Demonstrating Foundational AI/ML Principles

Donaven Bruce

West Los Angeles College

December 8, 2024

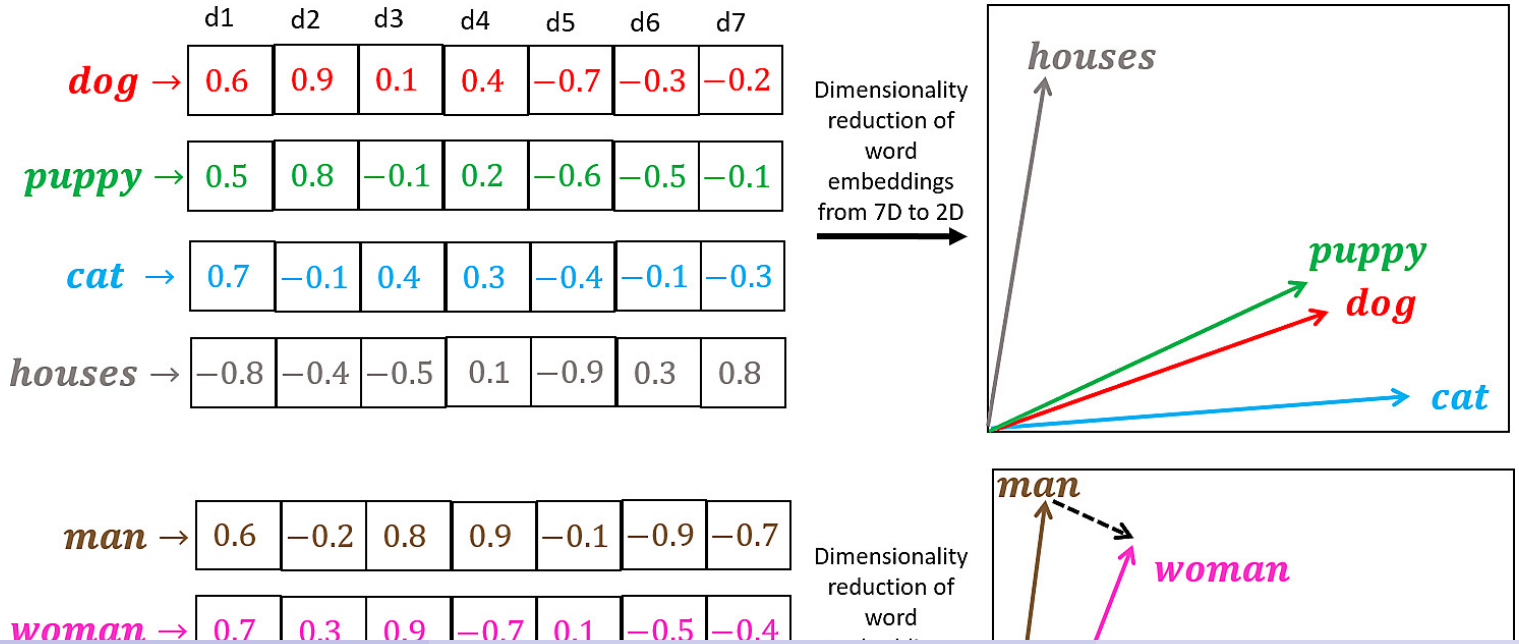
Introduction

Goal: Showcase how the Word Vector Visualizer project demonstrates foundational AI/ML concepts.

- **Challenge:** Improve pre-trained word embeddings using retrofitting.
- **Focus:** Align vectors with external knowledge while preserving original structure.
- **Why it Matters:** Enhanced embeddings improve downstream tasks like:
  - Sentiment analysis
  - Machine translation
  - Search engines

Problem Overview

- **Word Embeddings:** Numerical representations of words in a continuous vector space.
- **Semantic Gaps:** Pre-trained embeddings often fail to capture certain relationships (e.g., "cat" vs. "kitten").
- **Solution:** Retrofitting:
  - Adjust vectors to reflect semantic relationships defined in a lexicon.
  - Iterative updates to converge on improved representations.
- **Visualization:** The figure below demonstrates word embeddings as vectors in reduced dimensions, showing semantic relationships.



