Implementation of Special Token Removal

This documentation covers the changes made to handle special tokens (eg. </s>, <|eot_id|>, <|im_end|>) that were introduced by different LLM models in the Danswer application. Additionally the guide includes instructions for adding LLM models to the admin panel, providing the necessary information to ensure smooth integration.

We successfully implemented the removal of special tokens from LLM model responses and performed modifications to both the backend and frontend to ensure the tokens are removed across the application like General Chat and Knowledge Chat. Furthermore, detailed instructions for adding models to the Danswer application are provided based on the Hugging Face Inference API setup.

Server Implemented on:
(http://ec2-54-79-231-211.ap-southeast-2.compute.amazonaws.com)

(Site: http://54.79.231.211/).

1) List of LLM Models Added:

- ☑ **meta-llama/Meta-Llama-3-8B-Instruct**
- ☑ **meta-llama/Meta-Llama-3-70B-Instruct**
- ☑ **mistralai/Mixtral-8x7B-Instruct-v0.1**
- ☑ **NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO**
- ☑ **mistralai/Mistral-7B-Instruct-v0.2**

2) Admin Panel: Adding a New LLM Model:

➔ To **add** a **new model** in the **Danswer admin panel**, you **must provide details** such as **display name**, **provider name**, **API key**, **model name**, and a **default model**. These **details change** depending on the **LLM model** being **added**. Below is an example **configuration** based on **mistralai/Mixtral-8x7B-Instruct-v0.1:**

➔ Example Configuration:

- **Display Name: M2 (or your comfortable names)**
- **Provider Name: huggingface**
- **API Key**: (**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**)
- **Model Name: mistralai/Mixtral-8x7B-Instruct-v0.1**
- **Default Model: mistralai/Mixtral-8x7B-Instruct-v0.1**

➔ We **encountered special tokens** like **</s>, <|eot_id|>, and <|im_end|> returned** from these **LLM models**. To **remove** them, we made **changes** to both the **backend** and **frontend**.

3) (Message.tsx) Frontend:

Path: cd danswer/web/src/app/chat/message/Message.tsx

Summary: The **modifications** in the **frontend** were **primarily** made in the **Message.tsx file**, where we **added logic** to **remove special tokens** from the **response content** before **rendering**.

❖ Key Modification:

- **Message.tsx:** We **updated** the **processContent function** to **handle** and **remove special tokens**:

- **Code:**

```
const toolCallGenerating = toolCall && !toolCall.tool_result;
const processContent = (content: string | JSX.Element) => {
  if (typeof content !== "string") {
    return content;
  }

  // Use let instead of const to allow reassignment
  let cleanedContent = content.replace(/<\/s>|<\|eot_id\|>|<\|im_end\|>/g, "").trim();

  const codeBlockRegex = /```(\w*)\n[\s\S]*?```|```[\s\S]*?$/g;
  const matches = cleanedContent.match(codeBlockRegex);

  if (matches) {
    cleanedContent = matches.reduce((acc, match) => {
      if (!match.match(/```\w+/)) {
        return acc.replace(match, match.replace("```", "```plaintext"));
      }
      return acc;
    }, cleanedContent);

    const lastMatch = matches[matches.length - 1];
    if (!lastMatch.endsWith("```")) {
      return cleanedContent;
    }
  }
```

```
  return cleanedContent + (!isComplete && !toolCallGenerating ? " [*]() " : "");
};
```

- The **above code ensures** that **special tokens** are **stripped** out before **displaying** the **model's output** to the **user**.

- **We** also ensured that **code blocks** were **properly** handled by **adding** a **plaintext flag** where **needed**.

4) (process_message.py) Backend:

Path: cd danswer/backend/danswer/chat/process_message.py

Summary: In the **backend**, we made **changes** to handle the **special tokens similarly** when **processing** the **responses** from the **LLM models**.

❖ Key Modifications:

- We **modified** the **backend logic** that **processes** the responses from the **LLM models** to **remove** the **unwanted tokens**.

- **Code:**

```
gen_ai_response_message = partial_response(
   reserved_message_id=reserved_message_id,
   message=answer.llm_answer.replace("</s>", "").replace("<|eot_id|>",
"").replace("<|im_end|>", "").strip(),
   rephrased_query=(qa_docs_response.rephrased_query if qa_docs_response
else None),
   reference_docs=reference_db_search_docs,
   files=ai_message_files,
   token_count=len(llm_tokenizer_encode_func(answer.llm_answer)),
   citations=db_citations,
   error=None,
   tool_calls=[
      ToolCall(
         tool_id=tool_name_to_tool_id[tool_result.tool_name],
         tool_name=tool_result.tool_name,
         tool_arguments=tool_result.tool_args,
         tool_result=tool_result.tool_result,
      )
   ]
   if tool_result
```

```
    else [],
)
```

- This **code removes special tokens** from the **generated answer** before **saving** it to the **database** or **sending** it to the **frontend**.

- By using **.replace() functions**, we ensure that these **tokens** are **effectively cleaned** from all the **responses**.

5) Overall Summary:

- **Frontend:**

  File: Messages.tsx
  Path: danswer/web/src/app/chat/message/Messages.tsx
  Changes: Removal of special tokens from LLM responses in the frontend.

- **Backend:**

  File: process_message.py
  Path: danswer/backend/danswer/chat/process_message.py
  Changes: Removal of special tokens from the LLM responses at the backend level before they are sent to the frontend or stored in the database.

  To access the Danswer application as an admin, please log in as:
  Email: noelshallum@gmail.com
  Password: 12345

- **Ibrahim Sultan**