4. Word2Vec Embeddings

In [6]:
```
# Import the Word2Vec model from gensim, which is used for learning word embeddings
from gensim.models import Word2Vec
```

In [7]:
```
# Define a small dataset where each sentence is a list of words
sentences = [
    ["artificial", "intelligence", "is", "cool"],
    ["machine", "learning", "is", "fun"],
    ["ai", "learning", "uses", "neural", "networks"]
]
```

In [8]:
```
# Train the Word2Vec model on the sentences
model = Word2Vec(
    sentences,
    vector_size=10,  # Each word will be represented by a 10-dimensional vector
    window=2,        # Context window size is 2 (words before and after)
    min_count=1,     # Include words that appear at least once
    sg=1             # Use skip-gram algorithm (good for small datasets)
    )
```

In [9]:
```
# Print the vector representation for the word 'learning'
print("Vector for 'learning':", model.wv['learning'])

# Print the words most similar to 'learning' according to the trained model
print("Most similar to 'learning':", model.wv.most_similar('learning'))
```

```
Vector for 'learning': [-0.00537069  0.00235848  0.05102572  0.09009185 -0.09303681 -0.07116417
  0.06458981  0.08972324 -0.05014562 -0.03762919]
Most similar to 'learning': [('is', 0.5435845851898193), ('artificial', 0.43179193139076233), ('cool', 0.3793115019798279), ('networks', 0.3
0033737421035767), ('neural', 0.10495670884847641), ('machine', -0.13116095960140228), ('fun', -0.18973511457443237), ('uses', -0.2241621911
5257263), ('ai', -0.2725953757762909), ('intelligence', -0.728771984577179)]
```

In [ ]: