



Act like an External Examiner specializing in Information Retrieval (IR) and Quantum Artificial Intelligence (QAI) domains. You are an expert in assessing technical depth, conceptual clarity, and practical understanding during viva or oral examinations.

Your goal is to **analyze the uploaded IR and QAI Practical Manuals** and generate a comprehensive list of **oral-style technical questions and answers** for each practical. These questions should reflect what an external examiner would realistically ask in a viva session — with an emphasis on practical execution, experimental reasoning, and applied understanding.

Task:

Examine each practical experiment described in the manuals and produce **a minimum of 10–15 technical Q&A pairs per practical** that test a student's command over theory, procedure, logic, and analytical reasoning relevant to that specific topic.

Requirements:

1. Identify each practical title from the IR and QAI manuals and organize your output under separate headings for clarity.
2. For each practical:
 - o Generate questions primarily inspired by **oral/viva-type queries** commonly asked during practical exams.
 - o Include a mix of:
 - Conceptual questions (understanding principles)
 - Procedural questions (how the experiment is conducted)
 - Analytical questions (error handling, optimization, result interpretation)
 - Application questions (real-world or advanced applications of the concept)
3. Answers must be:
 - o Technically accurate and concise
 - o Strictly derived from the manuals' content
 - o Free from fabricated or external information

4. If any section lacks sufficient information, clearly note: "Insufficient data in manual for this question."

5. Maintain a professional tone consistent with academic external examination standards.

Context:

///

You will receive one or more PDF manuals containing details of **Information Retrieval (IR)** and **Quantum Artificial Intelligence (QAI)** practicals. These manuals include procedures, objectives, theory, and oral/viva questions. Your task is to transform this content into structured Q&A sets that simulate a realistic oral examination experience conducted by an external examiner.

///

Constraints:

- Format: Markdown or structured list
- Style: Academic, conversational (viva-style), technically rigorous
- Scope: Include all practicals with 10–20 oral-style Q&A each
- Reasoning: Carefully analyze manual content before crafting each question
- Self-check: Ensure completeness, technical accuracy, and relevance to each experiment.

Take a deep breath and work on this problem step-by-step.

Now I have comprehensive information from both manuals. Let me analyze the content and create structured Q&A sets for each practical experiment. Based on the manuals, I can see:

Information Retrieval Manual (5 practicals):

1. Text pre-processing (stop word removal, stemming)
2. Document retrieval using inverted files
3. Bayesian network for heart disease diagnosis
4. Agglomerative hierarchical clustering
5. PageRank algorithm

QAI Manual (6 practicals):

1. 16 Qubit Random Number Generator
2. Quantum Teleportation
3. 5 Qubit Quantum Fourier Transform
4. Tackle Noise with Error Correction
5. Tarrataca's quantum production system with 3-puzzle
6. Randomized Benchmarking Protocol (limited information)

Let me create comprehensive viva-style Q&A for all these practicals.

COMPREHENSIVE VIVA Q&A FOR INFORMATION RETRIEVAL AND QUANTUM AI PRACTICALS

INFORMATION RETRIEVAL (IR) PRACTICALS

IR Practical 1: Text Pre-Processing (Stop Word Removal & Stemming)

Q1: What is the primary purpose of text pre-processing in information retrieval systems?

Answer: Text pre-processing converts raw text data into a format that computers can understand and process efficiently. It filters out useless data, reduces dataset size, and focuses on relevant information to improve model efficiency. The main techniques include tokenization, stop word removal, stemming, lemmatization, and part-of-speech tagging.^[1]

Q2: Define stop words and explain why they are removed from text documents.

Answer: Stop words are commonly used words such as "the", "a", "an", and "in" that search engines are programmed to ignore during indexing and retrieval. They are removed because they take up valuable space in databases and processing time without contributing meaningful information to the content analysis. NLTK library in Python provides stop word lists in 16 different languages.^[1]

Q3: What is stemming and how does it differ from the original word form?

Answer: Stemming is the process of producing morphological variants of a root or base word by reducing words to their stem form. For example, "chocolates", "chocolatey", and "choco" reduce to "chocolate"; "retrieval", "retrieved", and "retrieves" reduce to "retrieve". It is an important part of natural language processing pipelines and helps normalize text for easier processing.^[1]

Q4: Explain the two types of errors that occur in stemming.

Answer: The two main types of stemming errors are over-stemming and under-stemming. Over-stemming (false-positives) occurs when two words from different stems are incorrectly stemmed to the same root, such as "arguing" becoming "argu" which is not a valid word. Under-stemming (false-negatives) occurs when two words that should have the same stem are not reduced to it, like "arguing" and "argument" being stemmed differently.^[1]

Q5: Compare Porter Stemmer and Snowball Stemmer.

Answer: Porter Stemmer, proposed in 1980, is the most widely used algorithm based on removing combinations of smaller suffixes, but it only works for English words. Snowball Stemmer is more advanced and supports multiple languages, making it a multi-lingual stemmer. Snowball is more aggressive than Porter and has greater computational speed due to improvements, which is why it's also called Porter2 Stemmer.^[1]

Q6: What are the advantages and limitations of the Porter Stemmer?

Answer: Advantages: Porter Stemmer produces the best output compared to other stemmers and has a lower error rate. Limitations: The morphological variants produced are not always real words, applications are limited to English only, and the output stem may not be meaningful even though stems are correctly mapped.^[1]

Q7: Describe the Lancaster Stemmer and its characteristics.

Answer: Lancaster Stemmer is more aggressive and dynamic compared to Porter and Snowball stemmers. It is faster in execution but the algorithm can be confusing when dealing with small words. It uses an iterative algorithm and saves rules externally, but it is not as efficient as Snowball Stemmers.^[1]

Q8: What is the N-Gram Stemmer approach?

Answer: N-Gram Stemmer uses a set of n consecutive characters extracted from a word, where similar words will have a high proportion of n-grams in common. For example, "INTRODUCTIONS" with n=2 becomes: I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S. It is based on string comparisons and is language dependent, but it requires significant space to create and index n-grams and is not time efficient.^[1]

Q9: What real-world applications use stemming algorithms?

Answer: Stemming is used in information retrieval systems like search engines to improve search results. It determines domain vocabularies in domain analysis and maps documents to common subjects by indexing. Sentiment analysis for product reviews uses stemming as a text-preparation step, and document clustering employs it for subject extraction and automatic document structuring. Google Search adopted word stemming in 2003.^[1]

Q10: How can over-stemming be avoided in practical implementations?

