University of Northampton

Week 5

Word2Vec & Word Embeddings

CSY3055 – Natural Language Processing

Dr Oluseyi Oyedeji

How can a computer tell that

'doctor' and 'nurse' are related,

or that

'king – man + woman = queen'?

Do machines actually understand meaning?

Today we'll see how they do it, through word embeddings, where words become numbers that carry meaning.

University of Northampton

Learning Goals for Week 5

By the end of week five, you should be able to:

- Explain **distributional semantics** ["*a word is known by the company it keeps*."]
- Describe how **Word2Vec** learns vector representations (**CBOW** & **Skip-gram**).
- Understand the **mathematics** behind Word2Vec's learning process.
- Compare **Word2Vec** and **GloVe embeddings**. **Gensim**
- **Train and test** a small Word2Vec model using.
- **Evaluate word similarities** and analogy tasks.
- Reflect on **bias** in embeddings and how to detect it.

Relevance

- Word embeddings form the foundation of modern NLP as they **transform text into numerical vectors** that **capture meaning**.
- Every **transformer** model (BERT, GPT, etc.) begins with this same idea.
- Understanding embeddings **bridges** your knowledge from ***classical NLP (TF-IDF) to deep learning***.

Today's Session

- Introduction
- Distributional Semantics
- Word2Vec
- GloVe
- Ethical reflection on bias

Introduction

From Words to Numbers

- Computers understand only **numbers**.
- **Earlier methods** (*Bag-of-Words, TF-IDF*) counted occurrences but **ignored meaning**:

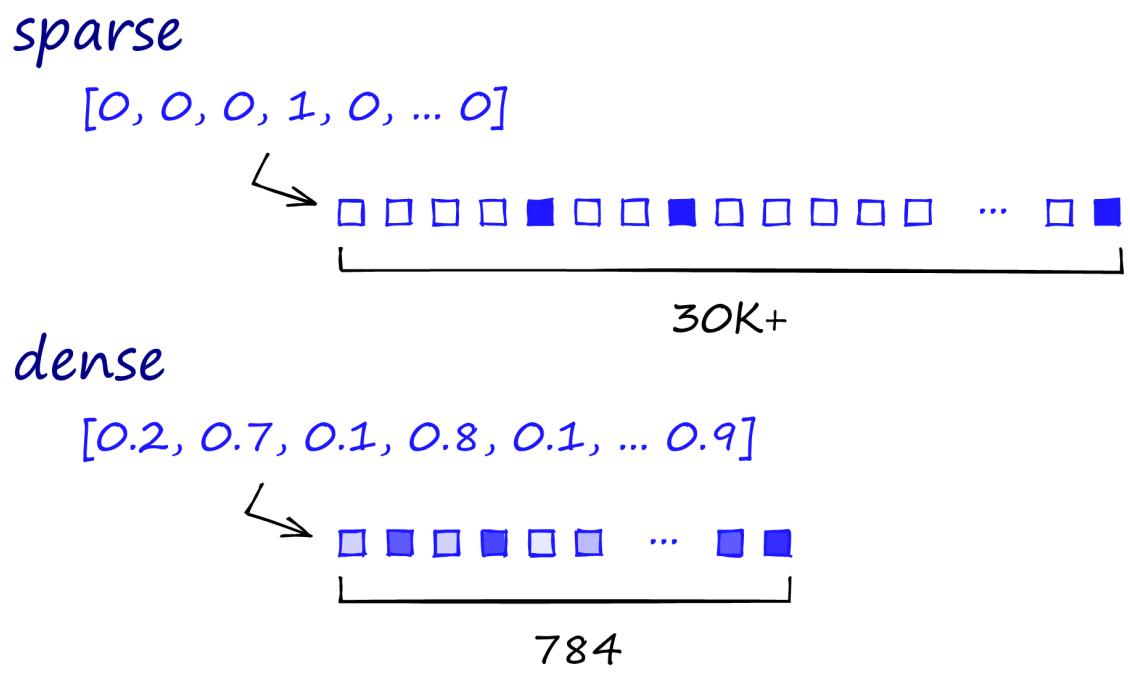
“cat”	“dog”	“car”
3	2	0

- These are **frequency counts**, not **relationships**.

From Words to Numbers contd.

- Word2Vec instead **learns how words occur together**, producing dense vectors such as

`cat = [0.23, −0.14, 0.88, …]` – this encode semantic similarity.

