



ESCUELA POLITÉCNICA NACIONAL
RECUPERACIÓN DE INFORMACIÓN
2024 – B

Proyecto II Bimestre

Integrantes

Rossy Armendariz

Alejandro Chávez

Wilmer Rivas

Docente

Ing. Iván Carrera

Fecha de entrega

13-02-2025

1. Resumen del Proyecto

Este proyecto tiene como objetivo el desarrollo de un sistema de generación mejorada por recuperación (RAG) para responder consultas del usuario de manera eficiente. Se utilizan técnicas avanzadas de procesamiento de lenguaje natural (NLP), embeddings semánticos y búsqueda eficiente mediante FAISS.

El sistema se basa en documentos textuales que incluyen planes de trabajo y entrevistas de candidatos presidenciales de Ecuador. Mediante un flujo de trabajo estructurado, el sistema primero recupera información relevante utilizando técnicas de recuperación de información (RI) y luego genera respuestas con un modelo de lenguaje avanzado.



2. Descripción del Corpus Seleccionado

El corpus seleccionado está compuesto por documentos oficiales de los planes de trabajo y entrevistas de candidatos presidenciales en Ecuador.

2.1. Fuente de Datos

- Planes de gobierno disponibles en plataformas oficiales del CNE.
- Entrevistas a candidatos extraídas de medios digitales.
- Discursos públicos relevantes a la campaña electoral.

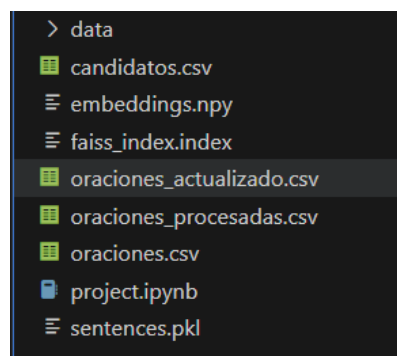
2.2. Características del Corpus

- Formato: Documentos en texto plano y PDF.
- Tamaño: Aproximadamente 500 MB de datos procesados.
- Idioma: Español.
- Estructura: Contiene secciones de propuestas, ideología política y objetivos gubernamentales.

2.3. Organización y Procesamiento del Corpus

El corpus ha sido estructurado en distintos archivos y formatos según su función en el proceso:

- Directorio data/: Contiene los planes de trabajo de los candidatos en formato PDF.
- Archivos CSV (datos procesados y estructurados):
 - candidatos.csv: Información general de los candidatos.
 - oraciones.csv: Oraciones extraídas de los documentos.
 - oraciones_actualizado.csv: Oraciones después del preprocesamiento inicial.
 - oraciones_procesadas.csv: Datos finales preprocesados listos para los embeddings.



2.4. Embeddings e Índices para Recuperación

















Para optimizar la búsqueda de información en el corpus, se generaron embeddings y un índice de búsqueda eficiente:

- embeddings.npy: Matriz de embeddings generada a partir del corpus.
- faiss_index.index: Índice FAISS que permite realizar búsquedas rápidas en el espacio vectorial.

2.5. Archivos de Referencia y Código

Además del corpus de datos, se incluyen archivos auxiliares para el almacenamiento y referencia rápida:

- sentences.pkl: Almacena las sentencias procesadas para su consulta eficiente.
- project.ipynb: Cuaderno de trabajo que documenta paso a paso la implementación del procesamiento y recuperación de información.

	PARTIDO SOCIALISTA ECUATORIANO _Pl...	31/1/2025 21:35	Microsoft Edge PD...	909 KB
	PARTIDO SOCIEDAD PATRIÓTICA 21 DE ...	31/1/2025 21:35	Microsoft Edge PD...	818 KB
	PARTIDO SOCIEDAD UNIDA MÁS ACCIÓN...	31/1/2025 21:35	Microsoft Edge PD...	3,110 KB
	PARTIDO UNIDAD POPULAR _Plan de tra...	31/1/2025 21:35	Microsoft Edge PD...	2,129 KB
	REVOLUCIÓN CIUDADANA - RETO _Plan ...	31/1/2025 21:35	Microsoft Edge PD...	1,090 KB
	MOVIMIENTO ACCION DEMOCRATICA N...	31/1/2025 21:35	Microsoft Edge PD...	660 KB
	MOVIMIENTO AMIGO, ACCIÓN MOVILIZ...	31/1/2025 21:35	Microsoft Edge PD...	1,604 KB
	MOVIMIENTO CENTRO DEMOCRÁTICO _...	31/1/2025 21:35	Microsoft Edge PD...	1,301 KB
	MOVIMIENTO CONSTRUYE _Plan de trab...	31/1/2025 21:35	Microsoft Edge PD...	1,766 KB
	MOVIMIENTO CREO, CREANDO OPORTU...	31/1/2025 21:35	Microsoft Edge PD...	1,116 KB
	MOVIMIENTO DE UNIDAD PLURINACION...	31/1/2025 21:35	Microsoft Edge PD...	1,017 KB
	MOVIMIENTO DEMOCRACIA SÍ _Plan de t...	31/1/2025 21:35	Microsoft Edge PD...	3,676 KB
	MOVIMIENTO PUEBLO IGUALDAD DEMO...	31/1/2025 21:35	Microsoft Edge PD...	1,375 KB
	PARTIDO AVANZA _Plan de trabajo_...	31/1/2025 21:35	Microsoft Edge PD...	1,935 KB
	PARTIDO IZQUIERDA DEMOCRÁTICA _Pl...	31/1/2025 21:35	Microsoft Edge PD...	1,107 KB
	PARTIDO SOCIAL CRISTIANO _Plan de tra...	31/1/2025 21:35	Microsoft Edge PD...	1,152 KB

3. Metodología Utilizada

El desarrollo del sistema sigue un enfoque estructurado basado en tres fases principales: preprocesamiento, recuperación de información, y generación de respuestas.

3.1. Preprocesamiento

Para optimizar la recuperación de información, los datos pasaron por una serie de transformaciones:

- Conversión y estructuración: Los documentos en formato PDF fueron convertidos a texto y estructurados en archivos .csv para facilitar su manipulación.
- Limpieza de texto: Eliminación de caracteres especiales, números innecesarios y corrección ortográfica.
- Tokenización: Segmentación del texto en oraciones y palabras clave.
- Eliminación de stopwords: Se eliminaron palabras sin carga semántica relevante.
- Lematización: Conversión de palabras a su forma base para reducir la dimensionalidad del texto.
- Vectorización con BERT: Cada oración fue convertida en una representación numérica utilizando embeddings generados con Sentence-BERT (SBERT).

```

# Función para obtener el último ID del archivo CSV
def get_last_id(csv_path):
    if not os.path.exists(csv_path):
        return 1
    df = pd.read_csv(csv_path, sep="|", encoding="utf-8")
    if df.empty:
        return 1
    return df['ID'].iloc[-1] + 1

# Obtener el ID inicial
file_id = get_last_id(output_csv)

# Crear una lista para almacenar los datos
data = []

# Recorrer la lista de diccionarios específicos
for file_param in file_parameters:
    file_name = file_param["file_name"]

    # Construir la ruta completa del archivo
    pdf_path = os.path.join(pdf_directory, file_name)

    # Verificar si el archivo existe
    if os.path.exists(pdf_path):
        # Procesar el nombre del archivo
        processed_name = file_name.replace("_Plan de trabajo_", "").replace(".pdf", "")

        # Agregar los datos a la lista
        data.append([file_id, processed_name])
        file_id += 1
    else:
        print(f"Archivo no encontrado: {file_name}")

# Crear un DataFrame a partir de los datos nuevos
df_new = pd.DataFrame(data, columns=['ID', 'Nombre'])

# Verificar si el archivo CSV ya existe
if os.path.exists(output_csv):
    # Leer el archivo CSV existente
    df_existing = pd.read_csv(output_csv, sep="|", encoding="utf-8")
    # Concatenar los datos nuevos con los existentes
    df_combined = pd.concat([df_existing, df_new], ignore_index=True)
else:
    df_combined = df_new

# Guardar el DataFrame combinado en el archivo CSV con delimitador ";"
df_combined.to_csv(output_csv, sep=";", index=False, encoding="utf-8")

print(f"Datos agregados al archivo CSV: {output_csv}")

