Perguntas e Respostas usando Modelos do Hugging Face

INTEGRANTES:
FELIPE RODRIGUES SANTANA
NICOLAS VIEIRA DOS SANTOS
VITORIA MARIA MENESES MOTA TEIXEIRA

DISCIPLINA: PROCESSAMENTO DE LINGUAGEM NATURAL

REPOSITÓRIO:
https://github.com/vitoriameneses/PLN_QA_LLM_2025_Vitoria_Nicolas_Felipe

Documento de Doenças Respiratórias

Seleção dos modelos

Foram utilizados três modelos gratuitos do Hugging Face ao longo da atividade. Nesse relatório, foi executado o modelo 1. Os modelos utilizados são brevemente descritos abaixo:

1. **PT-BR (Modelo 1 deste relatório)**
   pierreguillou/bert-base-cased-squad-v1.1-portuguese — Escolhido pelo suporte a língua portuguesa, possuindo maior facilidade de reconhecer termos clínicos e similares.

2. **Multilíngue (Modelo 2, para a comparação posterior)**
   AlexKay/xlm-roberta-large-qa-multilingual-finedtuned-ru — Escolhido pela alta robustez, sendo capaz de trabalhar bem com siglas e variação linguística.

3. **Inglês baseline SQuAD2 (Modelo 3, para a comparação posterior)**
   deepset/roberta-base-squad2 — Escolhido pela alta popularidade e boas referências.

Solução passo a passo

Nas seções abaixo, traremos um guia passo a passo da execução do modelo selecionado.

1) Ingestão e limpeza do PDF

- **Leitura por página** com PyMuPDF.

- **Limpeza**: Remoção dos hífens de fim de linha, remoção dos cabeçalhos e rodapés recorrentes (regex) e normalização dos espaços.

- **Segmentação**: detecção de capítulos e seções por regex. Criação de blocos semânticos.

- **Chunking**: As sentenças são empacotadas em janelas de aproximadamente 1200 caracteres com sobreposição de 150. O título do bloco é pré-fixado no 1º chunk para contexto.

Após a limpeza, foram retirados do texto os números de página e quebras de linha. Os blocos geraram chunks coesos e legíveis e não houve perda considerável de conteúdo por remoções de rodapé.

2) Recuperação por embeddings

- **Embeddings** com sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 (CPU/GPU auto).

- **Consulta**: para cada pergunta, é calculada a similaridade da pergunta com todos os chunks e selecionamos TOP_K = 3 trechos mais relevantes.

O que aconteceu: a consulta retornou consistentemente 3 prévias de trechos tematicamente alinhados ao assunto perguntado.

3) QA com janelas deslizantes

- **Modelo 1 executado**: pierreguillou/bert-base-cased-squad-v1.1-portuguese.

- **Janela deslizante** sobre (pergunta + contexto selecionado) com QA_MAX_LENGTH = 384 e QA_STRIDE = 128.

- **Seleção da resposta**: para cada janela, o modelo calcula distribuições de início/fim; escolhe-se o span com maior produto de probabilidades (softmax).

As perguntas escolhidas para esse documento foram:

- De que forma o diagnóstico da rinite alérgica é realizado e qual a importância do diagnóstico diferencial?
- Para que servem os medicamentos de atenção básica?
- Como a asma é classificada de acordo com a gravidade?

O que aconteceu (saída observada no console):

- O script imprimiu:

    - Resposta: seguido de um trecho textual curto (span extraído do contexto), em português coeso.

    - Score: um valor numérico entre 0 e 1 (produto de probabilidades de início/fim). Esse score não é uma probabilidade calibrada; serve apenas para ranquear a melhor janela.

    - 3 prévias dos chunks selecionados (≈200 caracteres cada), confirmando que o contexto alimentado ao QA era tematicamente adequado.

- **Comportamento da resposta:**

    - O texto retornado foi objetivo é citado, descrevendo como se dá o diagnóstico e mencionando a relevância do diagnóstico diferencial (sem

detalhes inventados fora do trecho).

    - Não houve tokens estranhos (p.ex., subpalavras não unidas), pois a decodificação do tokenizer recompôs as palavras corretamente.

- **Sensibilidade a parâmetros:**

    - Com TOP_K=3, a resposta já saiu plausível; aumentar TOP_K poderia ampliar recall, com custo extra.