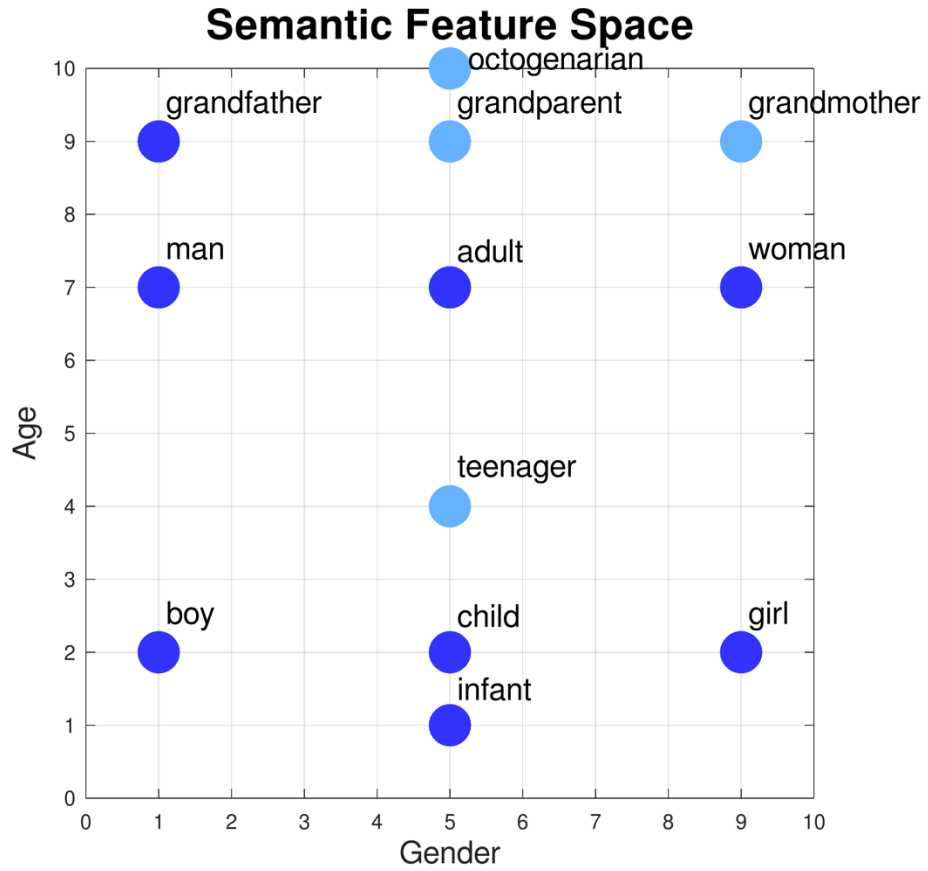


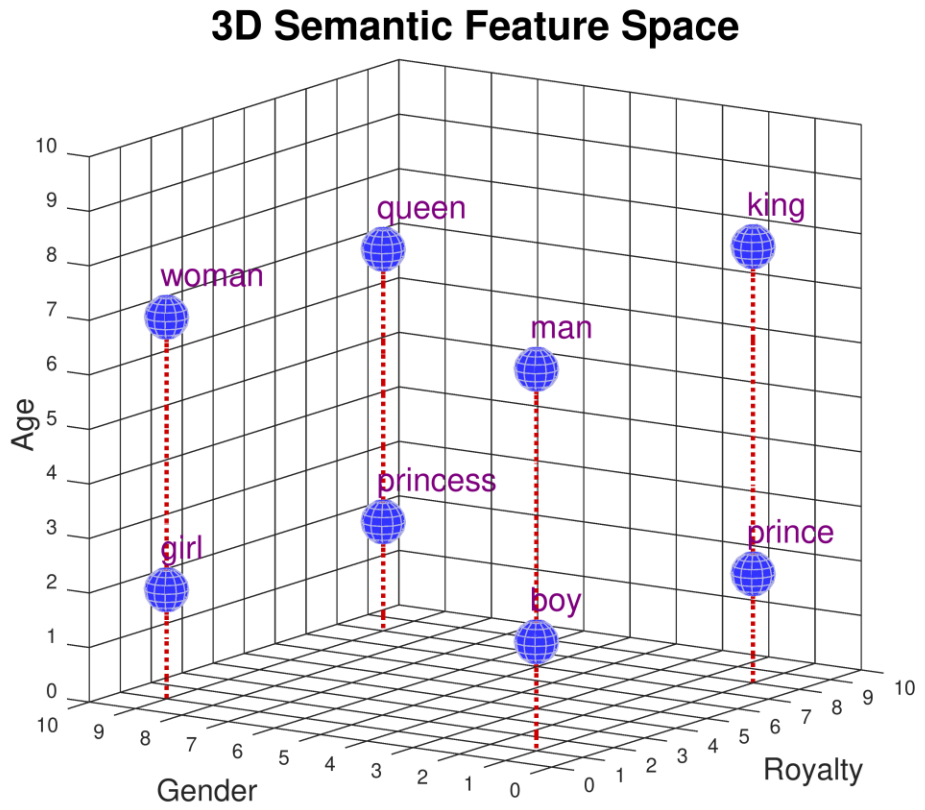
Distributional Semantics

"You shall know a word by the company it keeps." – J.R. Firth (1957)

- Words appearing in similar contexts tend to mean similar things.
  - "**dog**" and "**cat**" share contexts like **pet**, **food**, **cute**;
  - "**car**" and "**bus**" share **road**, **driver**, **wheel**.
