* <https://www.youtube.com/watch?v=n0uPzvGTFI0>

**RecursiveCharacterTextSplitter.ipynb**

* + Chunking işlemi için from langchain.text\_splitter import RecursiveCharacterTextSplitter kullandım.
  + Bu kod **text** **değişkenine** atanmış 3 paragraflık bir yazıyı parçalama işlemini ve **PDF.pdf** dosyasını text haline getirip parçalama işlemini yerine getirir.
* <https://python.langchain.com/docs/modules/data_connection/document_transformers/split_by_token>
  + **NLTK (The Natural Language Toolkit)**, Python programlama dilinde yazılmış İngilizce için sembolik ve istatistiksel doğal dil işleme (NLP) için bir kütüphanedir.
  + Sadece " " üzerine bölmek yerine, NLTK tokenizer'larına göre bölmek için NLTK'yı kullanılabilir. (Normalde text boşluklar (\n) ile split edilmiştir.)
  + Metin, NLTK tokenizer tarafından bölünür.
  + Chunk boyutu karakter sayısına göre ölçülür.

**NLTKTextSplitter.ipynb**

* + Chunking işlemi için from langchain.text\_splitter import NLTKTextSplitter kullandım.
  + Bu kod **text** **değişkenine** atanmış 3 paragraflık bir yazıyı parçalama işlemini ve **PDF.pdf** dosyasını text haline getirip parçalama işlemini yerine getirir.
* <https://python.langchain.com/docs/modules/data_connection/document_transformers/split_by_token>
  + **spaCy**, Python ve Cython programlama dillerinde yazılmış, gelişmiş doğal dil işleme için açık kaynaklı bir kütüphanedir.
  + NLTK'ya başka bir alternatif de **spaCy tokenizer** kullanır.
  + Metin, spaCy tokenizer tarafından bölünür.
  + Chunk boyutu karakter sayısına göre ölçülür.

**SpacyTextSplitter.ipynb**

* + Chunking işlemi için from langchain.text\_splitter import SpacyTextSplitter kullandım.
  + Bu kod **text** **değişkenine** atanmış 3 paragraflık bir yazıyı parçalama işlemini ve **PDF.pdf** dosyasını text haline getirip parçalama işlemini yerine getirir.
* https://www.pinecone.io/learn/chunking-strategies/
  + LaTeX, genellikle akademik makaleler ve teknik belgeler için kullanılan bir belge hazırlama sistemi ve biçimlendirme dilidir.
  + LaTeX komutlarını ve ortamlarını ayrıştırarak, içeriğin mantıksal organizasyonuna (örneğin, bölümler, alt bölümler ve denklemler) uyan parçalar oluşturabilir, böylece daha doğru ve bağlamsal olarak alakalı sonuçlar elde edilebilir.

**LatexTextSplitter.ipynb**

* + Chunking işlemi için from langchain.text\_splitter import LatexTextSplitter kullandım.
  + Bu kod **text** **değişkenine** atanmış 3 paragraflık bir yazıyı parçalama işlemini ve **PDF.pdf** dosyasını text haline getirip parçalama işlemini yerine getirir.
* <https://medium.com/@thallyscostalat/chunking-strategies-optimization-for-retrieval-augmented-generation-rag-in-the-context-of-e47cc949931d>
  + Content-Aware Splitting (Markdown, LaTeX, HTML)
    - Bu yöntem, özellikle Markdown, LaTeX veya HTML gibi yapılandırılmış belgelerde içerik türüne ve yapısına odaklanır. İçerik türlerinin yığınlar içinde karıştırılmamasını sağlayarak bütünlüğü korur. Zorluklar arasında belirli sözdiziminin anlaşılması ve yapılandırılmamış belgeler için uygun olmaması yer alır. Yapılandırılmış belgeler için kullanışlıdır ancak yapılandırılmamış içerik için önerilmez.

**MarkdownTextSplitter.ipynb**

* + Chunking işlemi için from langchain.text\_splitter import MarkdownTextSplitter kullandım.
  + Bu kod **text** **değişkenine** atanmış 3 paragraflık bir yazıyı parçalama işlemini ve **PDF.pdf** dosyasını text haline getirip parçalama işlemini yerine getirir.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parametreler** | **RecursiveCharacterTextSplitter** | **NLTKTextSplitter** | **SpacyTextSplitter** | **LatexTextSplitter** | **MarkdownTextSplitter** |
| chunk\_size = 500  chunk\_overlap = 0  length\_function = len | **text değişkeni** için **3** chunk,  **PDF.pdf** için **288** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **326** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **328** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **270** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **288** chunk |
| chunk\_size = 200  chunk\_overlap = 0  length\_function = len | **text değişkeni** için **9** chunk,  **PDF.pdf** için **814** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **684** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **722** chunk | **text değişkeni** için **7** chunk,  **PDF.pdf** için **684** chunk | **text değişkeni** için **9** chunk,  **PDF.pdf** için **814** chunk |
| chunk\_size = 500  chunk\_overlap = 20  length\_function = len | **text değişkeni** için **3** chunk,  **PDF.pdf** için **288** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **326** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **328** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **279** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **288** chunk |
| chunk\_size = 200  chunk\_overlap = 20  length\_function = len | **text değişkeni** için **9** chunk,  **PDF.pdf** için **817** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **684** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **724** chunk | **text değişkeni** için **7** chunk,  **PDF.pdf** için **743** chunk | **text değişkeni** için **9** chunk,  **PDF.pdf** için **817** chunk |
| chunk\_size = 500  length\_function = len | **text değişkeni** için **3** chunk,  **PDF.pdf** için **449** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **400** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **406** chunk | **text değişkeni** için **4** chunk,  **PDF.pdf** için **445** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **449** chunk |
| chunk\_size = 200  length\_function = len | **text değişkeni** için **64** chunk,  **PDF.pdf** için **1865** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **768** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **892** chunk | **text değişkeni** için **95** chunk,  **PDF.pdf** için **12612** chunk | **text değişkeni** için **64** chunk,  **PDF.pdf** için **1865** chunk |
| chunk\_size = 500 | **text değişkeni** için **3** chunk,  **PDF.pdf** için **449** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **400** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **406** chunk | **text değişkeni** için **4** chunk,  **PDF.pdf** için **445** chunk | **text değişkeni** için **3** chunk,  **PDF.pdf** için **449** chunk |
| chunk\_size = 200 | **text değişkeni** için **64** chunk,  **PDF.pdf** için **1865** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **768** chunk | **text değişkeni** için **8** chunk,  **PDF.pdf** için **892** chunk | **text değişkeni** için **95** chunk,  **PDF.pdf** için **12612** chunk | **text değişkeni** için **64** chunk,  **PDF.pdf** için **1865** chunk |

