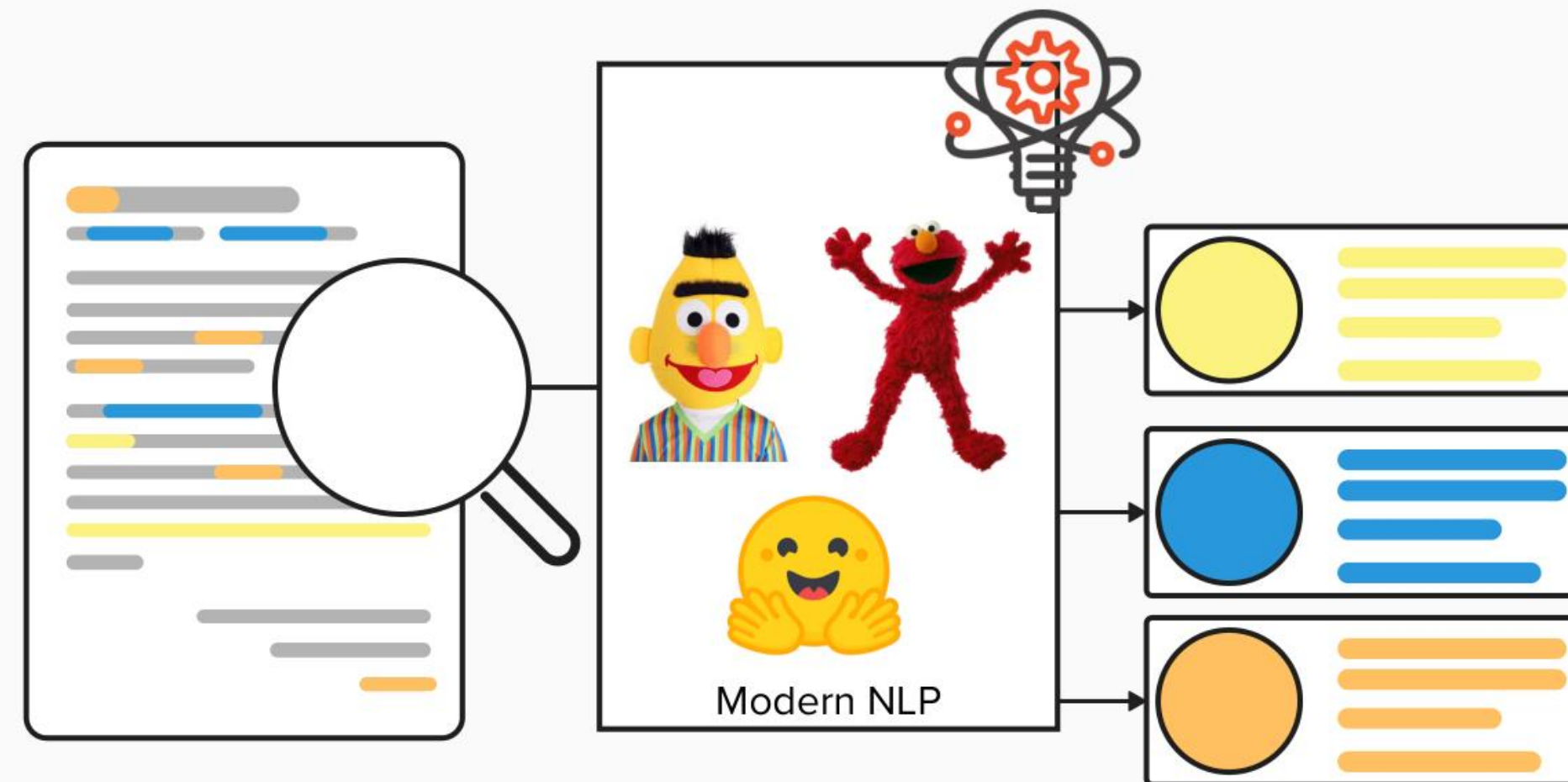


Modern Text Classification



Key Topics Covered:

1. Introduction

- a) How computers understand us?

2. Tokens and Embeddings

- a) What is tokenization
- b) Token Representation
- c) Word Embeddings introduction

3. Word Embeddings Implementation

- a. Word2VEC
- b. BERT
- c. Beyond BERT

4. Practical Workshop

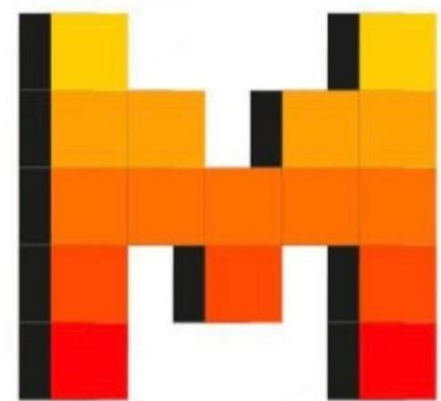
- a. Tokenizers and comparisons
- b. Embedding generation and visualization
- c. Sentence similarity
- d. Sentiment Analysis with embeddings
- e. Other applications with embeddings



Llama 3.1



ChatGPT



**MISTRAL
AI_**

The Gemini logo, featuring the word "Gemini" in a blue sans-serif font with a purple four-pointed star above the letter 'i'.

1 Embeddings – not only NLP



Computer Vision

- Image Embeddings with ResNet, VGG, CLIP
- Feature Embeddings (e.g., FaceNet)
- Visual similarity and retrieval



Recommendation Systems

- User behavior embeddings
- Item and content vectors



Social Internet of Things

- Device capability embeddings
- Social relationship vectors



Bioinformatics

- Protein sequence vectors (AlphaFold)
- Molecule embeddings (SMILES)
- Chemical property prediction



Audio Processing

- Speech recognition vectors
- Music similarity embeddings
- Speaker identification

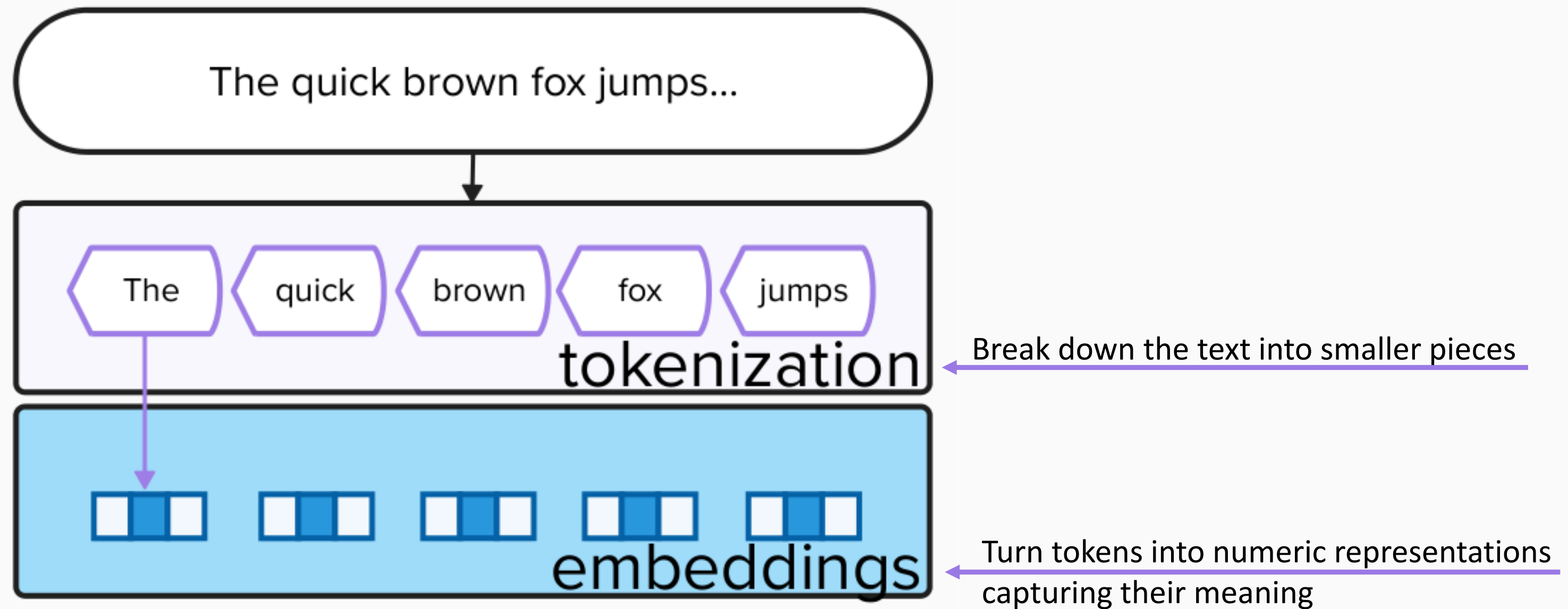


Robotics & Control

- State space embeddings
- Action vector spaces
- Navigation planning

1.1 Token and Embeddings

Learning object



2. Tokenization

How do we represent text?