- Word2Vec converts these contextual co-occurrences into positions in a semantic space.

Distributional Semantics contd.





Conceptual Semantic Space

Word2Vec

What is Word2Vec?

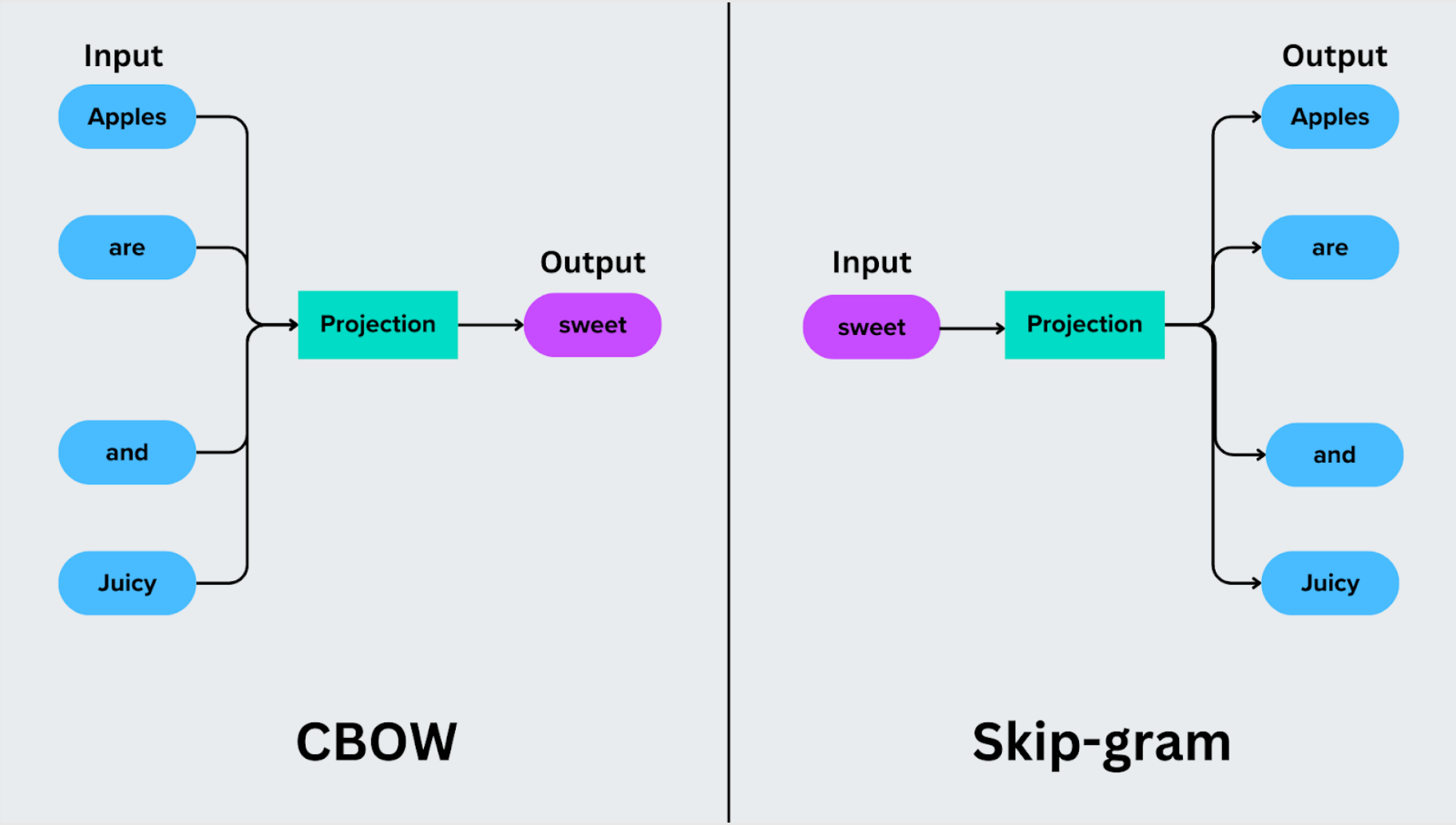
- **Word2vec** is a technique in natural language for obtaining vector representations of words.
- These vectors capture information about the **meaning of the word** based on the surrounding words.