Answer: To avoid over-stemming, use a stemmer appropriate for the specific task and language being processed. Test the stemmer on sample text to ensure it produces valid root forms. Consider using a lemmatizer instead of a stemmer, as it takes context into account and is less prone to errors. Additional techniques like semantic role labeling, sentiment analysis, and context-based information can help understand text context and make stemming more precise.^[1]

Q11: What is the difference between stemming and lemmatization?

Answer: While both reduce words to base forms, stemming uses heuristic algorithms to chop word endings without considering context, which may produce non-valid words. Lemmatization considers the context of words and reduces them to their dictionary form (lemma), always producing valid words. Lemmatization is more accurate but computationally more expensive than stemming.^[1]

Q12: In your practical implementation, which Python library is used for stop word removal?

Answer: NLTK (Natural Language Toolkit) is used for stop word removal in Python. NLTK has a built-in list of stopwords stored in 16 different languages, which can be found in the nltk_data/corpora/stopwords directory.^[1]

Q13: Why is tokenization performed before stemming?

Answer: Tokenization breaks down documents into individual words or tokens, which are required as input to the stemmer. The stemming process needs individual word units to apply morphological reduction algorithms. Without tokenization, the stemmer cannot identify word boundaries and apply appropriate suffix removal rules.^[1]

Q14: What is the practical significance of reducing dataset size through pre-processing?

Answer: Pre-processing reduces dataset size, which directly improves computational efficiency and processing speed. It helps focus on only useful and relevant data, leading to higher model accuracy. Removing irrelevant text prepares datasets for further processing or model building with better performance characteristics.^[1]

Q15: Can you explain the Krovetz Stemmer algorithm steps?

Answer: Krovetz Stemmer, proposed by Robert Krovetz in 1993, follows two main steps: First, it converts plural forms of words to singular forms (e.g., 'children' to 'child'). Second, it converts past tense to present tense and removes the suffix 'ing'. It is light in nature and can be used as a pre-stemmer for other stemmers, but it is inefficient for large documents.^[1]

IR Practical 2: Document Retrieval Using Inverted Files

Q1: What is an Inverted Index and why is it important in information retrieval?

Answer: An Inverted Index is a data structure that stores a mapping from content (words or numbers) to their locations in documents or sets of documents. It is organized by terms, where each term points to a list of documents containing that term. It is essentially a hashmap-like structure that directs from a word to a document or webpage, enabling efficient text search in search engines and database systems.^[1]

Q2: Explain the difference between Record-Level and Word-Level Inverted Index.

Answer: Record-Level Inverted Index contains a list of references to documents for each word without position information. Word-Level Inverted Index additionally contains the positions of each word within a document. The Word-Level index offers more functionality for phrase queries and proximity searches but requires more processing power and storage space to create.^[1]

Q3: Walk through the process of creating an inverted index with an example.

Answer: Consider two documents: Document 1: "The quick brown fox jumped over the lazy dog" and Document 2: "The lazy dog slept in the sun". First, tokenize both documents into terms. Then create an index where each term points to documents containing it: "The" → Document 1, Document 2; "quick" → Document 1; "lazy" → Document 1, Document 2; "dog" → Document 1, Document 2. When searching for "lazy", the system queries the index and retrieves both documents.^[1]

Q4: What are the main steps to build an inverted index from scratch?

Answer: The main steps are: 1) Fetch the Document; 2) Remove Stop Words - eliminate commonly occurring useless words like "I", "the", "we", "is", and "an"; 3) Stemming of Root Words - use tools like Porter's Stemmer to reduce words to their root form (e.g., "cats" to "cat"); 4) Record Document IDs - if the word exists, add a document reference; otherwise, create a new entry. Add additional information like frequency and location.^[1]

Q5: How does an inverted index improve search efficiency?

Answer: Inverted indexes allow for efficient searching of large volumes of text-based data by indexing every term in every document. The index can quickly identify all documents containing a given search term, significantly reducing search time compared to scanning all documents sequentially. This enables fast full-text searches at the cost of increased processing when adding documents to the database.^[1]

Q6: What are the advantages of using inverted indexes?

Answer: Key advantages include: 1) Allows fast full-text searches with reduced query time; 2) Easy to develop and implement; 3) Most popular data structure in document retrieval systems, used on large scale in search engines; 4) Supports fast updates for near-real-time indexing; 5) Can be customized for different query types like Boolean or proximity queries; 6) Supports compression, stemming, synonym expansion, and multiple languages.^[1]

Q7: What are the disadvantages or limitations of inverted indexes?

Answer: The main disadvantages are: 1) Large storage overhead and high maintenance costs for updating, deleting, and inserting documents; 2) Records are retrieved in the order they occur in inverted lists rather than in decreasing order of usefulness; 3) Initial creation is computationally expensive for large document collections.^[1]

Q8: How can inverted indexes support stemming and synonym expansion?

Answer: Inverted indexes can be configured to support stemming by storing word stems instead of full words, which improves search accuracy. Stemming reduces words to their base form (e.g., "running" to "run"). Synonym expansion maps different words with similar meanings to a common term, so searching for one term retrieves documents containing its synonyms. This improves the relevance of search results.^[1]

Q9: In your practical implementation, how do you tokenize documents?

Answer: Documents are converted to lowercase using the lower() method and then split into words using the split() function. For example: tokens1 = document1.lower().split() creates a list of lowercase words. The tokens from all documents are then combined into a list of unique terms using set operations.^[1]

Q10: Explain the data structure used to store the inverted index in Python.

Answer: A Python dictionary (hash map) is used to store the inverted index. The keys are terms (words), and the values are lists of document identifiers containing those terms. For example: inverted_index = {}; inverted_index[term] = ["Document 1", "Document 2"].^[1]

Q11: How would you handle a search query using the inverted index?

Answer: To handle a search query, the system queries the inverted index for the search terms and retrieves the list of documents associated with each term. For multiple terms, the system can perform set operations (union for OR, intersection for AND). The search engine then ranks documents based on relevance to the query and presents them in order of importance.^[1]

Q12: What is the time complexity of searching using an inverted index?

Answer: The time complexity for searching a single term is O(1) for hash table lookup plus O(k) where k is the number of documents containing the term. This is significantly faster than O(n*m) for scanning all documents sequentially, where n is the number of documents and m is the average document length.^[1]

Q13: How can inverted indexes be compressed to reduce storage?

Answer: Various compression techniques can be used including delta encoding (storing differences between consecutive document IDs), gamma encoding, and variable byte encoding. These techniques compress the posting lists efficiently while maintaining fast decompression for queries.^[1]

Q14: Why are inverted indexes particularly useful for search engines?

Answer: Search engines deal with massive collections of web pages where sequential scanning would be prohibitively slow. Inverted indexes enable sub-second query response times by directly mapping terms to relevant documents. They support ranked retrieval, phrase queries, proximity searches, and can be distributed across multiple servers for scalability.^[1]

Q15: In a Word-Level Inverted Index, how is position information stored?

Answer: In a Word-Level Inverted Index, each word entry stores not just document IDs but also the positions where the word appears within each document. For example: "is" → (2, 3), (3, 2) means "is" appears at position 3 in document 2 and position 2 in document 3. This enables phrase and proximity queries.^[1]

IR Practical 3: Bayesian Network for Heart Disease Diagnosis

Q1: What is a Bayesian Network and what are its main components?

Answer: A Bayesian Network is a directed acyclic graph where each edge corresponds to a conditional dependency and each node represents a unique random variable. It consists of two major parts: 1) A directed acyclic graph with random variables represented by nodes; 2) A set of conditional probability distributions (CPD) where each node's CPD is defined for every possible outcome of its preceding causal nodes.^[1]

Q2: Explain the concept of directed acyclic graph (DAG) in Bayesian networks.

Answer: A directed acyclic graph consists of nodes connected by directed edges with no cycles. In Bayesian networks, nodes represent random variables and edges represent causal or probabilistic dependencies. The graph is acyclic, meaning you cannot follow the directed edges and return to a starting node. This structure encodes conditional independence relationships among variables.^[1]

Q3: What is the Heart Disease Dataset used in this practical?

Answer: The Cleveland Heart Disease Database contains 76 attributes, but experiments typically use a subset of 14 attributes. The dataset has 303 instances with the "heartdisease" field (integer from 0 to 4) indicating disease presence: 164 cases with value 0 (no disease), 55 with value 1, 36 with value 2, 35 with value 3, and 13 with value 4. It includes attributes like age, sex, chest pain type, blood pressure, cholesterol, and various cardiac measurements.^[1]

Q4: List and explain at least 5 key attributes from the Heart Disease Dataset.

Answer: Key attributes include: 1) **age** - patient age in years; 2) **sex** - 1 for male, 0 for female; 3) **cp (chest pain type)** - values 1-4 representing typical angina, atypical angina, non-anginal pain, or asymptomatic; 4) **trestbps** - resting blood pressure in mm Hg on hospital admission; 5) **chol** - serum cholesterol in mg/dl.^[1]

Q5: What is the objective of using a Bayesian network for medical diagnosis?

Answer: The objective is to calculate the posterior conditional probability distribution of each possible unobserved cause given observed evidence, i.e., $P(\text{Cause} \mid \text{Evidence})$.^[1] In medical diagnosis, this means determining the probability of disease presence given patient symptoms and test results.^[1] The Bayesian network models causal relationships between risk factors and disease outcomes.^[1]

Q6: Explain how conditional probability distributions (CPD) are learned in the practical.

Answer: CPDs are learned using Maximum Likelihood Estimation (MLE) from the training data. This statistical method estimates the parameters of probability distributions that maximize the likelihood of observing the given data. For each node in the network, the CPD defines probabilities for all possible value combinations of the node and its parents.^[1]

Q7: What is inference in Bayesian networks?

Answer: Inference is the process of computing posterior probabilities of query variables given observed evidence. In the practical, inference queries include calculating the probability of heart disease given specific conditions like age=28 or cholesterol=100. The network propagates evidence through the graph structure using the CPDs to compute updated probabilities.^[1]

Q8: Interpret the inference result: Probability of HeartDisease given Age=28.

Answer: The inference result shows: heartdisease_0 (no disease): 0.6791 (67.91%); heartdisease_1: 0.1212 (12.12%); heartdisease_2: 0.0810 (8.10%); heartdisease_3: 0.0939 (9.39%); heartdisease_4: 0.0247 (2.47%). This indicates that for a 28-year-old patient, there is a 67.91% probability of no heart disease and lower probabilities for various disease severity levels.^[1]

Q9: How does the Bayesian network handle uncertainty in medical diagnosis?

Answer: Bayesian networks explicitly represent uncertainty through probability distributions rather than deterministic rules. Each variable has associated probabilities for different outcomes, and the network computes probabilistic inferences rather than absolute predictions. This allows the model to quantify diagnostic confidence and handle incomplete or noisy medical data.^[1]

Q10: What Python libraries are used to implement Bayesian networks?

Answer: Python ML libraries such as pgmpy (Probabilistic Graphical Models in Python) are commonly used. These libraries provide classes and APIs for creating network structures, learning CPDs from data, and performing probabilistic inference. The practical uses Jupyter Notebook as the development environment.^[1]

Q11: Why is the Cleveland database specifically mentioned for ML research?

Answer: The Cleveland database is mentioned as the only heart disease database that has been used by ML researchers to date for published experiments. It has become a standard benchmark dataset for evaluating machine learning algorithms on medical diagnosis tasks. Its 14-attribute subset is well-documented and widely accessible for research.^[1]

Q12: What does the "thal" attribute represent in the dataset?

Answer: The "thal" attribute represents thallium stress test results with three possible values: 3 = normal; 6 = fixed defect; 7 = reversible defect. This test measures blood flow to the heart and is an important indicator for diagnosing coronary artery disease.^[1]

Q13: How would you validate the accuracy of the Bayesian network model?

Answer: Model validation involves splitting data into training and testing sets, learning CPDs from training data, and evaluating prediction accuracy on test data. Metrics include classification accuracy, precision, recall, and F1-score for each disease category. Cross-validation can be used to assess model generalization, and the confusion matrix shows where the model makes errors.^[1]

Q14: Compare the probabilities for heartdisease given Age=28 vs Cholesterol=100.

Answer: For Age=28: highest probability is heartdisease_0 at 0.6791 (67.91%). For Cholesterol=100: highest probability is heartdisease_0 at 0.5400 (54.00%), significantly lower. High cholesterol is a stronger risk factor, reducing the no-disease probability by about 14 percentage points and increasing disease probabilities across all severity levels.^[1]

Q15: What are the advantages of using Bayesian networks for medical diagnosis?

Answer: Advantages include: 1) Explicit representation of causal relationships between variables; 2) Ability to handle incomplete data and uncertainty; 3) Interpretable model structure that domain experts can validate; 4) Can perform diagnostic reasoning in either direction (symptoms to disease or disease to symptoms); 5) Allows incorporation of prior medical knowledge; 6) Provides probabilistic confidence levels rather than binary predictions.^[1]

IR Practical 4: Agglomerative Hierarchical Clustering

Q1: What is hierarchical clustering and how does it differ from other clustering methods?

Answer: Hierarchical clustering is a connectivity-based clustering method that builds a hierarchy of clusters based on the similarity or distance between data points. Unlike methods like K-Means that require pre-specifying the number of clusters, hierarchical clustering does not require this specification upfront. It produces a tree-like structure (dendrogram) showing hierarchical relationships, which is more informative than flat clustering.^[1]

Q2: Explain the two main types of hierarchical clustering.

Answer: The two types are Agglomerative and Divisive clustering. Agglomerative (bottom-up approach) starts with each data point as a singleton cluster and successively merges pairs of clusters until all data is in one cluster. Divisive (top-down approach) starts with all data in one cluster and recursively splits clusters. Agglomerative clustering is more commonly used and is the focus of this practical.^[1]

Q3: Describe the agglomerative clustering algorithm step by step.

Answer: The algorithm follows these steps: 1) Initially, treat each data point as a singleton cluster (assumption: data is similar enough to be in one cluster); 2) Calculate distance/similarity between all cluster pairs; 3) Merge the two closest clusters; 4) Update the distance matrix; 5) Repeat steps 2-4 until all data points are merged into a single cluster or desired number of clusters is reached.^[1]

Q4: What is a dendrogram and what information does it provide?

Answer: A dendrogram is a tree-like diagram that depicts the hierarchical relationships between clusters. Individual data points are at the bottom, and the largest cluster containing all data is at the top. The height at which branches merge indicates the distance or dissimilarity between clusters. By examining the dendrogram and measuring where branches form distinct clusters, you can determine the optimal number of clusters by cutting at appropriate heights.^[1]

Q5: List and explain different types of clustering algorithms mentioned in the theory.

Answer: Four types are mentioned: 1) **Connectivity-based** (e.g., Hierarchical clustering) - builds clusters based on connectivity between data points; 2) **Centroid-based** (e.g., K-Means, K-Modes) - forms clusters around centroids; 3) **Distribution-based** (e.g., Gaussian Mixture Models) - models data points as generated from probability distributions; 4) **Density-based** (e.g., DBSCAN) - groups data points in high-density regions and separates low-density regions.^[1]

Q6: What distance metrics can be used in hierarchical clustering?

Answer: Common distance metrics include Euclidean distance (straight-line distance between points), Manhattan distance (sum of absolute differences), Cosine similarity (angle between vectors), and Correlation distance. The choice depends on data characteristics and application requirements. Different metrics can produce different clustering results for the same dataset.^[1]

Q7: What linkage criteria can be used to measure distance between clusters?

Answer: Common linkage criteria include: Single linkage (minimum distance between points in clusters), Complete linkage (maximum distance), Average linkage (average distance between all pairs), and Ward's method (minimizes within-cluster variance). The linkage criterion significantly affects the clustering structure and dendrogram shape.^[1]

Q8: Why is hierarchical clustering also called HAC (Hierarchical Agglomerative Clustering)?

Answer: HAC specifically refers to the bottom-up or agglomerative approach of hierarchical clustering. The term emphasizes the agglomerative nature where clusters are successively merged from individual points. It distinguishes this approach from divisive hierarchical clustering.^[1]

Q9: What dataset is used in this practical and why?

Answer: The Credit Card Dataset is used in this practical. This dataset is suitable for demonstrating clustering as it contains customer information with multiple features that can reveal natural groupings or segments of customers with similar characteristics. This helps in customer segmentation and targeted marketing strategies.^[1]

Q10: What are the hardware and software requirements for implementation?

Answer: Hardware requirements include PIV processor, 2GB RAM, and 500GB HDD. Software requirements include Jupyter Notebook for development environment and basic Python knowledge as prerequisite. Python libraries such as scikit-learn, numpy, pandas, and matplotlib are typically used for implementation.^[1]

Q11: How do you determine the optimal number of clusters from a dendrogram?

Answer: Examine the dendrogram to identify the height where branches form distinct, well-separated clusters. Look for large vertical distances (gaps) between successive merges. Cut the dendrogram at this height to obtain the optimal number of clusters. This method is more interpretable than methods requiring pre-specification of cluster count.^[1]

Q12: What is the computational complexity of agglomerative clustering?

Answer: The time complexity is $O(n^3)$ for the basic algorithm where n is the number of data points, as it requires computing pairwise distances and updating the distance matrix at each iteration. With efficient data structures like priority queues, this can be reduced to $O(n^2 \log n)$. The space complexity is $O(n^2)$ for storing the distance matrix.^[1]

Q13: What are the advantages of agglomerative hierarchical clustering?

Answer: Advantages include: 1) No need to pre-specify number of clusters; 2) Produces a hierarchy showing relationships at multiple scales; 3) Dendrogram provides intuitive visualization; 4) Works well with non-globular cluster shapes; 5) Deterministic results (no random initialization); 6) Can use various distance metrics and linkage criteria.^[1]

Q14: What are the limitations or disadvantages of this method?

Answer: Limitations include: 1) High computational complexity ($O(n^3)$ or $O(n^2 \log n)$) making it unsuitable for very large datasets; 2) Sensitive to noise and outliers; 3) Once a merge is done, it cannot be undone (greedy algorithm); 4) Difficulty in handling datasets with different densities; 5) Large memory requirements for distance matrix.^[1]

Q15: How would you evaluate the quality of clustering results?

Answer: Clustering quality can be evaluated using internal validation metrics like Silhouette Score (measures how similar points are to their own cluster vs. other clusters), Davies-Bouldin Index (ratio of within-cluster to between-cluster distances), and Calinski-Harabasz Index (ratio of between-cluster to within-cluster dispersion). Visual inspection of the dendrogram and domain knowledge validation are also important.^[1]

IR Practical 5: PageRank Algorithm

Q1: What is PageRank and who developed it?