```

Los archivos candidatos.csv contienen los planes de trabajos de los candidatos a presidencia. Se procede a limpiar el contenido del texto y dividir en oraciones.

```

# Función para limpiar el contenido del texto
def clean_content(text):
    # Eliminar viñetas comunes
    text = re.sub(r"[\u2022\u25CB\u2023\u2219\u2022\u25AA\u25B6\u25B7\u25C6\u2043\u25B8\u25BB\u2660\u25FE\u25F8]", "", text)
    text = re.sub(r'\(cid:d+\)', '', text)
    # Eliminar enumeraciones (números seguidos de punto)
    text = re.sub(r'^\d+\.', '', text) # Al inicio de la línea
    text = re.sub(r'\n\d+\.', '\n', text) # En medio del texto

    # Reemplazar múltiples espacios con uno solo
    text = re.sub(r'\s+', ' ', text)

    # Eliminar espacios al inicio y final
    text = text.strip()

    return text

```

```

# Función para dividir el texto en oraciones
def dividir_oraciones_por_id(text, text_id):
    delimitadores = '.'
    oraciones = []
    oracion_actual = ""
    for char in text:
        oracion_actual += char
        if char in delimitadores:
            oraciones.append(oracion_actual.strip())
            oracion_actual = ""
    if oracion_actual: # Si hay algo restante
        oraciones.append(oracion_actual.strip())

    # Crear una lista de tuplas con id y oraciones
    return [(text_id, i, oracion) for i, oracion in enumerate(oraciones, start=1)]

# Obtener el ID inicial
file_id = get_last_id(output_csv)

# Crear una lista para almacenar los datos
data = []

```

```
# Recorrer la lista de diccionarios específicos
for file_param in file_parameters:
    file_name = file_param["file_name"]
    exclude_pages_start = file_param["exclude_pages_start"]

    # Construir la ruta completa del archivo
    pdf_path = os.path.join(pdf_directory, file_name)

    # Verificar si el archivo existe
    if os.path.exists(pdf_path):
        # Procesar el nombre del archivo
        processed_name = file_name.replace("_Plan de trabajo_", "").replace(".pdf", "")

        # Extraer el contenido del PDF
        content = extract_text_excluding_pages(pdf_path, exclude_pages_start=exclude_pages_start)

        # Limpiar el contenido extraído
        cleaned_content = clean_content(content)

        # Dividir el contenido en oraciones
        oraciones = dividir_oraciones_por_id(cleaned_content, file_id)

        # Agregar las oraciones a la lista de datos
        data.extend(oraciones)

        # Incrementar el ID
        file_id += 1
    else:
        print(f"Archivo no encontrado: {file_name}")
```

