# Documents Report: Understanding the Interference Model and Belief Network

This report provides an explanation of the code and its functionality in the context of the two models: **Interference Model** and **Belief Network**. We will describe the purpose and implementation of each part of the code, discussing how it relates to ranking documents based on queries.

---

## 1. Loading Documents and Relevance Judgments

### Code:

```
def load_documents(file_path):
    with open(file_path, 'r') as file:
        return [line.strip() for line in file.readlines()]

def load_relevance_judgments(file_path):
    relevance_judgments = {}
    with open(file_path, 'r') as file:
        for line in file:
            query, doc, score = line.strip().split(',')
            relevance_judgments[(query.strip(), doc.strip())] =
int(score.strip())
    return relevance_judgments
```

- **Purpose**: This section defines functions to load documents and relevance judgments from files.
    - `load_documents`: Reads the documents from a file (where each line represents a document).
    - `load_relevance_judgments`: Reads relevance judgments (query-document pairs) and stores them as a dictionary, where the key is a tuple `(query, document)` and the value is an integer representing the relevance score (e.g., 0, 1, 2).
- **Explanation**: These functions allow for flexibility in loading input data from external text files, which may contain the documents and corresponding relevance scores for queries.

---

## 2. Calculating Probabilities for the Interference Model

### Code:

```
def calculate_probabilities(documents, queries, relevance_judgments):
    total_pairs = len(relevance_judgments)
    relevance_prob = {
        pair: relevance_judgments[pair] / total_pairs
        for pair in relevance_judgments
    }
    return relevance_prob
```

- **Purpose**: This function calculates the probability of relevance for each query-document pair.

    - The probability of relevance is determined by dividing the relevance score by the total number of relevance judgments.
- **Explanation**: This step is part of the **Interference Model**, where the goal is to estimate how likely a document is relevant to a given query based on previous relevance judgments. These probabilities will be used to rank documents based on their relevance to specific queries.

---

## 3. Ranking Documents for the Interference Model

### Code:

```
def rank_documents(query, documents, relevance_prob):
    scores = [
        (doc, relevance_prob.get((query, doc), 0))
        for doc in documents
    ]
    return sorted(scores, key=lambda x: x[1], reverse=True)
```

- **Purpose**: This function ranks documents for a given query based on their relevance probability.
    - It uses the `relevance_prob` dictionary to get the relevance score for each document-query pair. If no score is found, it defaults to 0.
- **Explanation**: The **Interference Model** ranks documents by their relevance probability to a query. Documents with higher relevance probabilities are ranked higher. The sorting is done in descending order to give priority to more relevant documents.

---

## 4. Defining a Belief Network

### Code:

```
belief_network = {
    "Query": ["Relevance"],
    "Document Features": ["Relevance"],
    "Relevance": []
}
```

- **Purpose**: This section sets up the structure of a belief network that models the relationships between queries, document features, and relevance.
    - The belief network defines that the "Query" and "Document Features" influence the "Relevance," but the "Relevance" does not affect anything else.
- **Explanation**: The belief network is a probabilistic graphical model used to infer the relevance of documents based on the features of the document and the query. The model assumes dependencies between query relevance and document features but assumes no reverse dependency.

---

## 5. Probability Functions for the Belief Network

### Code:

```python
def joint_probability(prior, likelihood):
    return prior * likelihood

def marginal_probability(joint_probs):
    return sum(joint_probs.values())

def bayes_theorem(prior, likelihood, evidence):
    return (likelihood * prior) / evidence
```

- **Purpose**: These functions compute different types of probabilities:
    - `joint_probability`: Calculates the joint probability of two events (prior and likelihood).
    - `marginal_probability`: Calculates the marginal probability from a set of joint probabilities.
    - `bayes_theorem`: Implements Bayes' theorem to compute the posterior probability given prior, likelihood, and evidence.
- **Explanation**: These functions are foundational for the **Belief Network**. They enable the computation of document relevance using a probabilistic approach where relevance is inferred from prior beliefs and evidence (document features).

---

## 6. Computing Document Relevance Using Bayes' Theorem

### Code:

```python
def compute_relevance(query, doc_features, prior_prob, feature_probs):
    joint_probs = {
        feature: joint_probability(prior_prob, feature_prob)
        for feature, feature_prob in feature_probs.items()
    }
    marginal = marginal_probability(joint_probs)
    return bayes_theorem(prior_prob, feature_probs.get(query, 0), marginal)
```

- **Purpose**: This function computes the relevance of a document to a given query using a **Belief Network** approach.
    - It calculates the joint probability for each document feature and then computes the posterior relevance using Bayes' theorem.
- **Explanation**: The **Belief Network** model uses the document's features and query information to compute the relevance score for each document. By using the joint and marginal probabilities, it infers the likelihood of a document being relevant to a query.

---

## 7. Ranking Documents for the Belief Network

### Code:

```
for query in queries:
    scores = []
    for doc in documents:
        score = compute_relevance(query, doc, prior_prob, feature_probs)
        scores.append((doc, score))
    ranked_docs = sorted(scores, key=lambda x: x[1], reverse=True)
    print(f"Query: {query} -> Ranked Documents: {ranked_docs}")
```

- **Purpose**: This block computes the relevance score for each document-query pair using the **Belief Network** model and ranks the documents accordingly.
- **Explanation**: For each query, the relevance scores for all documents are computed, and then the documents are ranked in descending order of relevance. This ranking is based on the probabilistic inference made by the belief network.

---

## 8. Comparison of the Two Models

### Code:

```
print("\nComparison of Models:")
for query in queries:
    interference_ranking = rank_documents(query, documents, relevance_prob)
    belief_ranking = sorted(
        [(doc, compute_relevance(query, doc, prior_prob, feature_probs))
         for doc in documents],
        key=lambda x: x[1], reverse=True
    )
    print(f"Query: {query}")
    print(f"  Interference Model: {interference_ranking}")
    print(f"  Belief Network: {belief_ranking}")
```

- **Purpose**: This section compares the results of the **Interference Model** and **Belief Network** for each query by ranking the documents based on their relevance scores.
- **Explanation**: Both models rank the documents for each query, and the results are printed side by side for comparison. This allows you to evaluate how each model ranks the documents differently based on their respective probabilistic approaches.

---

## Conclusion

The code demonstrates two different approaches for ranking documents based on queries:

- The **Interference Model** ranks documents based on the probability of relevance derived from previous relevance judgments. It uses simple statistical techniques to rank documents according to their relevance to the query.
- The **Belief Network** uses probabilistic reasoning (Bayes' theorem) to compute document relevance by considering the document's features and prior knowledge.

Both models are implemented with flexibility, allowing the user to input queries and generate rankings of documents based on their relevance to those queries. The comparison between the two models highlights their differences in approach, with the **Interference Model** relying on direct relevance probabilities and the **Belief Network** leveraging a more sophisticated probabilistic inference framework.