Comenius University in Bratislava

Sparse Sentence Embeddings

Master's Thesis

Bc. Tomáš Varga
Supervisor: Mgr. Vladimír Boža, PhD.

2026

Introduction & Motivation

Sentence Embeddings

- Map sentences → fixed dimensional vectors
- Semantic similarity = geometric proximity
- Foundation of modern NLP

Applications

- Semantic search engines
- Information retrieval
- Text clustering & classification
- Question answering systems

Key Models: Sentence-BERT (2019)

Built on transformer architectures, produces high-quality dense vector representations that effectively capture semantic meaning

Background: From Words to Sentences

Tokenization: The First Critical Step

Neural models don't process raw text. They operate on discrete units called tokens

Word-level

Pros:
Intuitive, semantic meaning
Cons:
Large vocabulary

Character-level

Pros:
Small vocabulary
Cons:
Long sequences, less semantic

Subword (BPE)

Pros:
Balanced approach, handles rare words
Cons:
Standard for modern LLMs

The Problem

Representational Capacity Bottleneck

As the number of distinct semantic concepts grows, fixed-dimensional dense vectors struggle to reliably separate them.
Weller et al. (2025)

Large-Scale Retrieval

- Millions of documents
- Subtle semantic distinctions
- Degraded retrieval performance

Theoretical Limitations

- Inherent scaling limits
- Reduced precision
- Fixed dimensionality constraint

The Alternative: Sparse Embeddings

Dense vs Sparse

Dense Embeddings	Sparse Embeddings
[0.23, -0.45, 0.12, 0.87, -0.33, ...]	[0, 0, 0.92, 0, 0, 1.45, 0, ...]
All dimensions carry information	Most values are exactly zero

Current Approach: SPLADE

- Uses Masked Language Model (MLM) predictions
- Assesses vocabulary token importance
- Limitation: Ties sparse embeddings to original vocabulary tokens

Our Approach: Thesis Goal

Design and evaluate a sparse pooling layer for BERT-like models

Directly learns to construct sparse sentence embeddings from token embeddings without using the MLM prediction head

Hypothesis

1. Learnable sparse pooling can produce sparse embeddings
2. Embeddings remain semantically meaningful
3. Independence from vocabulary-based projections

Experimental Setup

Base Encoder

- bert-base-uncased
- 12 transformer layers
- 768-dim embeddings
- Constant across experiments

Training Data

- MNLI dataset
- Entailment pairs only
- 50,000 training pairs

Evaluation

- STS-B validation set
- 1,500 sentence pairs
- Spearman correlation
- Sparsity statistics

Pooling Strategies

Baseline

- Mean pooling: average all tokens
- CLS pooling: use [CLS] token
- Max pooling: element-wise maximum

Learnable Pooling

- Attention: learned token weights
- Weighted: learned dimension weights
- Hierarchical: multi-head self-attention

Sparse Pooling (Our Focus)

Top-K Sparse Pooling

- Mean pool → Linear projection
- Keep only K largest dimensions
- Zero out all other dimensions
- $K \in \{50, 200\}$ out of 768

Attention Pooling Formula: $\alpha_i = w^T h_i$ → $a_i = \text{softmax}(\alpha_i)$ → $s = \Sigma\, a_i h_i$

where w is learnable weight vector, h are token embeddings, s is sentence embedding

Results & Analysis

Method	Type	Spearman	Sparsity
Attention (full)	Learnable	**0.811**	0%
Hierarchical (frozen)	Learnable	0.694	0%
Max pooling	Baseline	0.621	0%
Mean pooling	Baseline	0.593	0%
Sparse Top-K (k=200)	Sparse	0.593	**74.0%**
Sparse Top-K (k=50)	Sparse	0.580	**93.5%**
CLS pooling	Baseline	0.317	0%

Key Findings

Best: Attention (full training)

Spearman 0.811. Significantly outperforms all other methods

Sparse: Competitive at 74% sparsity

Top-K (k=200) matches Mean pooling while using only 26% of dimensions

Conclusion & Future Work

Summary

- Implemented benchmarking framework
- Compared baseline, learnable, sparse pooling
- Top-K sparse pooling achieves sparsity
- Trade-off: sparsity vs quality

Future Directions

- Different sparsity mechanisms
- Larger-scale evaluation
- Comparison with SPLADE variants

Thank you for your attention