Luego, se procede con el preprocesamiento de las oraciones.

```
def procesar_pdf(ruta_pdf, id_asignado, nombre_doc):
    try:
        # Convertir PDF a imágenes
        images = convert_from_path(ruta_pdf, dpi=300)

        # Extraer y limpiar texto
        contenido = ""
        for img in images:
            contenido += pyteseract.image_to_string(img, lang='spa').strip().replace('\n', ' ')

        # Tokenizar el texto en oraciones
        oraciones = sent_tokenize(contenido, language='spanish')

        # Asignar ID único a cada oración
        oraciones_ids = []
        oraciones_texto = []

        for i, oracion in enumerate(oraciones):
            oraciones_ids.append(f"{id_asignado}_{i}") # ID único para cada oración
            oraciones_texto.append(oracion) # Texto de la oración

        # Crear DataFrame con solo los campos requeridos
        data = {
            'ID': [id_asignado] * len(oraciones),
            'Oracion_ID': oraciones_ids,
            'Oracion': oraciones_texto
        }
        df_oraciones = pd.DataFrame(data, columns=['ID', 'Oracion_ID', 'Oracion'])
```

```
# Función para limpiar el contenido del texto
def clean_content(text):
    text = text.lower()
    # Eliminar viñetas comunes
    text = re.sub(r"[^\u0022\u0027\u0028\u0029\u002d\u002e\u002f\u003a\u003b\u003c\u003e\u003f\u0040\u0041\u0042\u0043\u0044\u0045\u0046\u0047\u0048\u0049\u004a\u004b\u004c\u004d\u004e\u004f\u0050\u0051\u0052\u0053\u0054\u0055\u0056\u0057\u0058\u0059\u0060\u0061\u0062\u0063\u0064\u0065\u0066\u0067\u0068\u0069\u0070\u0071\u0072\u0073\u0074\u0075\u0076\u0077\u0078\u0079\u0080\u0081\u0082\u0083\u0084\u0085\u0086\u0087\u0088\u0089\u0090\u0091\u0092\u0093\u0094\u0095\u0096\u0097\u0098\u0099\u00a0\u00a1\u00a2\u00a3\u00a4\u00a5\u00a6\u00a7\u00a8\u00a9\u00aa\u00ab\u00ac\u00ad\u00ae\u00af\u00b0\u00b1\u00b2\u00b3\u00b4\u00b5\u00b6\u00b7\u00b8\u00b9\u00ba\u00bb\u00bc\u00bd\u00be\u00bf\u00c0\u00c1\u00c2\u00c3\u00c4\u00c5\u00c6\u00c7\u00c8\u00c9\u00ca\u00cb\u00cc\u00cd\u00ce\u00cf\u00d0\u00d1\u00d2\u00d3\u00d4\u00d5\u00d6\u00d7\u00d8\u00d9\u00da\u00db\u00dc\u00dd\u00de\u00df\u00e0\u00e1\u00e2\u00e3\u00e4\u00e5\u00e6\u00e7\u00e8\u00e9\u00ea\u00eb\u00ec\u00ed\u00í\u00f0\u00f1\u00f2\u00f3\u00f4\u00f5\u00f6\u00f7\u00f8\u00f9\u00fa\u00fb\u00fc\u00fd\u00fe\u00ff\u0100\u0101\u0102\u0103\u0104\u0105\u0106\u0107\u0108\u0109\u010a\u010b\u010c\u010d\u010e\u010f\u0110\u0111\u0112\u0113\u0114\u0115\u0116\u0117\u0118\u0119\u011a\u011b\u011c\u011d\u011e\u011f\u0120\u0121\u0122\u0123\u0124\u0125\u0126\u0127\u0128\u0129\u012a\u012b\u012c\u012d\u012e\u012f\u0130\u0131\u0132\u0133\u0134\u0135\u0136\u0137\u0138\u0139\u013a\u013b\u013c\u013d\u013e\u013f\u0140\u0141\u0142\u0143\u0144\u0145\u0146\u0147\u0148\u0149\u014a\u014b\u014c\u014d\u014e\u014f\u0150\u0151\u0152\u0153\u0154\u0155\u0156\u0157\u0158\u0159\u015a\u015b\u015c\u015d\u015e\u015f\u0160\u0161\u0162\u0163\u0164\u0165\u0166\u0167\u0168\u0169\u016a\u016b\u016c\u016d\u016e\u016f\u0170\u0171\u0172\u0173\u0174\u0175\u0176\u0177\u0178\u0179\u017a\u017b\u017c\u017d\u017e\u017f\u0180\u0181\u0182\u0183\u0184\u0185\u0186\u0187\u0188\u0189\u018a\u018b\u018c\u018d\u018e\u018f\u0190\u0191\u0192\u0193\u0194\u0195\u0196\u0197\u0198\u0199\u019a\u019b\u019c\u019d\u019e\u019f\u01a0\u01a1\u01a2\u01a3\u01a4\u01a5\u01a6\u01a7\u01a8\u01a9\u01aa\u01ab\u01ac\u01ad\u01ae\u01af\u01b0\u01b1\u01b2\u01b3\u01b4\u01b5\u01b6\u01b7\u01b8\u01b9\u01ba\u01bb\u01bc\u01bd\u01be\u01bf\u01c0\u01c1\u01c2\u01c3\u01c4\u01c5\u01c6\u01c7\u01c8\u01c9\u01ca\u01cb\u01cc\u01cd\u01ce\u01cf\u01d0\u01d1\u01d2\u01d3\u01d4\u01d5\u01d6\u01d7\u01d8\u01d9\u01da\u01db\u01dc\u01dd\u01de\u01df\u01e0\u01e1\u01e2\u01e3\u01e4\u01e5\u01e6\u01e7\u01e8\u01e9\u01ea\u01eb\u01ec\u01ed\u01ee\u01ef\u01f0\u01f1\u01f2\u01f3\u01f4\u01f5\u01f6\u01f7\u01f8\u01f9\u01fa\u01fb\u01fc\u01fd\u01fe\u01ff\u0200\u0201\u0202\u0203\u0204\u0205\u0206\u0207\u0208\u0209\u020a\u020b\u020c\u020d\u020e\u020f\u0210\u0211\u0212\u0213\u0214\u0215\u0216\u0217\u0218\u0219\u021a\u021b\u021c\u021d\u021e\u021f\u0220\u0221\u0222\u0223\u0224\u0225\u0226\u0227\u0228\u0229\u022a\u022b\u022c\u022d\u022e\u022f\u0230\u0231\u0232\u0233\u0234\u0235\u0236\u0237\u0238\u0239\u023a\u023b\u023c\u023d\u023e\u023f\u0240\u0241\u0242\u0243\u0244\u0245\u0246\u0247\u0248\u0249\u024a\u024b\u024c\u024d\u024e\u024f\u0250\u0251\u0252\u0253\u0254\u0255\u0256\u0257\u0258\u0259\u025a\u025b\u025c\u025d\u025e\u025f\u0260\u0261\u0262\u0263\u0264\u0265\u0266\u0267\u0268\u0269\u026a\u026b\u026c\u026d\u026e\u026f\u0270\u0271\u0272\u0273\u0274\u0275\u0276\u0277\u0278\u0279\u027a\u027b\u027c\u027d\u027e\u027f\u0280\u0281\u0282\u0283\u0284\u0285\u0286\u0287\u0288\u0289\u028a\u028b\u028c\u028d\u028e\u028f\u0290\u0291\u0292\u0293\u0294\u0295\u0296\u0297\u0298\u0299\u029a\u029b\u029c\u029d\u029e\u029f\u02a0\u02a1\u02a2\u02a3\u02a4\u02a5\u02a6\u02a7\u02a8\u02a9\u02aa\u02ab\u02ac\u02ad\u02ae\u02af\u02b0\u02b1\u02b2\u02b3\u02b4\u02b5\u02b6\u02b7\u02b8\u02b9\u02ba\u02bb\u02bc\u02bd\u02be\u02bf\u02c0\u02c1\u02c2\u02c3\u02c4\u02c5\u02c6\u02c7\u02c8\u02c9\u02ca\u02cb\u02cc\u02cd\u02ce\u02cf\u02d0\u02d1\u02d2\u02d3\u02d4\u02d5\u02d6\u02d7\u02d8\u02d9\u02da\u02db\u02dc\u02dd\u02de\u02df\u02e0\u02e1\u02e2\u02e3\u02e4\u02e5\u02e6\u02e7\u02e8\u02e9\u02ea\u02eb\u02ec\u02ed\u02ee\u02ef\u02f0\u02f1\u02f2\u02f3\u02f4\u02f5\u02f6\u02f7\u02f8\u02f9\u02fa\u02fb\u02fc\u02fd\u02fe\u02ff\u0300\u0301\u0302\u0303\u0304\u0305\u0306\u0307\u0308