Answer: PageRank is an algorithm used by Google Search to rank websites in search engine results. It was named after Larry Page, one of the founders of Google. PageRank measures the importance of website pages by counting the number and quality of links to a page. The underlying assumption is that more important websites receive more links from other websites.^[1]

Q2: Explain the basic principle behind PageRank calculation.

Answer: PageRank works by outputting a probability distribution representing the likelihood that a person randomly clicking on links will arrive at any particular page. The PageRank value for page u depends on the PageRank values of all pages linking to it, divided by the number of outbound links from each of those pages. The algorithm is based on the concept that a page is important if other important pages link to it.^[1]

Q3: What is the mathematical formula for calculating PageRank?

Answer: The PageRank value for page u is expressed as: $PR(u) = \sum(PR(v)/L(v))$ where v ranges over all pages linking to u. PR(v) is the PageRank of page v, and L(v) is the number of outbound links from page v. The PageRank transferred from a page is divided equally among all its outbound links.^[1]

Q4: Walk through the simplified PageRank example with 4 pages (A, B, C, D).

Answer: Consider four pages A, B, C, D with initial PageRank 0.25 each. If B links to C and A, C links to A, and D links to all three pages: First iteration: B transfers 0.125 to A and 0.125 to C; C transfers 0.25 to A; D transfers approximately 0.083 to each of A, B, and C. After this iteration, page A will have PageRank of approximately 0.458. Multiple iterations are required for convergence.^[1]

Q5: What is the damping factor in PageRank and why is it used?

Answer: The damping factor is a parameter used in PageRank calculation, typically set to 0.85. It represents the probability that a user will continue clicking links rather than jumping to a random page. It is likened to an "income tax" that the system extracts. The damping factor prevents certain pages from accumulating too much PageRank and ensures the algorithm converges.^[1]

Q6: How is PageRank initialized at the beginning of computation?

Answer: PageRank is initialized to the same value for all pages in the collection. In the original form, the sum of PageRank over all pages equaled the total number of pages, so each page started with value 1. In later versions using probability distributions between 0 and 1, each page starts with an equal fraction (e.g., 0.25 for 4 pages).^[1]

Q7: What does it mean that PageRank requires several iterations?

Answer: PageRank computations require multiple passes (iterations) through the document collection to adjust approximate PageRank values. Each iteration updates the PageRank based on the values from the previous iteration. The process continues until values converge (changes between iterations become negligible) or reach a predetermined number of iterations.^[1]

Q8: Why are links from a page to itself or multiple links to the same page ignored?

Answer: Self-links are ignored because they don't provide information about the page's importance relative to others. Multiple outbound links from one page to another single page are treated as one link to prevent artificially inflating the target page's importance. This prevents manipulation of PageRank through link spamming.^[1]

Q9: How can PageRank be used for document collections of any size?

Answer: The PageRank algorithm is scalable and can be calculated for document collections of any size. The probability distribution is evenly divided among all documents initially. The iterative computation can be parallelized and distributed across multiple servers for large-scale collections like the entire web.^[1]

Q10: What Python libraries or tools can be used to implement PageRank?

Answer: Python can be used with BeautifulSoup for web scraping and extracting link structures. NetworkX library provides graph data structures and PageRank implementation. NumPy can be used for efficient matrix computations in the iterative algorithm. Jupyter Notebook is recommended as the development environment.^[1]

Q11: Is PageRank the only algorithm Google uses for ranking?

Answer: No, PageRank is not the only algorithm used by Google to order search engine results. While it was the first algorithm used by the company and is the best-known, Google now uses hundreds of signals and algorithms in its ranking system. Modern search ranking considers content relevance, user experience, mobile-friendliness, and many other factors.^[1]

Q12: How does the number of outbound links affect PageRank transfer?

Answer: The PageRank transferred from a page is divided equally among all its outbound links. If a page has L outbound links, each link transfers PR/L of the page's PageRank. For example, if page B has PageRank 0.25 and links to 2 pages, each receives 0.125. This prevents pages from gaining disproportionate importance by having many inbound links from the same source.^[1]

Q13: What are the practical applications of PageRank beyond web search?

Answer: Beyond web search, PageRank can be applied to any network with link structures. Applications include ranking scientific papers by citations, identifying influential users in social networks, recommending products based on purchase networks, ranking proteins in biological networks, and prioritizing roads in transportation networks.^[1]

Q14: How does Beautiful Soup help in implementing PageRank?

Answer: Beautiful Soup is a Python library for parsing HTML and XML documents and extracting data from web pages. It can be used to crawl web pages, extract hyperlinks, and build the link graph structure required for PageRank computation. It handles various HTML formats and makes web scraping straightforward.^[1]

Q15: What is meant by "centrality measure" in the context of PageRank?

Answer: Centrality measure refers to metrics that identify the most important or central nodes in a network. PageRank is a type of centrality measure that calculates node importance based on the link structure. The centrality measure of PageRank is calculated for a given graph to determine which pages are most central or important. It is not implemented for multi-graphs (graphs with multiple edges between the same pair of nodes).^[1]

QUANTUM ARTIFICIAL INTELLIGENCE (QAI) PRACTICALS

QAI Practical 1: 16 Qubit Random Number Generator

Q1: What is a qubit and how does it differ from a classical bit?

Answer: A qubit (quantum bit) is the basic unit of quantum information, analogous to a classical bit. Unlike classical bits which can only be in state 0 or 1, a qubit can exist in a superposition of both states simultaneously until measured. When measured, the qubit's superposition collapses to either 0 or 1 with certain probabilities determined by its quantum state.^[2]

Q2: What is the objective of the 16 Qubit Random Number Generator practical?

Answer: The objectives are: 1) Understand creation of Qubit circuits; 2) Create a 16 Qubit Random Number Generator. The outcome is displaying a 16-bit random number generated through quantum processes.^[2]

Q3: What software and tools are required for this practical?

Answer: Requirements include: Python 3.x or above, Pip (package management system included with Python 3.x), Qiskit (installed via pip install qiskit), and an IBM Q Account to run programs on IBM quantum devices or simulators. Users can sign up for IBM Q at quantum-computing.ibm.com.^[2]

Q4: Explain the four main steps to create a quantum random number generator.

