Explainable AI Methods for the Retrieval Phase of Retrieval-Augmented Generation

Carina Chen
Department of Information Systems
University of Maryland, Baltimore County
dk49490@umbc.edu

Abstract

This paper describes two explainable AI (XAI) methods for explaining the retrieval step in Retrieval Augmented Generation (RAG). The first method, prompt key token extraction, uses KeyBERT to discover and rank keywords from user prompts. The second method, fine-grained explanation, uses BERTopic to determine the semantic similarity between keywords and the retrieved content, illustrating how articles connect to prompts.

1. Introduction

Retrieval-Augmented Generation (RAG) is a technique for enhancing the output of large language models(LLMs). Unlike typical LLMs, which generate output from an established training database, RAG expands LLM capabilities to more particular fields by combining document storage and retrieval from the most up-to-date external resources. LLMs' output explanation has been a rising topic in the field, since LLMs continue to confront issues like as overconfidence, bias, and hallucination. Explainable AI (XAI) is a branch of study that aims to make AI systems understandable and explainable. This paper aims to design some XAI methods to explain the RAG retrieval step. The XAI method for explaining RAG can be divided into three steps: 1. retrieval step explanation 2. generation step explanation 3. impact of the retrieval step on the generation step.

2. Literature Review

Many researchers are researching the combined use of XAI and RAG; for example, Liu and Tsai (2024) integrated XAI and RAG into a clinical decision support system for nursing assistants. XAI provided transparent suggestions and explanations to increase trust in AI-driven decisions, whereas RAG pulled real-time patient data, medical guidelines, and historical case reports to enhance recommendations. Aside from that, Datta and Dickerson (2023) claim that there was no ground truth for XAI explanations, and that explanations for XAI decisions were subjective and context-dependent. Thus, using RAG to assist XAI in providing a better explanation was yet another approach to merging RAG and XAI. As an illustration, in Devansh Guttikonda et al.'s (2025) study, RAG was applied to XAI to improve its explainability. A novel strategy was

Prompt	Keyword Extraction	What to retrieve
“ What are the symptoms of flu?”	What, symptoms, flu	The articles related to the flu symptoms

Figure 1: Prompt Keyword Color Coding

developed that utilized RAG to improve the XAI method in many criteria, such as focus, scale, data, bias, and usage, hence making machine learning outputs more interpretable. However, although the fact that RAG and XAI work together or RAG is used on XAI, no research studies have been conducted to investigate the opposite direction, XAI on RAG.

3. Methodology

3.1 Prompt Key Token Extraction

The retrieval step is primarily reliant on the prompts that users submit; these prompts function as a key to open the retrieval step's door. Sometimes diverse wording represents the same meaning; nevertheless, similar wording with a minor variance may also imply another meaning. Therefore, comprehending and knowing the weight of the words in the prompts is critical. This Prompts Key Token Extraction method extracts the prompt's keyword(s) and seeks to distinguish between comparable prompts with various meanings and different wordings that have the same meaning. The method's fundamental premise is simple: when users provide a prompt, the method extracts the main keyword(s) from the prompt. Then rank each keyword according to its weight. As seen in Figure 1, each word is color-coded. The colors represent the token's impact: pink means general/low impact, green means medium impact, and yellow means extreme impact. RAG will retrieve documents based on the weights of the key tokens.

KeyBert is a tool that extracts keywords and keyphrases from text via BERT embeddings, as shown in Figure 2. It examines the semantic similarity of documents and key tokens to determine the most relevant phrases. This tool can do the majority of the tasks associated with the key token extraction approach, and the installation and sample code for KeyBert are provided in Appendix A. Four experimental cases were conducted on KeyBert. In each case, a code containing the prompt was inputted into KeyBert, which returned the keywords along with their relevance scores.

3.2 Fine-Grained Explanation

After extracting the keywords from the prompts, the following step is to get the relevant articles based on the keywords. Although RAG is an improved technique that increases LLMs' performance by including an external knowledge base, hallucination remains an unsolved problem for LLMs. To accomplish this step, we need to explain how the articles connect to the

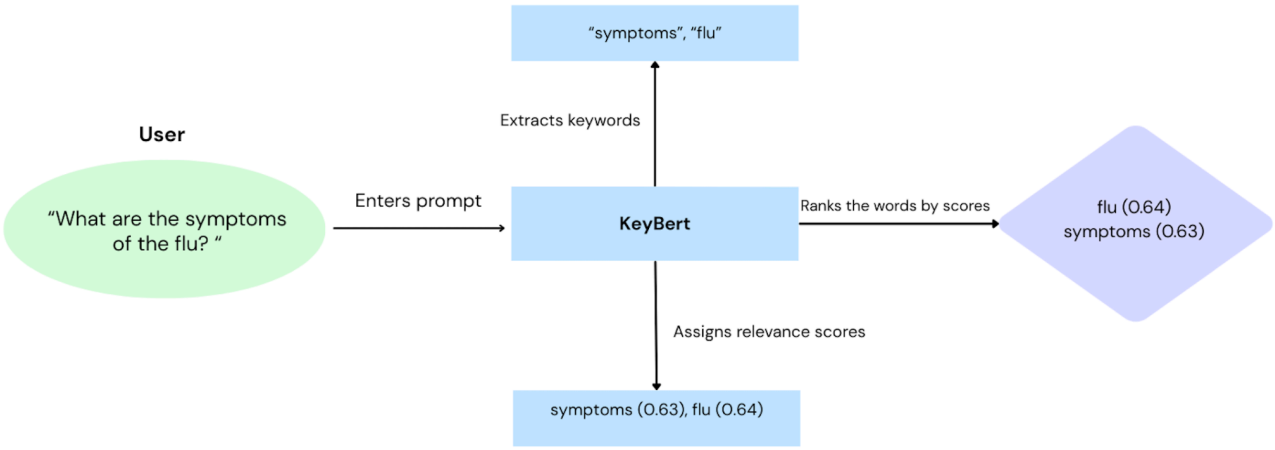

Figure 2: KeyBert Method Example

prompts. Similar to the previous method, there is another BERT-derived tool called BERTopic, which is a sentence transformer that uses BERT embeddings and clustering to extract interpretable topics from a collection of documents. Its implementation requires a sentence transformer model. BERTopic is more complicated to use than KeyBert. I designed a system that imported KeyBert; Figure 2 shows a code sample; a complete version of the system code is available in Appendix B. And Figure 4 is a simple illustration of the system.

