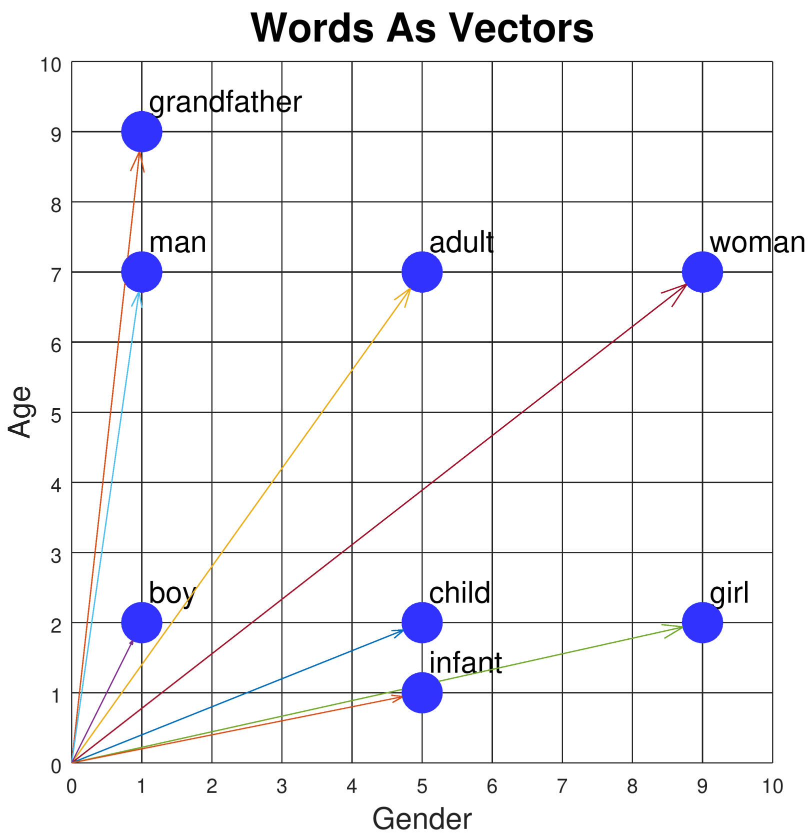
Word Embeddings

convert words → numbers (vectors)



Unlike Bag of Words or TF-IDF, these vectors:

- **Capture meaning** of the word,
- **Understand context** and similarity,
- Are **dense** and **compact** (not mostly zeros),
- Are learned from **large text data**.

> “Word embeddings give each word a unique, meaningful position in space.”

💡 **Each word is represented as a fixed-length vector (e.g., 100 or 300 dimensions) based on its context in large text corpora.**

🧱 Background: The Problem with Earlier Techniques

Technique	Major Limitation
One-Hot Encoding	No meaning, huge vectors
Bag of Words	No order, no context
TF-IDF	No similarity understanding

So all of them treat words like:

- "king" and "queen" → unrelated
- "India" and "Pakistan" → no connection
- "run" and "ran" → totally different

But **we know** those words are related.

Word Embeddings solve this.

✨ Example

Word	Embedding (simplified)
"king"	[0.21, -1.5, 0.3, ..., 0.8]
"queen"	[0.19, -1.4, 0.33, ..., 0.79]

Arithmetic relationships:

```
king - man + woman ≈ queen
```

Main Techniques:

Method	Description	
Word2Vec	Learns embeddings by predicting neighboring words	Google News (3 million words, 300-dim)
GloVe	Learns from global co-occurrence statistics	Wikipedia + Gigaword (6 billion tokens)
FastText	Like Word2Vec, but also looks at word-parts (subwords)	
ELMo	Learns context-dependent embeddings (deep LSTM)	
BERT	Learns context-aware embeddings using transformers	

🔁 How It Differs from Previous Methods

Feature	One-Hot / BoW / TF-IDF	Word Embedding
Fixed-size vector	✅ Yes	✅ Yes
Sparse	✅ Mostly zeros	❌ Dense
Understand similarity	❌ No	✅ Yes
Word meaning/context	❌ No	✅ Yes
Learns from data	❌ No (manual features)	✅ Yes (unsupervised)

📈 Use Cases

- Sentiment analysis
- Chatbots
- Question answering
- Document similarity
- Plagiarism detection
- Search engines

🧠 Trivia

- Word2Vec was created by Google in 2013.

- Word embeddings led to the “deep learning revolution” in NLP.
- Embeddings are like the **"first layer" of understanding** in NLP models.
- You can visualize them using **t-SNE** or **PCA** (to 2D or 3D).