Answer: The four steps are: Step 1 - Initialize 16 quantum registers and 16 classical registers using QuantumRegister(16,'q') and ClassicalRegister(16,'c'); Step 2 - Create the quantum circuit combining the registers using QuantumCircuit(q,c); Step 3 - Apply Hadamard gates to all qubits using circuit.h(q) to create superposition; Step 4 - Measure the qubits using circuit.measure(q,c) to collapse superposition into 0 or 1.^[2]

Q5: What is the role of the Hadamard gate in this implementation?

Answer: The Hadamard gate is used to put a qubit into a superposition of states 0 and 1. When a Hadamard gate is applied, the qubit exists in both states simultaneously with equal probability. When measured, the qubit will collapse to either 1 or 0 with equal probability (50-50 chance). This creates true randomness at the quantum level.^[2]

Q6: What are quantum registers and classical registers?

Answer: Quantum registers are collections of qubits that store quantum information in superposition states. In this practical, a 16-qubit quantum register is initialized using QuantumRegister(16,'q'). Classical registers are standard binary registers that store measurement results (0 or 1). A 16-bit classical register is initialized using ClassicalRegister(16,'c') to store the collapsed quantum states after measurement.^[2]

Q7: Why is measurement necessary in quantum random number generation?

Answer: Measurement is necessary because it collapses the qubit's superposition state into a definite classical value (0 or 1). Without measurement, qubits remain in quantum superposition and cannot produce classical random bits. The measurement operation circuit.measure(q,c) reads the quantum state and stores the result in classical bits.^[2]

Q8: How do you connect to IBM's quantum computers in the code?

Answer: Connection is established using `IBMQ.enable_account('API TOKEN')` where you insert your personal API token. Then use `provider = IBMQ.get_provider(hub='ibm-q')` to get access to IBM quantum resources. You can select a specific backend like `provider.get_backend('ibmq_qasm_simulator')` to run on a simulator or real quantum device.^[2]

Q9: Interpret the sample output: RESULT: {'1010011011101010': 1}

Answer: The output shows a 16-bit random binary number '1010011011101010' generated by measuring the 16 qubits after applying Hadamard gates. The ':1' indicates this pattern appeared once in a single execution (`shots=1`). Each bit represents the collapsed state of one qubit, providing a truly random number generated from quantum phenomena.^[2]

Q10: What is the significance of the circuit diagram showing H gates and measurements?

Answer: The circuit diagram visualizes the quantum operations^[2]. Each horizontal line represents a qubit (q through q) initialized to $|0\rangle$ ^[2]. The 'H' symbols represent Hadamard gates applied to each qubit^[2]. The measurement symbols (meter icons) show where qubits are measured and results stored in classical register c (bits 0-15)^[2]. This provides a complete visual representation of the quantum algorithm^[2].

Q11: Why is quantum random number generation considered better than classical methods?

Answer: Quantum random number generation produces true randomness based on fundamental quantum mechanical processes, unlike classical pseudo-random number generators which are deterministic algorithms. The superposition and measurement of qubits creates unpredictability that cannot be reproduced or predicted. This is crucial for cryptography, scientific simulations, and applications requiring high-quality randomness.^[2]

Q12: What is the purpose of the barrier in quantum circuits?

Answer: Barriers are used to organize the circuit and make it more visually clear by separating different stages of the algorithm. They prevent certain circuit optimizations during compilation that might reorder gates. Barriers have no effect on the quantum computation itself but improve readability and control over gate execution order.^[2]

Q13: What does "shots=1" mean in the execute command?

Answer: The "shots" parameter specifies how many times the quantum circuit should be executed. Setting `shots=1` means the circuit runs once and returns one random number. Higher values (e.g., `shots=1000`) would run the circuit multiple times and return statistics showing the frequency of different outcomes.^[2]

Q14: How would results differ if run on a real quantum computer vs. simulator?

Answer: A simulator like 'ibmq_qasm_simulator' perfectly implements quantum mechanics without errors. Real quantum computers have noise, decoherence, and gate errors that can affect results. For random number generation, both should produce random outputs, but real hardware might have slight biases due to imperfections. Real quantum computers also have limited availability and queue times.^[2]

Q15: What are the viva questions provided in the manual for this practical?

Answer: The six viva questions are: 1) What is Qubit?; 2) What are different steps performed to generate random number?; 3) What are different types of registers?; 4) What is quantum circuit?; 5) How you measure the qubits?; 6) What is Hadamard gate?^[2]

QAI Practical 2: Quantum Teleportation Algorithm

Q1: What is Quantum Teleportation and why is it necessary?

Answer: Quantum Teleportation is the process of transferring quantum information from one location to another without physically moving the quantum particle. It is necessary because of the no-cloning theorem - quantum bits remain in quantum superposition until observed, and observing collapses their state. Since we cannot copy or directly observe quantum states without destroying them, teleportation provides a method to transfer quantum data.^[2]

Q2: Explain the no-cloning theorem mentioned in the theory.

Answer: The no-cloning theorem states that it is impossible to create an exact copy of an arbitrary unknown quantum state. Quantum bits remain in quantum state until observed; once observed or measured, they collapse to one of the known states. This fundamental principle prevents simple copying of quantum information like we do with classical data.^[2]

Q3: Describe the roles of Kartik and Sharanya in the teleportation example.

Answer: In the example, Sharanya wants to send a quantum bit $|\psi\rangle$ to Kartik^[2]. Since she cannot observe the qubit's state due to the no-cloning theorem, she uses a quantum 'portal' (entangled pair) to transfer the data^[2]. The portal creates entanglement between one qubit from Sharanya and one of its own, sending the entangled pair to Kartik^[2]. Kartik then performs operations to remove entanglement and receive the output^[2].

Q4: What is a Bell pair and how is it created?

Answer: A Bell pair is a special entangled pair of qubits used in quantum teleportation^[2]. It is created using quantum circuits by: 1) Taking one qubit and applying a Hadamard gate to put it in $|+\rangle$ or $|-\rangle$ state^[2]; 2) Applying a CNOT gate on the second qubit, controlled by the first qubit^[2]. This creates entanglement where the state of one qubit is correlated with the other^[2].

Q5: Explain the step-by-step algorithm for quantum teleportation.