Tokenization: Splitting or breaking down a string into smaller pieces

Given a piece of text u , we can represent it as a sequence $\langle u_1, u_2, \dots, u_n \rangle$

The sentence “*The quick brown fox jumps over the lazy dog*” can be split into:

$u = \langle \text{The}, \text{quick}, \text{brown}, \text{fox}, \text{jumps}, \text{over}, \text{the}, \text{lazy}, \text{dog} \rangle$

What if the sentence becomes: “*The **quick-brown** fox jumps over the lazy dog*”?



2.1 Tokenization

How do we represent text?

Tokenization: Splitting or breaking down a string into smaller pieces

Given a piece of text u , we can represent it as a sequence $\langle u_1, u_2, \dots, u_n \rangle$

The sentence “*The quick brown fox jumps over the lazy dog*” can be split into:

$u = \langle \text{The}, \text{quick}, \text{brown}, \text{fox}, \text{jumps}, \text{over}, \text{the}, \text{lax}, \text{dog} \rangle$

What if the sentence becomes: “*The **quick-brown** fox jumps over the lazy dog*”?

An expected tokenized output can be:

$u_1 = \langle \text{The}, \text{quick} - \text{brown}, \text{fox}, \text{jumps}, \text{over}, \text{the}, \text{lax}, \text{dog} \rangle$

$u_2 = \langle \text{The}, \text{quick}, -, \text{brown}, \text{fox}, \text{jumps}, \text{over}, \text{the}, \text{lax}, \text{dog} \rangle$

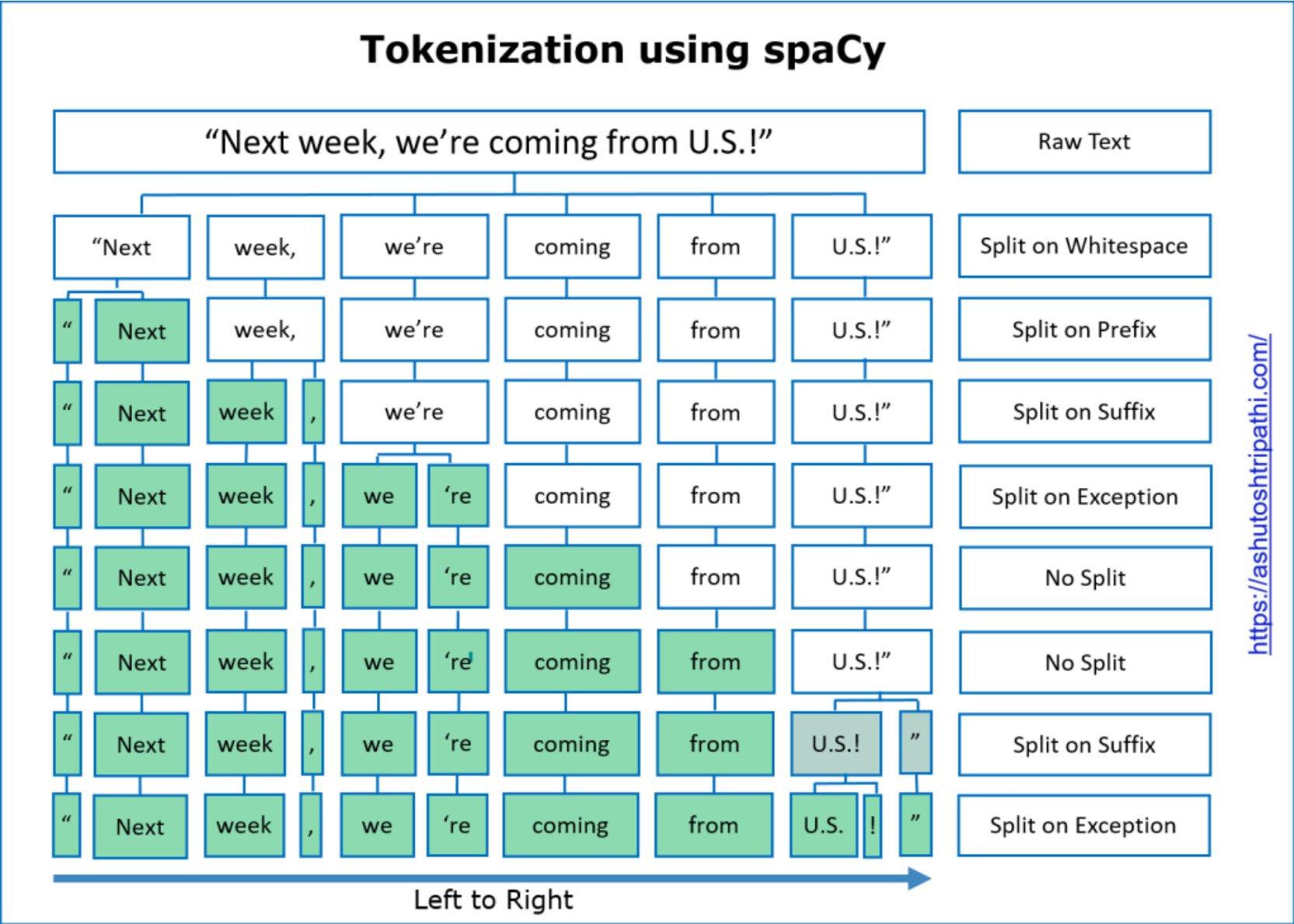
$u_3 = \langle \text{The}, \text{quick}, -, \text{brown}, \text{fox}, \text{jumps}, \text{over}, \text{the}, \text{lax}, \text{dog} \rangle$

...

Tokenization is not simple, and tokenizers require specialized rules



2.2 How tokenization works



2.3 Tokenization

GPT-4o & GPT-4o mini

GPT-3.5 & GPT-4

GPT-3 (Legacy)

Natural Language Processing, or NLP, enables machines to understand text. Tokenization splits words like preprocessing, embeddings, and transformers into smaller units. Even complex terms such as language-specific nuances get broken into meaningful pieces.

Clear

Show example

Tokens

48

Characters

256

Natural Language Processing, or NLP, enables machines to understand text. Tokenization splits words like preprocessing, embeddings, and transformers into smaller units. Even complex terms such as language-specific nuances get broken into meaningful pieces.

<https://platform.openai.com/tokenizer>

Definition according to IBM

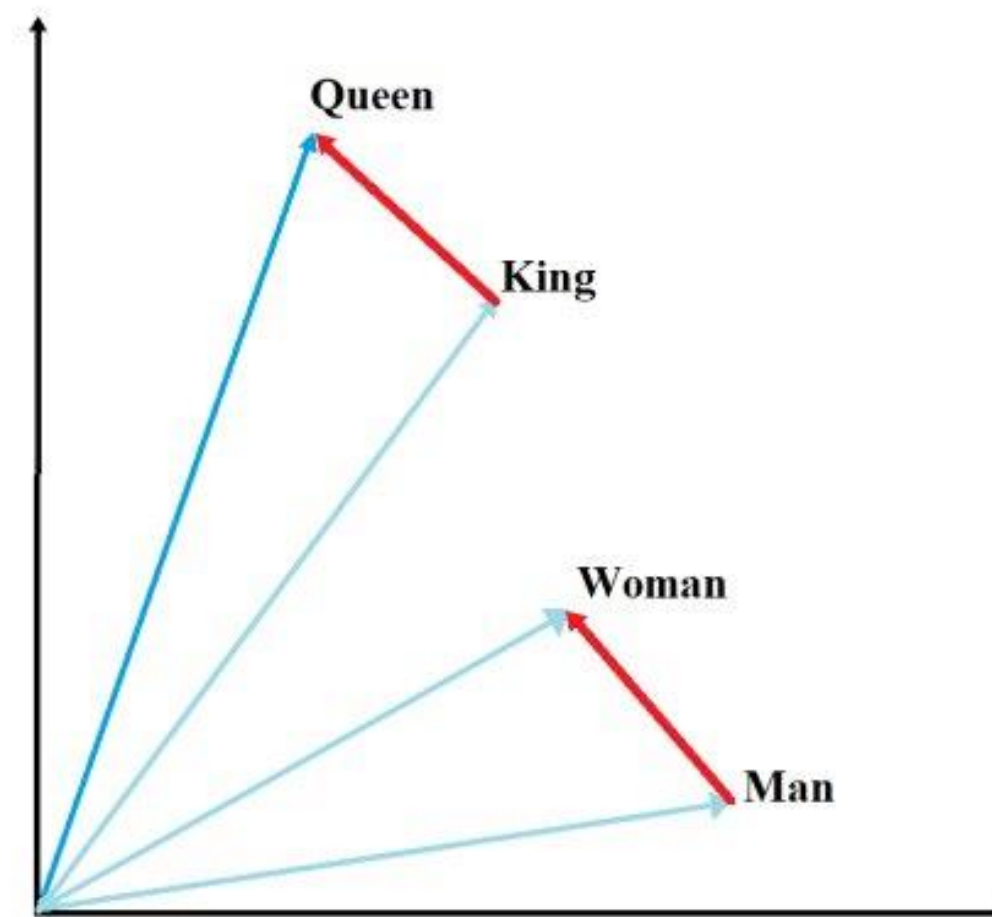
Word embeddings are a way of representing words as vectors in a multi-dimensional space, where the distance and direction between vectors reflect the similarity and relationships among the corresponding words.