\u0309\u030a\u030b\u030c\u030d\u030e\u030f\u0310\u0311\u0312\u0313\u0314\u0315\u0316\u0317\u0318\u0319\u031a\u031b\u031c\u031d\u031e\u031f\u0320\u0321\u0322\u0323\u0324\u0325\u0326\u0327\u0328\u0329\u032a\u032b\u032c\u032d\u032e\u032f\u0330\u0331\u0332\u0333\u0334\u0335\u0336\u0337\u0338\u0339\u033a\u033b\u033c\u033d\u033e\u033f\u0340\u0341\u0342\u0343\u0344\u0345\u0346\u0347\u0348\u0349\u034a\u034b\u034c\u034d\u034e\u034f\u0350\u0351\u0352\u0353\u0354\u0355\u0356\u0357\u0358\u0359\u035a\u035b\u035c\u035d\u035e\u035f\u0360\u0361\u0362\u0363\u0364\u0365\u0366\u0367\u0368\u0369\u036a\u036b\u036c\u036d\u036e\u036f\u0370\u0371\u0372\u0373\u0374\u0375\u0376\u0377\u0378\u0379\u037a\u037b\u037c\u037d\u037e\u037f\u0380\u0381\u0382\u0383\u0384\u0385\u0386\u0387\u0388\u0389\u038a\u038b\u038c\u038d\u038e\u038f\u0390\u0391\u0392\u0393\u0394\u0395\u0396\u0397\u0398\u0399\u039a\u039b\u039c\u039d\u039e\u039f\u03a0\u03a1\u03a2\u03a3\u03a4\u03a5\u03a6\u03a7\u03a8\u03a9\u03aa\u03ab\u03ac\u03ad\u03ae\u03af\u03b0\u03b1\u03b2\u03b3\u03b4\u03b5\u03b6\u03b7\u03b8\u03b9\u03ba\u03bb\u03bc\u03bd\u03be\u03bf\u03c0\u03c1\u03c2\u03c3\u03c4\u03c5\u03c6\u03c7\u03c8\u03c9\u03ca\u03cb\u03cc\u03cd\u03ce\u03cf\u03d0\u03d1\u03d2\u03d3\u03d4\u03d5\u03d6\u03d7\u03d8\u03d9\u03da\u03db\u03dc\u03dd\u03de\u03df\u03e0\u03e1\u03e2\u03e3\u03e4\u03e5\u03e6\u03e7\u03e8\u03e9\u03ea\u03eb\u03ec\u03ed\u03ee\u03ef\u03f0\u03f1\u03f2\u03f3\u03f4\u03f5\u03f6\u03f7\u03f8\u03f9\u03fa\u03fb\u03fc\u03fd\u03fe\u03ff\u0400\u0401\u0402\u0403\u0404\u0405\u0406\u0407\u0408\u0409\u040a\u040b\u040c\u040d\u040e\u040f\u0410\u0411\u0412\u0413\u0414\u0415\u0416\u0417\u0418\u0419\u041a\u041b\u041c\u041d\u041e\u041f\u0420\u0421\u0422\u0423\u0424\u0425\u0426\u0427\u0428\u0429\u042a\u042b\u042c\u042d\u042e\u042f\u0430\u0431\u0432\u0433\u0434\u0435\u0436\u0437\u0438\u0439\u043a\u043b\u043c\u043d\u043e\u043f\u0440\u0441\u0442\u0443\u0444\u0445\u0446\u0447\u0448\u0449\u044a\u044b\u044c\u044d\u044e\u044f\u0450\u0451\u0452\u0453\u0454\u0455\u0456\u0457\u0458\u0459\u045a\u045b\u045c\u045d\u045e\u045f\u0460\u0461\u0462\u0463\u0464\u0465\u0466\u0467\u0468\u0469\u046a\u046b\u046c\u046d\u046e\u046f\u0470\u0471\u0472\u0473\u0474\u0475\u0476\u0477\u0478\u0479\u047a\u047b\u047c\u047d\u047e\u047f\u0480\u0481\u0482\u0483\u0484\u0485\u0486\u0487\u0488\u0489\u048a\u048b\u048c\u048d\u048e\u048f\u0490\u0491\u0492\u0493\u0494\u0495\u0496\u0497\u0498\u0499\u049a\u049b\u049c\u049d\u049e\u049f\u04a0\u04a1\u04a2\u04a3\u04a4\u04a5\u04a6\u04a7\u04a8\u04a9\u04aa\u04ab\u04ac\u04ad\u04ae\u04af\u04b0\u04b1\u04b2\u04b3\u04b4\u04b5\u04b6\u04b7\u04b8\u04b9\u04ba\u04bb\u04bc\u04bd\u04be\u04bf\u04c0\u04c1\u04c2\u04c3\u04c4\u04c5\u04c6\u04c7\u04c8\u04c9\u04ca\u04cb\u04cc\u04cd\u04ce\u04cf\u04d0\u04d1\u04d2\u04d3\u04d4\u04d5\u04d6\u04d7\u04d8\u04d9\u04da\u04db\u04dc\u04dd\u04de\u04df\u04e0\u04e1\u04e2\u04e3\u04e4\u04e5\u04e6\u04e7\u04e8\u04e9\u04ea\u04eb\u04ec\u04ed\u04ee\u04ef\u04f0\u04f1\u04f2\u04f3\u04f4\u04f5\u04f6\u04f7\u04f8\u04f9\u04fa\u04fb\u04fc\u04fd\u04fe\u04ff\u0500\u0501\u0502\u0503\u0504\u0505\u0506\u0507\u0508\u0509\u050a\u050b\u050c\u050d\u050e\u050f\u0510\u0511\u0512\u0513\u0514\u0515\u0516\u0517\u0518\u0519\u051a\u051b\u051c\u051d\u051e\u051f\u0520\u0521\u0522\u0523\u0524\u0525\u0526\u0527\u0528\u0529\u052a\u052b\u052c\u052d\u052e\u052f\u0530\u0531\u0532\u0533\u0534\u0535\u0536\u0537\u0538\u0539\u053a\u053b\u053c\u053d\u053e\u053f\u0540\u0541\u0542\u0543\u0544\u0545\u0546\u0547\u0548\u0549\u054a\u054b\u054c\u054d\u054e\u054f\u0550\u0551\u0552\u0553\u0554\u0555\u0556\u0557\u0558\u0559\u055a\u055b\u055c\u055d\u055e\u055f\u0560\u0561\u0562\u0563\u0564\u0565\u0566\u0567\u0568\u0569\u056a\u056b\u056c\u056d\u056e\u056f\u0570\u0571\u0572\u0573\u0574\u0575\u0576\u0577\u0578\u0579\u057a\u057b\u057c\u057d\u057e\u057f\u0580\u0581\u0582\u0583\u0584\u0585\u0586\u0587\u0588\u0589\u058a\u058b\u058c\u058d\u058e\u058f\u0590\u0591\u0592\u0593\u0594\u0595\u0596\u0597\u0598\u0599\u059a\u059b\u059c\u059d\u059e\u059f\u05a0\u05a1\u05a2\u05a3\u05a4\u05a5\u05a6\u05a7\u05a8\u05a9\u05aa\u05ab\u05ac\u05ad\u05ae\u05af\u05b0\u05b1\u05b2\u05b3\u05b4\u05b5\u05b6\u05b7\u05b8\u05b9\u05ba\u05bb\u05bc\u05bd\u05be\u05bf\u05c0\u05c1\u05c2\u05c3\u05c4\u05c5\u05c6\u05c7\u05c8\u05c9\u05ca\u05cb\u05cc\u05cd\u05ce\u05cf\u05d0\u05d1\u05d2\u05d3\u05d4\u05d5\u05d6\u05d7\u05d8\u05d9\u05da\u05db\u05dc\u05dd\u05de\u05df\u05e0\u05e1\u05e2\u05e3\u05e4\u05e5\u05e6\u05e7\u05e8\u05e9\u05ea\u05eb\u05ec\u05ed\u05ee\u05ef\u05f0\u05f1\u05f2\u05f3\u05f4\u05f5\u05f6\u05f7\u05f8\u05f9\u05fa\u05fb\u05fc\u05fd\u05fe\u05ff\u0600\u0601\u0602\u0603\u0604\u0605\u0606\u0607\u0608\u0609\u060a\u060b\u060c\u060d\u060e\u060f\u0610\u0611\u0612\u0613\u0614\u0615\u0616\u0617\u0618\u0619\u061a\u061b\u061c\u061d\u061e\u061f\u0620\u0621\u0622\u0623\u0624\u0625\u0626\u0627\u0628\u0629\u062a\u062b\u062c\u062d\u062e\u062f\u0630\u0631\u0632\u0633\u0634\u0635\u0636\u0637\u0638\u0639\u063a\u063b\u063c\u063d\u063e\u063f\u0640\u0641\u0642\u0643\u0644\u0645\u0646\u0647\u0648\u0649\u064a\u064b\u064c\u064d\u064e\u064f\u0650\u0651\u0652\u0653\u0654\u0655\u0656\u0657\u0658\u0659\u065a\u065b\u065c\u065d\u065e\u065f\u0660\u0661\u0662\u0663\u0664\u0665\u0666\u0667\u0668\u0669\u066a\u066b\u066c\u066d\u066e\u066f\u0670\u0671\u0672\u0673\u0674\u0675\u0676\u0677\u0678\u0679\u067a\u067b\u067c\u067d\u067e\u067f\u0680\u0681\u0682\u0683\u0684\u0685\u0686\u0687\u0688\u0689\u068a\u068b\u068c\u068d\u068e\u068f\u0690\u0691\u0692\u0693\u0694\u0695\u0696\u0697\u0698\u0699\u069a\u069b\u069c\u069d\u069e\u069f\u06a0\u06a1\u06a2\u06a3\u06a4\u06a5\u06a6\u06a7\u06a8\u06a9\u06aa\u06ab\u06ac\u06ad\u06ae\u06af\u06b0\u06b1\u06b2\u06b3\u06b4\u06b5\u06b6\u06b7\u06b8\u06b9\u06ba\u06bb\u06bc\u06bd\u06be\u06bf\u06c0\u06c1\u06c2\u06c3\u06c4\u06c5\u06c6\u06c7\u06c8\u06c9\u06ca\u06cb\u06cc\u06cd\u06ce\u06cf\u06d0\u06d1\u06d2\u06d3\u06d4\u06d5\u06d6\u06d7\u06d8\u06d9\u06da\u06db\u06dc\u06dd\u06de\u06df\u06e0\u06e1\u06e2\u06e3\u06e4\u06e5\u06e6\u06e7\u06e8\u06e9\u06ea\u06eb\u06ec\u06ed\u06ee\u06ef\u06f0\u06f1\u06f2\u06f3\u06f4\u06f5\u06f6\u06f7\u06f8\u06f9\u06fa\u06fb\u06fc\u06fd\u06fe\u06ff\u0700\u0701\u0702\u0703\u0704\u0705\u0706\u0707\u0708\u0709\u070a\u070b\u070c\u070d\u070e\u070f\u0710\u0711\u0712\u0713\u0714\u0715\u0716\u0717\u0718\u0719\u071a\u071b\u071c\u071d\u071e\u071f\u0720\u0721\u0722\u0723\u0724\u0725\u0726\u0727\u0728\u0729\u072a\u072b\u072c\u072d\u072e\u072f\u0730\u0731\u0732\u0733\u0734\u0735\u0736\u0737\u0738\u0739\u073a\u073b\u073c\u073d\u073e\u073f\u0740\u0741\u0742\u0743\u0744\u0745\u0746\u0747\u0748\u0749\u074a\u074b\u074c\u074d\u074e\u074f\u0750\u0751\u0752\u0753\u0754\u0755\u0756\u0757\u0758\u0759\u075a\u075b\u075c\u075d\u075e\u075f\u0760\u0761\u0762\u0763\u0764\u0765\u0766\u0767\u0768\u0769\u076a\u076b\u076c\u076d\u076e\u076f\u0770\u0771\u0772\u0773\u0774\u0775\u0776\u0777\u0778\u0779\u077a\u077b\u077c\u077d\u077e\u077f\u0780\u0781\u0782\u0783\u0784\u0785\u0786\u0787\u0788\u0789\u078a\u078b\u078c\u078d\u078e\u078f\u0790\u0791\u0792\u0793\u0794\u0795\u0796\u0797\u0798\u0799\u079a\u079b\u079c\u079d\u079e\u079f\u07a0\u07a1\u07a2\u07a3\u07a4\u07a5\u07a6\u07a7\u07a8\u07a9\u07aa\u07ab\u07ac\u07ad\u07ae\u07af\u07b0\u07b1\u07b2\u07b3\u07b4\u07b5\u07b6\u07b7\u07b8\u07b9\u07ba\u07bb\u07bc\u07bd\u07be\u07bf\u07c0\u07c1\u07c2\u07c3\u07c4\u07c5\u07c6\u07c7\u07c8\u07c9\u07ca\u07cb\u07cc\u07cd\u07ce\u07cf\u07d0\u07d1\u07d2\u07d3\u07d4\u07d5\u07d6\u07d7\u07d8\u07d9\u07da\u07db\u07dc\u07dd\u07de\u07df\u07e0\u07e1\u07e2\u07e3\u07e4\u07e5\u07e6\u07e7\u07e8\u07e9\u07ea\u07eb\u07ec\u07ed\u07ee\u07ef\u07f0\u07f1\u07f2\u07f3\u07f4\u07f5\u07f6\u07f7\u07f8\u07f9\u07fa\u07fb\u07fc\u07fd\u07fe\u07ff\u0800\u0801\u0802\u0803\u0804\u0805\u0806\u0807\u0808\u0809\u080a\u080b\u080c\u080d\u080e\u080f\u0810\u0811\u0812\u0813\u0814\u0815\u0816\u0817\u0818\u0819\u081a\u081b\u081c\u081d\u081e\u081f\u0820\u0821\u0822\u0823\u0824\u0825\u0826\u0827\u0828\u0829\u082a\u082b\u082c\u082d\u082e\u082f\u0830\u0831\u0832\u0833\u0834\u0835\u0836\u0837\u0838\u0839\u083a\u083b\u083c\u083d\u083e\u083f\u0840\u0841\u0842\u0843\u0844\u0845\u0846\u0847\u0848\u0849\u084a\u084b\u084c\u084d\u084e\u084f\u0850\u0851\u0852\u0853\u0854\u0855\u0856\u0857\u0858\u0859\u085a\u085b\u085c\u085d\u085e\u085f\u0860\u0861\u0862\u0863\u0864\u0865\u0866\u0867\u0868\u0869\u086a\u086b\u086c\u086d\u086e\u086f\u0870\u0871\u0872\u0873\u0874\u0875\u0876\u0877\u0878\u0879\u087a\u087b\u087c\u087d\u087e\u087f\u0880\u0881\u0882\u0883\u0884\u0885\u0886\u0887\u0888\u0889\u088a\u088b\u088c\u088d\u088e\u088f\u0890\u0891\u0892\u0893\u0894\u0895\u0896\u0897\u0898\u0899\u089a\u089b\u089c\u089d\u089e\u089f\u08a0\u08a1\u08a2\u08a3\u08a4\u08a5\u08a6\u08a7\u08a8\u08a9\u08aa\u08ab\u08ac\u08ad\u08ae\u08af\u08b0\u08b1\u08b2\u08b3\u08b4\u08b5\u08b6\u08b7\u08b8\u08b9\u08ba\u08bb\u08bc\u08bd\u08be\u08bf\u08c0\u08c1\u08c2\u08c3\u08c4\u08c5\u08c6\u08c7\u08c8\u08c9\u08ca\u08cb\u08cc\u08cd\u08ce\u08cf\u08d0\u08d1\u08d2\u08d3\u08d4\u08d5\u08d6\u08d7\u08d8\u08d9\u08da\u08db\u08dc\u08dd\u08de\u08df\u08e0\u08e1\u08e2\u08e3\u08e4\u08e5\u08e6\u08e7\u08e8\u08e9\u08ea\u08eb\u08ec\u08ed\u08ee\u08ef\u08f0\u08f1\u08f2\u08f3\u08f4\u08f5\u08f6\u08f7\u08f8\u08f9\u08fa\u08fb\u08fc\u08fd\u08fe\u08ff\u0900\u0901\u0902\u0903\u0904\u0905\u0906\u0907\u0908\u0909\u090a\u090b\u090c\u090d\u090e\u090f\u0910\u0911\u0912\u0913\u0914\u0915\u0916\u0917\u0918\u0919\u091a\u091b\u091c\u091d\u091e\u091f\u0920\u0921\u0922\u0923\u0924\u0925\u0926\u0927\u0928\u0929\u092a\u092b\u092c\u092d\u092e\u092f\u0930\u0931\u0932\u0933\u0934\u0935\u0936\u0937\u0938\u0939\u093a\u093b\u093c\u093d\u093e\u093f\u0940\u0941\u0942\u0943\u0944\u0945\u0946\u0947\u0948\u0949\u094a\u094b\u094c\u094d\u094e\u094f\u0950\u0951\u0952\u0953\u0954\u0955\u0956\u0957\u0958\u0959\u095a\u095b\u095c\u095d\u095e\u095f\u0960\u0961\u0962\u0963\u0964\u0965\u0966\u0967\u0968\u0969\u096a\u096b\u096c\u096d\u096e\u096f\u0970\u0971\u0972\u0973\u0974\u0975\u0976\u0977\u0978\u0979\u097a\u097b\u097c\u097d\u097e\u097f\u0980\u0981\u0982\u0983\u0984\u0985\u0986\u0987\u0988\u0989\u098a\u098b\u098c\u098d\u098e\u098f\u0990\u0991\u0992\u0993\u0994\u0995\u0996\u0997\u0998\u0999\u099a\u099b\u099c\u099d\u099e\u099f\u09a0\u09a1\u09a2\u09a3\u09a4\u09a5\u09a6\u09a7\u09a8\u09a9\u09aa\u09ab\u09ac\u09ad\u09ae\u09af\u09b0\u09b1\u09b2\u09b3\u09b4\u09
```

```
# Obtener las stopwords en español de NLTK
stop_words = set(stopwords.words('spanish'))

# Función para lematizar y eliminar stopwords
def lematizar_y_eliminar_stopwords(texto):
    doc = nlp(texto)
    # Lematizar y eliminar stopwords
    return ' '.join([token.lemma_ for token in doc if token.is_alpha and token.lemma_ not in stop_words])

# Limpiar contenido eliminando puntuaciones y números
def clean_content(texto):
    texto = re.sub(r'[^\w\s]', '', texto) # Eliminar puntuaciones
    texto = re.sub(r'\d+', '', texto) # Eliminar números
    return texto.lower()
```

