**Supplementary File:**

**Strategy for Creating an External Database for Metal-Binding Sites**

**1. Data Overview**

**Metal-Binding Protein Classes:**

* **Zn**: 1647 training proteins, 211 testing proteins.
* **Mn**: 547 training proteins, 57 testing proteins.
* **Mg**: 1730 training proteins, 235 testing proteins.
* **Ca**: 1554 training proteins, 183 testing proteins.
* **Total Sequences=1968**

**2. Selection Criteria for Database Creation**

**Step 1: Protein Sampling**

* From each metal-binding protein class, select **500 representative proteins** from the training dataset for inclusion in the external database.
* For classes with fewer proteins (e.g., Mn), include all available training proteins.
* Check for overlapping protein sequences between the selected proteins and those in the training or testing datasets.
* Remove any overlapping sequences before finalizing the RAG database preparation.

**Total Sequences after removing Overlapping sequences=1948**

**Step 2: Diversity Assurance**

* Ensure proteins are selected to maximize diversity:
  + Sequence lengths.
  + Binding site locations.
* Use a stratified random sampling approach to ensure equal representation of different sequence lengths and binding site types.

**Step 3: Embedding Feature Representation**

* Extract **residue-level embeddings** using the ProtTrans model (primary embedding method).
* Incorporate **neighboring residue context** by using a **15-residue sliding window**:
  + Center residue as the target.
  + 7 residues on each side for context.
  + Embedding size: **15 x 1024 (ProtTrans)**.

**3. Embedding Generation Workflow**

**Step 1: Input Data Preparation**

1. Merge class-specific protein sequences (e.g., Zn, Mn) into a single FASTA file for each class.
2. Label each protein and residue with its corresponding class label (e.g., Zn, Mn).

**Step 2: Residue-Level Embedding Generation**

1. For each protein sequence, generate residue-level embeddings using the ProtTrans model.
2. Apply a **sliding window approach** to generate features for each residue:
   * For a protein with length **L**, create **L** samples.
   * Feature shape: **(15 x 1024)** for each residue.
3. Save embeddings and labels as .npy files for efficient storage.

**4. Database Construction**

**Step 1: Database Structuring**

* Combine residue-level embeddings for 500 selected proteins from each class into a single database.
* Include metadata for each residue:
  + **Protein ID**.
  + **Residue position**.
  + **Metal-binding class** (Zn, Mn, Mg, Ca).

**Step 2: File Organization**

* Save the database as two .npy files:
  1. **Database Features**:
     + Shape: **(N, 15, 1024)**.
     + Where **N** = total number of residues from 500 proteins/class.
  2. **Database Labels**:
     + Shape: **(N,)**.
     + Labels: 0 (non-binding) or 1 (binding).

To construct a balanced database for your RAG model, especially given the imbalance between positive (binding) and negative (non-binding) samples, it's essential to implement a strategy that ensures both classes are adequately represented. This balance enhances the model's ability to learn distinguishing features effectively.

**Proposed Strategy:**

1. **Data Collection:**
   * **Positive Samples:** Collect all available positive samples (binding residues) from your dataset.
   * **Negative Samples:** Select a subset of negative samples from the neighbors of the positive sample (non-binding residues) to achieve a desired ratio between positive and negative samples.
2. **Balancing Ratio:**
   * Aim for a ratio that provides sufficient representation of both classes. A common approach is to have a 1:1 ratio (equal number of positive and negative samples). However, depending on the specifics of your dataset and the importance of capturing variability in negative samples, you might opt for a different ratio, such as 1:2 (one positive for every two negatives).
3. **Implementation Steps:**
   * **Extract Positive Samples:** Identify and extract all positive samples from your dataset.
   * **Extract Negative Samples:** Select negative samples from the neighbors of the positive sample to match the desired ratio.
   * **Combine and Shuffle:** Combine the positive and selected negative samples, then shuffle the dataset to ensure random distribution.

**5. Database Integration for RAG**

**Step 1: Retrieval Strategy**

* To refine the representation of positive samples, a retrieval-augmented embedding strategy was employed. Each query embedding was compared against a precomputed database of protein embedding using the L2 norm as a similarity metric. The five most similar embeddings were identified, ensuring that only structurally relevant sequences contributed to the refinement process. This retrieval step effectively captured contextual similarities between proteins, allowing the incorporation of additional sequence-level information.

**Step 2: Feature Fusion**

* For each positive sample, the retrieved top 5 embeddings were averaged to generate representative contextual embedding. The final fused embedding was computed by integrating the original query embedding (70% weight) with the averaged retrieved embedding (30% weight), ensuring a balance between the intrinsic query features and external contextual information. This fusion mechanism enhanced feature representation by leveraging structural similarities while maintaining the integrity of the original embedding distribution.