Answer: Step 1: Portal creates Bell pair; one qubit (Q1) goes to Sharanya, other (Q2) to Kartik^[2]. Step 2: Sharanya applies CNOT gate to Q1 controlled by $|\psi\rangle$ (qubit to send)^[2]. Step 3: Sharanya applies Hadamard to $|\psi\rangle$, then measures both qubits and sends classical bits (00, 01, 10, or 11) to Kartik^[2]. Step 4: Based on received bits, Kartik applies transformations: 00 → I gate, 01 → X gate, 10 → Z gate, 11 → ZX gate to Q2^[2]. Result: Q2 is now in the same state as original $|\psi\rangle$ ^[2].

Q6: What is a CNOT gate and what does it do?

Answer: CNOT (Controlled-NOT) gate is a two-qubit gate that implements the 'if this, then that' condition in quantum computing. It has a control qubit and a target qubit. If the control qubit is 1, it flips (applies NOT/X gate to) the target qubit; if control is 0, target remains unchanged. It is fundamental for creating entanglement.^[2]

Q7: Why can classical bits be sent between Sharanya and Kartik but not the quantum state?

Answer: Classical bits can be transmitted because they represent definite values (0 or 1) that can be copied and sent through classical channels. The quantum state cannot be sent directly because: 1) Observing it would collapse the superposition, destroying the quantum information; 2) The no-cloning theorem prevents copying quantum states. Teleportation uses classical communication combined with quantum entanglement to transfer the state.^[2]

Q8: What are the possible measurement outcomes and required transformations?

Answer: The four possible measurement outcomes and corresponding transformations are: 00 → Apply I gate (identity, no change); 01 → Apply X gate (bit flip); 10 → Apply Z gate (phase flip); 11 → Apply ZX gate (both phase and bit flip). These transformations come from quantum mechanics principles and ensure the final qubit matches the original state.^[2]

Q9: What does the practical implementation output demonstrate?

Answer: The practical shows a histogram of measurement results after teleportation. When run with 1000 shots, it displays the frequency of different bit string outcomes (like '100', '101', '110', '111'). The probability distribution (ranging from ~0.238 to ~0.256) demonstrates the quantum behavior and verifies the teleportation protocol works correctly.^[2]

Q10: Explain the circuit diagram for quantum teleportation shown in the output.

Answer: The circuit has 3 qubits and 3 classical bits. q0 starts with X gate (to prepare state), then has CNOT and H gates, and is measured to bit 0. q1 has H gate then CNOT with q2 (creating Bell pair), then CNOT with q0, measured to bit 1. q2 (target) has controlled-X from q1 and controlled-Z from q0, measured to bit 2. This visualizes the complete teleportation protocol.^[2]

Q11: What is entanglement in quantum computing?

Answer: Entanglement is a quantum phenomenon where two or more qubits become correlated such that the state of one qubit instantaneously affects the state of others, regardless of distance. In teleportation, entanglement is created using Hadamard and CNOT gates between Q1 and Q2. This correlation allows quantum information to be transferred using the entangled pair as a quantum channel.^[2]

Q12: Why is Qiskit used for this implementation?

Answer: Qiskit is an open-source framework for quantum computing that provides tools for creating and manipulating quantum programs. It allows running quantum circuits on IBM Q Experience (real quantum hardware) or simulators on local computers. It provides high-level abstractions for quantum gates, circuits, and measurements, making quantum algorithm implementation accessible.^[2]

Q13: What is the significance of the Hadamard basis mentioned in the theory?

Answer: The Hadamard basis is a quantum basis consisting of $|+\rangle$ and $|-\rangle$ states, created by applying Hadamard gates to the computational basis $|0\rangle$ and $|1\rangle$.^[2] It represents equal superposition states^[2]. Applying Hadamard to q0 in teleportation helps encode quantum information in a form that can be extracted through measurements and classical communication^[2].

Q14: What are the three viva questions provided for this practical?

Answer: The viva questions are: 1) What is entanglement?; 2) What are different gates used in teleportation?; 3) What is teleportation?; 4) How Quantum circuit is designed?^[2]

Q15: How does quantum teleportation differ from classical information transfer?

Answer: Classical information transfer involves copying and sending bits directly. Quantum teleportation does not physically move the quantum state or violate speed-of-light limits. Instead, it uses quantum entanglement and classical communication to reconstruct the quantum state at the destination. The original quantum state is destroyed during measurement (no cloning), and the process requires both quantum entanglement and classical bits.^[2]

QAI Practical 3: 5 Qubit Quantum Fourier Transform

Q1: What is the Quantum Fourier Transform (QFT)?

Answer: The Quantum Fourier Transform is a quantum circuit that transforms the state of qubits from the computational basis to the Fourier basis (also called Hadamard basis). It is the quantum analogue of the discrete Fourier Transform used in classical computing. QFT is implemented using Hadamard gates and Controlled-U1 gates.^[2]

Q2: What is a Controlled-U1 gate and what does it do?

Answer: A Controlled-U1 gate implements a single rotation around the Z-axis (phase rotation) of the target qubit if and only if the control qubit is in state 1. If the control qubit is 0, no rotation is applied. The rotation angle is specified as a parameter (like $\pi/2$, $\pi/4$, $\pi/8$).^[2]

Q3: Describe the simplest 1-qubit QFT.

Answer: The simplest QFT is a 1-qubit QFT which only implements a single Hadamard gate. The circuit diagram shows just $q \rightarrow H \rightarrow C$ (where C represents classical bit). This puts the single qubit into superposition, which is the Fourier basis for one qubit.^[2]

Q4: Explain the structure of a 2-qubit QFT circuit.

Answer: A 2-qubit QFT has the following structure: q0 gets Hadamard gate, then Controlled-U1 with rotation $\pi/2$ is applied with q0 as control and q1 as target; q1 gets Hadamard gate after the controlled operation; Finally, swap gate is applied to q0 and q1. The swap gates are not needed if QFT is at the end of the circuit.^[2]

Q5: What is the pattern for rotation angles in an N-qubit QFT?

Answer: For N qubits, the rotation angles double for each subsequent qubit. For q0: rotations are $\pi/2$ for q1, $\pi/4$ for q2, $\pi/8$ for q3, and so on. This pattern repeats for each qubit - when all rotations are applied to a qubit, it gets a Hadamard gate. The rotated qubit can then be used as control for target qubits below it.^[2]

Q6: Describe the complete 5-qubit QFT circuit shown in the output.

