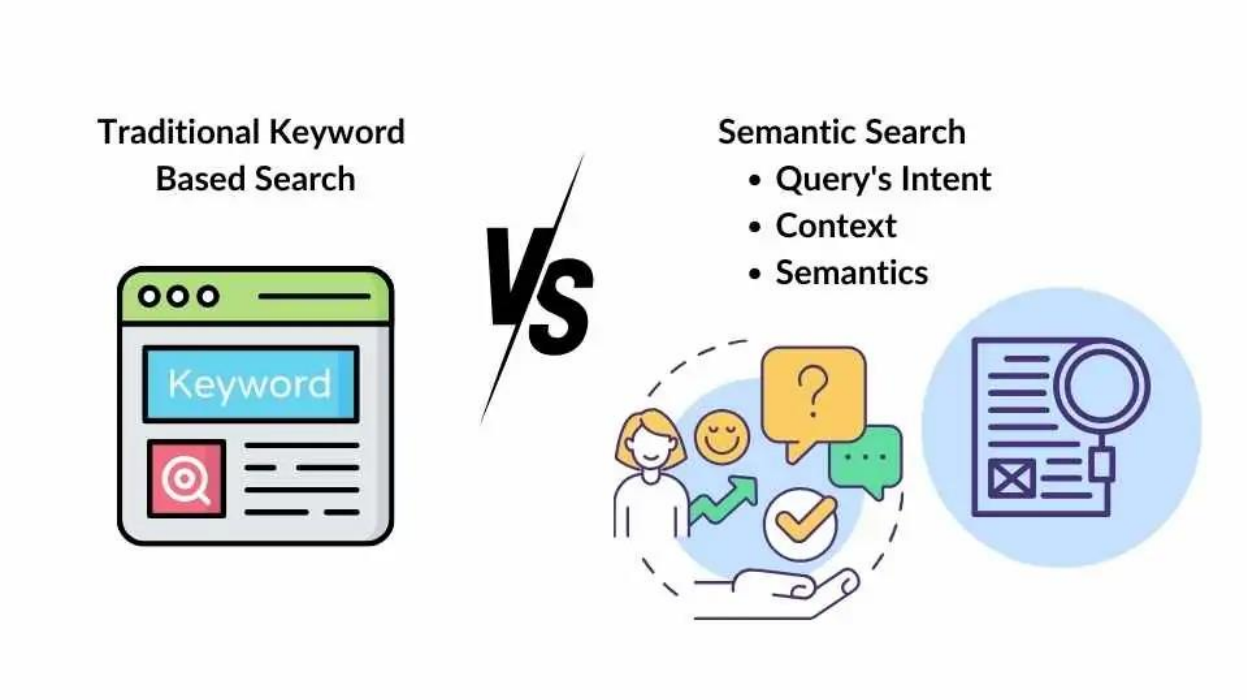
Semantic Search In Articles Using NLP



Contents

Figures

1.0 Introduction

Semantic search in articles aims to enhance information retrieval by understanding the context and meaning of words. This project focuses on developing a pipeline for searching specific words in English articles and extracting hot keywords. The objective is to create an efficient and accurate semantic search and keyword extraction system, optimizing the process and exploring various feature extraction techniques

2.0 Data Description

Medium is a prominent platform for disseminating knowledge across a wide range of topics. It is particularly renowned for articles on Machine Learning, Artificial Intelligence, and Data Science. This dataset comprises a collection of approximately 350 articles within these domains.

2.1 Content:

The dataset includes the following elements for each article:

- **Title:** The headline of the article.
- **Claps:** The number of claps (or likes) the article has received.
- **Link:** A URL linking to the full article on Medium.
- **Reading Time:** An estimate of the time required to read the article.

2.2 Data preprocessing

To ensure accurate and efficient extraction of search keys from the articles, the following preprocessing steps were implemented:

1. **Tokenize Article**
    - The text was divided into individual tokens or words, facilitating the analysis of each component separately.
2. **Normalize Words**
    - Words were converted to a uniform case (either lowercase or uppercase) to standardize the text and avoid discrepancies due to case differences.
3. **Remove Punctuation**

- Punctuation marks were removed from the text to concentrate solely on the words, eliminating distractions and potential noise.

4. **Remove Stop Words**

- Commonly used words (e.g., "the", "and") that do not contribute substantial meaning were excluded, focusing the analysis on more significant terms.

5. **Lemmatize the Tokens**

- Tokens were reduced to their base or root forms, which helps in consolidating variations of the same word (e.g., "running" to "run") and improving the consistency of the text.

These preprocessing steps enhance the clarity and relevance of the text data, setting the stage for more effective search key extraction and analysis.

2.3 Data Statistics

The dataset originally contains a total of 192,368 articles. For the purpose of testing the pipeline, a subset of 200 articles was selected. These 200 articles were further split into sentences, from which 1,000 sentences were chosen to create pairs for training a sentence transformer.

The statistics for the pairs created are as follows:

- **Training Set:** 400,400 pairs
- **Validation Set:** 90,090 pairs
- **Testing Set:** 10,010 pairs

To assess the similarity between pairs, the dataset utilizes Term Frequency-Inverse Document Frequency (TF-IDF) and cosine similarity matrices. These measures help in evaluating and refining the quality of the sentence pairs for the sentence transformer model.

3.0 Fine tune Sentence transformer model

The **all-MiniLM-L6-v2** model This is a sentence-transformers model: It maps sentences & paragraphs to a 768 dimensional dense vector space and can be used for tasks like clustering or semantic search. was fine-tuned using the Sentence Transformer library on a dataset comprising **500,500** pairs of sentences. During this fine-tuning process, the similarity scores between sentence pairs were used as the target evaluation metric. Cosine similarity was employed as the evaluator to measure and optimize the model's performance in capturing the semantic similarity between sentences.

Results:

Training Results:

Step	Training Loss	Validation Loss	Pearson Cosine	Spearman Cosine	Pearson Manhattan	Spearman Manhattan	Pearson Euclidean	Spearman Euclidean	Pearson Dot	Spearman Dot	Pearson Max	Spearman Max
800	0.004100	0.004052	0.772340	0.291697	0.880912	0.275734	0.891707	0.291697	0.772340	0.291699	0.891707	0.291699
1600	0.003700	0.003741	0.786892	0.302689	0.887709	0.286841	0.899778	0.302689	0.786892	0.302689	0.899778	0.302689
2400	0.003600	0.003562	0.794998	0.306467	0.891635	0.290215	0.903894	0.306467	0.794998	0.306466	0.903894	0.306467
3200	0.003700	0.003410	0.803787	0.317407	0.896580	0.301854	0.909131	0.317408	0.803787	0.317407	0.909131	0.317408
4000	0.003400	0.003342	0.806801	0.323789	0.897954	0.307302	0.910576	0.323789	0.806801	0.323789	0.910576	0.323789

Testing Results:

1. **Pearson Correlation Coefficient:**
    - **Cosine Similarity**: 0.767
    - **Manhattan Distance**: 0.878
    - **Euclidean Distance**: 0.895
    - **Dot Product**: 0.767
    - **Max Similarity**: 0.895
2. **Spearman Rank Correlation Coefficient:**
    - **Cosine Similarity**: 0.335
    - **Manhattan Distance**: 0.330
    - **Euclidean Distance**: 0.335
    - **Dot Product**: 0.335
    - **Max Similarity**: 0.335

4.0 Key Search pipeline

4.1 Data Preprocessing:

- **Tokenization:** The text is split into individual tokens or words.
- **Lowercasing:** All text is converted to lowercase to ensure uniformity.
- **Removal of Stop Words:** Common, non-informative words are removed.
- **Punctuation Removal:** Punctuation marks are eliminated to focus on meaningful words.

4.2 Vector Database:

- **Embedding Generation:** A vector database is created for the article texts using the FAISS library and the all-mpnet-base-v2 model from the Sentence Transformers library. This involves generating embeddings for each article using the Hugging Face API.

4.3 Search Process:

- **Query Embedding:** An embedding is generated for the search query (key).
- **Article Matching:** The most relevant articles are identified by finding the top N matches based on cosine similarity within the FAISS database.

4.4 Generate N-Hot Keywords:

- **Keyword Extraction:** From the most relevant articles, a TF-IDF/Count Vectorizer is used to extract the top N hot keywords.

4.5 Visualization:

- **Word Cloud:** A visual representation of the N-hot keywords is created using a word cloud to highlight the most significant terms.

4.6 Results:

- **Search Query:** "machine learning"
- **Top Matched Article:**
    - *"A machine learning method uncovered a hidden clue in people's language predictive of the later manifestation of psychosis: the frequent use of words associated with sound. A paper published by the journal npj Schizophrenia released the findings by scientists from Emory University and Harvard University. Hidden details The*

researchers developed a new machine-learning methodology to more precisely quantify the semantic richness of people's conversational language (a known indicator for psychosis). Their results indicated that automated analysis of the two language variables (more frequent use of words associated with sound and speaking with low semantic density, or vagueness) can predict if an at-risk person will later develop psychosis with an impressive 93 percent accuracy."

- **TF-IDF Top matched words:**

	word	tfidf
47	ai	0.159174
256	data	0.122734
1102	weather	0.104149
571	learning	0.102854
603	machine	0.099461
1062	using	0.077359
82	artificial	0.076220
530	intelligence	0.076220
639	min	0.074468
601	lstm	0.074468
684	norris	0.074468
910	series	0.074468
977	stauffer	0.074468
1022	time	0.070330
608	make	0.069066
141	business	0.068558
250	customer	0.064225
778	prediction	0.063718
1013	technology	0.062085
406	forecast	0.060297
809	psychosis	0.058661
167	claim	0.058083
1059	use	0.055799
560	language	0.055678
733	people	0.049939
1111	wind	0.049334
863	researcher	0.044740
915	severe	0.043852
477	human	0.043560
69	application	0.043002

