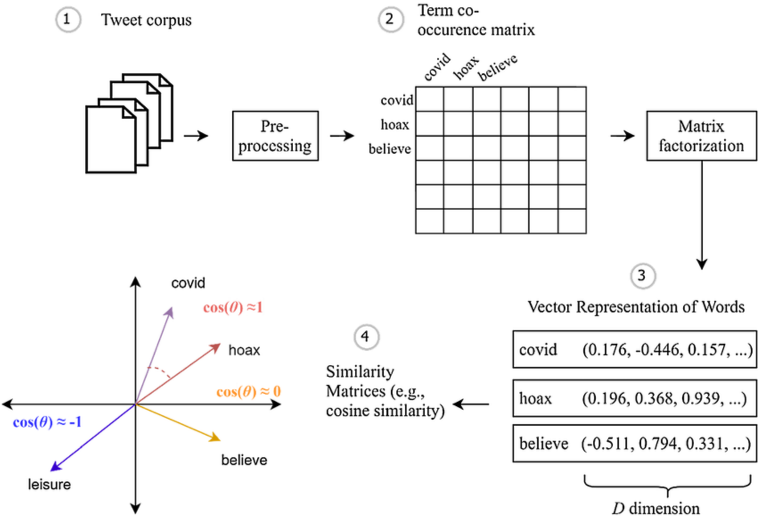
- X_{ij}: number of times word j occurs in the context of word i
- P_{ij}: probability of word j in the context of i
- $w_i, \tilde{w}_j$: word and context word vectors
- $b_i, \tilde{b}_j$: bias terms
- f: weighting function

GloVe – Global Vectors (cont.)





أكاديمية كاوست
KAUST ACADEMY
LMH
Lady Margaret Hall



fastText – Subword Embeddings

fastText – Key Ideas

- Developed by Facebook AI (2016)
- Builds on Word2Vec by incorporating character n-grams
- Useful for morphologically rich languages and rare words
- Better handling of OOV (out-of-vocabulary) words

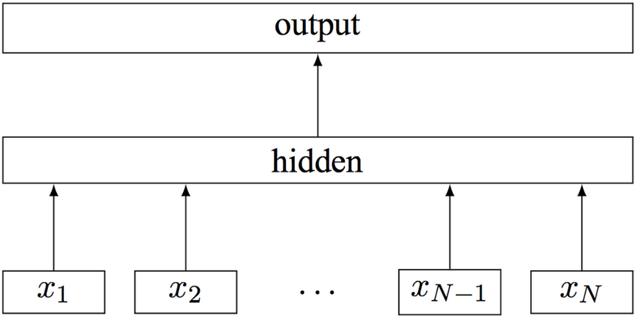
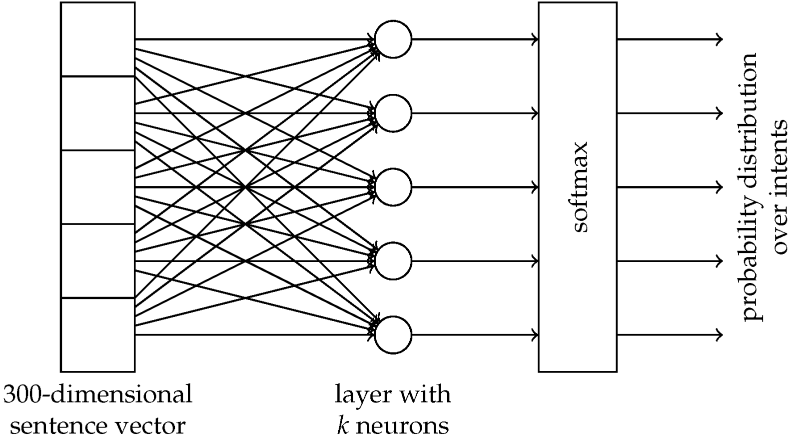


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.



FastText-Based Intent Detection for Inflected Languages

- Word: `playing`
- Character n-grams (n=3): `["pla", "lay", "ayi", "yin", "ing"]`
- Word vector = sum of n-gram vectors

Comparison: Word2Vec vs. GloVe vs. fastText

Feature	Word2Vec	GloVe	fastText
Local Context	✓	–	✓
Global Info	–	✓	✓(partial)
Subword Info	–	–	✓
Handles OOV	–	–	✓

- **High Dimensionality:**
  - Vectors can become very high-dimensional, leading to computational inefficiency.
  - Curse of dimensionality: distance metrics become less meaningful.
- **Sparsity:**
  - One-hot vectors are sparse, leading to inefficiencies in storage and computation.
  - Dense vectors mitigate this but still require large datasets for effective training.
- **Lack of Context:**
  - Traditional VSMs do not capture word context effectively.
  - Same word can have different meanings in different contexts (polysemy).

Limitations of Vector Space Models (cont.)

- **Semantic Limitations:**
  - Cannot capture complex relationships like negation or antonymy.
  - Similar words may not always be semantically related (e.g., "bank" vs. "river bank").
- **Scalability:**
  - As vocabulary size increases, the term-document matrix becomes larger and more sparse.
  - Requires significant computational resources for training and inference.

Summary

Future Directions

- **Contextual embeddings** (ELMo, BERT, GPT): Word vectors depend on sentence context.
- **Multilingual embeddings** and cross-lingual models.
- **Graph-based embeddings** (e.g., knowledge graph completion).
- **Hybrid embeddings**: Combining structured and unstructured data.

Key Takeaways

- Vector Space Models (VSMs) are foundational for modern NLP.
- Word embeddings capture semantic relationships in a continuous space.
- Word2Vec, GloVe, and fastText are key methods for generating word vectors.
- Dense embeddings outperform one-hot vectors in capturing meaning and relationships.
- Future work focuses on contextual, multilingual, and hybrid embeddings.

References

References

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*.

[2] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *EMNLP*.

[3] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *TACL*.

[4] Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd Ed Draft).

References (cont.)

[5] Stanford NLP Slides.
https://web.stanford.edu/class/cs224n

[6] Facebook AI Research (FAIR) – fastText.
https://fasttext.cc

[7] Goldberg, Y. (2016).
A Primer on Neural Network Models for NLP.
JMLR.

[8] Chrupała, G. (2019).
Symbolic and Subsymbolic Representations in NLP – Deep Learning Lectures.

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.