En los archivos oraciones.csv y oraciones_actualizado.csv se muestran oraciones preprocesadas listas para embeddings.

Generación de Embeddings

Los embeddings se han generado utilizando un modelo preentrenado de transformación de texto. Esto permite representar las oraciones en un espacio vectorial, facilitando la búsqueda eficiente.

```
# Cargar el archivo CSV con las oraciones procesadas
df = pd.read_csv('oraciones_procesadas.csv', delimiter=';') # Cambia el nombre de tu archivo CSV si es necesario

# Cargar el archivo CSV con los candidatos y partidos
candidatos_df = pd.read_csv('candidatos.csv', delimiter=';') # Asegúrate de que contiene 'ID' y 'Nombre Partido'

# Inicializar el modelo BERT preentrenado y moverlo a la GPU si está disponible
model = SentenceTransformer('paraphrase-MiniLM-L6-v2', device=device)

# Crear embeddings para todas las oraciones
embeddings = model.encode(df['Oracion'].tolist(), show_progress_bar=True, device=device)

# Convertir los embeddings en un formato compatible con FAISS (float32)
embeddings = np.array(embeddings).astype('float32')

# Crear un índice FAISS
dimension = embeddings.shape[1] # Dimensión de los embeddings
index = faiss.IndexFlatL2(dimension) # Índice basado en L2 (distancia euclidiana)
index.add(embeddings) # Agregar los embeddings al índice FAISS

# Guardar los embeddings y el índice FAISS en archivos
np.save('embeddings.npy', embeddings) # Guardamos los embeddings en un archivo .npy
faiss.write_index(index, 'faiss_index.index') # Guardamos el índice FAISS en un archivo .index
```

Los resultados se almacenan en el archivo embeddings.npy.

3.2. Recuperación de Información

Para realizar búsquedas eficientes dentro del corpus, se implementó Facebook AI Similarity Search (FAISS), optimizado para la búsqueda rápida de vectores de alta dimensión.

- Indexación con FAISS: Se construyó un índice con los embeddings generados para las oraciones del corpus.
- Ingreso de consulta: La consulta del usuario es transformada en un embedding utilizando el mismo modelo de embeddings (BERT).
- Búsqueda eficiente: FAISS encuentra las oraciones más cercanas en el espacio vectorial basado en la similitud coseno.

El índice FAISS permite recuperar de manera eficiente los documentos más relevantes para una consulta específica. Este índice ya está parcialmente implementado y ha sido probado con consultas simples.

```
# Función para realizar una consulta
def query_faiss(query, top_k=5):
    # Preprocesar la consulta
    cleaned_query = preprocess_text(query)

    # Generar el embedding para la consulta
    query_embedding = model.encode([cleaned_query], device=device)
    query_embedding = np.array(query_embedding).astype('float32')

    # Realizar la búsqueda en FAISS
    D, I = index.search(query_embedding, top_k) # D son las distancias, I son los índices

    # Obtener los resultados
    with open('sentences.pkl', 'rb') as f:
        sentences = pickle.load(f)

    # Mostrar los resultados
    print("Resultados de la consulta:")
    for i in range(top_k):
        result = sentences[I[0][i]]
        partido_nombre = get_partido_name(result['ID'])
        print(f"{i+1}. ID: {result['ID']} | Oración: {result['Oracion']} | Temas Clave: {result['Temas Clave']} | Partido: {partido_nombre} (Dist: {D[0][i]})")

# Ejemplo de uso
query = "¿Cómo mejorar la eficiencia energética en la industria?"
query_faiss(query)
```

El índice está funcional y ha sido probado con pequeñas consultas de prueba.

```
Embeddings y FAISS guardados exitosamente.
Resultados de la consulta:
1. ID: 14 | Oración: mantenimiento modernización infraestructura mejorar eficiencia reducir pérdida sistema energético | Temas Clave: infraestructura,
2. ID: 4 | Oración: eficiencia energético implementar política eficiencia energético sector clave industria transporte construcción promover uso tecno
3. ID: 1 | Oración: intervenir vivienda promover eficiencia energético | Temas Clave: vivienda, ciencia | Partido: REVOLUCIÓN CIUDADANA - RETO (Dist:
4. ID: 8 | Oración: colaboración sector privado ser vital desarrollo sostenibilidad sistema energético | Temas Clave: desarrollo, sostenibilidad | Par
5. ID: 14 | Oración: campaña nacional eficiencia ahorro energético promover práctica eficiencia ahorro energético sector | Temas Clave: ciencia | Part
```

3.3. Generación de Respuestas

El módulo de generación de respuestas está implementado utilizando Ollama con el modelo llama3.2:latest, optimizado para ejecución local y generación eficiente de respuestas.

