Exploring Bias in the Design of AI Applications

Eleftheria Polychronidou, PhD
Technical Product Owner @ Applied AI team, Causaly
Generative AI (GenAI), can create diverse content based on learned patterns. Large Language Models (LLMs), a form of GenAI, specialize in generating human-like text by learning from extensive textual data.

Large Language models (LLMs) serve as a representation of how language works, learning from diverse datasets to predict what words or sequences of words are likely to come next in a given context.
LLM REAL-WORLD USE-CASES

Content Generation
Examples: Marketing platforms, social media management tools, content creation platforms, advertising agencies

Language Translation
Examples: Translation services, global communication platforms, international business applications

Text Summarization
Examples: Research tools, news aggregators, content curation platforms

Information Retrieval
Examples: Search engines, database systems, knowledge management platforms

Question Answering and Chatbots
Examples: Customer support systems, chatbots, virtual assistants, educational platforms

Content Moderation
Examples: Social media platforms, online forums, community management tools

Educational Tools
Examples: E-learning platforms, educational chatbots, interactive learning applications

Created By: Aishwarya Naresh Reganti
AI application example:

Talk to my Notion documents
Exploring bias

Definition: AI bias is an anomaly in the output of ML algorithms, due to the prejudiced assumptions made during the algorithm development process or prejudices in the training data.

through the “Talk to my Notion documents” reference conceptual architecture
Step: Text chunking | Potential Bias Risks: medium

Text chunking is a technique in NLP that divides text into smaller segments.

### Popular methods for text chunking:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Limitation/Bias</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLTK Sentence Tokenizer</td>
<td>Splits text into sentences.</td>
<td>Language dependency, issues with abbreviations and punctuation, lacks semantic understanding.</td>
<td>handle abbreviations/punctuation post-processing.</td>
</tr>
<tr>
<td>Spacy Sentence Splitter</td>
<td>Uses linguistic rules for sentence tokenization</td>
<td>May not generalize well across different text styles and languages.</td>
<td>Update rules and incorporate machine learning models.</td>
</tr>
<tr>
<td>Langchain</td>
<td>Recursively divides text at specific characters.</td>
<td>Ignores semantic context.</td>
<td>Combine with semantic analysis</td>
</tr>
<tr>
<td>KMeans Clustering</td>
<td>Groups sentences based on semantic similarity using sentence embeddings and K-means algorithm.</td>
<td>Loss of sentence order, computationally intensive.</td>
<td>Use fair pre-trained embeddings, apply bias correction techniques.</td>
</tr>
<tr>
<td>Clustering Adjacent Sentences</td>
<td>Groups consecutive sentences based on similarity.</td>
<td>Assumes adjacent sentences are always related</td>
<td>Validate clusters with context-aware methods, use additional semantic checks</td>
</tr>
</tbody>
</table>

### Tokenization using spaCy

- **NLTK Sentence Tokenizer**
  - Splits text into sentences.
  - Language dependency, issues with abbreviations and punctuation, lacks semantic understanding.
  - Handle abbreviations/punctuation post-processing.

- **Spacy Sentence Splitter**
  - Uses linguistic rules for sentence tokenization.
  - May not generalize well across different text styles and languages.
  - Update rules and incorporate machine learning models.

- **Langchain**
  - Recursively divides text at specific characters.
  - Ignores semantic context.
  - Combine with semantic analysis.

- **KMeans Clustering**
  - Groups sentences based on semantic similarity using sentence embeddings and K-means algorithm.
  - Loss of sentence order, computationally intensive.
  - Use fair pre-trained embeddings, apply bias correction techniques.

- **Clustering Adjacent Sentences**
  - Groups consecutive sentences based on similarity.
  - Assumes adjacent sentences are always related.
  - Validate clusters with context-aware methods, use additional semantic checks.

> How to Chunk Text Data — A Comparative Analysis
Embedding is the numerical representation of a chunk

Embedding is a way of representing data as points in space where the locations are semantically meaningful

Bias in Embeddings often stems from the biases present in the training dataset.

OpenAI Embeddings  Cohere Embeddings  Google Embeddings  Microsoft’s E5 Embeddings  BGE-M3

Step: Embedding Type (Tuning) | Potential Bias Risks: high

**Generative AI on Vertex AI**

**Vertex AI Tuning Workflow**
- Prepare your model tuning dataset (ensure diversity to prevent bias).
- Upload the model tuning dataset to a Cloud Storage bucket.
- Configure your project for Vertex AI Pipelines.
- Create a model tuning job (consider diverse hyperparameter choices).
- Deploy the tuned model to a Vertex AI endpoint of the same name.

**Vertex AI Dataset Size Requirements**
- Queries: 9 - 40,000 (ensure varied and representative queries).
- Documents: 9 - 500,000 (include diverse text sources).
- Labels: < 500,000 (use balanced label distributions).

**Bias Mitigation Strategies**
- **Diverse Training Data**: Ensure datasets are representative of all groups.
- **Bias Detection Tools**: Regularly evaluate embeddings for biases (e.g., Fairlearn, IBM AI Fairness 360).
- **Bias Correction Methods**: Use techniques like counterfactual data augmentation and fairness constraints.