Figure 1 Top 30 Words in machine learning articles TF-IDF

- **Word Cloud Visualization TF-IDF:**

Figure 2 Word Cloud TFIDF

- **Count vectorizer:**

ai	55
data	34
learning	25
artificial	25
artificial intelligence	25
intelligence	25
machine	24
machine learning	22
weather	19
claim	16
customer	16
technology	14
research	14
science	14
use	13
business	12
department	12
researcher	12
university	11
forecast	11

Figure 3 TOP-20 Words in machine learning articles Count vectorizer

- **Word Cloud Visualization Count visualizer:**

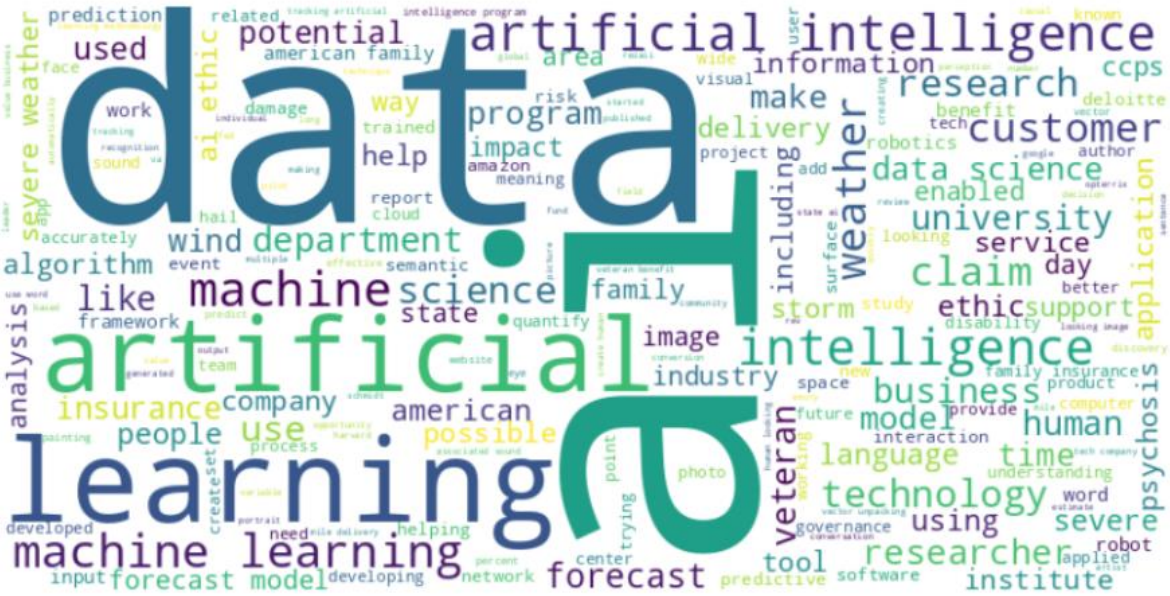


Figure 4 Word Cloud Count vectorizer

6.0 Tools and Resources

6.1 Tools:

1. **Python:** The main programming language used for developing the pipeline.
2. **Jupyter Notebook:** For interactive development, visualization, and code documentation.
3. **Sentence Transformers:** Used to generate sentence embeddings and train models:
4. **Pandas:** For data manipulation and processing with DataFrames.
5. **NLTK (Natural Language Toolkit):** For text processing tasks
6. **Scikit-learn**: TfidfVectorizer, CountVectorizer: For converting text to numerical features, train_test_split: For splitting data into training and testing sets.
7. **FAISS (Facebook AI Similarity Search)**: For efficient similarity search and clustering of vectors.
8. **Datasets:** For managing and processing datasets.
9. **NumPy**: For numerical computations and handling matrix operations.
10. **pandas**: For data manipulation and analysis, particularly for handling the datasets.

6.2 Resource

External Resources Used: all-mpnet-base-v2 This is a sentence-transformers model: It maps sentences & paragraphs to a 768 dimensional dense vector space and can be used for tasks like clustering or semantic search

7.0 Biggest Challenge Faced

7.1 Finding the Best Embeddings Model and Dataset:

- Identifying the most effective embeddings model was a significant challenge. It involved evaluating various models to determine which one best suited the needs of the project. Additionally, selecting an appropriate dataset that aligns with the model's requirements and the project's goals was crucial for achieving optimal performance.

8.0 Lessons Learned

8.1 Transformer Embeddings:

- Gained a deep understanding of how transformer embeddings work, including their ability to capture contextual relationships between words. This knowledge highlighted the advantages of transformer-based models over traditional embeddings in capturing nuanced meanings and context.

8.2 Differences in Embeddings:

- Learned about the distinctions between transformer embeddings and other types of embeddings. Transformers offer a more sophisticated approach to representing text compared to older methods, providing richer and more accurate representations.

8.3 Vector Database:

- Acquired insights into using vector databases for efficient similarity search and retrieval. Understanding how to leverage vector databases like FAISS to store and query embeddings effectively has been valuable in optimizing search processes.

8.4 Training Sentence Transformers:

- Developed skills in training Sentence Transformers, including creating effective training pairs, defining loss functions, and fine-tuning models to improve their performance on specific tasks.