SPORTPLUS REPORT

Exalte Wembonyama & Maxime mizero

2025

Table of Contents

[INTRODUCTION 2](#_Toc194343992)

[Dataset 2](#_Toc194343993)

[Method 3](#_Toc194343994)

[RESULTS 4](#_Toc194343995)

[REPRODUCIBILITY 5](#_Toc194343996)

[CHALLENGES AND SOLUTIONS 6](#_Toc194343997)

[Discussion and Future Work 6](#_Toc194343998)

Abstract

This project presents a sports news Question Answering (QA) system that uses a retriever-reader pipeline powered by Haystack and a pre-trained RoBERTa model. The system answers natural language questions based on recent sports news articles collected from Yahoo Sports. It leverages the InMemoryBM25Retriever for semantic search and an extractive reader to identify answer spans. We achieved an average accuracy score of 0.80 on a custom evaluation set.

# INTRODUCTION

The rise in digital sports content has made it increasingly difficult for fans to quickly extract specific facts from large volumes of articles. Our research aims to build an automated QA system capable of answering natural language sports questions by retrieving and extracting answers from real news articles. The core research question is: **Can a retriever-reader pipeline accurately respond to sports questions based on recent news articles?** This work is motivated by the growing need for intelligent sports assistants. Our approach uses a BM25 retriever with a RoBERTa-based extractive QA model.

# Dataset

The dataset used in this project was constructed from publicly available articles published on Yahoo Sports, focusing primarily on NBA-related content. Yahoo Sports offers a rich trove of up-to-date reporting, opinion pieces, and statistical analysis on players, games, and broader league events. Our goal was to create a compact yet diverse corpus of sports-related articles that would serve as the knowledge base for our QA system.

We successfully scraped and cleaned 120 articles, each representing a distinct document within our QA pipeline. These articles include coverage of recent games, trade rumors, athlete interviews, and league developments. To prepare the data for use in our pipeline, we performed an extensive preprocessing pipeline:

**Preprocessing Steps**

* **Emoji & Special Character Removal**: Emojis and other non-textual symbols common in sports reporting (especially from social media embeds) were stripped out.
* **Byline & Timestamp Removal**: Patterns like “By John Smith · March 2025 · 5 min read” were removed using regex to eliminate noise and metadata.
* **Stopword and Junk Pattern Cleaning**: Redundant line breaks, social media tags (e.g., pic.twitter.com), and branding phrases were removed.
* **HTML Tags & Formatting**: Although not heavily present, basic HTML or formatting artifacts were cleaned as a precaution.
* **Extraction of Key Metadata**: We extracted:
  + **Title**: Used in visualization and metadata.
  + **Full Description**: Used for QA model ingestion.
  + **Short Description**: The first meaningful sentence, useful for interface summarization.
  + **Category**: Most frequently “NBA” or “Fantasy Basketball.”
  + **Source URL**: Tracked for attribution and retrieval transparency.

The cleaned articles were serialized into a JSON file, processed\_articles.json, and later converted to Haystack-compatible Document objects. These were written into an InMemoryDocumentStore for use in our Retriever-Reader QA pipeline.

**Dataset Statistics**

Based on our cleaning process and analysis:

* **Total Articles**: 120
* **Average Word Count per Article**: ~656 words (mean), with a long-tail distribution up to 4000+ words.
* **Average Title Length**: ~12.6 words.
* **Top Keywords in Titles**: "Draft", "Scouting", "NBA", "Report", "Lakers", "Return", "Playoff".
* **Most Mentioned Athletes**: LeBron James, Luka Dončić, Damian Lillard, Jimmy Butler, Connor Show, Anthony Davis, Cooper Flagg.

**Visual Insights**

A close-up of a graph

AI-generated content may be incorrect.

The first image contains a word cloud showing the most common words in article titles (left) and a bar chart ranking the top 10 most frequently mentioned players (right). LeBron James is the clear leader in mentions, indicating heavy coverage of his activities. Words like “Draft,” “Scouting,” and “NBA” dominate the title space, showing that a lot of content is focused on pre-season scouting and player movement.

A screenshot of a graph

AI-generated content may be incorrect.

The second image provides a quantitative breakdown of content and title lengths. The distribution of article content length is right-skewed with a peak around 500-800 words, while title lengths show a tighter distribution centered around 10-13 words. This suggests consistent editorial formatting but a variety of article depths and reporting styles.

**Final Output Format**

Each cleaned article was saved as a structured dictionary with the following keys:

json

A black screen with white text

AI-generated content may be incorrect.

The articles were stored in processed\_articles.json and later loaded into the Haystack InMemoryDocumentStore via code for inference and evaluation.

## Method

We used the Haystack 2.x architecture:

**Components:**

* **Document Store**: InMemoryDocumentStore
* **Retriever**: BM25-based keyword retriever (InMemoryBM25Retriever)
* **Reader**: Transformer-based extractive QA model deepset/roberta-base-squad2

**Key Functions:**

* Texts are ingested, cleaned, and converted to Document objects
* BM25 retrieves top relevant passages
* RoBERTa extracts answer spans with confidence scores
* Filter low-confidence predictions (threshold: 0.7)

A black screen with white text

AI-generated content may be incorrect.