Why it matters:

- Represent a word as a point in a multidimensional semantic space
- Space itself is constructed from distribution of word neighbours

Key Properties

- Captures semantic similarity and relatedness
- Dense vectors with continuous values
- Learned from large text corpora



3.1 Word2Vec (2013)

Limitation of BoW

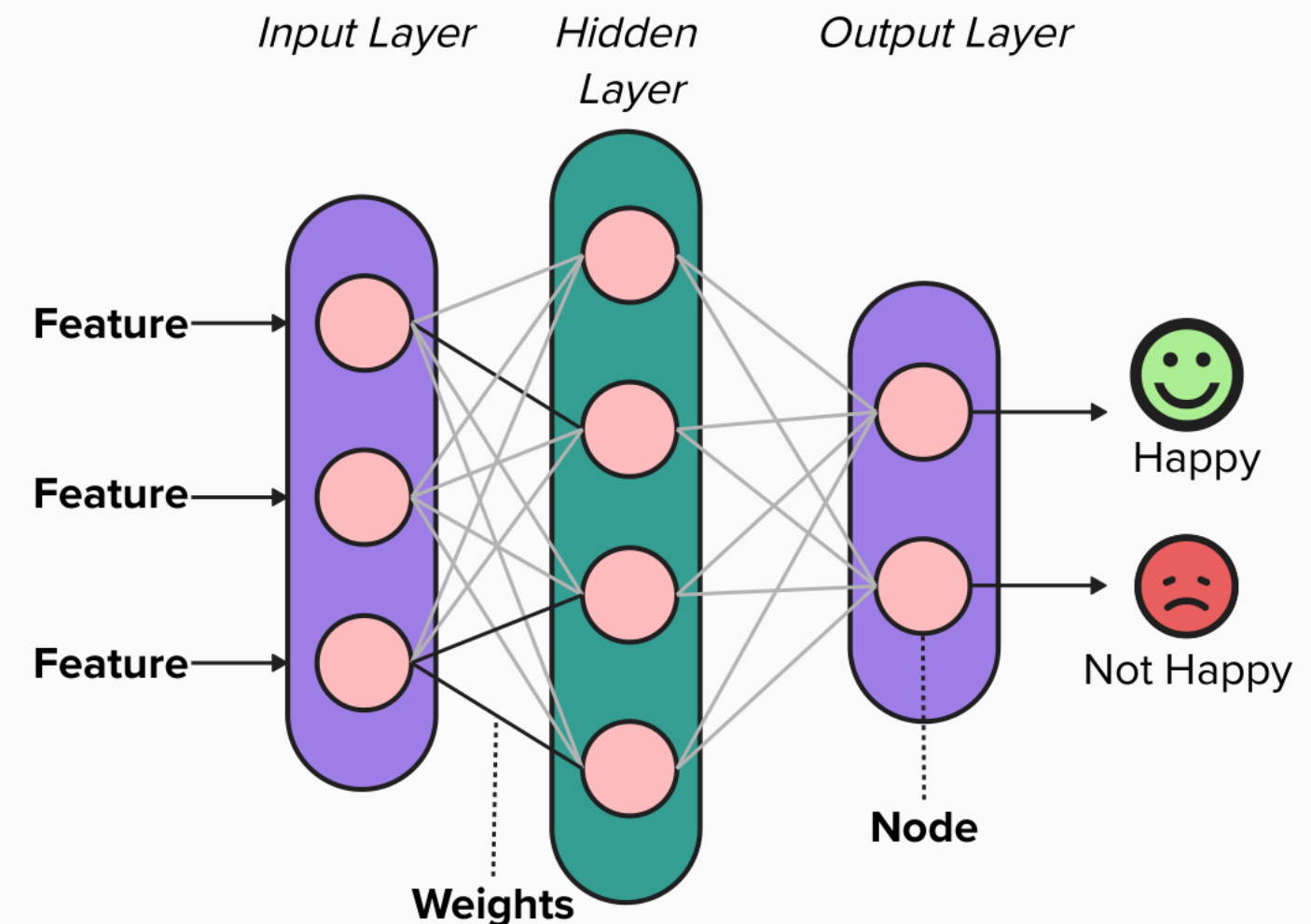
- Words are treated as independent tokens
- No semantic relationship captured
- Order and context are lost
 - “Dog bites man” or “man bites dog” are the same
- Cannot Capture meaning nuances
- Sparse representation

Word2Vec (2013)

- Dense vector embeddings instead of sparse vectors
- Learns from massive text corpora (e.g., Wikipedia)
- Captures semantic relationships between words
- ☒ Similar words have similar vectors
- ☒ Enables arithmetic with words!

How it works

- Uses neural networks to learn word representation
 - Key Components
 - Input layer
 - Hidden layer(s)
 - Output layer



3.2 Word2Vec (2013)

Limitation of BoW

- Words are treated as independent tokens
- No semantic relationship captured
- Order and context are lost
 - “Dog bites man” or “man bites dog” are the same
- Cannot Capture meaning nuances
- Sparse representation

Word2Vec (2013)

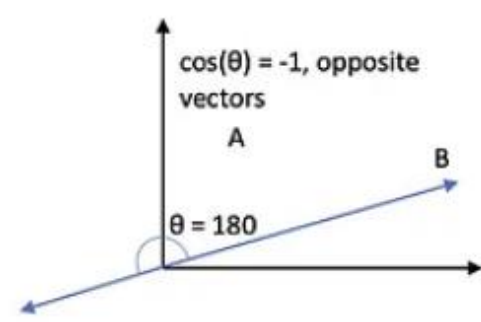
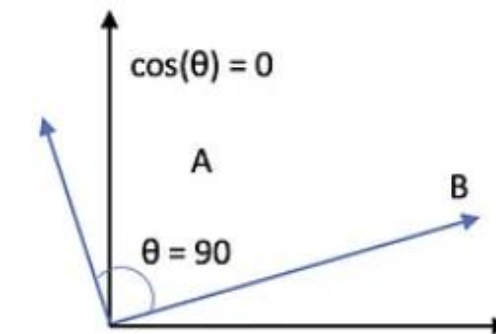
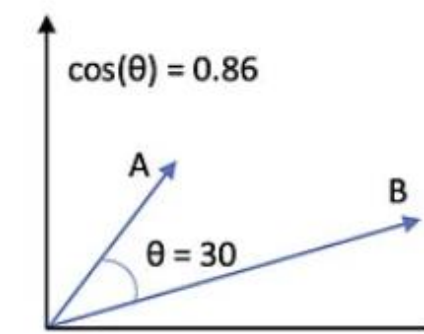
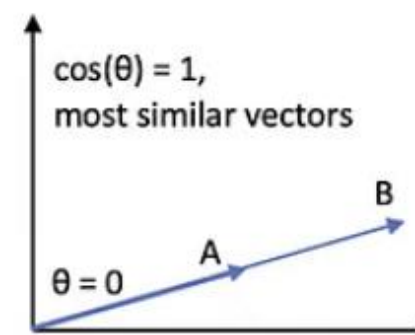
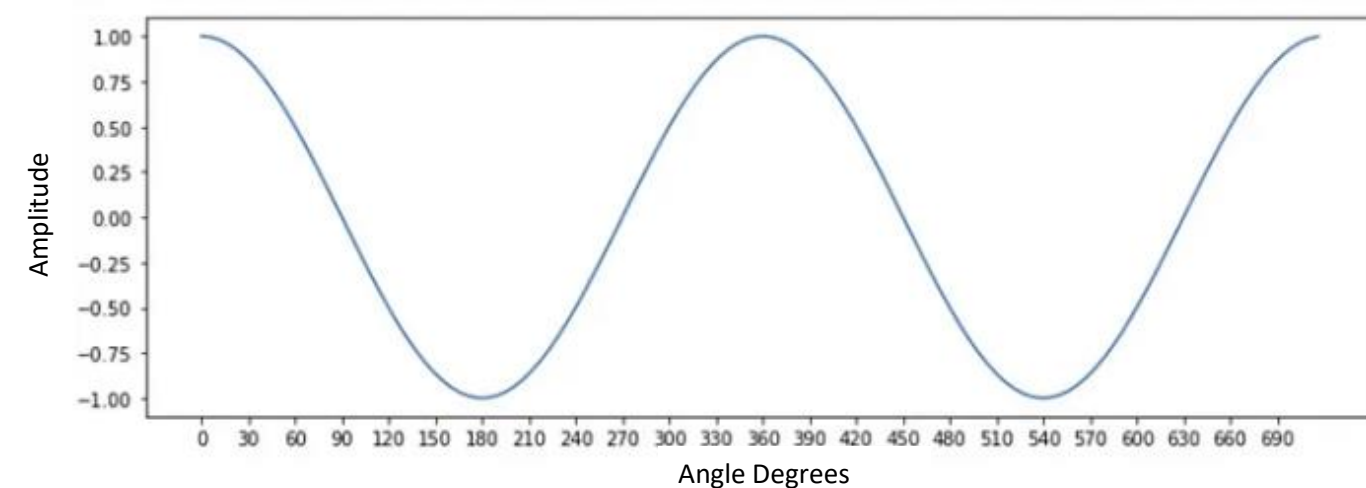
- Dense vector embeddings instead of sparse vectors
- Learns from massive text corpora (e.g., Wikipedia)
- Captures semantic relationships between words
- ☒ Similar words have similar vectors
- ☒ Enables arithmetic with words!