Word2Vec Architectures

- Continuous Bag of Words (**CBOW**)
  - Predicts the **target word** based on the **context words** around it.
- **Skip-Gram**
  - Predicts the **context words** based on the **target word**.

Model	Predicts	Description	When to use
CBOW	Target from context	Predict missing word from its neighbours	Faster; good for large data
Skip-gram	Context from target	Predict nearby words from the centre	Better for small or sparse data

Word2Vec Architectures contd.

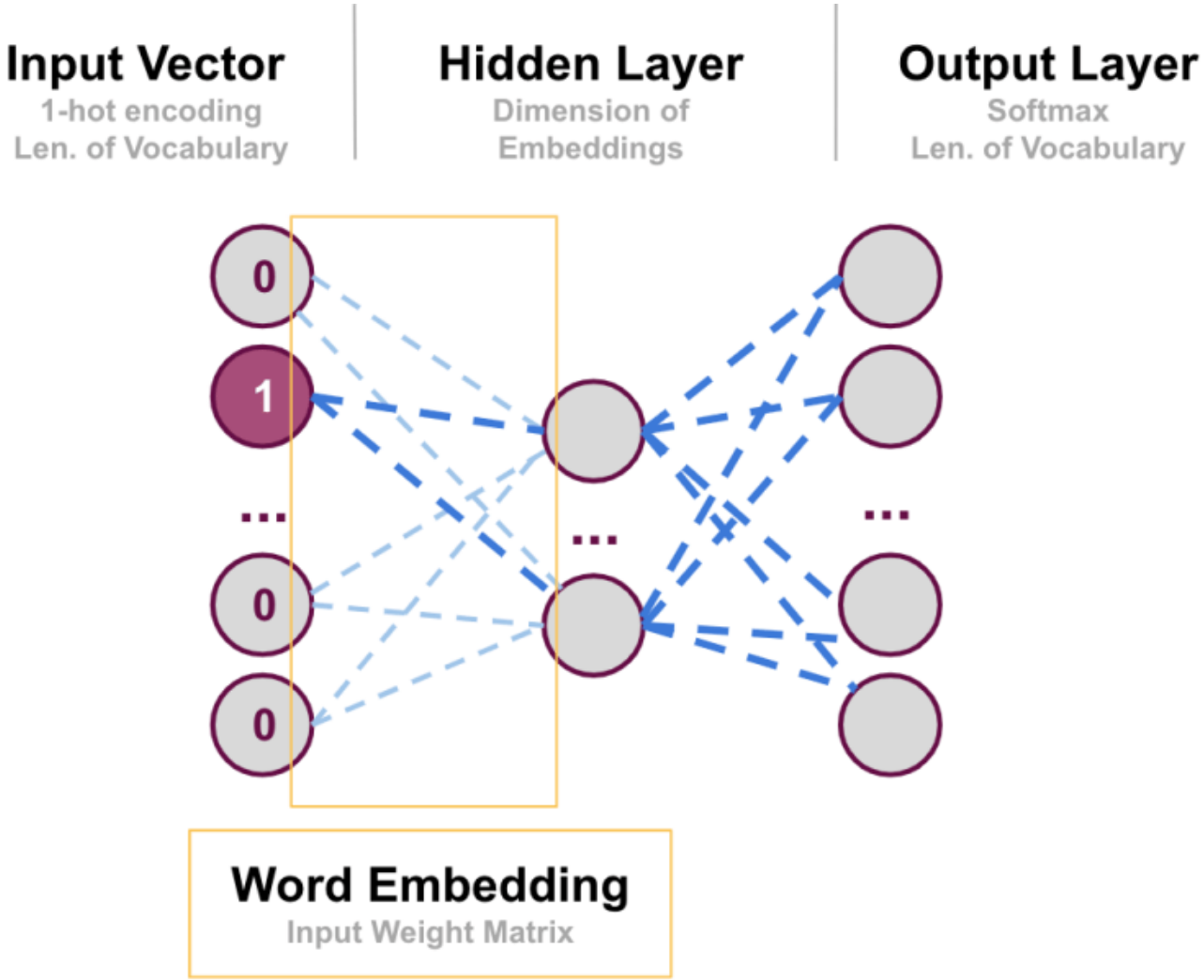


Inside the Word2Vec Neural Network (Skip-Gram)