First Principles of AI/ML

Key Concepts Demonstrated:

- **Vector Representations:** Words encoded as multi-dimensional numerical arrays.
- **Cosine Similarity:** Measures closeness between word vectors.

$$\text{cosine\_similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

- **Iterative Optimization:** Gradual refinement of word vectors over multiple iterations:

$$\mathbf{v}_{\text{new}} = \frac{\alpha \mathbf{v}_{\text{orig}} + \beta \sum_i \mathbf{v}_i}{\alpha + \beta n}$$

- **Normalization:** Ensures consistent vector magnitudes.
- **Semantic Frames:** Represents conceptual structures describing events, relationships, or entities and their participants.

Retrofitting Workflow

Steps in the Retrofitting Process:

1. **Input:** Load pre-trained word embeddings and a lexicon mapping words to neighbors.
2. **Initialization:** Use original embeddings as starting points for refinement.
3. **Iteration:**
   - Update vectors using a weighted average of original and neighbor vectors.
   - Evaluate convergence by measuring similarity changes.
4. **Normalization:** Scale vectors to ensure uniform magnitudes.
5. **Output:** Generate retrofitted embeddings aligned with lexical knowledge.

Results and Evaluation

Quantitative Metrics:

- Pre- and post-retrofitting cosine similarity comparison.
- Improved alignment for tracked words (e.g., "cat", "kitten").

Visualization:

- Scatter plots of word vectors before and after retrofitting.

Summary of Outputs

Important Results:

Word Pair	Pre-Similarity	Post-Similarity	Improvement
kitten, cat	0.0862	0.6596	+0.5734
bird, cat	-0.2670	0.5411	+0.8081
dog, puppy	-0.0807	0.6656	+0.7463
cat, dog	-0.1114	0.6736	+0.7850

Table: Cosine Similarity Improvements

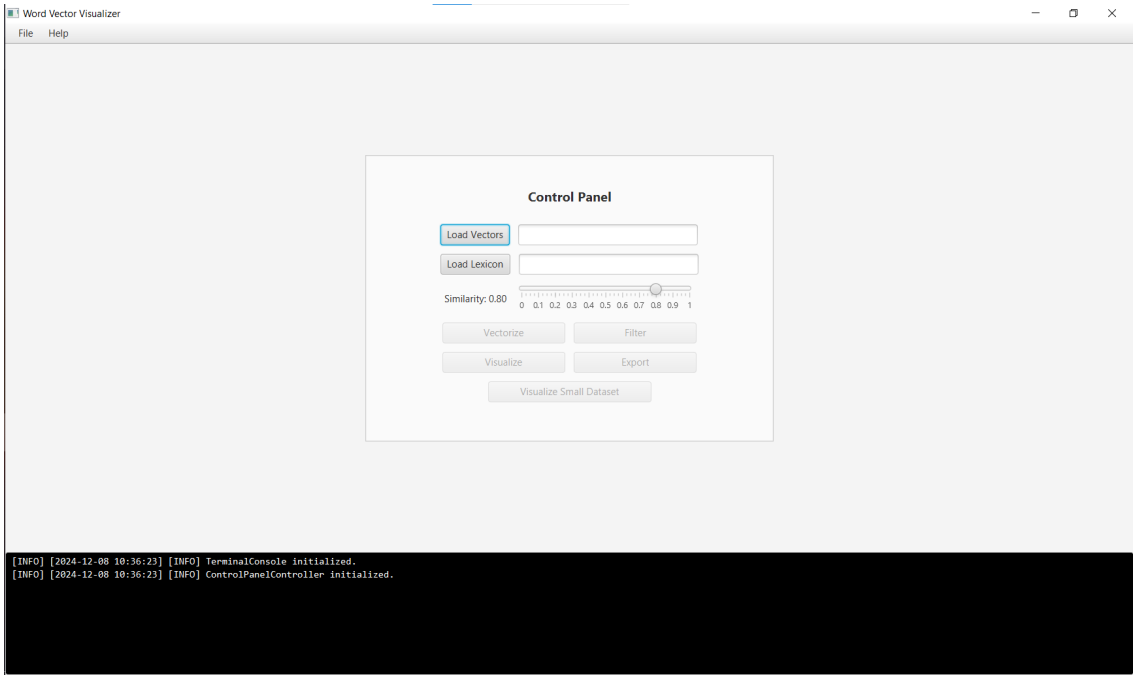
Top Words Meeting Threshold:

- rabbit (1.0000), rock (1.0000), fish (1.0000), tree (1.0000)

Human-Computer Interaction

GUI Features:

- File upload for embeddings and lexicons.
- Adjustable similarity thresholds via sliders.
- Interactive scatter plot visualizations.



Broader Implications

Applications in AI:

- Improved embeddings enhance tasks like NLP, search engines, and conversational agents.
- Scalable for larger datasets using parallelization frameworks like CUDA.

Research Extensions:

- Apply retrofitting principles to sentence or graph embeddings.
- Integrate with deep learning pipelines for dynamic adjustments.

Conclusion

Takeaways:

- Demonstrated AI/ML principles: vector spaces, similarity metrics, optimization.
- Real-world impact: enhanced word embeddings improve downstream NLP tasks.
- Future work: scaling, integration with advanced models, and user-friendly improvements.

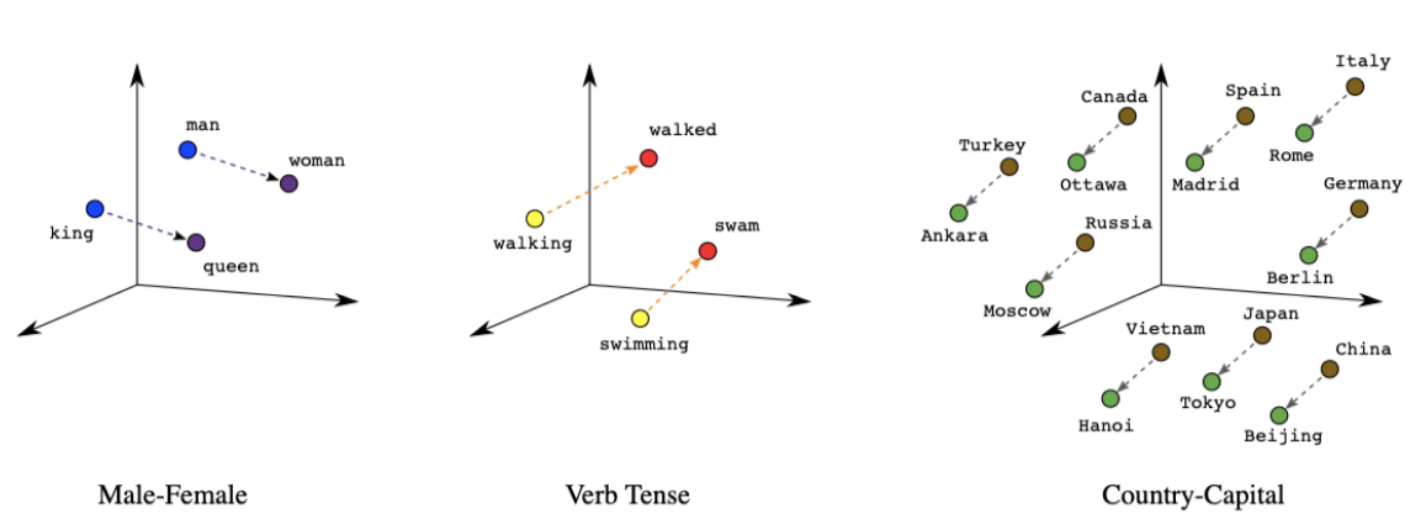


Image Source: (Embeddings: Translating to a Lower-Dimensional Space) by Google.

Thank you! Questions?