Incorporating Social Media Content into Digital Twin Information Retrieval

To enhance Digital Twin Information Retrieval, we can integrate social media content from platforms like Twitter (X), LinkedIn, Reddit, and GitHub. This allows us to:

- **Track real-time expertise updates from professionals.**
- **Enhance knowledge retrieval with public discussions and research.**
- **Detect emerging trends in a given field.**

1. Components of Social Media Integration

1.1 Data Source

- **Twitter (X): Extract professional tweets, hashtags, and expert mentions.**
- **LinkedIn: Fetch professional profiles, endorsements, and posts**
- **Reddit: Analyze discussions on knowledge-intensive topics.**
- **GitHub: Monitor project contributions, repositories, and discussions.**

1.2 Methods of Integration

- **APIs: Use official APIs (Twitter API, LinkedIn API, etc.) to pull data.**
- **Web Scraping: Extract structured information when APIs are limited.**
- **Natural Language Processing (NLP): Extract expertise topics from text.**

2. Implementation: Social Media Integration for Digital Twin Retrieval

2.1 Install Required Libraries

```
pip install tweepy beautifulsoup4 requests fastapi sqlite3 pydantic elasticsearch transformers
```

2.2 Twitter API Integration for Expertise Extraction

We fetch **tweets from experts and update digital twins based on their posts.**

```
import tweepy
import json

# Twitter API credentials (replace with actual keys)
```

```
API_KEY = "your_api_key"
API_SECRET = "your_api_secret"
ACCESS_TOKEN = "your_access_token"
ACCESS_SECRET = "your_access_secret"

# Authenticate with Twitter API
auth = tweepy.OAuthHandler(API_KEY, API_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
api = tweepy.API(auth, wait_on_rate_limit=True)

def fetch_expert_tweets(username, count=10):
    """Fetch recent tweets from an expert."""
    try:
        tweets = api.user_timeline(screen_name=username, count=count, tweet_mode="extended")
        return [{"tweet": tweet.full_text, "date": tweet.created_at} for tweet in tweets]
    except Exception as e:
        print(f"Error fetching tweets: {e}")
        return []
```

2.3 LinkedIn Scraper for Professional Expertise

LinkedIn API access is restricted, so **web scraping is used where applicable.**

```
import requests
from bs4 import BeautifulSoup

def scrape_linkedin_profile(linkedin_url):
    """Extract expertise from a LinkedIn profile."""
    headers = {"User-Agent": "Mozilla/5.0"}
    response = requests.get(linkedin_url, headers=headers)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, "html.parser")
        expertise_section = soup.find("div", {"class": "experience-section"})
        expertise = expertise_section.get_text(strip=True) if expertise_section else "No expertise 
found"
        return expertise
    else:
        return "Failed to fetch profile"
```

2.4 Updating Digital Twin with Social Media Data

We integrate **Twitter and LinkedIn data into the Digital Twin retrieval system.**

```
from fastapi import FastAPI
from pydantic import BaseModel
import sqlite3
import json

app = FastAPI()
conn = sqlite3.connect("digital_twins.db", check_same_thread=False)
cursor = conn.cursor()

class SocialMediaData(BaseModel):
    twin_id: str
    twitter_handle: str
    linkedin_url: str

@app.post("/update_twin_with_social/")
def update_twin_social(data: SocialMediaData):
    """Fetches social media data and updates the digital twin."""
    tweets = fetch_expert_tweets(data.twitter_handle)
    linkedin_expertise = scrape_linkedin_profile(data.linkedin_url)

    social_content = {"tweets": tweets, "linkedin_expertise": linkedin_expertise}

    cursor.execute("UPDATE digital_twins SET expertise=? WHERE id=?",
(json.dumps(social_content), data.twin_id))
    conn.commit()

    return {"message": "Digital Twin updated with social media data", "data": social_content}
```

3. AI-Based Expertise Extraction from Social Media

To **extract expertise from social media, we use NLP with BERT.**

```
from transformers import pipeline

nlp_extractor = pipeline("ner", model="dbmdz/bert-large-cased-finetuned-conll03-english")

def extract_expertise_from_tweets(tweets):
    """Extracts expertise topics from tweets using NLP."""
    all_text = " ".join([tweet["tweet"] for tweet in tweets])
    entities = nlp_extractor(all_text)
    keywords = [ent["word"] for ent in entities if ent["entity"] in ["B-MISC", "I-MISC"]]
    return list(set(keywords))
```

4. Enhancements & Future Work

- **Semantic Search:** Use embeddings to improve expertise retrieval.
- **Predictive Analytics:** Detect **emerging fields** from social media trends.
- **Sentiment Analysis:** Evaluate **expert credibility** based on public perception.

Would you like me to integrate **predictive expertise trends** using AI?