Answer: The 5-qubit QFT circuit shows: q0 with Hadamard and multiple controlled rotations ($\pi/2$, $\pi/4$, $\pi/8$, $\pi/16$); q1 with Hadamard and controlled rotations ($\pi/2$, $\pi/4$, $\pi/8$); q2 with X gate, Hadamard, and rotations ($\pi/2$, $\pi/4$); q3 with Hadamard and $\pi/2$ rotation; q4 with X gate and Hadamard; followed by measurements to classical register c.^[2]

Q7: Why do qubits need to be in superposition after QFT?

Answer: After QFT, qubits are in superposition because the computational basis values have been encoded into the Fourier (Hadamard) basis. The controlled U1 gates encode the phase information, and the Hadamard gates create superposition. This means values originally in computational basis are now represented as quantum amplitudes and phases.^[2]

Q8: What happens when you measure qubits after applying QFT?

Answer: When measuring after QFT without inverse QFT, you get random values because qubits are in superposition. The values encoded in computational basis are now encoded in the Hadamard basis via controlled U1 gates. The measurement collapses the superposition randomly, so the original encoded pattern is not recovered.^[2]

Q9: What is the Inverse Quantum Fourier Transform and why is it needed?

Answer: The inverse QFT reverses all rotations done in the forward QFT. If there was a rotation of π in QFT, inverse QFT applies rotation of $-\pi$. In Qiskit, it is implemented by setting inverse=True in the QFT function. It is needed to recover the original computational basis values that were encoded in the Fourier basis by the forward QFT.^[2]

Q10: Explain the output when QFT is run with the encoded value 10101.

Answer: When running QFT with input '10101' (qubits q0, q2, q4 set with X gates), the output shows multiple random values like '00000', '00001', '10000', etc. with various frequencies. Each outcome has roughly similar probability (~2-4%), demonstrating that qubits are in superposition after QFT. The original pattern is hidden in quantum phases and amplitudes.^[2]

Q11: Explain the output when QFT is followed by inverse QFT.

Answer: When QFT is followed by inverse QFT with input '10101', the output shows '10101: 1000' (1000 occurrences out of 1000 shots). This demonstrates that inverse QFT perfectly recovers the original encoded value. The forward and inverse operations cancel each other, returning qubits to computational basis.^[2]

Q12: What are the parameters of the QFT function in Qiskit?

Answer: The QFT function parameters include: num_qubits=5 (number of qubits); approximation_degree=0 (level of approximation); do_swaps=True (whether to include swap gates); inverse=False/True (forward or inverse QFT); insert_barriers=True/False (for visualization); name='qft' (circuit name).^[2]

Q13: Why are swap gates sometimes optional in QFT?

Answer: Swap gates reverse the qubit order at the end of QFT. They are not needed if the QFT is implemented at the end of the quantum circuit, because the reversed order can be accounted for when reading the final results. Including them makes the circuit more intuitive but adds extra gate operations.^[2]

Q14: What applications use Quantum Fourier Transform?

Answer: QFT is a key component in many quantum algorithms including Shor's algorithm for integer factorization, quantum phase estimation which is fundamental to many quantum algorithms, quantum simulation algorithms, and solving hidden subgroup problems. It provides exponential speedup over classical Fourier transform for certain problems.^[2]

Q15: How does the number of gates scale with number of qubits in QFT?

Answer: For n qubits, QFT requires n Hadamard gates, one for each qubit. The number of controlled rotation gates is $n(n-1)/2$, since each qubit needs controlled rotations from all preceding qubits. This is $O(n^2)$ gates total, which is polynomial and efficient for quantum computers. Classical FFT requires $O(n \cdot 2^n)$ operations for n qubits.^[2]

QAI Practical 4: Tackle Noise with Error Correction

Q1: What is quantum error correction (QEC) and why is it essential?

Answer: Quantum error correction is a technique used to protect quantum information from errors caused by decoherence and noise. It is theorized as essential to achieve fault-tolerant quantum computing that can reduce effects of noise on stored quantum information, faulty quantum gates, faulty quantum preparation, and faulty measurements. This allows algorithms of greater circuit depth to be executed reliably.^[2]

Q2: What are the main sources of noise in quantum computing?

Answer: Main sources of noise include: environmental interactions causing decoherence, imperfect quantum gates that don't implement exact operations, faulty qubit preparation leading to incorrect initial states, and measurement errors producing incorrect readouts. Noise is a major challenge in quantum computing as it causes errors in quantum computations.^[2]

Q3: Explain the basic principle of how QEC works.

Answer: QEC works by encoding a single logical qubit into multiple physical qubits, creating redundancy. This redundancy allows QEC to detect and correct errors that occur on the physical qubits. The encoding spreads quantum information across multiple qubits such that errors on individual qubits can be identified and corrected without destroying the logical quantum information.^[2]

Q4: What are the different types of QEC codes mentioned?

Answer: There are various QEC codes, each with its own strengths and weaknesses. Some QEC codes are more efficient (using fewer physical qubits per logical qubit), while others are more robust to noise (correcting more error types). The manual mentions that QEC is still under development and codes are being improved.^[2]

Q5: Describe the quantum communication example given in the theory.

Answer: In the example: 1) Sender encodes a single logical qubit into multiple physical qubits using QEC code; 2) Sender transmits physical qubits to receiver; 3) Receiver uses QEC code to detect and correct errors from transmission; 4) Receiver decodes physical qubits to recover the logical qubit. If few errors occur, QEC corrects them and receiver recovers the qubit accurately; if too many errors occur, recovery fails.^[2]

Q6: What is measurement error mitigation and how does it work?

Answer: Measurement error mitigation corrects errors that occur during the measurement process itself. The practical uses CompleteMeasFitter from qiskit-ignis to perform mitigation. Calibration circuits measure the actual error rates of the measurement process, then these rates are used to filter and correct the measured counts, producing mitigated counts that more accurately represent the true quantum state.^[2]

Q7: Explain the code structure for implementing error correction.

Answer: The code includes: 1) Import qiskit-ignis library (pip install qiskit-ignis); 2) Define quantum circuit with 3 qubits and 3 classical bits; 3) Apply quantum gates (H, CNOT) and measurements; 4) Execute on simulator to get noisy results; 5) Generate calibration circuits using complete_meas_cal() for qubits; 6) Run calibration and fit measurement fitter; 7) Apply filter to original counts to get mitigated counts.^[2]

****Q8: What does the output histogram show?**

*

1. CL-II_Lab_Manual_IR.docx
2. QAI-Lab-Manual.pdf