MA

```
import { RecursiveCharacterTextSplitter } from 'langchain/text_splitter';
import { OpenAIEmbeddings } from 'langchain/embeddings';
import { PineconeStore } from 'langchain/vectorstores';
import { pinecone } from '@/utils/pinecone-client';
import { processMarkDownFiles } from '@/utils/helpers';
import { PINECONE_INDEX_NAME, PINECONE_NAME_SPACE } from '@/config/pinecone';

/* Name of directory to retrieve files from. You can change this as required */
const directoryPath = 'Notion_DB';

export const run = async () => {
  try {
    /*load raw docs from the markdown files in the directory */
    const rawDocs = await processMarkDownFiles(directoryPath);

    /* Split text into chunks */
    const textSplitter = new RecursiveCharacterTextSplitter({
      chunkSize: 1000,
      chunkOverlap: 200,
    });

    const docs = await textSplitter.splitDocuments(rawDocs);
    console.log('split docs', docs);

    console.log('creating vector store...');
    /*create and store the embeddings in the vectorStore*/
    const embeddings = new OpenAIEmbeddings();
    const index = pinecone.Index(PINECONE_INDEX_NAME); //change to your own index
name
    await PineconeStore.fromDocuments(
      index,
      docs,
      embeddings,
      'text',
      PINECONE_NAME_SPACE, //optional namespace for your vectors
    );
  } catch (error) {
    console.log('error', error);
    throw new Error('Failed to ingest your data');
  }
};

(async () => {
  await run();
  console.log('ingestion complete');
})();
```

Tambahkan fiture untuk mengekstrak pdf atau txt

To extend the functionality of the script to extract text from PDF and TXT files, you can use the `pdf-parse` library for PDF files and the native `fs` module for TXT files. First, install the `pdf-parse` library by running:

bash — Copy code

```
npm install pdf-parse
```

Next, update your code to include the necessary imports and modifications:

javascript — Copy code

```
import { RecursiveCharacterTextSplitter } from 'langchain/text_splitter';
import { OpenAIEmbeddings } from 'langchain/embeddings';
import { PineconeStore } from 'langchain/vectorstores';
import { pinecone } from '@/utils/pinecone-client';
import { processMarkDownFiles } from '@/utils/helpers';
import { PINECONE_INDEX_NAME, PINECONE_NAME_SPACE } from '@/config/pinecone';
import * as fs from 'fs/promises';
import pdf from 'pdf-parse';

const directoryPath = 'Notion_DB';

const processPDFFiles = async (directoryPath) => {
  // Get all PDF files in the directory
  const pdfFiles = (await fs.readdir(directoryPath)).filter((file) =>
    file.endsWith('.pdf'),
  );

  return await Promise.all(
    pdfFiles.map(async (pdfFile) => {
      const dataBuffer = await fs.readFile(`${directoryPath}/${pdfFile}`);
      const pdfData = await pdf(dataBuffer);
      return {
        title: pdfFile.slice(0, -4),
        text: pdfData.text,
      };
    }),
  );
};

const processTextFiles = async (directoryPath) => {
  const textFiles = (await fs.readdir(directoryPath)).filter((file) =>
    file.endsWith('.txt'),
  );

  return await Promise.all(
    textFiles.map(async (txtFile) => {
      const text = await fs.readFile(`${directoryPath}/${txtFile}`, 'utf-8');
      return {
        title: txtFile.slice(0, -4),
        text: text,
      };
    }),
  );
};

export const run = async () => {
  try {
    const rawMarkdownDocs = await processMarkDownFiles(directoryPath);
    const rawPDFDocs = await processPDFFiles(directoryPath);
    const rawTextDocs = await processTextFiles(directoryPath);
    const rawDocs = [...rawMarkdownDocs, ...rawPDFDocs, ...rawTextDocs];

    const textSplitter = new RecursiveCharacterTextSplitter({
      chunkSize: 1000,
      chunkOverlap: 200,
    });

    const docs = await textSplitter.splitDocuments(rawDocs);
    console.log('split docs', docs);

    console.log('creating vector store...');
    const embeddings = new OpenAIEmbeddings();
    const index = pinecone.Index(PINECONE_INDEX_NAME);
    await PineconeStore.fromDocuments(
      index,
      docs,
      embeddings,
      'text',
      PINECONE_NAME_SPACE,
    );
  } catch (error) {
    console.log('error', error);
    throw new Error('Failed to ingest your data');
  }
};

(async () => {
  await run();
  console.log('ingestion complete');
})();
```

This updated script includes two new functions, `processPDFFiles` and `processTextFiles`, which handle extracting text from PDF and TXT files, respectively. The extracted data is then combined with the existing Markdown data before being processed by the rest of the script.