CONTEXUAL SEARCH ANATOMY:

We are developing a contextual search system where, upon entering a user query (a single keyword), the **Query Processing module** extracts keywords from the input to comprehend intent, creating a structured representation**. Semantic Relationship Mapping** constructs a graph to capture relationships between keywords. A sub-process prunes unnecessary edges for efficiency. The final module integrates a knowledge graph, enhancing contextual understanding by mapping semantic relationships onto broader informational spaces and aligning the results with one of the contextual spaces.

The user initiates the process by entering a single-word query. Using the Wikipedia database, the system fetches keyword-specific multidomain documents. Subsequently, keywords are extracted from these documents using five different Python packages: **Rake, Yake, SpaCy, KeyBERT, and TextBlob.** Extracted keywords undergo further processing by removing stopwords. The processed set of keywords from all packages is then intersected to identify the most repeated 15 keywords. This set forms the basis for further processing. A **complete graph** is constructed using the Python package 'networkx.' After graph construction, data is extracted for each node in a structured JSON format, containing Wikipedia data and infobox data. The final document consists of structured key-value data for all nodes. With the above process we are able to capture keyword specific multi-domain data.

To obtain the knowledge graph, edges are pruned from the complete graph. Edge pruning is based on similarity values calculated between each pair of nodes using the following measures: **Normalized Levenshtein distance, Wu Palmer similarity, Path similarity, Jaro-Winkler similarity**

Edges with weights below a set threshold are pruned, resulting in the creation of the knowledge graph.

Next, we apply graph algorithms and traversals on the knowledge graph to gain more insights:

1.BFS Traversal: Systematically explores nodes at each level, helping to discover relationships between related concepts in a knowledge graph. It aids in identifying clusters by highlighting densely connected groups of nodes at different levels of abstraction.

2.DFS Traversal: Explores a branch of nodes deeply before backtracking, revealing hierarchical relationships and uncovering clusters that share common ancestry. It is particularly useful for identifying cohesive subgroups within a knowledge graph.

3.Cluster Analysis: Examines clusters of nodes that appear together in the traversal, indicating subtopics or closely related concepts. It helps to understand the thematic coherence within the knowledge graph.

4.Centrality Analysis: Includes Betweenness Centrality to identify nodes acting as bridges, connecting different clusters or communities, and Closeness Centrality to find nodes efficient in information retrieval and dissemination.

5.Path Analysis: Uses shortest path algorithms to find the quickest connections between different concepts, aiding in understanding relationships and dependencies within the knowledge graph.

6.Discover Related Concepts: As the traversal progresses, each level reveals nodes directly connected to the starting node. We explore these nodes to discover related concepts or keywords, helping the user understand the broader context around their initial query.

Next, we use the traversal results to generate contextual suggestions for the user. For instance, if the user started with a specific keyword, we suggest related terms or topics that might be of interest. This final step involves mapping the results obtained to one of the predefined contextual spaces:

Pratyaksha: Direct/Immediate knowledge. The user receives precisely what they asked for, matching the keywords in their query.

Anumana: Knowledge derived indirectly through reasoning or inference. The search engine uses reasoning to infer what the user might be looking for based on context, delivering relevant results beyond explicit query terms.

Upamana: Knowledge gained through comparison or analogy. The search engine recognizes patterns in user behavior or preferences, offering content analogous or similar to what others with similar queries have found useful.

Contextual spaces defined:

Pradhana: All possible information.

Prakriti: The process that finds and organizes information.

Maha-tattva: The basic elements that make up the information found.

The system incorporates three qualities:

Rajas (Passion/Activity): Reflects the dynamic, ever-changing nature of real-time search results and trending topics.

Tamas (Ignorance/Inertia): Represents static or outdated information that may persist despite newer and more relevant content being available.

Sattva (Goodness/Harmony): Signifies accurate, relevant, and well-organized information aligning with the user's intent.

Additionally, we are using OpenAI to generate contextual suggestions for the user.