- Word2Vec is a shallow neural network with three layers:

Layer	Description	Example (Skip-gram)
Input layer	Takes one word (as a one-hot vector).	e.g. cat = [0,0,1,0,0,...]
Hidden (embedding) layer	A linear layer that converts the one-hot input into a dense vector.	Multiplies input by a weight matrix → produces the word embedding.
Output layer	Predicts which context words appear nearby.	Gives a probability distribution over all words (softmax).

Inside the Word2Vec Neural Network



Foundation and Formula

- **Goal**
  - Predict the **probability** of each context word given the **target word**.

$$P(w_O \mid w_I) = \frac{e^{v'_{w_O} \cdot v_{w_I}}}{\sum_{w=1}^{V} e^{v'_w \cdot v_{w_I}}}$$

- v_{WI} : input vector
- $v`_{WO}$: output vector
- V = Vocabulary size
- Numerator = “target similarity”; denominator = “sum over all words”

Predicting Nearby Words (Skip-gram)

- Word2Vec learns to predict **context words** given a **target word**.
- Example: *"The cat sat on the mat."*
  - **Target** = cat
  - **Context(Window = 2)** = {the, sat, on}
- **Training pairs**
  - `(cat → the)`
  - `(cat → sat)`
  - `(cat → on)`

Predicting Nearby Words (Skip-gram)

- The model should assign higher probability to real contexts and lower to random ones

The cat sat on the mat

- Across many sentences:

(cat → mat)

Across many sentences, “cat” and “mat” often co-occur, so the model still learns them as semantically related.

Worked Example [Conceptual]

- Compute Similarity (dot product)

Pair	Dot Product
cat·mat	0.86
cat·car	-0.08

- Apply **Softmax** (convert to probabilities)
  - “mat” is 72 % likely near “cat”; “car” is 28 %.
  - That’s meaning emerging from numbers.

