Semantic Representation

Lecture 4: Word Embeddings

Harito

September 15, 2025

The Limits of Frequency-Based Models

TF-IDF is powerful, but it has major limitations:

- **High Dimensionality & Sparsity**: Vocabulary can have >100,000 terms, leading to huge, mostly zero vectors.
- **No Semantic Understanding**: The model has no idea that ‘car‘ is similar to ‘automobile‘, or that ‘king‘ and ‘queen‘ share a relationship. They are just different, unrelated dimensions in the vector space.

Goal: Can we create representations that capture *meaning*?

The Distributional Hypothesis

The core idea behind modern embeddings was formulated in 1957 by J.R. Firth:

"You shall know a word by the company it keeps."

- This means that words that appear in similar contexts are likely to have similar meanings.
- Example: If you see the sentences:
  - "I need to fill up my car with **gasoline**."
  - "I need to fill up my car with **petrol**."
- You can infer that 'gasoline' and 'petrol' are semantically similar.

Word Embeddings

The goal is to represent each word as a **dense**, **low-dimensional** vector that captures its meaning.

Sparse (TF-IDF)

- Dimension: 50,000+
- Mostly zeros
- Explicit: each dimension is a specific word

‘[0, 0, ..., 1, ..., 0, 0]‘

Dense (Word2Vec)

- Dimension: 50-300
- All non-zero values
- Implicit: dimensions don’t have an obvious meaning

‘[0.23, -0.1, 0.6, ..., 0.9]‘

Similarity between words is measured by the **cosine similarity** between their vectors.

How Word2Vec Learns (Intuition)

Word2Vec is a predictive model. It's a neural network that tries to solve a fake task. In the process of learning to solve the task, it learns excellent word embeddings.

Example Task (Skip-gram model): Given a center word, predict the words that appear in its context window.

The network's internal weights, learned during this prediction task, become the word embeddings!

The Magic of Word Embeddings

Because embeddings capture meaning, we can perform mathematical operations on them that have semantic results.
The most famous example:

$$\text{vec}('king') - \text{vec}('man') + \text{vec}('woman') \approx \text{vec}('queen')$$

This shows the model has learned the concept of gender and royalty.

Using Pre-trained Models

Training a Word2Vec model from scratch requires a massive amount of text data (billions of words) and significant computation time.

Common Practice

Download and use models that have already been trained by researchers on huge datasets (e.g., Google News, Wikipedia, Common Crawl).

The ‘gensim‘ library in Python makes this incredibly easy: python import gensim.downloader as api
Load pre-trained vectors (e.g., 50-dim GloVe) This will download the model if not already present model = api.load('glove-wiki-gigaword-50')
Get vector for a word $vec_king = model['king']$
Get most similar words similar = $model.most_similar('king')$

From Word to Document Embeddings

How do we get a single vector for a whole sentence or document?

Simple but effective method: Averaging

1. Tokenize the document. 2. Get the word vector for each token. 3. Take the element-wise average of all the vectors.

This average vector provides a good summary of the document's overall meaning.

Next Steps

Time for Lab 4!

Objective:

- Load a pre-trained Word2Vec model using ‘gensim‘.
- Explore word similarities.
- Implement document embedding by averaging word vectors.