How it works

- Uses neural networks to learn word representation
 - Key Components
 - Input layer
 - Hidden layer(s)
 - Output layer

Cosine Similarity

$$\cos(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| |\vec{B}|} = \frac{\sum_{i=1}^N A_i B_i}{\sqrt{\sum_{i=1}^N A_i^2} \sqrt{\sum_{i=1}^N B_i^2}}$$



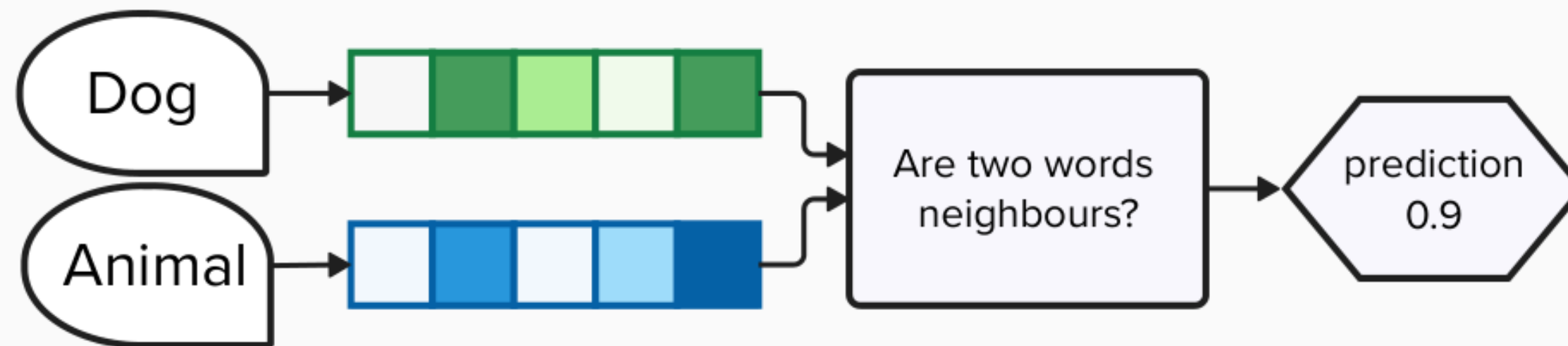
3.3 Word2Vec (2013)

Training Process

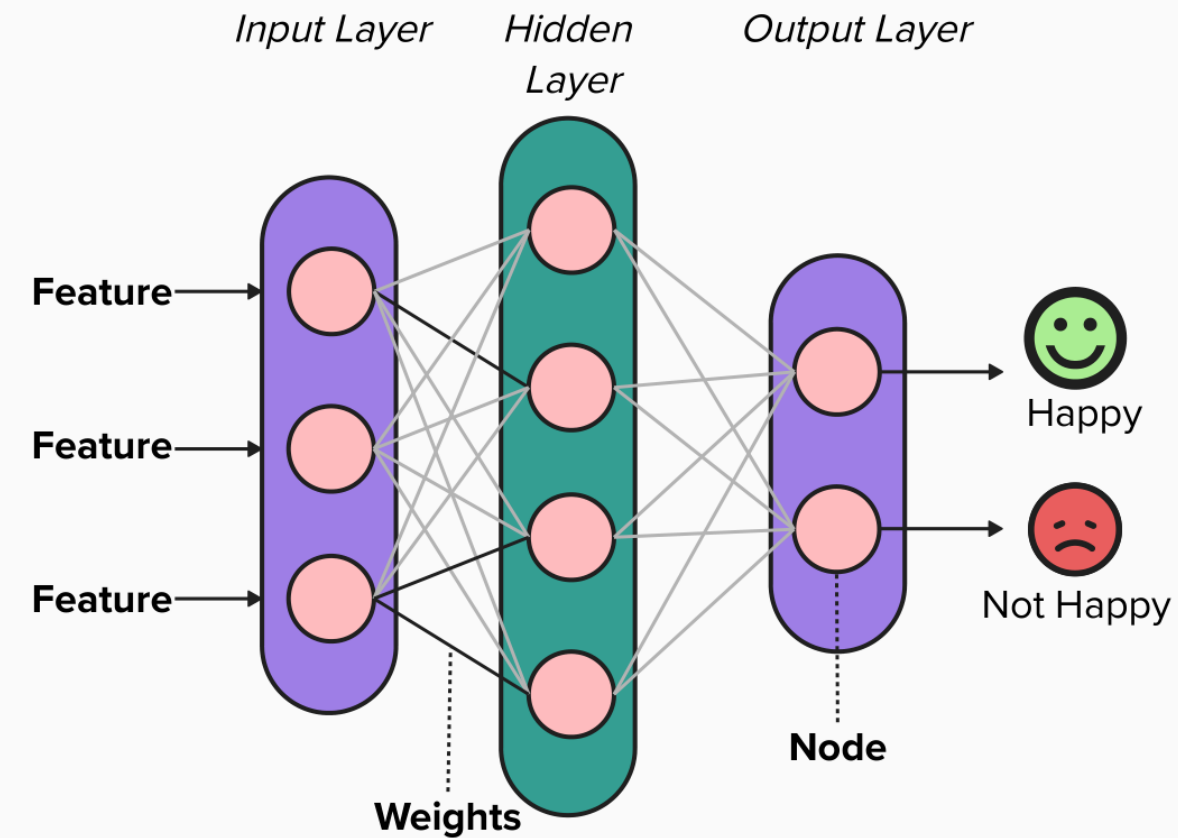
1. Initialize random vectors for each word
 2. Predict if words are neighbors in sentences
 3. Update embeddings based on predictions
 4. Repeat until convergence
- ✓ Words that appear in similar contexts get similar embeddings

Words are represented as fixed-length vectors

Each dimension captures semantic properties



Limitations?