$$e^{0.86} = 2.36, \quad e^{-0.08} = 0.92$$

$$P(\text{mat}|\text{cat}) = \frac{2.36}{2.36 + 0.92} = 0.72$$

$$P(\text{car}|\text{cat}) = \frac{0.92}{2.36 + 0.92} = 0.28$$

From Training to Embeddings

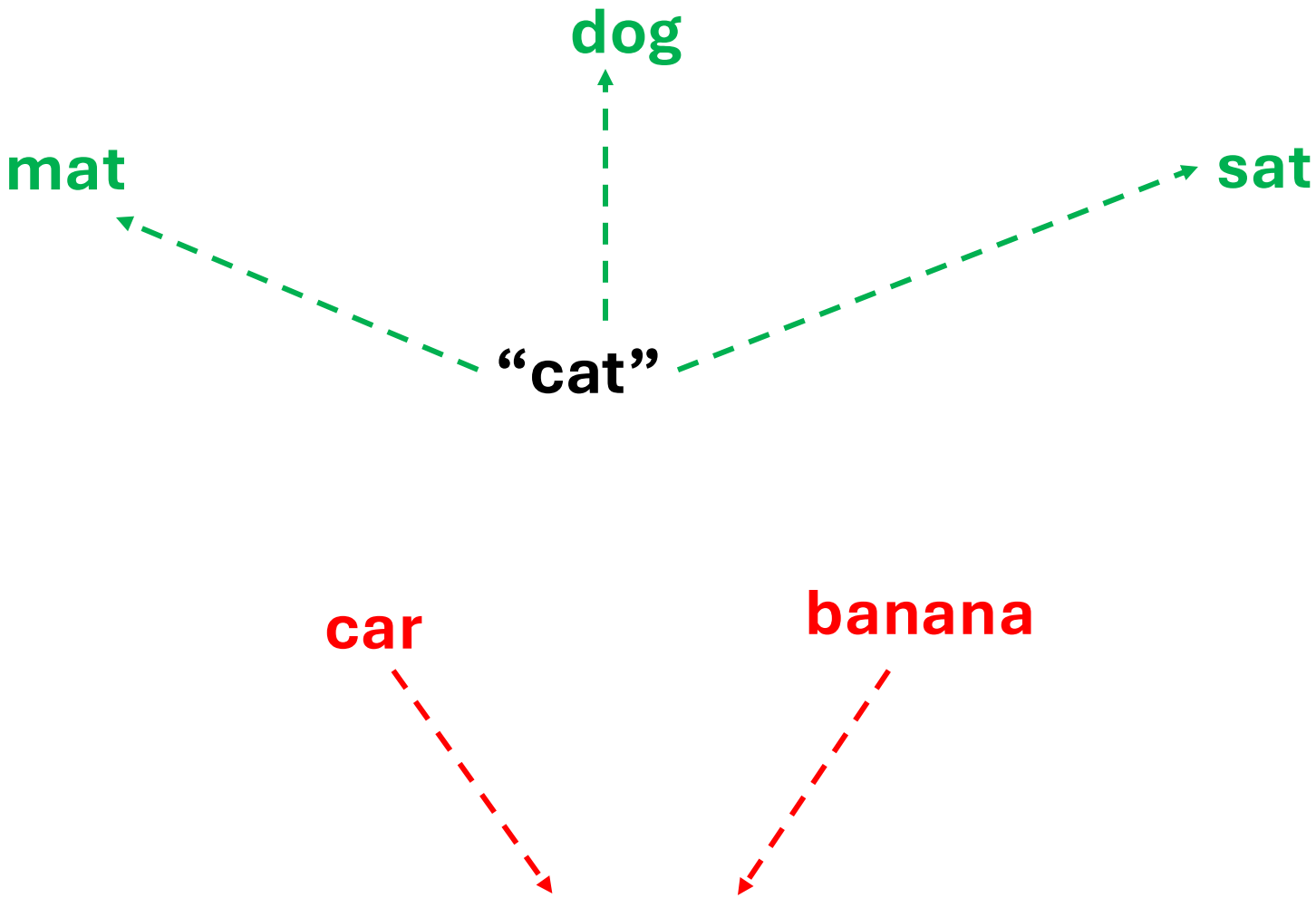
- The **output layer** is **discarded**; we only keep the **hidden layer weights**.
- Each word now has its own **embedding vector**.
- *Words that appear in similar contexts have similar embeddings.*

Word	Embedding (simplified)
cat	[0.35, 0.77, 0.12, …]
dog	[0.33, 0.79, 0.14, …]
car	[0.10, –0.55, 0.82, …]

How Word2Vec Learns from Data

- Word2Vec improves by comparing real word pairs vs random ones and adjusting its internal weights to get better at prediction.
  - The model starts with **random numbers** for every word.
  - Each time it predicts correctly (**e.g., cat → mat**), it slightly strengthens that connection.
  - Each time it predicts wrongly (**e.g., cat → car**), it weakens that connection.
  - Thousands of small adjustments make the model better at spotting meaningful relationships.

How Word2Vec Learns from Data contd.