Tune text embeddings, [https://cloud.google.com/vertex-ai/generative-ai/docs/models/tune-embeddings](https://cloud.google.com/vertex-ai/generative-ai/docs/models/tune-embeddings)
**Definition:** Stores data as high-dimensional vectors.

**Advantages:** Enables fast and accurate similarity search based on vector distance or similarity.

**Functionality:**
- Indexes vectors generated by embeddings to find comparable assets.
- Supports hybrid searches combining keywords and vectors.

**Accurate Similarity Searches:** Efficient retrieval of similar data points.

**Versatile Indexing:** Handles diverse data types for comprehensive searches.

---

**Steps: Vector DB and indexing | Potential Bias Risks: low**

---

**Vector Databases**

**Open-Source Vector Databases**

Chroma, Milvus, Faiss, Weaviate
Steps: Vector DB and Indexing | Potential Bias Risks: Low

Vector indexing can help to significantly increase the speed of the search process of similarity search with a trade-off in search accuracy, or efficiently store many subsets of data in a small memory footprint.

**Flat Indexing** stores vectors in their original form without modifications, ensuring perfect accuracy but being relatively slow.

**Approximate Nearest Neighbour (ANN)** techniques balance search accuracy and computational efficiency by quickly finding approximately similar data points using optimized data structures.

**Locality Sensitive Hashing (LSH)** uses hashing techniques like SimHash to efficiently group similar data points, enabling fast similarity searches in high-dimensional datasets.

**Hierarchical Navigable Small World (HNSW)** organizes data in a multi-layered structure, enhancing scalability and performance by combining skip list and navigable small world algorithms.

**ANNOY (Approximate Nearest Neighbors Oh Yeah)** is a memory-efficient algorithm for fast nearest neighbor retrieval in high-dimensional spaces, using a forest structure for quick indexing.

**Product Quantization** compresses high-dimensional vectors into smaller sub-vectors, retaining enough information for accurate similarity comparisons.

**Loss of Transparency and Hidden Bias:**
- Vector search can make their operations opaque, complicating the explanation and justification of search results.
- **Impact:** Raises concerns about bias and unfairness, and hinders identification and correction of biases.
- **Mitigation:** Enhance transparency by documenting model behavior and decisions, and regularly auditing for biases.


Searching of nearest neighbours using HNSW: Image from Pinecone
Step: LLM Selection | Potential Bias Risks: High

LLMs Challenges

Deployment Challenges
- Scalability
- Latency
- Monitoring and Maintenance
- Integration
- Cost Management
- Interoperability
- User Feedback Incorporation
- Regulatory Compliance
- Dynamic Content Handling

Technical Challenges
- Computational Resources
- Interpretability
- Evaluation
- Fine-tuning Challenges
- Contextual Understanding
- Robustness to Adversarial Attacks
- Long-Term Context

Data Challenges
- Data Bias
- Hallucination
- Training Data Quality

Ethical Challenges
- Ethical Concerns
- Bias Amplification
- Legal and Copyright Issues
- User Privacy Concerns

Created by: Aishwarya Naresh Reganti
In the realm of language models, "prompting" refers to the art and science of formulating precise instructions or queries provided to the model to generate desired outputs.

It's the input—typically in the form of text—that users present to the language model to elicit specific responses.

The effectiveness of a prompt lies in its ability to guide the model's understanding and generate outputs aligned with user expectations.

The effectiveness and fairness of prompt engineering techniques can be compromised by inherent biases in the design and implementation of prompts.
Bias mitigation: RAG Evaluation towards TruLens triad of evaluations:

- **Context Relevance**: A scoring system based on embeddings retrieval. Each retrieval is scored individually, with the final score being an average. This evaluates the effectiveness of embeddings retrieval/ranking systems but also guides potential enhancements. Can be used to test the Top X retrievals quality and improve the ranking system.

- **Groundedness**: This evaluates how well the response, broken down into chunks, aligns with the retrieved context. Trace how each chunk back to the source. It’s a tool to measure potential hallucinations and assess how well the RAG's summaries are grounded in the original documents or sentences.

- **Response Relevance**: Here, the evaluation of the relevance score of the answers to the posed questions is measured. The framework provides a score (0-1) and accompanying evidence as feedback, critically reviewing the LLMs' output at all stages (input, output, intermediate). It is essentially a feedback function.
Bias mitigation: continuous monitoring and evaluation

