DepoIndex: Automating Deposition Table of Contents Generation

---

1. Introduction: The Challenge of Deposition Transcripts

The Problem

- **Manual & Time-Consuming:** Creating accurate tables of contents for lengthy deposition transcripts is a labor-intensive, manual process for paralegals and legal professionals.
- **Inconsistency:** Human error can lead to inconsistencies in topic labeling and indexing.
- **Efficiency Bottleneck:** Delays in indexing slow down legal review processes.

Our Solution: DepoIndex

- **Leveraging AI:** An intelligent system that automates the generation of a precise, high-quality table of contents for deposition PDFs.
- **Key Benefit:** Saves significant time and resources, enhances accuracy, and streamlines legal document review.

---

2. Project Overview & Architecture

DepoIndex Workflow

1. **Input:** Raw deposition PDF transcript.
2. **Preprocessing:** Cleans and prepares the text using NLP techniques.
3. **Topic Extraction:** A Generative AI model identifies and labels key topics.
4. **Output:** Structured Table of Contents (JSON, Markdown, DOCX).

Core Components

- **PyPDF:** For extracting text from PDF documents.
- **NLTK (Natural Language Toolkit):** For robust text preprocessing.
- **Google Gemini (Generative AI):** The intelligent core for topic identification and structuring.

---

3. Deep Dive into Text Preprocessing

Why Preprocess?

- Raw text from PDFs often contains noise (headers, footers, timestamps) and requires standardization for effective AI processing.

Steps Involved

- **Text Extraction:** Using PyPDF to read and extract text page by page.
- **Lowercasing:** Standardizing all text to lowercase to reduce vocabulary size and improve matching.

- **Noise Removal (Regex):**
  - re.sub(pattern=r'\b\d{2}:\d{2}\b', string=text, repl=''): Removes timestamps like 01:32.
  - re.sub(pattern=r"page (\d+)",repl='',string=text): Removes "page X" indicators.
  - re.search(pattern='witness signature', string=text): Identifies the end of the main transcript.
- **Tokenization:** Breaking down text into individual words or phrases (nltk.word_tokenize).
- **Lemmatization:** Reducing words to their base form (e.g., "running" to "run," "better" to "good") using WordNetLemmatizer and pos_tag for accurate Part-of-Speech (POS) tagging (nltk.download('wordnet'), nltk.download('omw-1.4')).
  - Helper function get_wordnet_pos maps NLTK POS tags to WordNet's format.
- **Stop Word Removal:** Eliminating common words that don't add significant meaning (e.g., "the", "is", "a") using nltk.corpus.stopwords.
- **POS Filtering:** Removing specific parts of speech (Determiners, Prepositions, Conjunctions) that are less relevant for topic identification (pos_to_remove = {'DT', 'IN', 'CC'}).

Code Snippet (Illustrative)

Python

```
# NLTK Downloads
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')


def processText(path:str):
    # ... (rest of your processText function)
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    tagged_tokens_for_lemmatization = pos_tag(tokens)
    lemmatized_words = []
    for word, tag in tagged_tokens_for_lemmatization:
        lemmatized_words.append(lemmatizer.lemmatize(word, pos=get_wordnet_pos(tag)))
```

```
    tokens = lemmatized_words

    pos_tags = pos_tag(tokens)

    pos_to_remove = {'DT', 'IN', 'CC'}

    filtered_tokens_pos = [word for word, tag in pos_tags if tag not in pos_to_remove and word not in stopwords_set and word not in {',',';',':'} ]

    # ...

    return inputText
```

---

4. Generative AI for Topic Extraction

The LLM's Role

- After preprocessing, the clean text is fed to a large language model (LLM).
- The LLM's task is to act as an "expert paralegal" to identify and label new topics, along with their precise page and line numbers.

Prompt Engineering

- **Role-Playing:** Instructing the LLM to act as "DepoIndex, an expert paralegal."
- **Input Format Clarification:** Clearly stating the input is "lemmatized text, and removed stop words," with page and line number markers.
- **Output Constraint: Crucially, forcing JSON output** (response_mime_type="application/json") for structured data.
- **Key Instruction:** "Re-use the **exact same topic label** (case-sensitive) if a new section appears similar in content or is a continuation of a previous topic. This maintains consistency in the output." This is vital for coherent TOCs.
- **Examples (Few-Shot Learning):** Providing one or two well-chosen examples significantly guides the LLM to produce desired output formats and quality. These examples demonstrate the expected topic granularity and line/page accuracy.
- **Line Number Constraint:** Emphasizing that the "line number It MUST be between 1 and 25" (as per typical deposition page formats).

Code Snippet (Illustrative)

Python

```
import google.genai as genai

from google.genai import types


def promptLLM(processed_Text:str):

    prompt = f"""You are DepoIndex, an expert paralegal that labels *new* topics...
```

```
...
Here is the actual input text:
<<<
{processed_Text}
>>>
Now extract topics and return ONLY the JSON output.
"""
client = genai.Client(api_key="YOUR_API_KEY") # Replace with actual key or environment variable
configuration = types.GenerateContentConfig(
    temperature=0.2, # Lower temperature for more deterministic, factual output
    response_mime_type="application/json"
)
response = client.models.generate_content(
    model="gemini-2.5-flash", # Or other suitable Gemini model
    contents=prompt,
    config=configuration
)
return response
```

---

5. Output Generation & Validation

Multiple Output Formats

- The extracted topics (topics, page, line) are saved into:
    - **JSON file:** For programmatic use and easy data interchange.
    - **Markdown file:** For human-readable, plain-text tables of contents.
    - **DOCX (Word) document:** For professional presentation and easy editing by legal staff.

Validation Process

- **Purpose:** To ensure the accuracy and reliability of the AI-generated table of contents.
- **Methodology (Sampled Validation):**
    1. Randomly sample a subset of generated topics (e.g., 10 topics).
    2. For each sampled topic, retrieve the original text excerpt at the predicted page and line number.

3. Manually (or via a second AI prompt for auto-validation) verify if the *topic label* accurately describes the content *starting precisely* at the *given line number*.

- **Metrics:** Calculate **accuracy** as (number of correct items / total number of items).

Validation Notebook Insights

- The validation notebook demonstrates a systematic approach to quality assurance.
- It includes a loop for manual human review (input("type y or n...")) as well as an automated AI-driven validation prompt for continuous integration and testing.
- **Chain of Thought (CoT):** The CoT prompt (Give a chain-of-thought reasoning...) is crucial for debugging and understanding the LLM's decision-making process, ensuring the topic is indeed initiated at the specified line. This aligns with Deep Learning and explainable AI concepts you are studying.

Sample Validation Result (from your output)

Validation Accuracy: 90.00%

(This is a good result, indicating high precision!)

---

6. Project Significance & Future Enhancements

Impact

- **Efficiency:** Drastically reduces the time required for preparing deposition summaries.
- **Accuracy:** Minimizes human error in indexing.
- **Cost Savings:** Frees up legal professionals for higher-value tasks.
- **Scalability:** Can process large volumes of transcripts quickly.