Removing hallucinations – embeddings perspective

Ritesh Modi

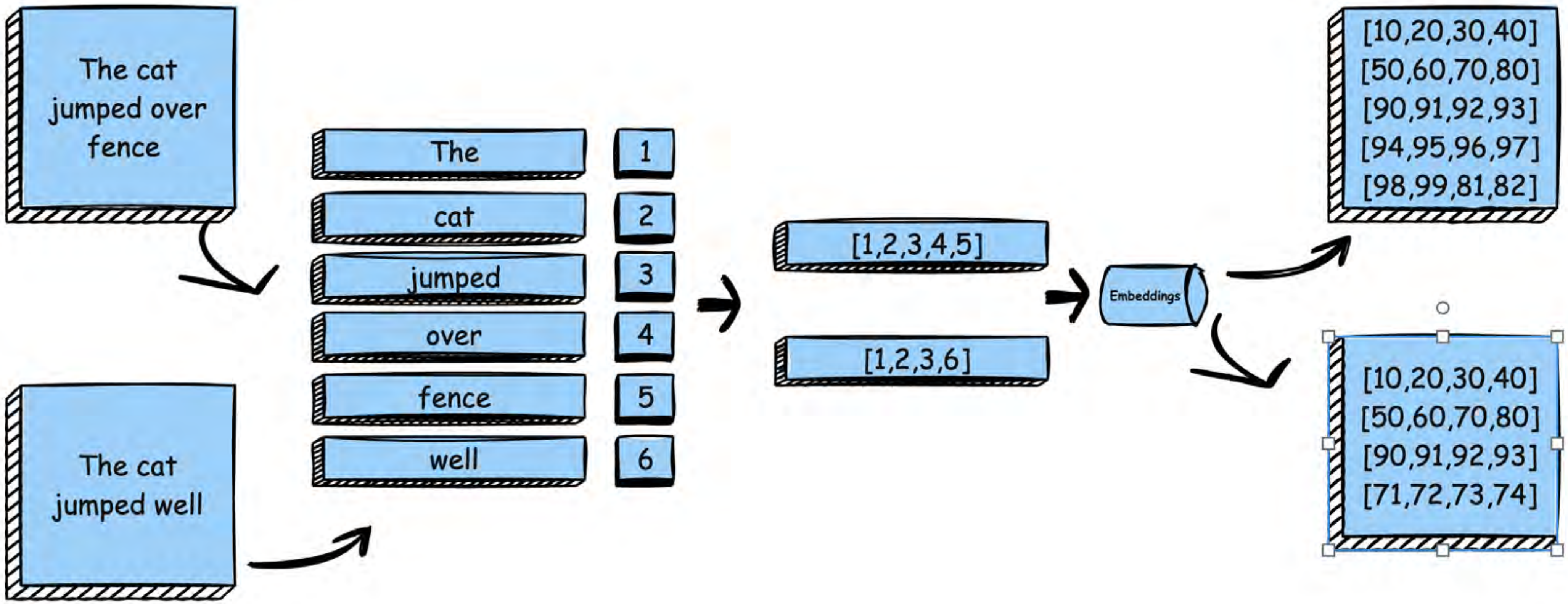
Principal Engineer, Microsoft

What Are Embeddings?

- Dense numerical vectors that translate human concepts into mathematical spaces
- Similar concepts positioned closer together in the vector space
- Enable mathematical operations on meaning (king - man + woman ≈ queen)
- Powers AI applications like search, recommendations, and language understanding
- Bridge between human language and machine processing

Why We Shouldn't Trust Embeddings from Foundation Models without Experimentation and Evaluation

Embedding process



Important Concepts - **Dimensionality**

Number of floating-point values in each embedding vector

Typical ranges: 32-4096 dimensions (100-768 most common)

Insights:

- Higher dimensions → better semantic capture but more computation
- Lower dimensions → faster processing but potential information loss

Selection factors: Dataset size, complexity of semantic relationships, computational resources

Examples: BERT-base (768 dimensions), GPT embeddings (1536 dimensions), Word2Vec (300 dimensions)

Important Concepts – **Max Sequence Length**

Maximum number of tokens an embedding model can process at once

Importance: Determines the context window for understanding relationships

Typical ranges:

- Word embeddings: single tokens
- Sentence models: 128-512 tokens
- Document models: 1024-8192+ tokens

Context truncation: Longer sequences get cut off, potentially losing critical information

Computational impact: Quadratic relationship with attention-based models ($O(n^2)$)

Important Concepts – **vocabulary + Size**

Number of unique tokens the embedding model recognizes

Relevance: Affects tokenization granularity and out-of-vocabulary handling

Typical sizes: 30,000-50,000 tokens for language models

Trade-offs: Larger vocabularies capture more nuance but increase model size

Embeddings - Use cases

Semantic Search

- Finding relevant documents beyond keyword matching
- Retrieving information based on meaning rather than exact terms
- Powering RAG (Retrieval-Augmented Generation) systems

Recommendation Systems

- Product recommendations in e-commerce
- Content recommendations (articles, videos, music)
- "People also viewed" features based on item similarity

Natural Language Processing

- Text classification (sentiment analysis, topic categorization)
- Named entity recognition
- Question answering systems

Information Retrieval

- Document clustering and organization
- Duplicate detection
- Contextual search filtering

Embedding Comparisons

Cosine Similarity	Euclidean Distance	Manhattan Distance (L1 Norm)	Dot Product
• Measures the angle between vectors, not magnitude • Range: -1 (opposite) to 1 (identical) • Advantage: Normalizes for vector length, focusing on direction • Popular for text embeddings where magnitude is less important	• Straight-line distance between points in vector space • Intuitive for physical space analogies • Works well when magnitude matters • Less common for high-dimensional embeddings due to "curse of dimensionality"	• Sum of absolute differences between vector components • More robust to outliers than Euclidean • Useful in grid-like spaces	• Simple multiplication and sum of corresponding values • Not normalized, so sensitive to magnitude • Quick computation but less interpretable

Find Embeddings using OpenAI and SentenceTransformers

Compare Embeddings

A small Quiz

Statement1: "The treatment was completely ineffective against the disease."
Statement2: "The treatment was absolutely effective against the disease."