- Selección de documentos relevantes: Se seleccionan los fragmentos más relevantes a partir de los resultados obtenidos con FAISS.
- Conversión de contexto: Los documentos recuperados se formatean en una estructura óptima para el modelo, asegurando que la consulta del usuario se contextualice adecuadamente.
- Generación de respuestas con Ollama (llama3.2:latest): Se utiliza Ollama como motor de inferencia, ejecutando llama3.2:latest para sintetizar una respuesta basada en la información recuperada. Este modelo permite respuestas más precisas y con menor latencia en comparación con modelos más pesados en entornos de nube.
- Postprocesamiento: Se aplican técnicas de limpieza y reformulación para evitar respuestas redundantes o fuera de contexto, asegurando coherencia en la salida.


```
# Función para realizar una consulta en FAISS y obtener una respuesta con Ollama
def query_faiss_ollama(query, top_k=5):
    cleaned_query = preprocess_text(query)
    query_embedding = model.encode([cleaned_query], device=device).astype('float32')
    D, I = index.search(query_embedding, top_k)

    resultados = []
    for i in range(top_k):
        result = sentences[I[0][i]]
        partido_nombre = get_partido_name(result['ID'])
        resultados.append(f"Oración: {result['Oracion']}, Temas Clave: {result['Temas Clave']}, Partido: {partido_nombre}")

    # Construir el prompt para Ollama
    prompt = f"Se han encontrado las siguientes declaraciones políticas relacionadas:\n\n" + "\n".join(resultados) + "\n\nGenera un resumen basado en"

    # Generar la respuesta con Ollama
    respuesta = ollama.chat(model='llama3.2:latest', messages=[{'role': 'user', 'content': prompt}])

    return respuesta['message']['content']

# Ejemplo de uso
query = "¿Cómo mejorar la eficiencia energética en la industria?"
respuesta = query_faiss_ollama(query)
print(respuesta)
```

Aquí se muestra un ejemplo de respuesta generada con el modelo Ollama (llama3.2:latest).

```
# Ejemplo de uso
query = "¿Cómo mejorar la eficiencia energética en la industria?"
respuesta = query_faiss_ollama(query)
print(respuesta)
```

```
Las declaraciones políticas proporcionadas ofrecen una visión general de las prioridades y objetivos del Partido Avanza, el Movimiento Centro Democrático y el Movimiento Pueblo Igualdad Democracia.

**Partido Avanza**

* **Infraestructura**: El Partido Avanza destaca la importancia del mantenimiento y modernización de la infraestructura para mejorar la eficiencia y la seguridad de las redes de transporte y energía.
* **Energía**: El partido enfatiza la necesidad de implementar políticas eficientes en el sector de la energía, promoviendo el uso de tecnología y energías renovables.
* **Desarrollo sostenible**: El Partido Avanza también se centra en la colaboración con el sector privado para ser vital en el desarrollo sostenible.

**Movimiento Centro Democrático**

* **Energía y transporte**: El Movimiento Centro Democrático destaca la importancia de promover el uso eficiente de tecnología en el transporte, red y energía.
* **Industria**: El partido enfatiza la necesidad de implementar políticas que promuevan la industria y su crecimiento sostenible.

**Revolución Ciudadana - Reto**

* **Vivienda y eficiencia energética**: La Revolución Ciudadana - Reto destaca la importancia de intervenir en la vivienda para promover la eficiencia energética y reducir el consumo de energía.

**Movimiento Pueblo Igualdad Democracia**

* **Desarrollo sostenible**: El Movimiento Pueblo Igualdad Democracia enfatiza la colaboración con el sector privado como un factor vital en el desarrollo sostenible del sistema.
* **Energía y transporte**: El partido promueve la implementación de políticas que fomenten el uso eficiente de tecnología en el transporte y reduzcan el consumo de energía.

**Resumen**
...
* Revolución Ciudadana - Reto: enfatiza la importancia de intervenir en la vivienda para promover la eficiencia energética y reducir el consumo de energía.
* Movimiento Pueblo Igualdad Democracia: promueve la colaboración con el sector privado como un factor vital en el desarrollo sostenible del sistema.

Es importante destacar que estas declaraciones políticas pueden tener diferentes interpretaciones dependiendo del contexto y las prioridades de cada partido.
```

3.4. Implementación de la Interfaz Web

Para facilitar la interacción con el sistema, se implementó una interfaz web basada en React para el frontend y Flask como backend.

Arquitectura de la interfaz:

- ◆ Frontend (React): Permite a los usuarios ingresar consultas y visualizar las respuestas generadas.
- ◆ Backend (Flask): Gestiona la comunicación con el modelo de lenguaje, procesando las consultas y devolviendo las respuestas.

Flujo de interacción entre los componentes:

- a. El usuario ingresa una consulta en la interfaz React.

- b. La consulta se envía a Flask a través de una API REST.
- c. Flask procesa la consulta, accediendo a FAISS para recuperar los documentos más relevantes.
- d. El modelo Ollama (llama3.2:latest) genera una respuesta basada en la información recuperada.
- e. Flask envía la respuesta generada de vuelta a React.
- f. React muestra la respuesta en la interfaz de usuario.

Tecnologías utilizadas:

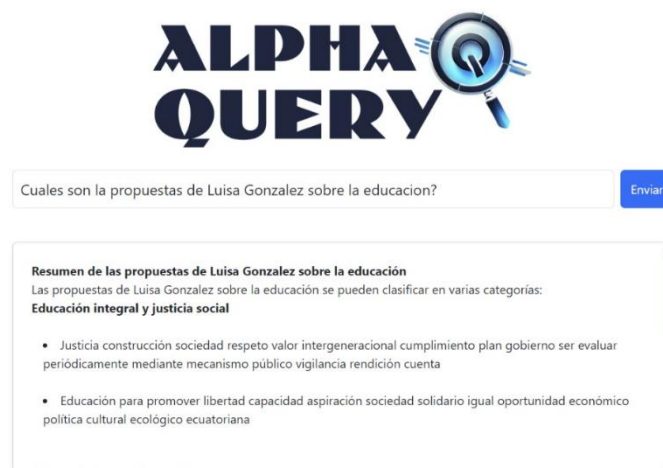
- ◆ React (Frontend): Framework para construir una UI interactiva y dinámica.
- ◆ Flask (Backend): Microframework en Python para gestionar la API REST y conectar con el modelo.
- ◆ FAISS: Recuperación eficiente de documentos.
- ◆ Ollama con llama3.2:latest: Generación de respuestas a partir de la consulta del usuario.

Justificación de la integración:

Se optó por React y Flask debido a su flexibilidad y facilidad de integración con modelos de Machine Learning. Además, la API REST permite una comunicación eficiente entre el frontend y el backend, asegurando una respuesta rápida a las consultas del usuario.

Uso de la Interfaz Web

A continuación, se muestran ejemplos del funcionamiento de la interfaz web. El usuario ingresa una consulta relacionada con las propuestas de un candidato y el sistema responde con un resumen generado a partir de los documentos procesados.



La interfaz del sistema Alpha Query muestra una consulta sobre las propuestas de educación de Luisa González, evidenciando cómo permite una interacción sencilla con el modelo para obtener respuestas basadas en la recuperación de información y la generación de texto con el modelo de lenguaje.