3.4 Word2Vec (2013)

Training Process

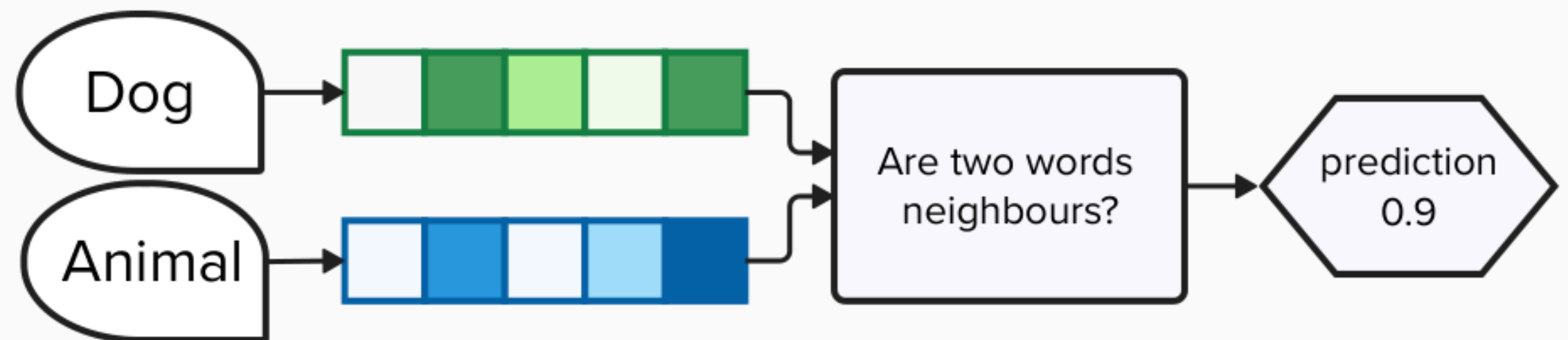
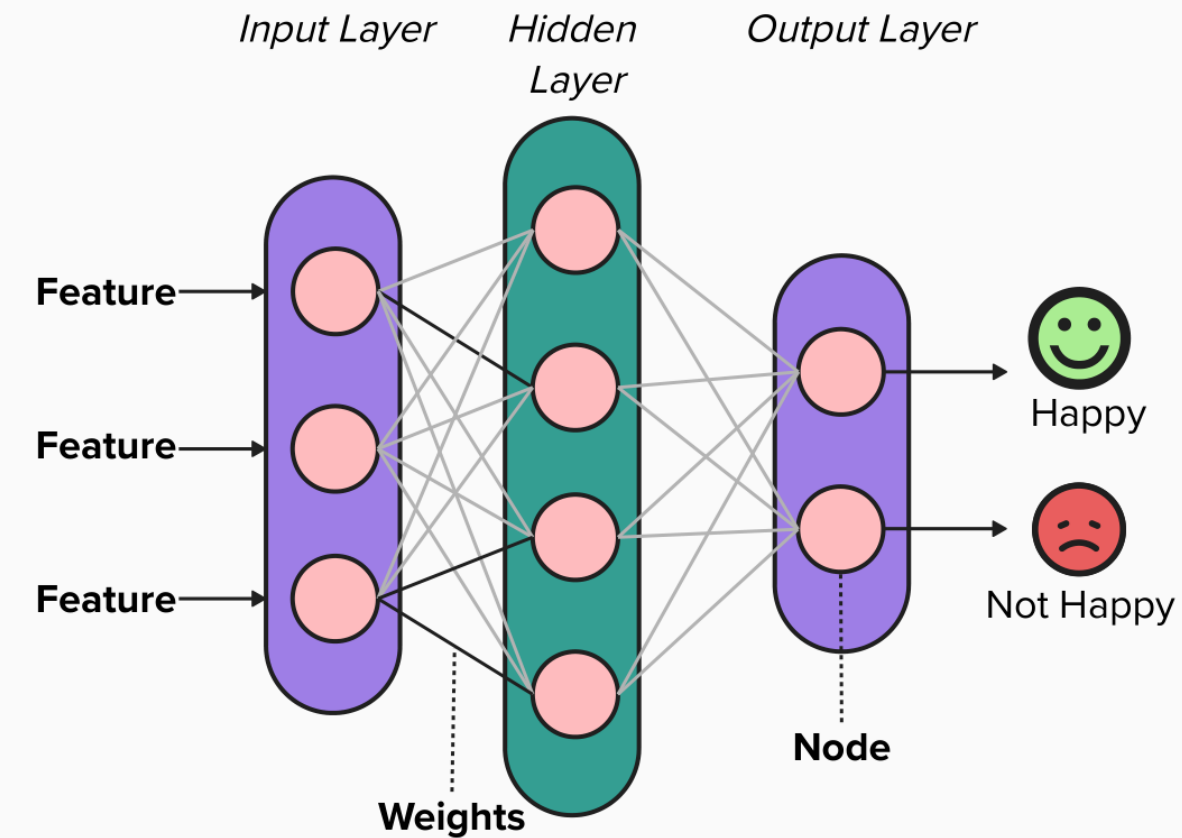
1. Initialize random vectors for each word
 2. Predict if words are neighbors in sentences
 3. Update embeddings based on predictions
 4. Repeat until convergence
- ✓ Words that appear in similar contexts get similar embeddings

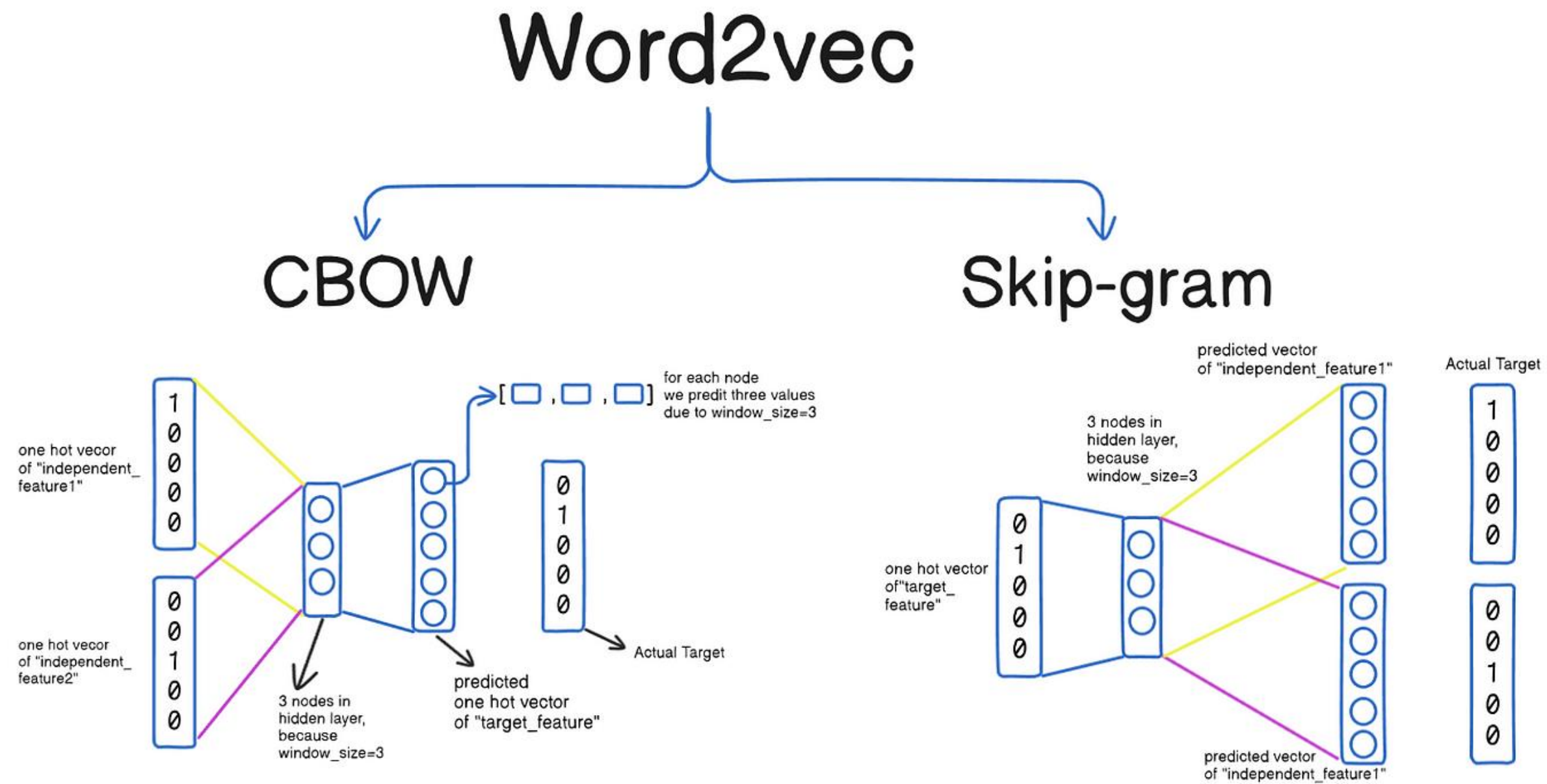
Words are represented as fixed-length vectors

Each dimension captures semantic properties

Limitations?

1. Static Embeddings (one vector per word)
2. No context awareness
3. Cannot handle polysemy





Continuous BoW

- **Training Phase**

1. The CBOW model starts with no knowledge of words or their meanings.
2. It is trained on a large text corpus (e.g., millions of sentences).
3. During training, the model learns to predict a target word (e.g., "boat") based on the surrounding words (the context).

- **Model Structure**

1. **Input Layer:** The context words (e.g., "The", "sailed", "across", "the", "ocean") are converted into numerical vectors (initial random embeddings).
2. **Output Layer:** The model tries to predict the target word ("boat") using these vectors.

- **Model Structure**

1. The model calculates a probability for each word in the vocabulary (e.g., "boat", "ship", "car", "plane", etc.) to be the correct target word.
2. This probability is determined by a mathematical function (softmax) that compares how "close" the context (surrounding words) is to each word in the vocabulary.
3. Initially, the probabilities are random, but during training, the model "learns" to adjust the word vectors to maximize the probability of the correct words.

If “boat” and “ship” often appear in similar contexts, the model learns that their vectors must be similar to produce similar probabilities.

Skip-Gram

- **Objective**

1. It predicts the context words (the surrounding words) based on a target word.
2. The model tries to predict each context word based on the target word.
3. Example:

Sentence: "The cat sat on the mat."

Target word: "sat"

Context: ["The", "cat", "on", "the", "mat"]

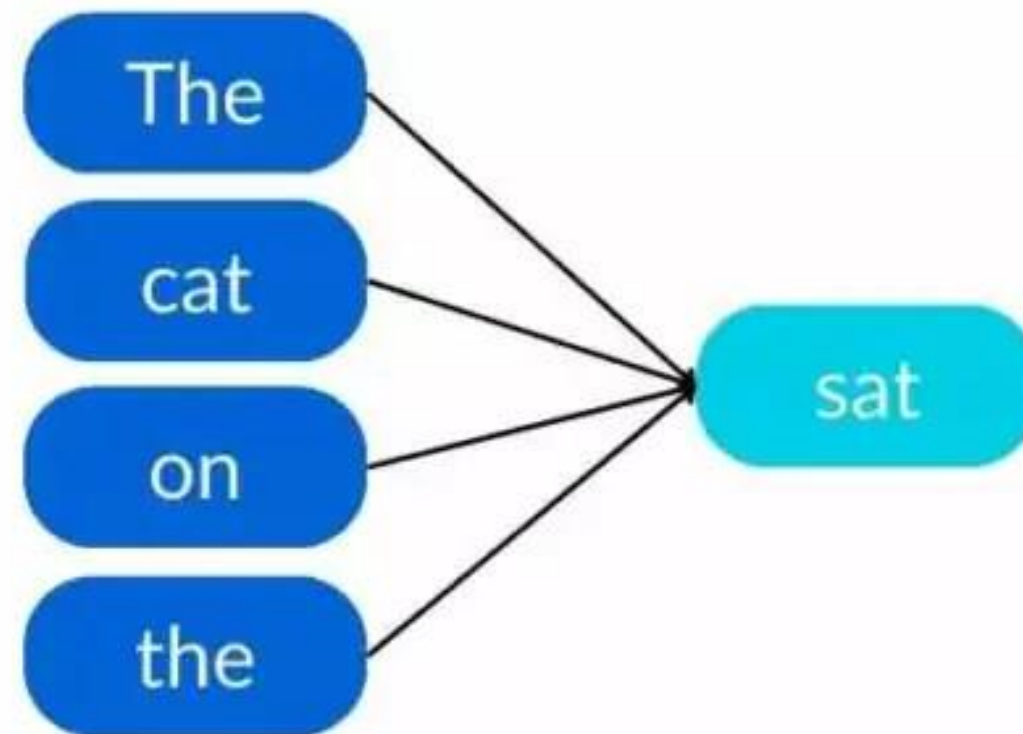
- **Advantages**

1. Skip-gram performs better with rare or infrequent words because it focuses on one word at a time and learns to predict its context.
2. It is particularly useful when the text corpus is large and contains many rare words.

Example Sentence: The cat **sat** on the mat.

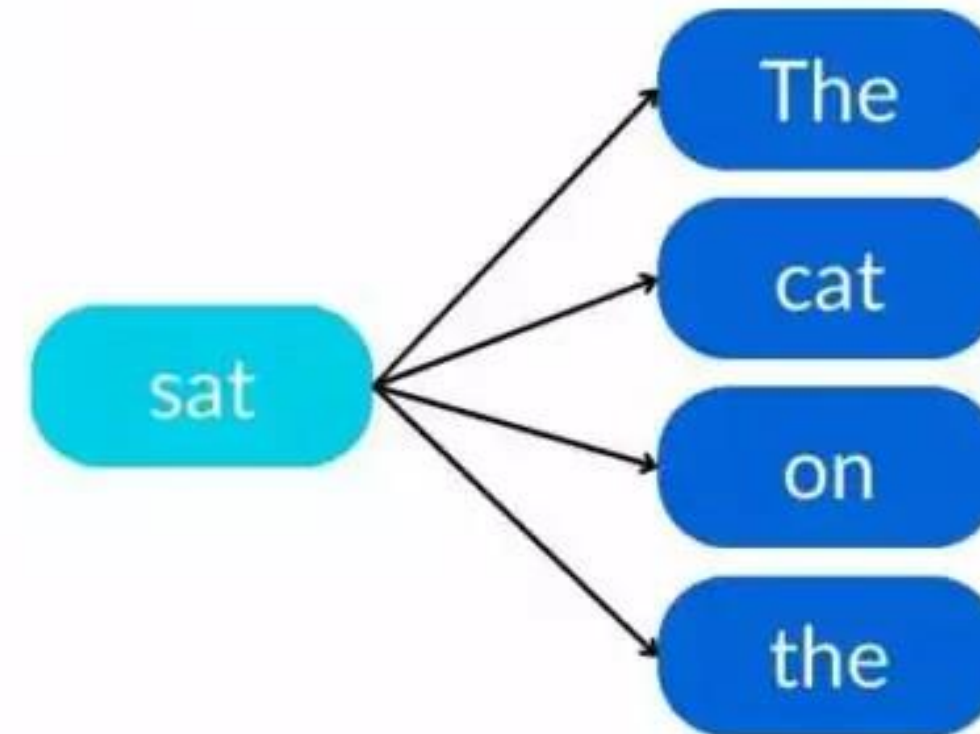
Continuous Bag-of-Words (CBOW)

Goal: Given context words,
predict the target word.



Skip-gram Model

Goal: Given a word,
predict the surrounding context words.



GPT-1

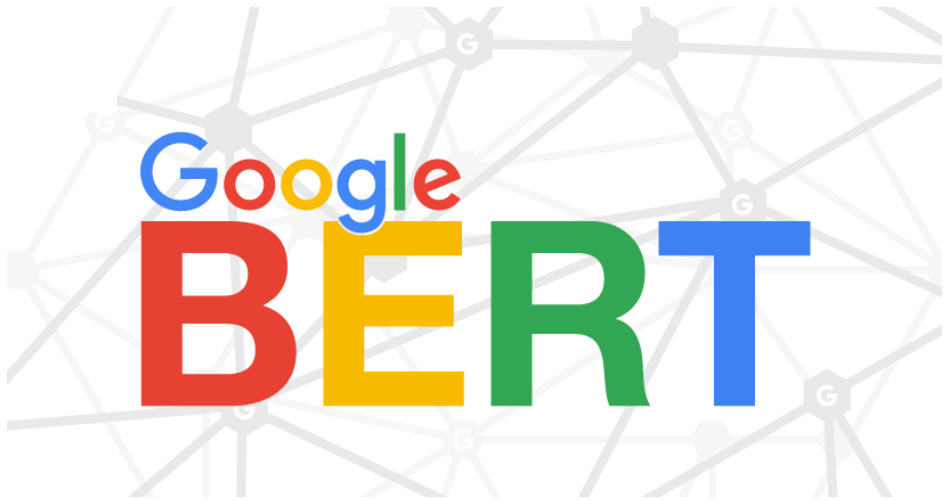


[PDF] Improving language understanding by generative pre-training
A Radford - 2018 - hayate-lab.com
Improving language understanding by generative pre-training Improving language understanding by generative pre-training ...
☆ Salva Cita Citato da 12485 Articoli correlati

ELMO



arXiv
https://arxiv.org > cs · Traduci questa pagina
[1802.05365] Deep contextualized word representations
di ME Peters · 2018 · Citato da 16432 -- We introduce a new type of deep contextualized word representation that models both (1) complex characteristics of word use (eg, syntax and semantics),



[CITAZIONE] Bert: Pre-training of deep bidirectional transformers for language understanding
J Devlin - arXiv preprint arXiv:1810.04805, 2018
☆ Salva Cita Citato da 125300 Articoli correlati

