Word Embeddings and Modern Language Models

From Static Vectors to Contextualized Representations

Introduction to Embeddings

Embeddings map words or tokens to vectors in continuous space, capturing semantic and syntactic relationships. Imagine placing words on a map—words with similar meanings end up close by. These representations power many NLP tasks by converting text into numbers that models can understand.

1. GloVe (Global Vectors)

Concept: Learns word vectors by factorizing global co-occurrence counts in a corpus.
Key idea: Words that co-occur frequently share similar contexts, so their vectors should have small distance.

Equation:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2$$

- X_{ij}: co-occurrence count of words i and j [1].

Reflection:

- Captures global statistics.
- Produces static embeddings (same vector for a word in any context).
- Good for semantic similarity and as input to downstream models.

2. Word2Vec: Skip-Gram and CBOW

Concept: Predicts surrounding words (context) given a target word (skip-gram) or vice versa (CBOW).

Skip-Gram Example (Pseudo-Code):

```
# w: target word index, context: list of context indices
emb_w = W[w]                          # get target embedding
scores = emb_w @ W_context.T         # score each context word
loss = cross_entropy(scores, context) # train to predict actual context words
```

Reflection:

- Fast to train; captures local context.
- Produces static embeddings; context-independent.
- Effective for semantic and syntactic tasks.

3. FastText

Concept: Extends Word2Vec by representing each word as a bag of character n-grams.
Key idea: Handles rare and out-of-vocabulary words by composing subword vectors.

Reflection:

- Better for morphologically rich languages.
- Still static embeddings, but more robust to unseen words.

4. ELMo (Embeddings from Language Models)

Concept: Contextual embeddings from a pretrained bidirectional LSTM.
Key idea: Word representations depend on the entire sentence.

Equation:

$$\mathrm{ELMo}_k = \gamma \sum_{j=0}^{L} s_j h_{k,j}$$

- $h_{k,j}$: hidden state of layer j at position k [1].

Reflection:

- Produces different vectors for the same word in different contexts.
- Improves tasks like question answering and named entity recognition.

5. BERT (Bidirectional Encoder Representations)

Concept: Deep bidirectional Transformer pre-trained with masked language and next-sentence prediction.
Key idea: Learns context from both left and right simultaneously.

Usage Example:

```
from transformers import BertTokenizer, BertModel

tok = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
inputs = tok("The cat sat on the mat", return_tensors='pt')
outputs = model(**inputs)
embeddings = outputs.last_hidden_state  # contextualized token embeddings
```

Reflection:

- Powerful contextual embeddings; static fine-tuning yields state-of-the-art on many tasks.
- Computationally heavy; requires GPUs.

6. RoBERTa

Concept: Improved BERT with more data, longer training, and removal of next-sentence task.
Reflection:

- Stronger performance on benchmarks.
- Same usage as BERT with different pretrained weights.

7. GPT Embeddings

Concept: Unidirectional Transformer trained as a language model.
Key idea: Predict next token given previous context.

Reflection:

- Good for generation tasks; embeddings reflect preceding context.
- Less suited for tasks needing full bidirectional context.

8. T5 (Text-to-Text Transfer Transformer)

Concept: Unified text-to-text framework: all tasks cast as text generation.
Key idea: Pre-trained on a multi-task mixture of unsupervised and supervised tasks.

Reflection:

- Flexibility: translation, summarization, Q&A framed as text input → text output.
- Large model size and compute demands.

Overall Reflection

- **Static vs. Contextual:** Word2Vec, GloVe, FastText produce static vectors; ELMo, BERT, RoBERTa, GPT, and T5 produce contextual embeddings.
- **Bidirectional vs. Unidirectional:** BERT family uses bidirectional attention; GPT is unidirectional.
- **Task framing:** T5's text-to-text simplifies multi-task pipelines.
- **Computational trade-offs:** More context and layers mean better performance but higher resource needs.

Download the PDF above for a fully formatted guide on word embeddings and modern pretrained models.