Statement 1: "Place the specimen in the refrigerator at exactly 4°C."
Statement 2: "The sample must be stored at precisely 4 degrees Celsius in the cooling unit."

Statement 1: " The results showed statistical significance"
Statement 2: " The findings indicated a significant effect"

Statement 1: “The patient shows hypertension.”
Statement 2: " The patient shows hypotension."

Why is this happening???

Bag-of-words influence:	• Many embedding models, especially earlier ones, are influenced by bag-of-words approaches where word presence matters more than word order or negations.
Shared vocabulary:	• Both sentences share almost all their tokens ("the", "movie", "was", "good", "and", "I", "enjoyed", "it") - only differing by the word "not".
Contextual similarities:	• The overall context of both sentences is about movie watching and enjoyment.
Positivity bias:	• Both sentences contain the positive sentiment word "enjoyed" which contributes strongly to the vector representation.
Negation handling weakness:	• Embedding models often struggle with negations ("not good") because negation fundamentally changes meaning while only adding/changing minimal text.

Scenario’s

1. Capitalization 2. Whitespace variations 3. Negations 4. Special characters 5. Word order 6. Synonyms and paraphrasing 7. Spelling errors and typos 8. Named entity variations 9. Grammatical variations 10. Fille ar words and verbosity 11. Contractions and expansions 12. Named entity variations 13. Missing information 14. Grammatical variations 15. Language mixing and code-switching 16. temporal_direction 17. quant_threshold 18. hypo_fact 19. scaler_inversion 20. medicine_domain_based 21. legal_domain_based 22. attribution 23. unit of time 24. unit_conversion 25. speed and miles 26. exact vs range 27. domain_significance 28. Percentages 29. Date and time 30. statistics 31. Counterfactual 32. Taxonomic 33. Procedural 34. Comparison 35. Metaphorical 36. Presupposition 37. References 38. Extensional

Solutions

Solution

1. There is not one solution, and these solutions can be combined or used in isolation

2. Use a Model that is based on your domain i.e. use domain specific model

3. Use preprocessing steps on your data to add additional context.
    1. Add entity type along with entity i.e., Add Brand or Fruit along with Apple depending on context
    2. Check for abbreviations
    3. Expand numbers into words
    4. Expand date-time into sentences

4. Fine-tune an existing foundation model

Fine-tuning

Steps for fine-tuning

Set up the environment

Preparing training, validation and test data for fine-tuning model

Pick a base model to fine-tune from huggingface

Provide configuration and hyper-parameters for fine-tuning the model

Train the model and save it

Evaluate the model and compare with base model

Choose a Model

Domain alignment: Choose a base model that's conceptually close to your target domain. For medical text, clinical BERT variants may perform better than general models.

Size vs. performance trade-off: Larger models generally perform better but require more compute resources for fine-tuning and deployment.

Inference speed requirements: If you need real-time embeddings in production, a smaller model might be preferable despite slightly lower quality.

Training stability: Some models fine-tune more reliably than others. Models from the sentence-transformers library are specifically designed for fine-tuning.

Community support: Models with active maintenance and large user bases tend to have better documentation and fewer unexpected behaviors.

Choose a Model

Model	Size	Strengths	Best for
sentence-transformers/all-MiniLM-L6-v2	80MB	Fast, compact, good general performance	Resource-constrained environments, mobile applications
sentence-transformers/all-mpnet-base-v2	420MB	Excellent general performance, handles longer text	General-purpose embeddings with good quality-speed tradeoff
sentence-transformers/multi-qa-mpnet-base-dot-v1	420MB	Optimized for retrieval, handles questions and answers	RAG systems, Q&A applications
intfloat/e5-large-v2	1.3GB	State-of-the-art performance, rich semantic understanding	When quality is the top priority
BAAI/bge-large-en-v1.5	1.3GB	Strong on retrieval benchmarks, works well with Chinese and English	Multilingual applications, search systems

Configuration

Important hyper parameters to tune for effective fine-tuning

Helps avoid over-fitting and under-fitting

- **train_objectives**
- evaluator
- epochs
- warmup_steps
- optimizer_params: learning rate and weight decay.
- scheduler
- output_path
- evaluation_steps
- save_best_model
- use_amp
- checkpoint_path
- checkpoint_save_steps
- checkpoint_save_total_limit
- show_progress_bar

Key Takeaways

- know your data well
- Identify the model to use
- **Embedding models aren't magic**
- **Data quality trumps quantity**
- fine-tune if it is required
- **Systematic evaluation is essential -**
- **Fine-tuning is iterative.**

Questions

https://www.linkedin.com/in/ritesh-modi

https://github.com/ritesh-modi

https://www.x.com/automationnext