Model and Database Plan for Article Similarity and Recommendation System

This document tries to outline the architecture and database plan for building a high-performance system that extracts the semantic meaning of articles, searches through a database of over 24,000 articles, and identifies related articles in real-time. The proposed solution aims to provide a fast retrieval of the articles.

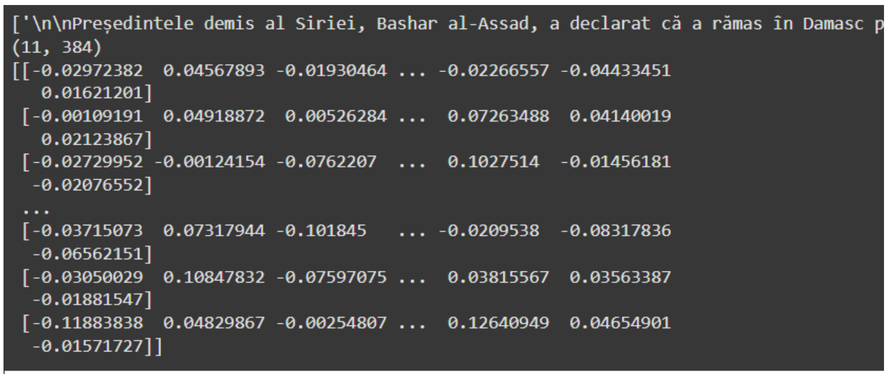
Objectives

1. **Extract Semantic Meaning:** Use an AI model to convert articles into vector embeddings that capture their semantic meaning.
2. **Efficient Search:** Implement a fast similarity search mechanism to find related articles from a large dataset.
3. **Scalability:** Ensure the system can handle incremental updates and scale with the growth of the article database.
4. **Real-Time Performance:** Provide sub-second response times for user queries.

System Components

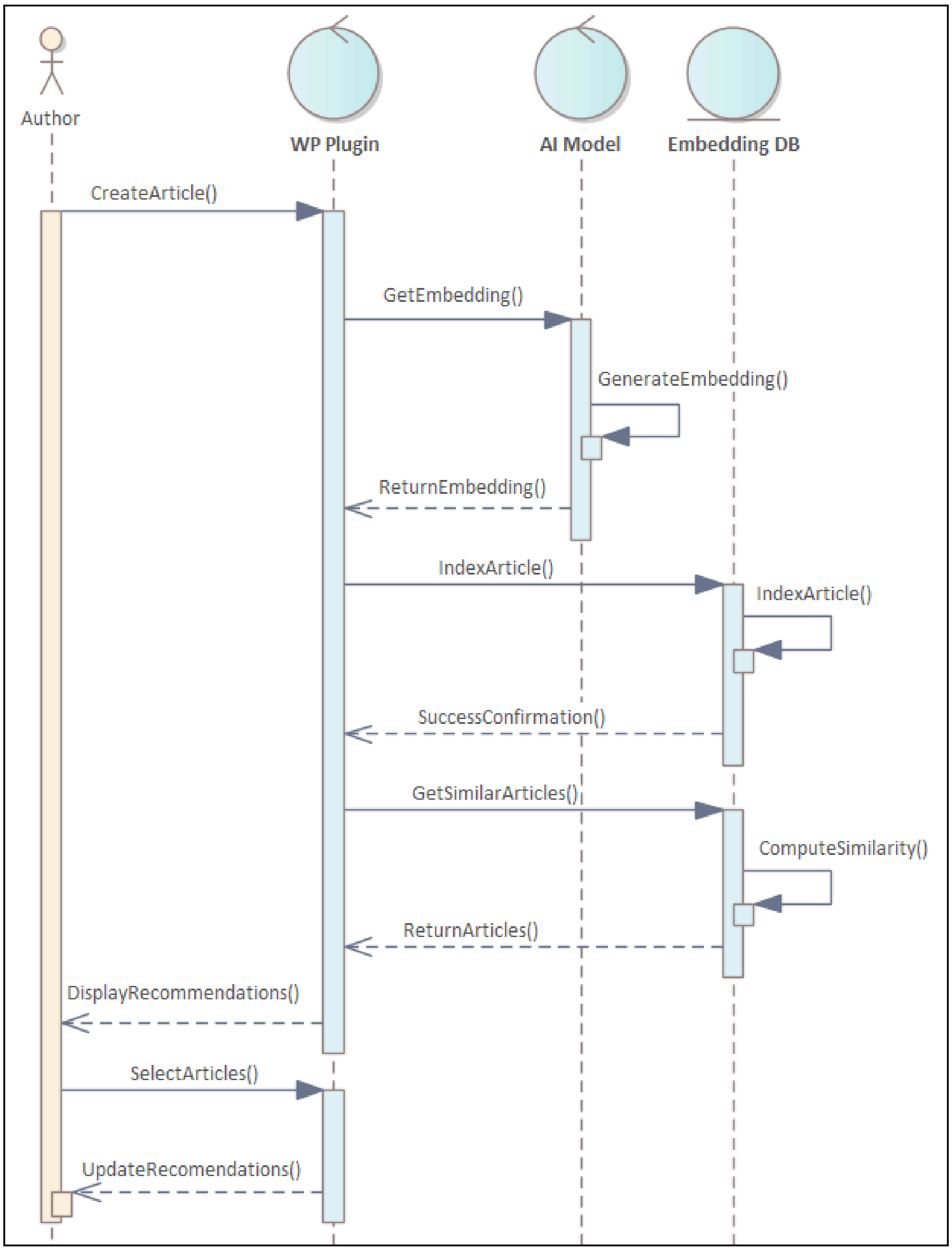
1. Embedding Model

- **Purpose:** Convert articles into high-dimensional vector embeddings.
- **Model:**
  - **Sentence-BERT (SBERT):** SBERT, is a pre-trained transformer model optimized for sentence embeddings.
  - **Example:** Taking a recent article from the publication, the whole content can be vectorized as such:



```
['\n\nPreședintele demis al Siriei, Bashar al-Assad, a declarat că a rămas în Damasc p
(11, 384)
[[-0.02972382  0.04567893 -0.01930464 ... -0.02266557 -0.04433451
   0.01621201]
 [-0.00109191  0.04918872  0.00526284 ...  0.07263488  0.04140019
   0.02123867]
 [-0.02729952 -0.00124154 -0.0762207  ...  0.1027514  -0.01456181
  -0.02076552]
 ...
 [-0.03715073  0.07317944 -0.101845   ... -0.0209538  -0.08317836
  -0.06562151]
 [-0.03050029  0.10847832 -0.07597075 ...  0.03815567  0.03563387
  -0.01881547]
 [-0.11883838  0.04829867 -0.00254807 ...  0.12640949  0.04654901
  -0.01571727]]
```

- **Workflow:**
  - Batch process all existing articles to generate and store embeddings.
  - Generate embeddings for new articles dynamically as they are published, and update the database.



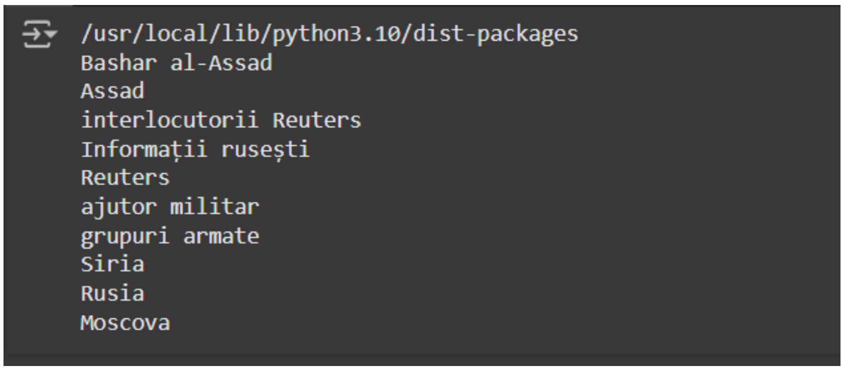
2. Vector Database (with metadata link)

- **Purpose:** Store and retrieve article embeddings efficiently.
- **Tools:**
  - **Elasticsearch:** Stores data in JSON formats, making it easier to store vector representations together with metadata in the same place.

- **Features:**
    - Supports **approximate nearest neighbor (ANN)** searches for vectors (e.g., embeddings) through plugins like **Elasticsearch KNN** or native vector fields.
    - Stores metadata (e.g., titles, URLs) alongside embeddings.
    - Could also store summaries if summarization shall be implemented down the line.

3. Alternative Models

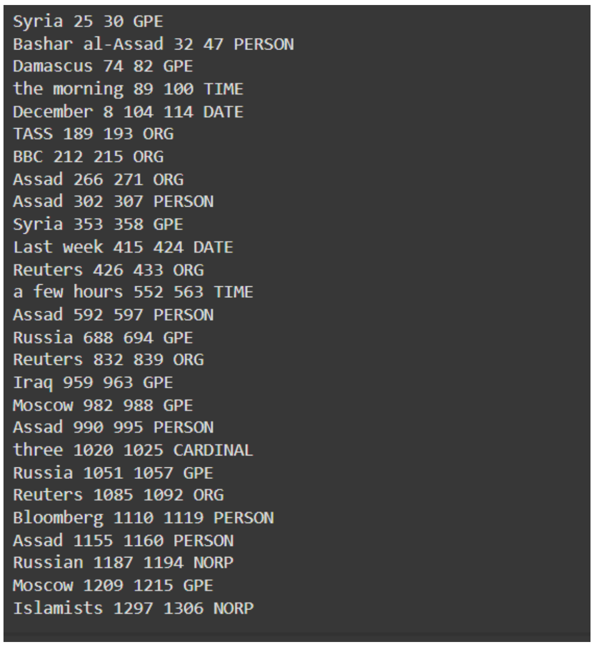
- If the vectorization approach fails, other models with alternative approaches can be employed, such as keyword extraction using **TestRank**:



```
/usr/local/lib/python3.10/dist-packages
Bashar al-Assad
Assad
interlocutorii Reuters
Informații rusești
Reuters
ajutor militar
grupuri armate
Siria
Rusia
Moscova
```

Here the TestRank algorithm analyzes the same article as before, but instead of vectorizing the text, it extracts the most important words.

- Similar to **TestRank,** to extract information, the **Named Entity Recognition (NER)** model can be used:



```
Syria 25 30 GPE
Bashar al-Assad 32 47 PERSON
Damascus 74 82 GPE
the morning 89 100 TIME
December 8 104 114 DATE
TASS 189 193 ORG
BBC 212 215 ORG
Assad 266 271 ORG
Assad 302 307 PERSON
Syria 353 358 GPE
Last week 415 424 DATE
Reuters 426 433 ORG
a few hours 552 563 TIME
Assad 592 597 PERSON
Russia 688 694 GPE
Reuters 832 839 ORG
Iraq 959 963 GPE
Moscow 982 988 GPE
Assad 990 995 PERSON
three 1020 1025 CARDINAL
Russia 1051 1057 GPE
Reuters 1085 1092 ORG
Bloomberg 1110 1119 PERSON
Assad 1155 1160 PERSON
Russian 1187 1194 NORP
Moscow 1209 1215 GPE
Islamists 1297 1306 NORP
```

In this case, **NER** identifies and classifies entities within a text into predefined categories such as the names of persons, organizations, locations, dates, etc.