## RESULTS

We tested the system with 10 domain specific questions, With a sample score of 1 (correct), 0.5 (partially correct), 0 (incorrect)

|  |  |  |  |
| --- | --- | --- | --- |
| **Question** | **Answer** | **Confidence** | **Correctness** |
| What happened between LeBron and Stephen A. Smith? | verbal confrontation | 0.67 | 1 |
| What did LeBron say on McAfee’s show? | He got personal | 0.74 | 1 |
| Who left the $130k rental in disrepair? | Ja Morant | 0.88 | 1 |
| Who was out indefinitely? | Damian Lillard | 0.81 | 1 |
| What was said about Bronny James? | LeBron clapped back | 0.66 | 0.5 |
| What caused the Bulls comeback? | Turnover by LeBron | 0.72 | 1 |
| What is the weather today? | madness | 0.55 | 0 |
| Who got a win over the Lakers? | Clippers | 0.78 | 1 |

We scored an accuracy of 80%

# REPRODUCIBILITY

Our project was developed on a Windows 11 machine equipped with an Intel Core i7 processor and 16GB of RAM. However, the system is also compatible with other operating systems including macOS and Linux. The Python environment was managed using Anaconda, and all development was conducted using Jupyter Notebook. The project was implemented in Python 3.10, and the required dependencies include haystack-ai, transformers, torch, pandas, numpy, sentence-transformers, and streamlit. These packages can be installed using pip to replicate the environment setup.

To reproduce our results, a user should begin by acquiring the raw Yahoo Sports dataset (e.g., yahoo\_sports\_articles\_2025-03-28\_21-48-27-197.json). The first step is to run the data cleaning and preparation notebook (cleaning\_and\_pipeline.ipynb), which removes noise such as emojis, journalist bylines, and irrelevant metadata, and generates two cleaned files: cleaned\_articles.json and processed\_articles.json. These files are then used to create a DocumentStore for the retriever component in the Haystack pipeline.

Next, users can build the full retriever-reader pipeline using the BM25 retriever and an extractive reader model (deepset/roberta-base-squad2). The pipeline is instantiated and executed through a simple Haystack Pipeline object where the retriever is connected to the reader. Once the pipeline is built, users can run predefined test queries to observe the system’s outputs and conduct evaluations.

For evaluation purposes, we provided a set of 10 domain-relevant questions. The system returns answers along with confidence scores, and these were manually assessed for correctness using a scoring system (1 = correct, 0.5 = partially correct, 0 = incorrect). The results, including questions, system answers, confidence scores, and human-assigned correctness scores, were compiled into a table. The average accuracy score was computed as a summary metric of system performance.

All source code and scripts are provided, including the data cleaning functions, pipeline construction, evaluation logic, and a Streamlit web application (app\_streamlit.py) that enables users to interact with the QA model through a friendly interface. In addition, we provide a JSON file (models/documents.json) containing the serialized Haystack Document objects for easy reuse, along with intermediate files such as cleaned\_articles.json and processed\_articles.json.

# CHALLENGES AND SOLUTIONS

During the development,we encountered several technical challenges and addressed them with targeted solutions. One significant issue was that Haystack 2.x no longer supports the save\_to\_yaml() method, which limited our ability to serialize the entire pipeline. To resolve this, we opted to serialize only the processed documents using JSON, allowing us to reload them into memory during evaluation or deployment. Another challenge involved the system returning irrelevant or out-of-domain answers when asked general knowledge questions. We implemented a confidence threshold filter to suppress low-confidence outputs and display a fallback message such as “Sorry, I couldn’t find any answer related to that topic in my knowledge base.” Finally, we faced a serialization error using dill when attempting to save the pipeline due to the presence of unpicklable objects. To overcome this, we abandoned pickling and adopted a safer approach by saving document-level data only, ensuring reproducibility without serialization errors.leaned\_articles.json and processed\_articles.json.

# Discussion and Future Work

The system demonstrated strong performance on domain-specific queries, particularly when relevant information existed in the article corpus. The combination of a BM25 retriever and an extractive reader like RoBERTa allowed the system to provide accurate, context-aware answers for many sports-related questions. However, there is room for improvement in both the retrieval and response generation phases. Future work could involve integrating an abstractive reader such as T5, which would allow the system to generate more natural, paraphrased responses rather than simply extracting text spans. Additionally, switching from the in-memory document store to more scalable solutions like ElasticSearch or FAISS could enhance retrieval efficiency, especially with larger datasets. Expanding the dataset to include more teams, sports leagues, and historical seasons would also broaden the system’s knowledge base. Finally, incorporating user feedback mechanisms to adjust ranking scores or retrain the model over time could help the system continuously improve and stay up-to-date with evolving sports narratives.

# REFERENCES

* Haystack Docs: https://docs.haystack.deepset.ai
* HuggingFace Models: https://huggingface.co/models
* Yahoo Sports Dataset: Custom scraped using Python scripts