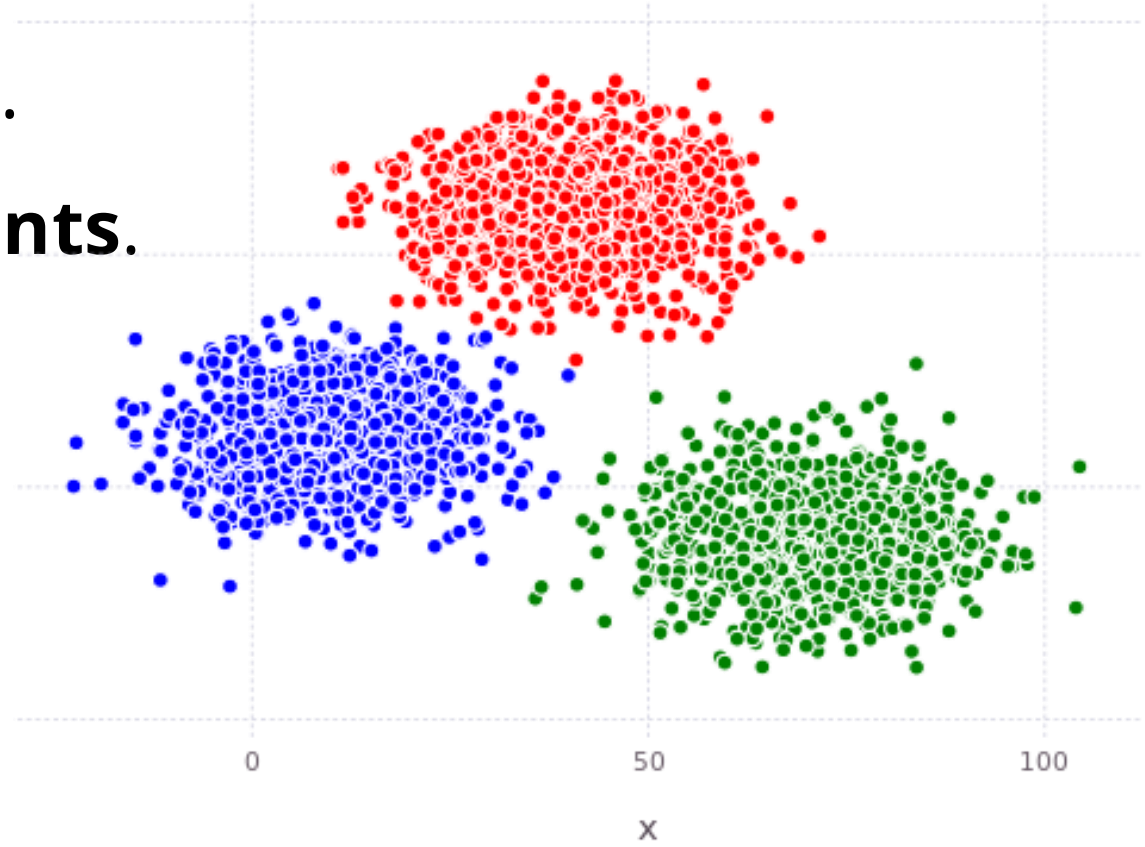
Training Faster with Negative Sampling

- Predicting across **100,000** words is too **slow**.
- Instead, we train with
  - **1 true context word**
  - **a few random "noise" words (negatives)**
- The model **learns** to tell **true from fake.**

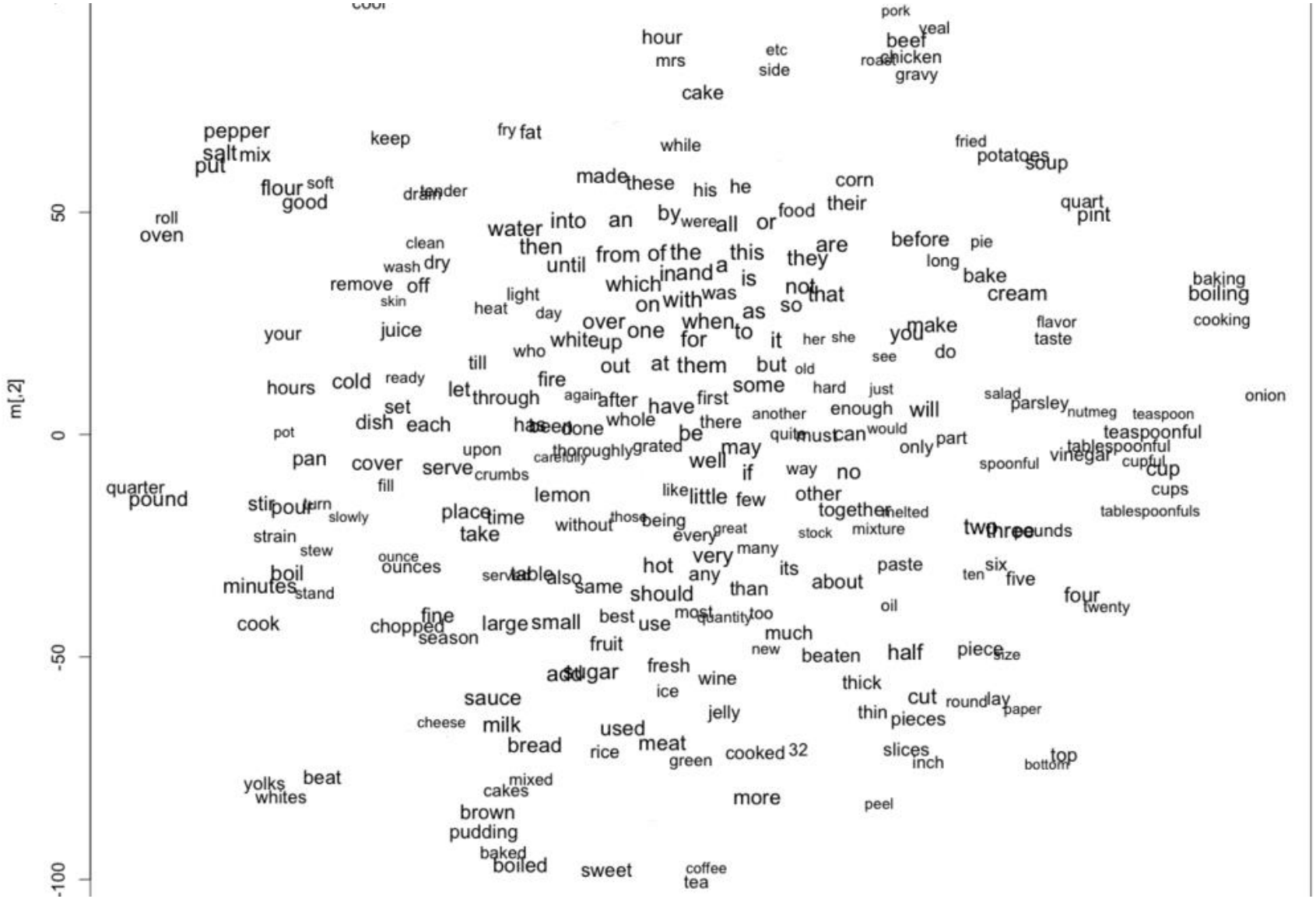
Target Word	Real Context	Fake Contexts
cat	mat 🟢	car 🔴, sky 🔴, banana 🔴

