PTLog 1_8

Function Demo

```
def transform(self, df: pd.DataFrame) -> pd.DataFrame:

        # BACKUP  plan
        #return pd.concat([self.transform_one(query_df) for query_df in
])

        state_active_queries = []
        for _, row in df.iterrows():
            state = {
                'qid': row['qid'],
                'query': row['query'],
                'context': self.prompt.format(question=row["query"]) if
self.prompt else row["query"],
                'search_history': [],
            }
            state_active_queries.append(state)
        state_finished_queries = []

        for turn in range(self.max_turn):
            #1. call the LLM for each query still active
            # outputs = self.generate([q['context'] for q in
state_active_queries])
            batch_answers = self.check_answers(outputs)
            outputs = self.generate([q['context'] for q in
state_active_queries])  # outputs: List[str]
            #2. check for answer in each of the :
            # if we see the question has been answered:
                # extract the answer
                # remove this query from state_active, add to
state_finished list
            batch_answers = self.check_answers(outputs)  # List[answer
or None]
            batch_querying = []
            batch_queries = []
            for i, answer in enumerate(batch_answers):
                if answer is not None:
                    finished_query = state_active_queries[i]
                    finished_query['qanswer'] = answer
                    finished_query['output'] = outputs[i]
```

```
                    state_finished_queries.append(finished_query)
                else:
                    batch_querying.append(state_active_queries[i])

 batch_queries.append(self.get_search_query(outputs[i]))  # 提取检索query

            #3. check for retrieve requirements in each of the outputs
            # build up BATCH of queries (df) to execute
            #3a. check for outputs with no ansewr and no retrieval -
that is error condition
            for i, q in enumerate(batch_querying):
                if batch_queries[i] is None or batch_queries[i] == "":
                    # 标记为异常，直接结束
                    q['qanswer'] = None
                    q['output'] = outputs[i]
                    q['error'] = "No answer and no retrieval request"
                    state_finished_queries.append(q)
            # 只保留需要检索的
            batch_querying = [q for i, q in enumerate(batch_querying) if
batch_queries[i] is not None and batch_queries[i] != ""]
            batch_queries = [q for q in batch_queries if q is not None
and q != ""]

            # 4. 执行批量检索
            #4. exectute queries
            # all_results = (self.retriever % self.top_k)(batch_queries)
            if batch_queries:
                all_results = (self.retriever %
self.top_k).search(batch_queries)
            else:
                all_results = []

            #5. get their results and add to the next context for the
right question
            # replace state_active_queries with batch_querying
            # 5. 将检索结果加入到下轮context
            for i, q in enumerate(batch_querying):
                if batch_queries:
                    docs_str = self.format_docs(all_results[i]) if
len(all_results) > i else ""
                    q['context'] += self.wrap_search_results(docs_str)
                    q['search_history'].append(batch_queries[i])
            state_active_queries = batch_querying

            # 6. if no queries left in state_active, then break
            if not state_active_queries:
                break
```

```
        # 7. combine state_finished into results_df, and anything left
in state_active that
        results = state_finished_queries + state_active_queries
        return pd.DataFrame(results)
```