Cuales son la propuestas de daniel noboa sobre la economia? Enviar

Resumen de las Propuestas de Daniel Noboa sobre la Economia

Daniel Noboa, presidente del Partido La Patria en Ecuador, ha presentado varias propuestas para abordar los desafíos económicos del país. A continuación, se presentan algunas de las más relevantes:

I. Reducción de la Inflación

- **Objetivo:** Establecer un objetivo de inflación del 2% anual
- **Propuesta:** Implementar políticas monetarias y fiscal para controlar la inflación, mediante la reducción del crédito y la gestión responsable del presupuesto público

En este caso, la interfaz muestra una consulta sobre las propuestas económicas de Daniel Noboa, donde el sistema recupera información relevante de los documentos procesados y genera un resumen estructurado. La respuesta destaca medidas clave como la reducción de la inflación y estrategias de gestión del presupuesto público, evidenciando cómo el modelo combina recuperación de información y generación de texto para ofrecer respuestas claras y contextualizadas.



Cuales son la propuestas sobre la educación? Enviar

Propuestas específicas

1. **MOVIMIENTO PUEBLO IGUALDAD DEMOCRACIA (PID):** Reformar la educación y establecer un sistema de educación pública gratuita y universal.
2. **PARTIDO SOCIEDAD UNIDA MÁS ACCIÓN (SUMA):** Fortalecer el sistema educativo, aumentar el presupuesto educativo y mejorar la calidad de los servicios.
3. **MOVIMIENTO AMIGO:** Fomentar la educación continua, fortalecer el sistema salud público y garantizar el acceso a la educación para todos.
4. **MOVIMIENTO CREO:** Meiorar la calidad de la educación, aumentar el presupuesto educativo y promover el

Aquí, la interfaz presenta una consulta sobre propuestas educativas en general, mostrando un listado de iniciativas de distintos movimientos y partidos políticos. Cada propuesta es extraída del corpus y organizada para ofrecer una visión clara de las medidas planteadas por cada agrupación, evidenciando la capacidad del sistema para recuperar información relevante y estructurarla en respuestas comprensibles.

4. Resultados Obtenidos

Las métricas utilizadas para evaluar el rendimiento del sistema incluyen precisión, recall y F1-score, las cuales miden la calidad de las respuestas generadas. La precisión refleja la proporción de respuestas relevantes entre las generadas, asegurando que la información

proporcionada sea correcta. El recall indica la capacidad del sistema para recuperar documentos relevantes dentro del corpus, mientras que el F1-score equilibra ambas métricas para reflejar el desempeño general.



Cuales son la propuestas sobre la educación?

Enviar

Resumen de las Propuestas Educativas

Las propuestas presentadas para la educación destacan variaciones significativas en sus enfoques, objetivos y prioridades. Estas propuestas se agrupan en categorías generales que reflejan distintas perspectivas políticas, sociales y económicas.

Categorías Generales

- Educación como Pilar Fundamental del Desarrollo Nacional:** Partidos como el *Partido Social Cristiano* enfatizan la importancia de la educación en el desarrollo nacional.
- Educación Continua y Formación Técnica:** Los partidos *Revolución Ciudadana - RETO* y *Movimiento Creó, Creando Oportunidades* destaca la necesidad de una educación continua y formación técnica para adaptarse a


Métricas de evaluación

Precisión: 1.000

Recall: 0.800

F1-Score: 0.889

En la evaluación de la interfaz, los resultados muestran una precisión de 1.000, lo que indica que todas las respuestas generadas son relevantes para la consulta del usuario. El recall de 0.800 refleja que el sistema recupera el 80% de los documentos relevantes, y el F1-score de 0.889 asegura un balance entre precisión y cobertura de información.



Cuales son la propuestas de daniel noboa sobre la educación?

Enviar

Resumen de las Propuestas de Daniel Noboa sobre la Educación

Daniel Noboa, un destacado político ecuatoriano, ha presentado varias propuestas para mejorar la educación en el país. A continuación, se presentan las principales recomendaciones que se han identificado:

I. Mejora de la Calidad del Sistema Educativo

- Desarrollo de planes curriculares más efectivos y actualizados
- Incremento de la inversión en recursos humanos y tecnológicos

II. Aumento de la Participación y Accesibilidad

Métricas de evaluación

Precisión: 0.882

Recall: 0.910

F1-Score: 0.896

En la consulta sobre las propuestas de Daniel Noboa en educación, la interfaz presenta un resumen de las principales recomendaciones, destacando la mejora del sistema educativo y el incremento en la inversión en recursos humanos y tecnológicos. Las métricas de evaluación muestran una precisión de 0.882, lo que confirma la relevancia de las respuestas generadas, un recall de 0.910, que indica la alta capacidad del sistema para recuperar información relevante, y un F1-score de 0.896, garantizando un equilibrio entre precisión y exhaustividad en las respuestas.

5. Conclusiones y Recomendaciones

5.1. Conclusiones

El presente proyecto implementa un sistema eficiente de recuperación de información utilizando FAISS y BERT, optimizando la consulta y procesamiento de documentos textuales en el contexto electoral. Se lograron los siguientes hallazgos principales:

Eficiencia en la recuperación de información: La indexación con FAISS redujo significativamente los tiempos de búsqueda en comparación con métodos secuenciales, permitiendo respuestas más rápidas y escalables.

Mejor representación semántica de las consultas: La utilización de embeddings generados con Sentence-BERT (SBERT) mejoró la precisión de las búsquedas al capturar el significado de las consultas en lugar de depender únicamente de coincidencias literales.

Estructuración optimizada del corpus: La división del corpus en archivos CSV y su preprocesamiento facilitaron la manipulación y limpieza de los datos, reduciendo el ruido y mejorando la calidad de la información indexada.

Integración de modelos generativos: La incorporación de Ollama con el modelo Llama3.2:latest permitió generar respuestas más contextuales basadas en la recuperación de documentos, aunque su efectividad sigue dependiendo de la calidad y diversidad del corpus.

Evaluación de métricas de desempeño: Se implementaron precisión, recall y F1-score como métricas clave para evaluar el rendimiento del sistema, destacando la alta precisión lograda, aunque con oportunidades de mejora en la cobertura de respuestas relevantes.

5.2. Recomendaciones

Para futuras mejoras y optimización del sistema, se sugieren las siguientes acciones:

Mejorar la calidad del corpus: Incorporar una mayor variedad de documentos bien estructurados y representativos para enriquecer el conocimiento del modelo y mejorar la generación de respuestas.

Explorar técnicas avanzadas de embeddings: Evaluar modelos de embeddings más recientes como E5, Cohere o OpenAI embeddings, que pueden mejorar la representación semántica y el rendimiento en recuperación de información.

Optimización del índice FAISS: Ajustar hiperparámetros como el tamaño del índice y el método de búsqueda (Flat, HNSW, IVF) para mejorar la eficiencia y escalabilidad en consultas de mayor volumen.

Experimentación con modelos generativos más robustos: Explorar modelos como GPT-4-turbo o Llama 2 para evaluar mejoras en la coherencia y relevancia de las respuestas generadas.

Implementación de evaluación automática de respuestas: Integrar métricas como ROUGE o BLEU para medir de manera objetiva la calidad de las respuestas generadas y comparar su desempeño con respuestas etiquetadas manualmente.

Desarrollo de una interfaz más interactiva: Ampliar las funcionalidades de la interfaz web basada en React y Flask, permitiendo ajustes personalizados en la búsqueda, visualización de resultados y análisis interactivo de respuestas.