3.7 Bidirectional Encoder Representations from Transformers (BERT)

What is BERT

1. Developed by Google in 2018, it reads text simultaneously from left-to-right and right-to-left, capturing richer contextual meaning than previous models
2. Transformer architecture

Core Features

1. Pre-trained on massive amounts of text using two key tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)
2. Uses attention mechanisms to weigh the importance of different words based on their context within the sentence
3. Can understand subtle nuances in language by considering the full context around each word

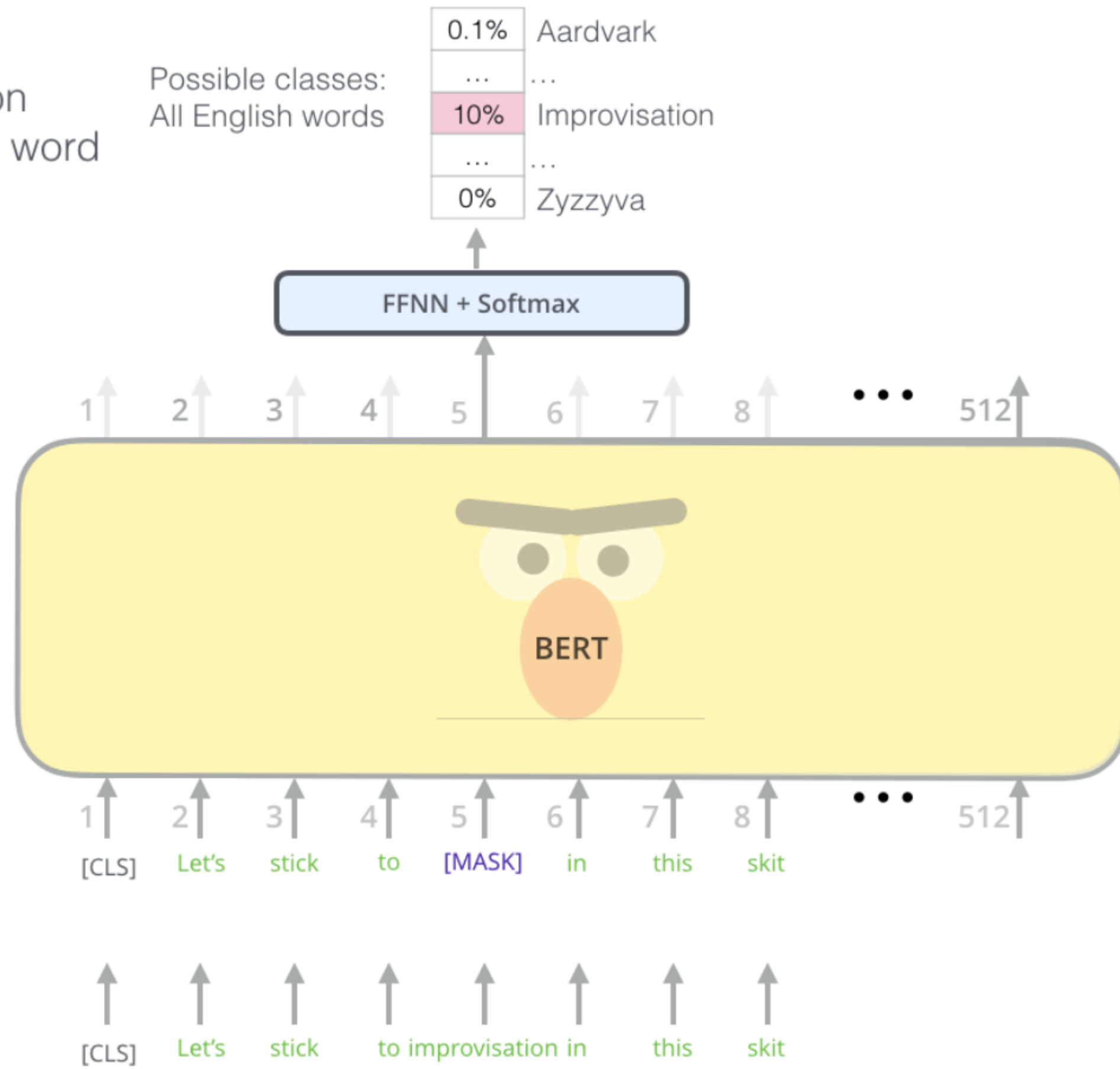


3.8 BERT

Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

Input





1

Semi-supervised Learning

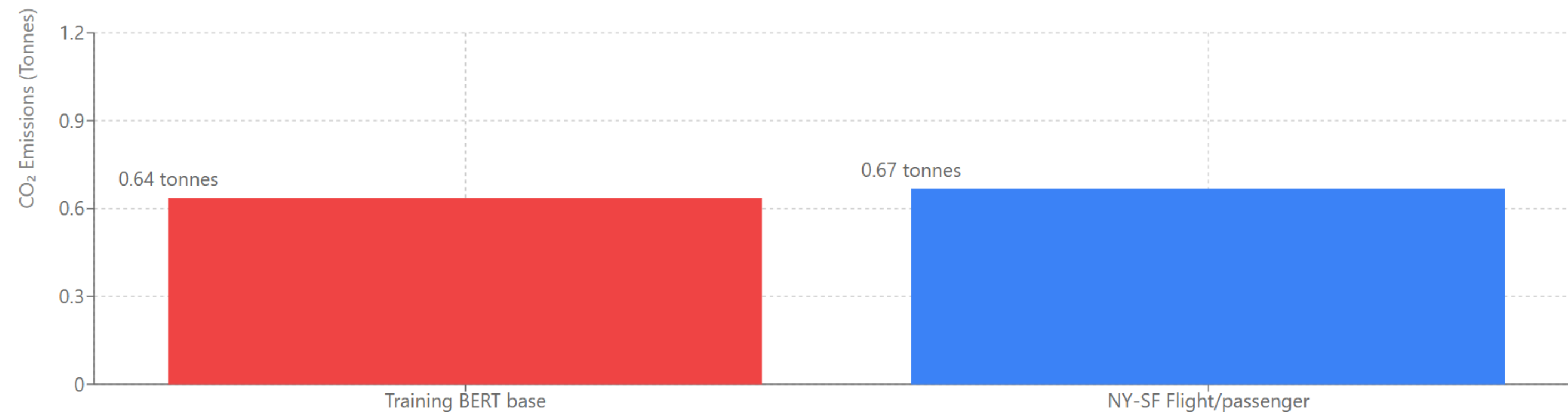
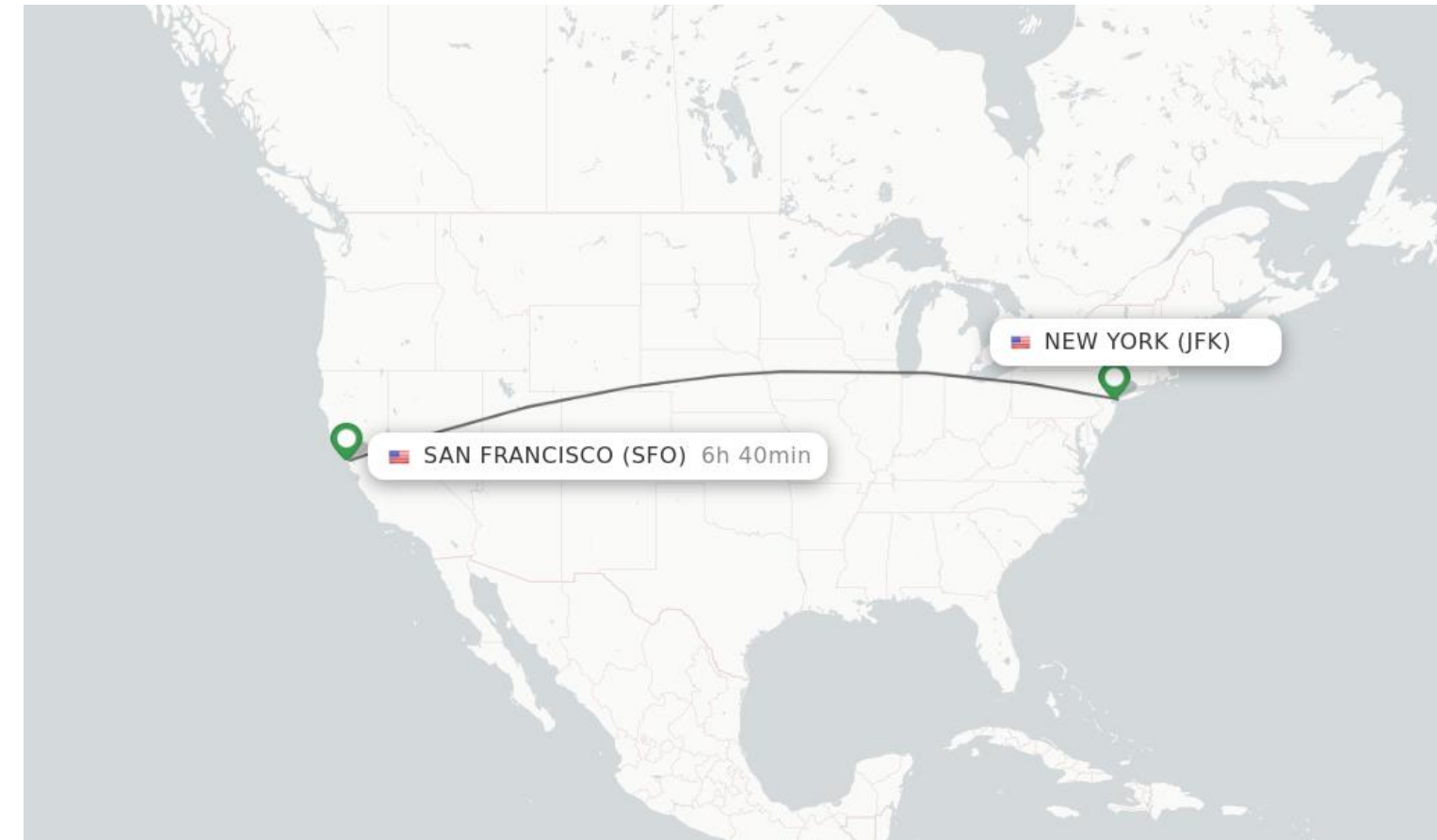
Model	BERT	
Dataset	Wikipedia	 WIKIPEDIA L'enciclopedia libera
Task	Predict the masked word	The ___ brown fox

