# **Entity Extraction Workflow Documentation**

## **Overview**

The Entity Extraction workflow consists of two main tabs: "Entities from XML" and "Entities from Narrative." This process involves extracting specific information from XML files and narrative text, utilizing distinct methods for each tab.

## **Entities from XML**

### **Purpose**

The "Entities from XML" tab focuses on extracting relevant data from XML files. This section is designed to chunk the required data and extract specific information from XML sources.

**Workflow Steps:**

* **Input XML File:**
  + Start by uploading the XML file containing the targeted data.
* **XML Parsing:**
  + The system undertakes parsing of the XML file to discern its structure and constituent elements.
* **Data Chunking:**
  + The workflow identifies and chunks the requisite data from the XML file based on predefined criteria or user specifications.
* **Information Extraction:**
  + Utilizing the Llama2 7b model, the chunked data is processed to extract specific information, adhering to user-defined parameters or discerned patterns.
* **Output:**
  + The extracted information is presented as output, providing a structured representation of the pertinent data obtained from the XML source.

**Entities from Narrative**

**Purpose:** The "Entities from Narrative" tab is dedicated to extracting entities from narrative text, aiming to discern and extract pertinent information embedded within textual narratives.

**Workflow Steps:**

* **Input Narrative Text:**
  + Users input the narrative text containing the desired information.
* **Text Analysis:**
  + The system conducts comprehensive text analysis to discern entities present within the narrative.
* **Entity Identification:**
  + Leveraging predefined rules or algorithms, the workflow identifies entities within the narrative text.
* **Information Extraction:**
  + Utilizing the Llama2 7B model, the narrative text undergoes processing to extract entities and relevant information, adhering to specified criteria or discerned patterns.
* **Output:**
  + The final output comprises a curated list of extracted entities along with associated information gleaned from the provided narrative text.

**Conclusion:** The Entity Extraction workflow offers a structured and efficient approach to extract specific information from both XML files and narrative text. By utilizing the respective tabs tailored to different data sources, users can achieve precise and accurate information retrieval, thereby enhancing overall efficiency and productivity.

**Mapping the Product Code**

**Purpose:** The **product code** endpoint serves the purpose of mapping product codes based on medicinal product names. This functionality aids in associating relevant product codes with corresponding medicinal products, facilitating efficient data handling and retrieval.

**Workflow Steps:**

* **Input Medicinal Product Details:**
  + Users provide details including the medicinal product for which the product code needs to be mapped.
* **Data Processing:**
  + The provided medicinal product details are processed to prepare the data for mapping.
* **Product Code Mapping:**
  + The system employs TF-IDF vectorization and cosine similarity to map the medicinal product to the most similar product in the dataset.
* **Output Generation:**
  + The resulting mapping includes details such as the medicinal product name, associated product description (**MAKTX**), and the corresponding product code (**MATNR**).
* **Output Delivery:**
  + The mapped product codes along with relevant details are returned as a response to the user query.

**Conclusion:** The **/productcode** endpoint offers a streamlined approach to map product codes based on medicinal product names. By leveraging advanced text analysis techniques, users can efficiently retrieve accurate product code mappings, enhancing overall data management and processing capabilities.

**Narrative Generation**

**Purpose:** The narrative generation functionality aims to produce comprehensive medical narratives based on provided input data. It ensures factual accuracy, clarity, completeness, logical structure, and contextual awareness while adhering to medical terminology and patient confidentiality.

**Workflow Steps:**

* **Input Data Preparation:**
  + Users provide input data in the form of a dictionary containing patient information.
* **Prompt Update:**
  + The system generates a prompt using the provided input data, setting guidelines for narrative generation.
* **Model Invocation:**
  + The narrative generation model (Llama 2.7B) is invoked using the prepared prompt and model parameters.
* **Response Parsing:**
  + The response from the model is parsed to extract the generated narrative text.
* **Output Delivery:**
  + The generated narrative text, conforming to the specified guidelines and parameters, is returned as the output response.

**Conclusion:** The narrative generation functionality provides a robust solution for generating medical narratives from input data. By leveraging advanced natural language processing models, users can obtain accurate and contextually relevant narratives, enhancing documentation and communication in medical contexts.

**Documentation Packages**

**XML Extraction Package:**

from fastapi import FastAPI

from pydantic import BaseModel

import xml.etree.ElementTree as ET

from sagemaker.jumpstart.model import JumpStartModel

from dotenv import load\_dotenv

from botocore.config import Config

import os

import boto3, Json

import uvicorn

import re

from fastapi.responses import JSONResponse

**Narrative Extraction Package:**

from sagemaker.jumpstart.model import JumpStartModel

import os

import boto3, Json

from dotenv import load\_dotenv

import xml.etree.ElementTree as ET

import re

**Narrative Generation Package:**

from fastapi import FastAPI

from pydantic import BaseModel

import xml.etree.ElementTree as ET

from sagemaker.jumpstart.model import JumpStartModel

from botocore.config import Config

import os

import boto3, Json

import uvicorn

**Creating Documentation Points:**

* **XML Extraction:**
  + The XML extraction package provides functionalities for parsing and extracting specific information from XML files. It utilizes FastAPI for building APIs, Pydantic for data validation, and SageMaker JumpStart for model interactions.
* **Narrative Extraction:**
  + The narrative extraction package facilitates the extraction of entities from textual narratives. It incorporates SageMaker JumpStart for model interactions and handles data from both XML files and narrative texts.
* **Narrative Generation:**
  + The narrative generation package focuses on generating comprehensive medical narratives based on input data. It uses FastAPI for API development, Pydantic for data validation, and interacts with the Llama 2 7B model for generating accurate and contextually relevant narratives.

**Conclusion:** The documentation packages encapsulate the required functionalities for XML extraction, narrative extraction, and narrative generation. Developers can refer to these packages to understand and implement the respective features in their applications.

import pandas as pd from sklearn.metrics.pairwise import cosine\_similarity from sklearn.feature\_extraction.text import TfidfVectorizer

**Product code Documentation packages :**

* **Pandas (pd):**
  + Pandas is a versatile library for data manipulation and analysis in Python. It offers data structures like DataFrame and Series, along with functions for data cleaning, transformation, and analysis. In the context of product coding, Pandas can be used to organize and manipulate data efficiently.
* **Cosine Similarity (from sklearn.metrics.pairwise):**
  + Cosine similarity is a metric used to measure the similarity between two vectors in a high-dimensional space. In product coding, cosine similarity can be employed to determine the similarity between medicinal product names, aiding in matching products based on their names.
* **TF-IDF Vectorizer (from sklearn.feature\_extraction.text):**
  + TF-IDF (Term Frequency-Inverse Document Frequency) is a technique to quantify the importance of terms in a collection of documents. The TF-IDF vectorizer converts text data into numerical vectors, with each dimension representing the importance of a term in a document relative to the entire corpus. In product coding, TF-IDF vectorization can be utilized to extract features from medicinal product names for similarity comparison.

**Conclusion:** These packages provide essential functionalities for product coding tasks, including data manipulation with Pandas and text similarity calculation using cosine similarity and TF-IDF vectorization. By leveraging these tools, developers can efficiently process and analyze medicinal product data for accurate product coding.