
RAG EDUCATIONAL CHATBOT

Background Section

Dr. Hager Saleh

Amro Ismail – Mohamed Algazwani

1. Natural Language Processing (NLP)

Natural Language Processing (NLP) is an interdisciplinary area of computer science, artificial intelligence and computational linguistics. It is about computers' ability to process, analyze and generate human language in a semantically rich way. Modern NLP tasks vary from simple text classification to complex generative and reasoning workflows.

1.1. Basic NLP Tasks and Taxonomy

Modern NLP tasks can be roughly classified into the following distinct functional areas:

- **Natural Language Understanding (NLU):** Involves the assessment of semantic intent, syntactic structure, and contextual context from text. Main applications are text classification, sentiment analysis, named entity recognition (NER) and intent detection.
Natural Language Generation (NLG): Concentrates on creating coherent, grammatically correct and contextually relevant human text. Applications of interest include abstractive summarization, dialogue generation and automated report generation.
- **Sequence-to-Sequence Processing:** It processes an input sequence and produces a corresponding output sequence in a different domain or modality, e.g., neural machine translation, speech recognition.

1.3. End-to-End NLP Pipeline Architecture

The standard NLP workflow translates unstructured human discourse to numerical representations as below:

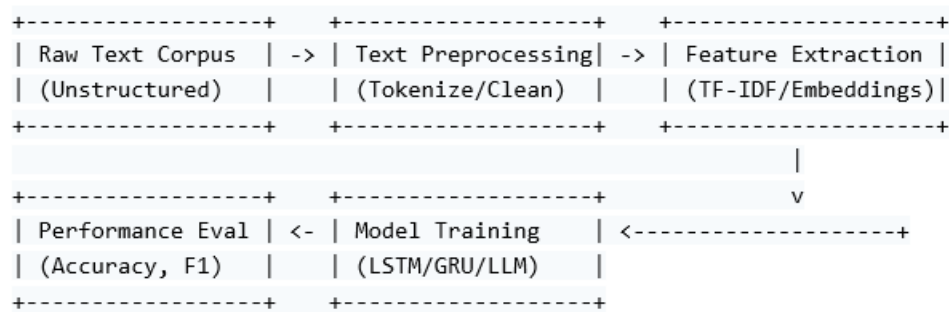


Figure 1: High-level overview of an end-to-end Natural Language Processing Pipeline.

2. Text Pre-processing & Normalization

Raw text data from real-world environments is inherently noisy; Pre-processing of text constitutes a mandatory preparatory phase of the pipeline that purifies raw corpora, reduces vocabulary redundancy and ensures uniform input structures for downstream machine learning algorithms.

Linguistic Pre-processing Methods

- **Tokenization:** The fundamental operation of segmenting a continuous stream of text into discrete, indexable semantic units called tokens:
 - Word-Level Tokenization: Tokenizes text on white spaces and explicit punctuation marks. It is

simple, but suffers from out-of-vocabulary words and complex compound words.

- Subword-Level Tokenization: Rare words are divided into common subword units (e.g., Byte-Pair Encoding [BPE], WordPiece, Unigram). With this method, models can preserve small vocabulary sizes and yet be able to construct arbitrary or out-of-vocabulary tokens.
- **Stop-Word Removal:** filter and remove the stop words (e.g. “the”, “is”, “at”, “which”) using standardized domain stop-word lists. Removing these words reduces the dimensionality and concentrates model parameters on semantically salient words.
- **Normalization Strategies (Stemming vs. Lemmatization):**
 - Stemming: A heuristic process that it is a rule-based (e.g. Porter Stemmer, Lancaster Stemmer) that chops off ends of words to reduce inflected words to a common base form. Stemming is a fast computational process, but it often leads to non-words, due to over-stemming or under-stemming.
 - Lemmatization: A morphologically valid approach (e.g., WordNet Lemmatizer) which considers the Part-of-Speech (POS) tag and the grammatical context of a token to map it to its corresponding root form (or lemma) in the valid dictionary.
- **Text Regularization:** It covers the conversion of characters to lowercase, removal of HTML tags, removal of non-alphanumeric symbols, expansion of contractions and handling of numerical entities.

3. Text Representation (Feature Extraction)

Text inputs have to be converted into numerical representations as mathematical and machine learning models can only operate with numerical tensors. In modern NLP, the feature extraction methods can be divided into sparse traditional representation and dense continuous representation.