Supervised Learning

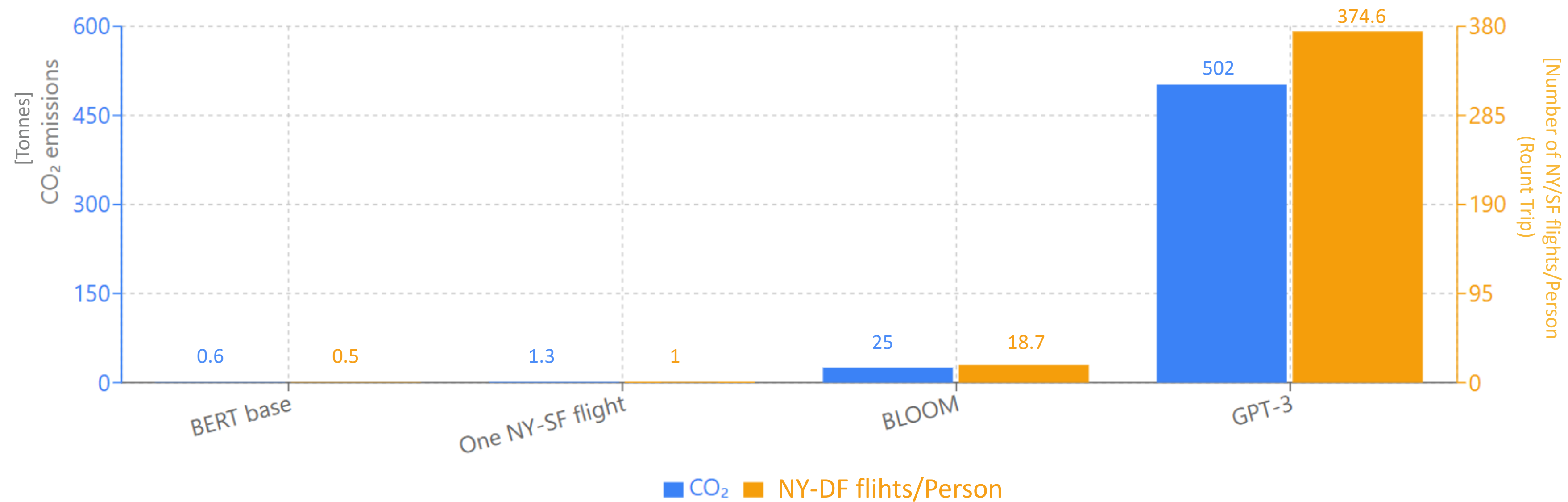
Model	Pretrained BERT	
Dataset	Text	Sentiment
	I hate this	 Not Happy
	I love this	 Happy
Task	Predict the sentiment	

2

What about environment?



3.92 BERT



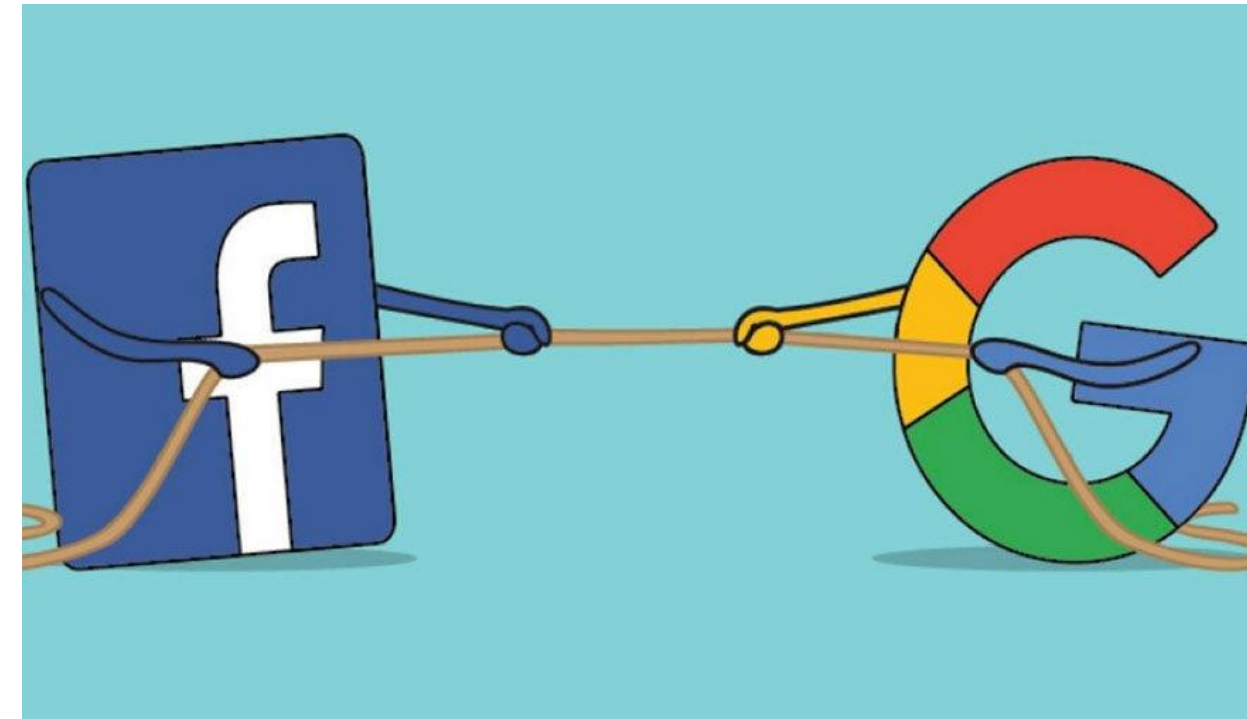
3.93 Beyond BERT

RoBERTa (2019)

- by Meta AI - improved BERT by modifying key training choices:
 1. Removed Next Sentence Prediction and trained with larger batches
 2. Transformer architecture

And also ...

- ALBERT (2019): made BERT more efficient with parameter sharing techniques, using far less memory while maintaining performance
- DistilBERT (2019): reduced BERT's size by 40% while retaining 97% of its performance through knowledge distillation
- DeBERTa (2020): enhanced BERT's attention mechanism with disentangled attention matrices and enhanced position encoding





Objective

- **Exploring Different Tokenizers**
 - Compare BERT (WordPiece) and RoBERTa (BPE) tokenization on various text inputs
 - Analyze tokenization of different text types
- **Understanding BERT Embeddings**
 - Generate and visualize contextual embeddings from BERT
 - Examine the shape and structure of BERT's output vectors
 - Compare embeddings for similar and different sentences
- **Practical Application**
 - Build a sentiment analysis pipeline using pre-trained
- **Dataset:**
 - Rotten Tomatoes movie reviews
 - Binary classification: positive (1) vs negative (0) reviews
 - Training set + Test set for evaluation



**Rotten
Tomatoes®**