The Word Embedding Space

- Each word → **one dense vector**.
- **Similar meanings → closer points**.
- Examples:
  - cat, dog, rabbit → cluster together
  - car, bus, train → another cluster
  - apple, orange, banana → another



The Word Embedding Space contd.

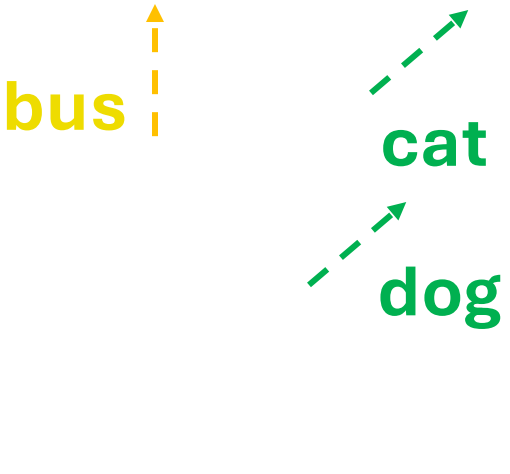


University of Northampton

How We Measure Word Similarity

- **Cosine similarity** looks at direction, not length.
- *Two words pointing the same way* = **similar meaning**.
- Example:
  - cat–dog: very similar

    cat–car: not similar
  - car–bus: similar



Bias in Embeddings

- Because models learn from human text, they absorb **human stereotypes.**
- Example analogy:
  - **man** : computer programmer :: **woman** : homemaker
  - Bias can influence **translation**, **hiring systems**, and **chatbots**.
- Mitigation
  - balanced corpora, bias detection tests, documentation, or debiasing algorithms.

Summary - Word2Vec

Step	Description
1	Text → one-hot vectors
2	Model predicts nearby words
3	Adjusts weights , pulls similar, pushes different
4	Keep hidden layer as word embeddings
5	Measure meaning with cosine similarity

Glove

What is GloVe?

- Glove means **Global Vectors for Word Representation**
- GloVe learns word meanings by studying **global co-occurrence patterns** across the entire text, not just nearby words like Word2Vec.

	Data	is	next	oil	future
Data	0	2	0	0	0
is	2	0	1	0	1
future	0	1	0	0	0
next	0	1	0	1	0
oil	0	0	1	0	0

What is GloVe?

- Developed by **researchers at Stanford**.
- Uses statistics of word co-occurrence, how often words appear together in the whole corpus.
- Learns embeddings so that the ratios of co-occurrence probabilities capture meaning.

GloVe vs Word2Vec

Feature	Word2Vec	GloVe
Approach	Predicts context words using a neural net	Learns from co-occurrence matrix (count-based)
Focus	Local context (sliding window)	Global statistics (entire corpus)
Learning Objective	Predict probabilities	Factorize co-occurrence ratios
Strengths	Efficient, good for streaming data	Captures long-range meaning better
Output	Dense word vectors	Dense word vectors

Reference

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). **Efficient estimation of word representations in vector space**. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). **Distributed representations** of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26, 3111–3119.
- Pennington, J., Socher, R., & Manning, C. D. (2014). **GloVe: Global vectors for word representation**. Proceedings of EMNLP 2014, 1532–1543.

Practical & Wrap-Up

Thank you