Text Representation	
+-----+-----+	
Traditional (Sparse Vectors)	Distributed (Dense Vectors)
- Bag of Words (BoW)	- Word2Vec (CBOW / Skip-gram)
- N-grams	- GloVe
- TF-IDF	- FastText
	- Contextual Embeddings (BERT/LLMs)

3.1. Traditional Methods (Sparse Representations)

- **Bag of Words (BoW):** Maps a document to a high dimensional vector corresponding to the size of the vocabulary. Each vector index reflects the frequency of a word within the document. BoW ignores completely the order of tokens and the grammatical syntax.
- **N-grams:** Preserves local context by looking at a contiguous sequence of items (e.g. Unigrams, Bigrams, Trigrams). With n-grams models can distinguish between different phrases such as “not bad” vs. “bad”.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** A statistical weighting scheme for the

relative importance of a term within a given document, relative to its global frequency within an entire corpus. TF-IDF penalizes common generic words and emphasizes domain-specific keywords.

3.2. Distributed Representations (Static Word Embeddings)

Sparse models are high dimensional and do not capture semantics similarities between words (e.g. “cat” and “feline” are treated as completely orthogonal vectors). Distributed representations project words into small continuous real-valued vector spaces where semantically similar words are close to each other.

- **Word2Vec**: a neural framework proposed by Mikolov et al. which learns word embeddings over two different shallow architectures:
 - Continuous Bag-of-Words (CBOW): Predicts a target word from the context tokens.
 - Skip-gram: Predicts the tokens in the context of a single target word.
- **GloVe** (Global Vectors): An unsupervised embedding algorithm that utilizes the global co-occurrence matrix to learn the embeddings, and thus combines the advantages of global matrix factorization methods and local context window methods.
- **FastText**: An extension of Word2Vec by Facebook AI Research, which represents words as bags of character n-grams. Rare, misspelt or out-of-vocabulary words are represented by aggregating subword vectors in FastText.

4. Deep Learning Models

Traditional Feedforward Neural Networks treat text instances as independent vectors, and do not capture sequential context and temporal dependencies. Recurrent architectures are explicitly designed for maintaining the state memory over sequential inputs.

4.1. Long Short-Term Memory (LSTM) Networks

Standard Recurrent Neural Networks (RNNs) suffer from severe vanishing and exploding gradients during backpropagation through time preventing them from learning long-range context dependencies. To solve this problem, Hochreiter and Schmidhuber introduced the (LSTM) architecture. The LSTM architecture introduces a dedicated Cell State controlled by three specialized gating mechanisms:

- **Forget Gate**: It takes the input and previous hidden state and decides what information to be thrown away from cell state.
- **Input Gate**: Determines which new candidate information to add to update the current cell state.
- **Output Gate**: Determines the fraction of the internal cell state to be emitted as the current hidden state output to downstream layers.

4.2. Gated Recurrent Unit (GRU)

GRUs is a simplified version of LSTM architecture. The GRU merges the cell and hidden state into a single hidden state. It has two gates:

- **Reset Gate:** Decides to what extent the previous hidden state memory needs to be dropped while calculating the candidate state values.
- **Update Gate:** It functions as an input and forget gate at the same time and determines the amount of new information to be added to the current memory state.

GRUs have fewer parameters than LSTMs, so they can be trained faster and require less computational memory, while performing comparably on many tasks.

5. Large Language Models (LLMs)

LLMs constitute a significant milestone in NLP, transitioning from narrow, task-specific models to broad, unified foundational architectures trained over enormous quantities of unstructured text. Rather than training separate pipelines for classification, translation and summarization, a single LLM learns general language capabilities, contextual reasoning and broad world knowledge that can be adapted to a wide range of downstream tasks.

5.1. Architectural Foundations of Large Language Models

5.1.1 The Transformer Paradigm

Modern LLMs are based on Transformer architecture. Prior to Transformers, long text sequences suffered from severe performance bottlenecks when processed sequentially by recurrent networks (e.g., LSTMs and GRUs). The Transformer removed recurrent connections and instead used Self-Attention Mechanisms which process all the words in a sequence at the same time in parallel.

This model uses self-attention to determine dynamically how much any token in a text is related to any other token. This allows the model to capture long-range contextual dependencies, resolve pronouns and understand complex syntactic relations irrespective of the distance between words in a document.

