Assistente LLM de Política Portuguesa

# 1. Objetivo Principal

O objetivo principal deste projeto pessoal é explorar o desenvolvimento de um assistente baseado em Modelos de Linguagem Amplos (LLM - Large Language Models) com um foco específico na política portuguesa. A meta inicial é utilizar dados concretos das Eleições Legislativas de 2025 para criar uma ferramenta que ajude a compreender de forma mais clara e acessível as propostas e posições dos diferentes partidos políticos para essas eleições. Pretende-se dar um pequeno contributo para o esclarecimento dos cidadãos.

# 2. Funcionalidades Exploradas

Com o conjunto de dados disponível nesta fase inicial, focado nos materiais das eleições de 2025, as funcionalidades a serem exploradas e desenvolvidas serão centradas neste material. As ideias incluem:

* **Resumos dos Programas:** Gerar resumos concisos do conteúdo dos programas eleitorais dos partidos para 2025.
* **Informação por Tema:** Apresentar as propostas ou posições dos partidos em temas específicos, utilizando os documentos temáticos criados manualmente a partir dos programas.
* **Perfil Básico de Partidos:** Criar uma descrição sucinta de cada partido baseada estritamente na informação contida nos seus programas de 2025.
* ***Matching*:** Investigar a possibilidade de, com base em poucas características ou interesses indicados pelo utilizador, sugerir partidos cujas propostas nos programas de 2025 pareçam alinhar-se, **sublinhando sempre que é uma correspondência limitada e baseada apenas nos documentos de 2025.**

# 3. Público-Alvo

Este projeto destina-se principalmente a cidadãos portugueses que procuram uma forma simples de aceder à informação sobre os partidos e os seus programas para as Eleições de 2025. É também para qualquer pessoa interessada em ver uma aplicação prática de LLMs a um domínio específico como a política portuguesa, mesmo que de forma limitada.

# 4. Fontes de Dados

Para este projeto, o foco é a utilização de dados diretamente relacionados com as Eleições Legislativas de Maio de 2025.

* 1. **Fonte de Dados Principal:**

Os Programas Eleitorais completos dos partidos políticos Portugueses que concorreram às Eleições Legislativas de Maio de 2025. Estes documentos foram obtidos a partir dos sites oficiais de cada partido.

Documentos de resumo e comparação organizados por tema, criados manualmente e com ajuda de uma llm exterior para decompor de forma organizada a informaçãi – foi retificado) a partir da extração e análise do conteúdo dos programas eleitorais completos. (Tal como visível na pasta 'data' com ficheiros como programa\_eleitoral\_2025\_tema\_saude.pdf, programa\_eleitoral\_2025\_tema\_trabalho.pdf, etc.)

*Nota:* Links como o do Observador ([https://observador.pt/interativo/guia-eleitoral-interativo-as-propostas-dos-partidos-para-10-areas-centrais/#/partidos](https://www.google.com/url?sa=E&q=https%3A%2F%2Fobservador.pt%2Finterativo%2Fguia-eleitoral-interativo-as-propostas-dos-partidos-para-10-areas-centrais%2F%23%2Fpartidos)) foram úteis para identificar partidos e temas relevantes, mas a recolha dos documentos completos e a criação dos resumos temáticos foram trabalho manual baseado nos programas oficiais.

* 1. **Potenciais Fontes de Dados Futuras (para uma evolução do projeto):**

Resultados eleitorais históricos. Informação dos sites oficiais da CNE e Assembleia da República. Arquivo selecionado de notícias e análises políticas.

# 5. Pilha Tecnológica (a Explorar)

* Linguagem: Python
* Ferramentas para LLMs: Bibliotecas como transformers da Hugging Face, PyTorch ou TensorFlow. Exploração de técnicas como Embeddings e Bases de Dados Vetoriais para gerir e pesquisar a informação dos programas.
* Tratamento de Dados: Bibliotecas Python para leitura de PDFs e processamento de texto (e.g., PyPDF2, spaCy, NLTK).
* Gestão do Código: Git e GitHub para controlo de versões e organização do projeto.
* Interface (Ideia para o futuro): Uma pequena interface web (usando Flask ou similar) para interagir com o assistente.