    - A sobreposição de 150 chars evitou cortes de resposta entre janelas.

4) Limitações observadas (e como lidamos)

- **Score não calibrado:** o Score alto nem sempre implica exatidão semântica perfeita. Por isso, avaliamos visualmente com as prévias e o trecho de suporte.

- **Dependência do recorte:** se a pergunta mistura duas ideias (ex.: “como + por que”), a resposta pode não cobrir ambas em uma única janela. Para mitigar, podemos garantir que o chunk inclua toda a explicação relevante (ou aumentar TOP_K).

- **Variação do PDF:** elementos de layout/tabelas não textuais não entram; o pipeline é texto-centrado.

5) Conclusão parcial

5.1) Modelo 1 (PT-BR) no PDF

- O Modelo 1 gerou respostas curtas, mas direta e alinhada ao tema da pergunta, o que faz sentido por ser um modelo de pergunta/resposta ao invés de texto generativo.

- Utilizamos a ideia passada pelo professor sobre a utilização de embeddings + chunks e mantivemos a janela do QA configurado no padrão (384/128) que foi o que apresentou uma melhor resposta para as 3 perguntas.

5.2) Modelo 2 - Multilíngue

- Nesse modelo ele também consegue gerar uma resposta, quase idêntica ao primeiro, mas para a pergunta “Como a asma é classificada de acordo com a gravidade?” percebemos que ele retorna palavra a mais do que o primeiro. O que deve ser uma pequena diferença na hora gerar os tokens.

5.3) Modelo 3 - Inglês

- Mesmo com a utilização de embeddings, o modelo não conseguiu encontrar resposta para as perguntas, o que gerava um retorno de “<s>” até para a pergunta mais simples.
- Foi mantido os valores de max_length e stride para os 3 três modelos.

Documento de Dicionário de Dados

Já para o segundo documento, o dicionário de dados, a tentativa de solução foi baseada em três etapas principais:

1. **Extração de tabelas do documento Word (.docx)** – realizada com a biblioteca python-docx, que percorreu cada tabela do arquivo e converteu em DataFrames.

2. **Construção de um banco SQLite** – cada tabela extraída foi armazenada em uma instância SQLite local (dicionario.db).

3. **Conversão de perguntas em SQL com LLM** – empregamos o modelo google/flan-t5-base para transformar perguntas em linguagem natural em queries SQL, buscando automaticamente dentro do banco.

Solução passo a passo

1) Ingestão e carregamento do DOCX

- O código percorreu todas as tabelas contidas no arquivo de dados.

- Cada tabela foi convertida em DataFrame e inserida no banco como tabela_0, tabela_1 etc.

- Foi possível listar as tabelas e inspecionar amostras de linhas, confirmando que os dados estavam corretamente estruturados.

2) Tentativa de recuperação com SQL gerado por IA

- Implementamos uma função gerar_sql que recebe a pergunta em linguagem natural e, a partir do esquema das tabelas, cria um prompt para o modelo de linguagem (Flan-T5).

- O modelo então retorna uma instrução SQL que deveria ser executada sobre o banco SQLite.

- As perguntas selecionadas no classroom e testadas com no script foram:

    - Quais são os domínios do campo TP_SITUACAO da tabela NFCES109?
    - Qual a descrição do campo CNS_PROF da tabela NFCES110?
    - Quantas tabelas tem o campo UNIDADE_ID?

3) O que aconteceu na prática

- As queries SQL geradas pelo modelo não conseguiram localizar corretamente os dados.

- Muitas vezes o SQL retornado fazia referência a tabelas ou colunas inexistentes (provavelmente porque o modelo não conseguiu interpretar adequadamente os nomes das tabelas geradas no SQLite).

- Ao executar as queries, os erros mais comuns foram:

    - no such table (referência a tabela inexistente)

    - no such column (referência a campo inexistente)

Portanto, não obtivemos respostas satisfatórias às perguntas feitas, mesmo com as tabelas corretamente carregadas no banco.

Conclusão parcial

- O pipeline conseguiu extrair e estruturar o conteúdo do .docx em SQLite, confirmando que a ingestão estava funcional.

- Entretanto, a etapa de busca via perguntas em linguagem natural não foi satisfatória, pois o modelo não conseguiu produzir queries SQL válidas para responder às perguntas do prompt.