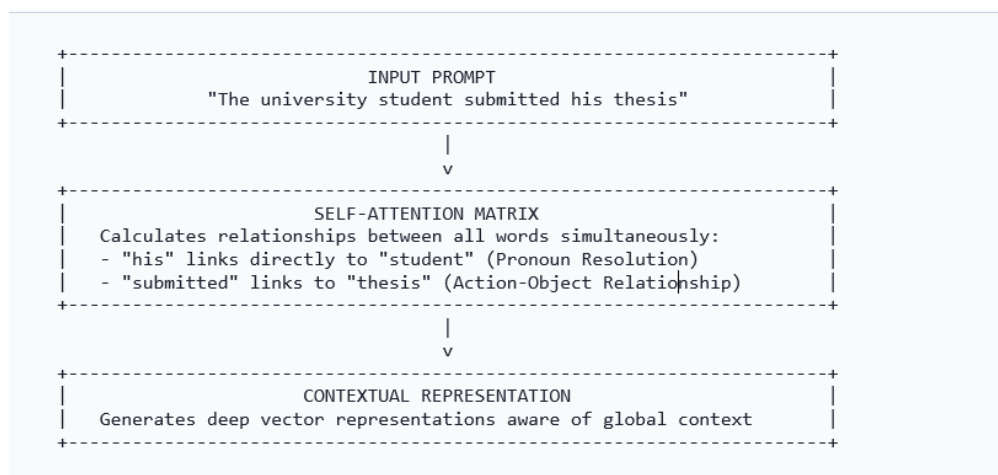


Figure 1: Conceptual Diagram of Self-Attention Capturing Inter-Token Relationships.

5.1.2 Transformer Taxonomy and Variations

The Transformer architectures can be divided into three main classes depending on the implementation of the attention mechanism:

- **Decoder-Only (Autoregressive Models):** Great at generating text. The model predicts the next word in a sequence only from the previous words (causal masking). These models are at the heart of large generative assistant architectures such as GPT-4, LLaMA and Mistral.
- **Encoder-Only (Autoencoding Models):** For understanding text in both directions. The model considers both the words before and after a target token. These models (BERT, RoBERTa) are good for document classification, named entity recognition and dense search retrieval but not text generation.
- **Encoder-Decoder (Sequence-to-Sequence Models):** The encoder processes the input prompt, and the decoder generates the output sequence. Such architectures (e.g., T5, BART) are useful for structural transformations such as machine translation and abstractive document summarization.

5.2. Complete LLM Training & Alignment Lifecycle

Developing a production-ready LLM involves a structured multi-stage training pipeline, transforming raw, unstructured text into an aligned conversational system.

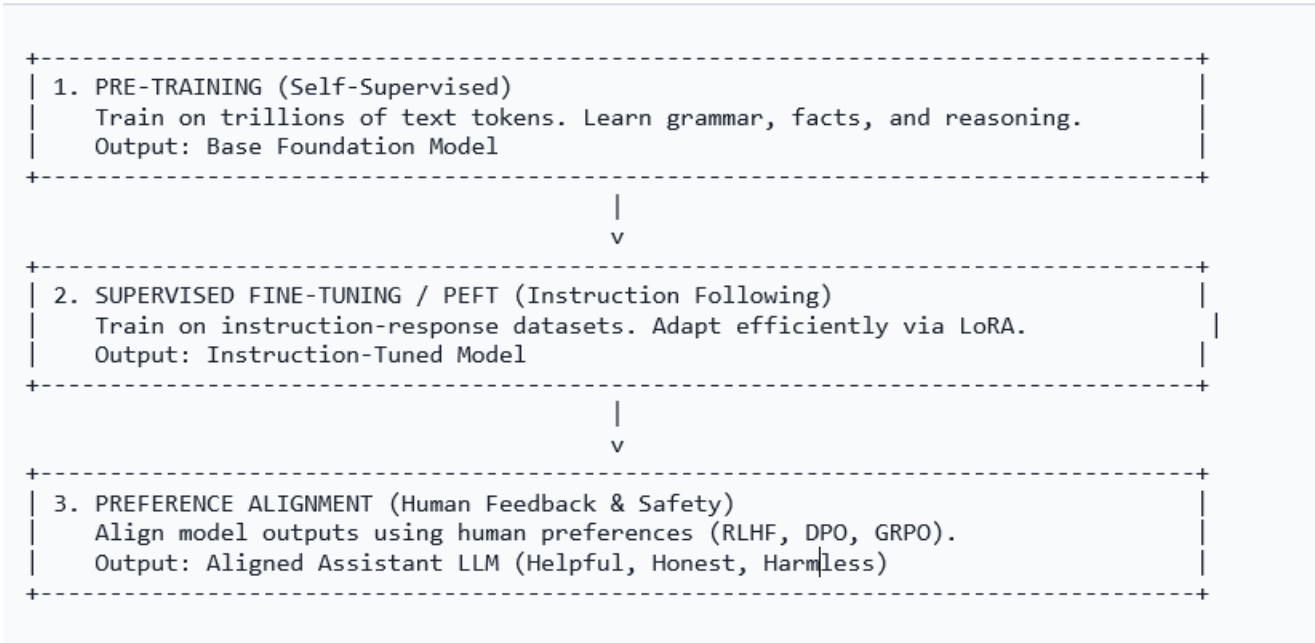


Figure 2: The Multi-Stage Lifecycle of Modern Large Language Model Training.