Chunking strajejilerinde parametrelere göre elde edilen chunk sayıları:

* + **chunk\_size:** Metnin kaç karakterlik (harf veya işaret) parçalara bölüneceğini belirler.
  + **chunk\_overlap:** Parçalar arasındaki karakter örtüşmesini belirler. Örtüşme olması durumunda, her parça bir önceki parçanın belirli bir kısmını içerir. 0 ise, parçalar arasında örtüşme yok.
  + **length\_function:** Metnin uzunluğunu hesaplayan bir fonksiyon belirler. "len" fonksiyonu belirlenirse, metnin karakter sayısı uzunluğunu ifade eder.
* <https://docs.llamaindex.ai/en/stable/examples/node_parsers/semantic_chunking/> **semantic\_chunker.ipynb**
  + Semantic chunker, metni sabit bir yığın boyutuyla yığınlamak yerine, gömme benzerliğini kullanarak cümleler arasındaki kesme noktasını uyarlamalı olarak seçer. Bu, bir "chunk" 'ın anlamsal olarak birbiriyle ilişkili cümleler içermesini sağlar.
  + **Data Setup** kısmında PDF.pdf dosyasındaki veri ***PDF.text*** haline çevirip txt dosyasına yazdırdım. Ayrıca fransa hakkında 3 paragraflık bir yazıyı da ***fransa.txt*** dosyasına yazdırdım.
  + **Chunking işlemi için:**
    - **Define Semantic Splitter With OpenAi** ksımında from llama\_index.embeddings.openai import OpenAIEmbedding kullanıldığında 'You exceeded your current quota, please check your plan and billing details.’ hatası alıyorum.
    - Bu yüzden **Define Semantic Splitter With Hugging Face** kısmındafrom llama\_index.embeddings.huggingface import HuggingFaceEmbedding kullandım.
    - **fransa.txt için:** ***splitter*** ile **2 chunk** (node) elde edildi, ***base\_splitter*** ile **1 chunk** (base\_node) elde edildi ve ***parser*** ile **1 chunk** (node) elde edildi.
    - **PDF.pdf için:**
      * ***splitter*** ile **49 chunk** (node) elde edildi
      * ***base\_splitter*** ile **116 chunk** (base\_node) elde edildi
      * ***parser*** ile **44 chunk** (node) elde edildi.
    - ***splitter*** *(buffer\_size=1, breakpoint\_percentile\_threshold=95, embed\_model=embed\_model)* ***SemanticSplitterNodeParser()*** *classından,* ***base\_splitter*** *(chunk\_size=512)* ve ***parser*** ise ***SentenceSplitter()*** classından oluşturulmuş farklı metin bölme stratejilerini uygulayan text splitter (metin bölücü) nesneleridir.
    - **buffer\_size=1:** Bu parametre, işlenmeden önce bellekte tutulan düğüm sayısını kontrol eder. Bunu 1'e ayarlamak, bölücünün her düğümü belleğe almadan bağımsız olarak işleyeceği anlamına gelir.
    - **breakpoint\_percentile\_threshold=95:** Bu parametre, muhtemelen bölücünün metin düğümleri içinde anlamsal sınırları nasıl belirlediğini etkiler. 95'lik bir değer, bir kesme noktasının (bölme noktası), ardışık metin segmentleri arasındaki gömüm mesafesi tüm gömüm mesafelerinin 95. persentilini aştığında ortaya çıkabileceğini önerir. Bu, düğümü anlamsal olarak farklı bölümlerde bölmeye yardımcı olur.
    - **embed\_model=embed\_model:** Bu parametre, daha önce oluşturulan `embed\_model` örneğini `SemanticSplitterNodeParser`'a aktarır. Bu, bölücünün gömüm modelini kullanarak gömüm mesafelerini hesaplamasına ve metin düğümleri içinde anlamsal kopmaları belirlemesine olanak tanır.
  + **Setup Query Engine** kısmında yine nedenini bilmediğim halde kota hatası aldım, çözümünü bulamadım.