Title: Semantic Vector Field Architecture: A Structured, Low-Parameter Alternative to LLMs

Abstract: This whitepaper presents a new language model architecture based on semantic vectors and positions in multidimensional Cartesian spaces. Unlike traditional transformer-based LLMs, this approach focuses on the explicit semantic organization of tokens and their conceptual relationships, using fewer parameters and a simpler neural structure. The result is a more interpretable, efficient model with greater generalization capabilities without literal memorization.

1. Introduction

- Problems of traditional LLMs: high computational cost, opacity, memorization of examples.
- Proposal: Architecture based on vectorized tokens distributed in semantic Cartesian spaces.
- Introductory example: A model with 370,000 tokens and 500 Cartesian spaces would have about 185 million parameters, capable of understanding practically all words in the Portuguese language with high conceptual and semantic accuracy.

2. Token Representation

- Each token has:
  - Numeric ID (int8 or int16)
  - Meaning vector: IDs of other tokens that conceptually describe it
  - Positions in N Cartesian spaces (e.g., verbs, nouns, numbers, etc.)
- The vector functions as an internal and mutable conceptual dictionary, which can be initialized by another AI and refined during training.

2.1 Token Representation Examples

- Token: "bola" (ball)
  - ID: 102
  - Meaning Vector: [201 (object), 305 (spherical), 410 (sport)]
  - Positions:
    - Space 1 (Grammatical Category): (x=10, y=5)
    - Space 2 (Thematic Domain): (x=3, y=8)
    - Space 3 (Physical Form): (x=7, y=2)

- Token: "elétron" (electron)
  - ID: 150
    - Meaning Vector: [311 (particle), 325 (energy), 408 (charge), 420 (negative), 503 (electromagnetism), 610 (physics), 702 (quantum)]

3. Semantic Cartesian Spaces

- Each space represents a semantic dimension (e.g., time, emotion, syntactic function)
- Tokens are positioned in coordinates reflecting their use and meaning
- Hundreds of spaces possible, with 256x256 or more positions per dimension

4. Neural Network Architecture

- Layer 1: Fixed embeddings based on space coordinates
- Layer 2: Semantic comparator between neighboring tokens
- Layer 3: Token clusterer with correlated positions
- Layer 4: Context and inference based on local history
- Layer 5: Final decision on the next token

- The model does not use traditional backpropagation. Updates occur based on logs of hits and errors, frequencies, positions, and sequential context per epoch.

5. Learning and Updating

- Each epoch logs:
  - Token occurrences
  - Relative and absolute positions
  - Token sequence and frequent transitions
  - Prediction errors and hits
- At the end of the epoch:
  - Meaning vectors may be expanded or restructured
  - Positions in Cartesian spaces are adjusted
  - Epoch data is discarded after use
- Possible use of auxiliary AI to construct initial vectors based on dictionaries and Wikipedia

6. Training Data

- Highly clean, structured, and varied data is essential
- The model handles medium to long texts well due to its structured semantic architecture
- Token coverage is critical: unknown tokens can disrupt understanding
- Example: A model with 370,000 tokens would have a low chance of encountering unknown terms

7. Comparison with Transformers/LLMs

Criterion	Vector Model	Transformer LLM

| Parameters | Low (10-200M) | High (100M-175B) |

| Interpretability | High | Low |

| Literal Memorization| Low | High |

| Training Cost | Low | High |

| Generalization | High with good semantics | High with big data |

8. Advantages of the Approach

- Significant reduction in model size
- High conceptual intelligibility
- Explicit control over associations and semantic vectors
- Trainable on mobile devices
- Capable of understanding meaning and context with less data

9. Limitations

- Reduced capacity for highly abstract and technical concepts
- Accuracy depends on space structure and coverage
- Requires auxiliary AI for initial semantic description

Appendix A: Experiments and Technical Notes

A.1 Example Model Configuration

- 512 semantic spaces, 6 functional layers, 834 total neurons
- 50,000 tokens with unique IDs, vectors and spatial coordinates

A.2 Training Phases

1. Pre-training (42MB of basic text)
2. Main training (Portuguese Wikipedia)
3. Post-training (full Wiktionary)
4. Fine-tuning (prompts and structured inputs)

A.3 Gradient-based Learning

- Updates based on Euclidean distance between prediction and ground truth
- Learning rate adapts per token (e.g., $0.9 \rightarrow 0.01$)

A.4 Inference Example

Input: 'gravity affects time'

$\rightarrow$ Output candidates: ['relativity', 'Einstein', 'time dilation']

A.5 Multimodal Extension

- Images: pattern vectors like 'eye', 'face'
- Audio: phoneme structures mapped to semantic coordinates

A.6 Efficiency Comparison

SVF: ~30,000 tokens/s, ~$0.001/million tokens

GPT-3.5: ~3,000 tokens/s, ~$0.60/million tokens

A.7 Observed Limitations

- Sensitive to unknown tokens
- Requires large token vocabulary for robustness

A.8 Future Potential

- Video input compatibility
- Symbolic-reasoning fusion
- Multilingual embeddings