5.2.1 Pre-training in a self-supervised manner

In the pre-training phase the base model ingests vast amounts of data, including trillions of words from books, academic papers, code repositories and web content. Pre-training involves self-supervised learning, where the model learns by trying to predict missing or next tokens, and does not require human annotated data. At this point the model accumulates a basic vocabulary, grammar, contextual understanding and knowledge of the world.

5.2.2 Supervised Fine-Tuning (SFT)

The base model is good at predicting what word is likely to come next, but not so good at following instructions or directly answering a user's questions. Supervised Fine-Tuning fine-tunes the base model on curated datasets of high-quality instruction-response pairs, to teach the model to act as a helpful conversational assistant.

5.2.3 Parameter Efficient Fine Tuning (PEFT/LoRA)

Memory and computation intensive is fine-tuning a multi-billion parameter model by updating all the parameters. This is addressed by Low-Rank Adaptation (LoRA), which freezes the main weights of the base model and instead injects small, lightweight adapter layers into key attention modules. During fine-tuning only these small adapters are trained. This reduces the number of trainable parameters by more than 99 % while matching the performance of fine-tuning the full model.

5.2.4 Approaches for Human Alignment

Models are aligned using data of human preferences. This helps keep model outputs helpful, truthful, and safe on of them is:

- **RLHF (Reinforcement Learning from Human Feedback):** - A separate Reward Model is trained on preference rankings from humans.

5.3. External Knowledge Integration: Retrieval-Augmented Generation (RAG)

5.3.1 The Need for Grounded Retrieval

Despite their strong reasoning capabilities, standalone LLMs face inherent limitations: static training cutoffs, lack of access to private institutional data, and a tendency to generate plausible-sounding but incorrect information (hallucinations). **Retrieval-Augmented Generation (RAG)** addresses these issues by pairing the generative LLM with an external document retrieval pipeline.

5.3 Incorporating External Knowledge: Retrieval-Augmented Generation (RAG)

5.3.1 Why Grounded Retrieval is Required

Standalone LLMs, while powerful in their reasoning abilities, have inherent limitations tendency to generate plausible-sounding but incorrect information (hallucinations). Retrieval-Augmented Generation (RAG) solves these issues by coupling the generative LLM with an external document retrieval pipeline.

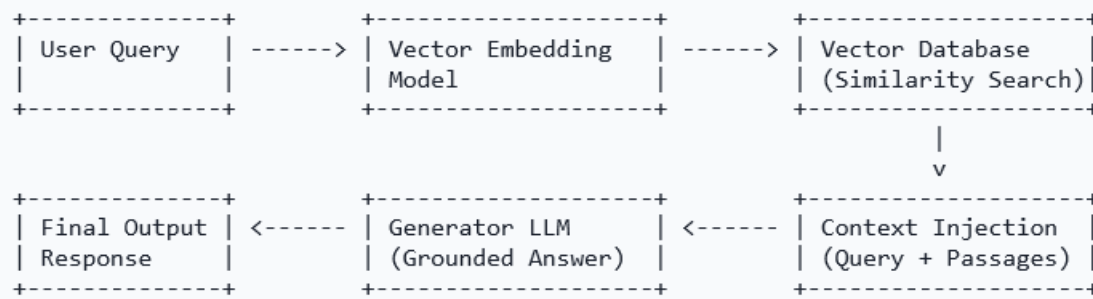


Figure 3: Operational Pipeline of a Grounded Retrieval-Augmented Generation (RAG) System.