# 6. Fases Iniciais do Projeto

1. **Configuração:** Estruturar o repositório no GitHub.
2. **Processamento de Dados:** Assegurar que o texto dos programas de 2025 e dos documentos temáticos está extraído e organizado de forma utilizável.
3. **Aprendizagem Prática:** Começar a usar bibliotecas LLM em Python para carregar dados e experimentar tarefas básicas (leitura, resumo) com base nos documentos.
4. **Implementação Mínima:** Desenvolver uma primeira versão muito simples de uma funcionalidade (ex: obter o resumo de um programa, ou a secção de um tema).
5. **Testes e Iteração:** Testar o que foi implementado e melhorar gradualmente, focando-se em como o LLM interage com os dados fornecidos.

# 7. Considerações Importantes: Âmbito e Limitações

É crucial ter presente que este projeto é um esboço inicial, de cariz pessoal e com um âmbito deliberadamente limitado nesta fase.

O assistente baseia-se quase exclusivamente nos programas eleitorais e nos resumos temáticos das Legislativas de Maio de 2025. Devido a esta limitação de dados, o assistente não oferece uma visão completa ou dinâmica da política portuguesa. Faltam elementos essenciais que estariam num sistema mais robusto, como: resultados históricos, atividade parlamentar atual, cobertura noticiosa, a evolução das posições dos partidos ao longo do tempo, ou contexto histórico detalhado. As capacidades serão diretamente condicionadas pela informação contida nos programas de 2025. Ele poderá falar sobre o que está nesses documentos, mas não sobre o que aconteceu *depois* ou o que não foi incluído nos programas.

Este projeto serve como um ponto de partida para aprender e aplicar conceitos de LLMs a um domínio específico. As limitações atuais definem o foco, mas também abrem portas para futuras expansões, caso se decida continuar a desenvolver.

Passo a passo das coisas que fiz:

1. Fui procurar documentos, tudo em pdf, serão os nossos dados como já referi anteriormente.
2. Transformei os em txt (processed data)
3. Depois process\_and\_summarize.py
   1. A sumarização deve ser diferente entre os documentos de programa eleitoral e os de tema, pois o primeiro contém os dados em bruto, e o segundo não dessa forma. Precisam de atenção diferente.
4. –
   1. **Resposta Curta:** **Não, não deve usar remoção de *stop words* ou outras técnicas de pré-processamento de texto clássicas (como stemming ou lematização) quando trabalha com LLMs modernos e modelos de embedding como o que estamos a usar.**
   2. Vamos perceber porquê.
   3. O Mundo "Antigo" vs. o Mundo dos LLMs
   4. Técnicas Clássicas (Pré-LLMs)
   5. No passado, quando os modelos de PLN eram mais simples (como TF-IDF, Word2Vec, etc.), eles não entendiam o **contexto** de uma frase. Eles viam o texto como um "saco de palavras" (bag of words). Nesse cenário, era crucial fazer um pré-processamento rigoroso do texto:
   6. ***Stop Words* (Palavras de Paragem):** Palavras como "o", "a", "de", "para", "é", "que" são extremamente comuns, mas (neste modelo antigo) não carregavam muito significado sozinhas. Removê-las ajudava o modelo a focar-se nas palavras-chave mais importantes (substantivos, verbos, adjetivos).
   7. **Lematização e Stemming:** Reduzir palavras à sua raiz. Por exemplo, "correr", "correu", "correndo" seriam todas transformadas em "corr". Isto ajudava a agrupar palavras com o mesmo significado base, já que o modelo não saberia que elas estavam relacionadas.
   8. **Remoção de Pontuação e Minúsculas:** Simplificar o texto ao máximo para reduzir a "complexidade" para o modelo.
   9. **O objetivo era:** Limpar o "ruído" e deixar apenas as palavras "importantes" para que algoritmos mais simples pudessem funcionar melhor.
   10. O Mundo dos LLMs e Modelos de Embedding (o que estamos a usar)
   11. Modelos como Sentence-Transformers e T5 são radicalmente diferentes. Eles foram treinados em biliões de frases da internet e aprenderam as complexas relações entre as palavras. **Para eles, o contexto é tudo.**
   12. ***Stop Words* são Essenciais para o Contexto:** A diferença entre "aumento **de** impostos" e "aumento **nos** impostos" pode ser subtil, mas existe. "Dar dinheiro **para** o SNS" é diferente de "Dar dinheiro **do** SNS". Os LLMs entendem estas nuances, e remover estas palavras **destrói o significado** da frase para o modelo.
   13. **Lematização e Stemming Destroem a Informação:** O modelo sabe a diferença entre "proponho" (futuro/intenção) e "propus" (passado). Reduzir ambas a "prop" faria com que o modelo perdesse esta informação temporal e gramatical crucial.
   14. **A Pontuação e a Capitalização Carregam Significado:** Uma pergunta termina com ?. Uma sigla como "NATO" ou "PCP" em maiúsculas é diferente da palavra "nato" ou "pcp". Os modelos modernos usam esta informação.
   15. **Conclusão:** Remover *stop words* ou aplicar outras técnicas de pré-processing clássico a textos que vão ser usados por modelos de embedding modernos é **contraproducente e prejudicial**. É como pegar numa fotografia a cores de alta resolução e convertê-la para preto e branco com baixa resolução antes de a mostrar a alguém. Você perde detalhe e contexto que são essenciais para a compreensão.
   16. Então, quando é que se faz pré-processamento agora?
   17. O "pré-processamento" no mundo dos LLMs é muito mais subtil e focado em **estrutura e limpeza**, não em alteração de palavras:
   18. **Limpeza de "Lixo" de OCR:** Remover cabeçalhos e rodapés repetidos, números de página, artefactos estranhos da conversão de PDF para TXT. (Você já fez isto ao extrair o texto, o que é ótimo).
   19. **Divisão Lógica (Chunking):** Dividir o texto em parágrafos ou secções com significado, em vez de cortar a meio de uma frase. A nossa abordagem no ask\_the\_assistant\_final.py de dividir por partido e tema já é uma forma inteligente de *chunking*.
   20. **Normalização de Espaços:** Remover múltiplos espaços ou linhas em branco para ter um texto mais "limpo", mas sem alterar o conteúdo.
   21. **Tratamento de Casos Específicos:** Se houvesse muitos erros de ortografia, poder-se-ia pensar em corrigi-los, mas geralmente os modelos são robustos a isso.
   22. **Em suma:** O seu colega está absolutamente correto no contexto de PLN clássico, e é ótimo que ele tenha levantado esse ponto. No entanto, para a arquitetura que estamos a construir (RAG com Sentence Transformers e T5), a regra de ouro é: **quanto mais original e contextualizado o texto, melhor**.
   23. **A resposta final é:** Não, não precisa de adicionar um passo de remoção de *stop words*. A forma como o script ask\_the\_assistant\_final.py já processa os documentos (dividindo por tema e partido) é o tipo de "pré-processamento" correto e mais eficaz para esta tecnologia. Já está a fazer da forma certa

Metodologia

O ponto de partida foi a recolha de dois tipos de documentos:

1. **Programas Eleitorais Completos:** Documentos longos em formato PDF, obtidos dos sites oficiais dos partidos.
2. **Documentos Temáticos:** Resumos por tema (ex: Defesa, Habitação), criados manualmente para organizar e comparar as propostas de todos os partidos.

O primeiro passo técnico consistiu em extrair o texto de todos os documentos PDF e guardá-los em formato de texto simples (.txt) na pasta *processed\_data*. Este passo foi crucial para tornar os dados legíveis para os nossos scripts em Python.