### Dimensions for Evaluating RAG Pipelines

<table>
<thead>
<tr>
<th>Retrieval Dimensions</th>
<th>Generation Dimensions</th>
<th>Applicable Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context Recall</strong></td>
<td><strong>Faithfulness</strong></td>
<td>• Accuracy</td>
</tr>
<tr>
<td>Evaluates how well the retrieved context matches the ground-truth answer. It relies on both the ground truth and the retrieved context.</td>
<td>Measures the factual consistency of the generated response in relation to the provided context. It is derived from both the answer and the retrieved context.</td>
<td>• Exact Match</td>
</tr>
<tr>
<td><strong>Context Precision</strong></td>
<td><strong>Answer Relevance</strong></td>
<td>• Recall</td>
</tr>
<tr>
<td>Evaluates whether all ground-truth items within the contexts are prioritized higher in ranking compared to other items.</td>
<td>Measures the relevance of the generated answer to the provided prompt. Answers with incomplete or redundant information receive lower scores and vice versa.</td>
<td>• Precision</td>
</tr>
<tr>
<td><strong>Context Relevance</strong></td>
<td><strong>Negative Rejection</strong></td>
<td>• R-Rate</td>
</tr>
<tr>
<td>Measures the relevance of the retrieved context, it is determined by analyzing both the question and the provided contexts.</td>
<td>Evaluates the model's ability to refrain from providing a response when the retrieved documents lack the necessary information to answer a question.</td>
<td>• Cosine Similarity</td>
</tr>
<tr>
<td><strong>Context Entity Recall</strong></td>
<td><strong>Information Integration</strong></td>
<td>• Hit Rate</td>
</tr>
<tr>
<td>Measures the recall of the retrieved context by comparing the number of entities present in both the ground truths and the contexts to the number of entities present in the ground truths alone.</td>
<td>Assesses the model's skill in combining information from various documents to answer complex questions.</td>
<td>• MRR</td>
</tr>
<tr>
<td><strong>Noise Robustness</strong></td>
<td><strong>Counterfactual Robustness</strong></td>
<td>• NDCG</td>
</tr>
<tr>
<td>Evaluates the model's capability to manage noise documents that are question-related but lack substantive information.</td>
<td>Evaluates the model's capacity to identify and disregard established inaccuracies within documents, even when informed about potential misinformation.</td>
<td>• BLEU</td>
</tr>
</tbody>
</table>

**Frameworks & Benchmarks**

- **Frameworks**
  - RAGas
  - TruLens
  - ARES
  - DeepEval
  - Tonic Validate
  - LangFuse

- **Benchmarks**
  - RGB
  - Recall
  - CRUD
Do we really want completely unbiased AI applications?

What is the impact of heuristics in AI applications?

1. **Domain Knowledge Bias**: If your heuristics favor well-known or commonly used terms over scientific names like, you are introducing a bias based on what you deem more relevant or appropriate.

2. **Selection Bias**: By consistently selecting certain terms over others, your pipeline may ignore less common but equally correct terms, thus not providing a comprehensive view of possible UMLS concepts.

3. **Confirmation Bias**: If your heuristics are designed to confirm pre-existing knowledge or expectations, you may reinforce the selection of certain terms, which may not always align with the most accurate or updated medical terminology.

**Implications of Biasing the Pipeline**

- **Improved Precision but Reduced Recall**: Your selections might become more precise, but you might miss out on valid alternatives, thus reducing recall.

- **Reduced Generalization**: The model might perform well in familiar scenarios but struggle with new, unseen data that doesn’t fit the heuristics.

- **Human Intervention Dependency**: The pipeline becomes more dependent on the quality and relevance of the heuristics applied, which requires ongoing updates and validation.

**Balancing Bias and Heuristics**

1. **Transparent Heuristics**: Clearly document the heuristics and the rationale behind their selection.

2. **Regular Updates**: Periodically review and update heuristics to align with the latest medical standards and terminology.

3. **Hybrid Approach**: Use a combination of automated model predictions and heuristics, ensuring a feedback loop where heuristics can be adjusted based on model performance.

4. **Evaluation Metrics**: Continuously monitor the impact of heuristics on the model's precision, recall, and overall performance using standard evaluation metrics.
Chat GPT 4o response in “Will AI ever be completely unbiased?”

Achieving complete unbiasedness in AI is a complex and challenging goal,

**Human Bias in Data**: AI systems learn from data that is often generated by humans. If this data contains biases, the AI is likely to inherit and perpetuate those biases.

**Algorithmic Bias**: The algorithms themselves can introduce biases. The design choices made by developers, such as which features to consider and how to weigh them, can embed biases into the AI system.

**Context and Subjectivity**: What is considered biased or unbiased can be subjective and context-dependent.

**Bias Detection and Mitigation**: Identifying and correcting biases is a non-trivial task. While there are techniques to detect and mitigate bias, these methods are not perfect and can sometimes introduce new biases or fail to address all aspects of the problem.

**Complexity of Real-World Scenarios**: Real-world scenarios are often complex and multifaceted, making it difficult for AI systems to avoid biases entirely.

**Evolution of Bias**: As societal norms and values evolve, so do perceptions of bias.

While **complete unbiasedness may be unattainable**, there are ongoing efforts to reduce bias in AI:

**Diverse and Inclusive Data Collection**: Ensuring that training data is representative of all relevant groups and perspectives.

**Fairness-Aware Algorithms**: Developing algorithms designed to minimize bias and promote fairness.

**Bias Audits and Monitoring**: Regularly auditing AI systems for biases and implementing mechanisms for continuous monitoring and improvement.

**Transparency and Accountability**: Promoting transparency in AI development and decision-making processes, along with accountability mechanisms for addressing bias-related issues.