5.3.2 Advanced Institutional Document Pipeline

- **Document Parsing and Semantic Chunking:** Complex lists and formal clauses are found in structured institutional documents (academic regulations, bylaws, course catalogues, PDFs). Fixed size chunking can destroy meaningful context. Semantic Chunking Strategies split documents into sections according to headings, paragraphs, and logical boundaries, and thus each chunk contains a full administrative rule or concept.
- **Vector Indexing & Dense Retrieval:** Text chunks are transformed into high dimensional vector spaces, and then indexed in vector databases (e.g. Chroma DB, FAISS, Qdrant). When a user asks a question, the system retrieves the most relevant passages according to their semantic similarity (e.g. Cosine Similarity).
- **Re-Ranking Mechanisms for Accurate Retrieval:** The initial retrieval stage frequently produces multiple candidate passages with keyword overlaps but no precise relevance. A secondary Cross-Encoder Reranker scores and re-orders the retrieved chunks, so that the exact regulatory passage is at the very top of the context window before it reaches the LLM.
- **Contextual Prompt Building:** The top-ranked passages are directly injected into the prompt, along with the user query, giving the LLM the exact information it needs to generate the response.

5.4. Hallucination mitigation, guard-rail and multi-language support

5.4.1 Prompting and Mitigating Hallucinations

Providing wrong information in an institutional setting can cause serious confusion. To prevent model hallucinations, strict guardrails are embedded into the system:

- **Strict Context Guard Rails:** System prompts explicitly instruct the LLM to respond only based on the retrieved passages of context. If the answer is not in the context the model is strictly conditioned to decline politely (e.g. "This information is not available in the provided documents")
- **Citation & Source Attribution:** Modern RAG architectures provide explicit metadata (file name, section title, page number) to every retrieved passage. The LLM is prompted to cite these sources directly in its answer, allowing users to verify the validity of the response against the official document.

5.4.2 Multilingual NLP for Various User Requests

Institutional settings often require bilingual support (e.g. Arabic and English) and have to cope with a variety of language registers from formal terminology to colloquial expressions:

- **Multilingual Embedding Models:** Uses specialized vector space models that can map different languages into shared semantic spaces, matching queries in one language to document passages in another.
- **Cross-Lingual Language Understanding:** Advanced multilingual LLMs can handle enquiries posed in local dialects or informal formats and accurately associate them with formal institutional documents.

5.5. Evaluation Frameworks for Knowledge-Grounded Systems

Evaluating document-grounded, conversational AI systems requires specialized metrics beyond standard classification measures. Frameworks such as **RAGAS** evaluate the pipeline across several key dimensions:

Evaluation Metric	Evaluated Pipeline Stage	Objective & Meaning
Faithfulness (Groundedness)	Generated Response vs. Retrieved Context	Verifies that every claim in the response is directly supported by the retrieved official documents, guaranteeing zero hallucinations.
Answer Relevance	Generated Response vs. User Query	Ensures the response directly answers the user's specific question without adding off-topic information.
Context Precision	Retrieved Passages vs. User Query	Measures whether the vector store fetched relevant document sections while filtering out background noise.

Evaluation Metric	Evaluated Pipeline Stage	Objective & Meaning
System Latency & Performance	End-to-End Pipeline	Tracks end-to-end response time to ensure fast and responsive performance for user interaction.

References

- **Chowdhury, G. G. (2003).** Natural language processing. *Journal of Documentation*, 59(1), 51-68.
- **Kaur, J., & Saini, J. R. (2016).** Traditional and nature-inspired techniques for text pre-processing: A review. *International Journal of Computer Applications*, 137(10), 18-25.
- **Vijayarani, S., Ilamathi, J., & Nithya, C. (2015).** Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1), 7-16.
- **Manning, C. D., Raghavan, P., & Schütze, H. (2008).** Introduction to information retrieval. Cambridge University Press.
- **Ramos, J. (2003).** Using tf-idf to determine word relevance in document queries. *Proceedings of the First Instructional Conference on Machine Learning*, 242(1), 29-48.
- **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- **Pennington, J., Socher, R., & Manning, C. D. (2014).** GloVe: Global vectors for word representation. *Proceedings of EMNLP*, 1532-1543.
- **Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017).** Enriching word vectors with subword information. *TACL*, 5, 135-146.
- **Hochreiter, S., & Schmidhuber, J. (1997).** Long short-term memory. *Neural Computation*, 9(8), 1735-1780.
- **Cho, K., et al. (2014).** Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- **Vaswani, A., et al. (2017).** Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 5998-6008.
- **Devlin, J., et al. (2019).** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 4171-4186.
- **Sokolova, M., & Lapalme, G. (2009).** A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.
- **Zhao, W. X., et al. (2023).** A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- **Hu, E. J., et al. (2021).** LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- **Rafailov, R., et al. (2024).** Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 36.
- **Es, S., et al. (2023).** Ragas: Automated Evaluation of Retrieval Augmented Generation. *arXiv preprint arXiv:2309.15217*.
- **Lewis, P., et al. (2020).** Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 9459-9474.