A solução foi desenvolvida em **Python**, utilizando um ecossistema de bibliotecas de código aberto da Hugging Face e da Meta AI. A pilha tecnológica implementada inclui:

* **Linguagem:** Python 3.11
* **Bibliotecas Principais:** transformers, sentence-transformers, torch, faiss-cpu.
* **Modelos de Linguagem (LLMs):**
  + **Sumarização:** csebuetnlp/mT5\_multilingual\_XLSum, um modelo multilingual otimizado para gerar resumos a partir de textos longos.
  + **Geração de Embeddings (para RAG):** distiluse-base-multilingual-cased-v1, um modelo eficiente para converter texto em vetores semânticos.
  + **Geração de Respostas (para RAG):** google/flan-t5-base, um modelo de texto-para-texto versátil e rápido, ideal para formular respostas com base num contexto.
* **Base de Dados Vetorial:** FAISS (Facebook AI Similarity Search), uma biblioteca de alta performance para pesquisa de similaridade em vetores.

Para gerar os resumos dos programas eleitorais completos, foi implementada uma técnica de **sumarização hierárquica (MapReduce)** no script process\_and\_summarize.py. Dado que os programas são demasiado longos para serem processados de uma só vez por um LLM, o processo foi o seguinte:

1. **Map (Mapeamento):** O documento completo é dividido em pedaços de texto mais pequenos e sobrepostos (*chunks*). Cada *chunk* é então sumarizado individualmente pelo modelo mT5.
2. **Reduce (Redução):** Todos os resumos intermédios gerados no passo anterior são concatenados num único documento. Este documento, muito mais curto que o original, é então sumarizado uma última vez para produzir o resumo final e coeso.
3. **Armazenamento:** Os resumos finais são guardados na pasta ./summaries/, criando uma base de dados de resumos pronta a ser utilizada.

O núcleo do assistente foi implementado no script ask\_the\_assistant\_final.py usando a arquitetura **Retrieval-Augmented Generation (RAG)**. Esta abordagem garante que o assistente responde com base nos factos contidos nos documentos, minimizando o risco de "alucinações" ou informação incorreta.

1. **Construção da Base de Conhecimento:**
   * O script lê os **documentos temáticos** e os **resumos dos programas** (gerados na fase anterior).
   * O conteúdo é dividido em unidades de informação lógicas (propostas por partido/tema, resumos por partido).
   * O modelo sentence-transformers converte cada uma destas unidades num vetor numérico (embedding) que capta o seu significado semântico.
   * Todos os vetores são armazenados num índice **FAISS**, que funciona como um "catálogo" de conhecimento altamente eficiente e pesquisável.
2. **Ciclo de Pergunta-Resposta:**
   * **Retrieval (Recuperação):** Quando o utilizador faz uma pergunta, a mesma é convertida num vetor. O FAISS é usado para encontrar os k pedaços de texto mais semanticamente relevantes na base de conhecimento.
   * **Augmentation (Aumento):** Os textos recuperados são adicionados como "contexto" a um *prompt* cuidadosamente construído, juntamente com a pergunta original do utilizador.
   * **Generation (Geração):** O *prompt* completo é enviado para o modelo flan-t5-base, que tem a tarefa de sintetizar a informação do contexto e formular uma resposta clara, concisa e fiel às fontes.

Durante o desenvolvimento, foi considerada a utilização de técnicas clássicas de PLN, como a remoção de *stop words* e *stemming*. No entanto, após análise, concluiu-se que estas técnicas são **prejudiciais** para o desempenho dos modelos de embedding modernos. Estes modelos dependem do contexto completo da frase, incluindo *stop words* e pontuação, para compreenderem o significado. Portanto, optou-se por um pré-processamento mínimo, focado apenas na limpeza estrutural dos dados e na sua divisão lógica (*chunking*), preservando a integridade do texto original.