In this study, I used all-MiniLM-L6-v2 as the sentence transformer model for converting articles into semantic embeddings. I initially gathered papers relevant to the prompt and then provided a list of pre-defined topics. The system would download the articles from the internet and save them in text format, after which BERTopic would encode the embeddings of both the articles and topics. Following that, a function named sentence_transformers.util computed the cosine similarity of the article and topic embeddings. The cosine similarity scores represented the final similarity scores.

```
# Load the SentenceTransformer model
model = SentenceTransformer('all-MiniLM-L6-v2')
# Encode documents and topics
doc_embeddings = model.encode(documents, convert_to_tensor=True)
topic_embeddings = model.encode(topics, convert_to_tensor=True)
# Compute similarity
cosine_scores = util.cos_sim(doc_embeddings, topic_embeddings)
```

Figure 3: BERTopic Sample Code

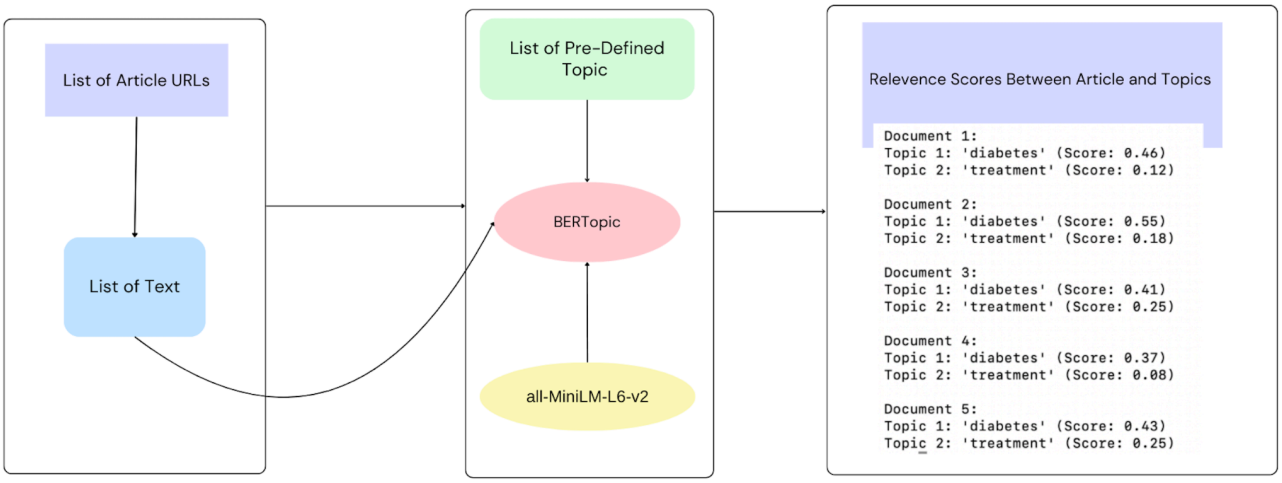

Figure 4: BERTopic System Illustration

"flu"; KeyBert was able to extract these and produce a relatively good relevance score. In case 2, the question featured another keyword, "causes," which KeyBert was also able to detect with its relevance score. That being said, KeyBert can tell the difference between similar prompts. In cases 3 and 4, the prompts "What are the treatments for flu symptoms?" and "How to cure the flu symptoms?" have the same meaning but use different terms. As a result, KeyBert effectively extracted all of the keywords such as "treatment," "cure," and "flu" and assigned similar relevance scores to the words "treatment" (0.55) and "cure" (0.44) because their meanings are similar based on context.

4.2 BERTopic

Two experimental trials were conducted in this system; the computed similarity scores for each case are shown in Tables 2 and 3. In the first case, five depression-related articles were collected, with a pre-defined topic list of ["depression", "symptoms", "causes"]. Table 2 illustrates the

Case	Input Query	Top Keywords (with Scores)
1	What are the symptoms of the flu?	flu (0.64), symptoms (0.63)
2	What are the causes of flu symptoms?	flu (0.67), symptoms (0.57), causes (0.36)
3	What are the treatments for flu symptoms?	flu (0.6186), treatments (0.5551), symptoms (0.4289)
4	How to cure the flu symptoms?	flu (0.5872), cure (0.4415), symptoms (0.3663)

Table 1: Extracted Keywords and Scores for Sample KeyBert Input Queries

Document URL	Depression	Symptoms	Causes
CDC - Depression	0.46	0.32	0.06
MedlinePlus - Depression	0.55	0.37	0.12
MedlinePlus - Medical Encyclopedia	0.4	0.08	-0.06
CDC - Tobacco & Depression	0.63	0.32	0.1
PubMed Article	0.55	0.27	0.1

Table 2: Cosine Similarity Scores — Topics: Depression, Symptoms, Causes

Document URL	Diabetes	Treatment
NIDDK - Diabetes Treatments	0.46	0.12
MedlinePlus - Diabetes Care	0.41	0.25
PubMed - Diabetes Research Article	0.37	0.08
CDC - Public Health & Diabetes	0.43	0.25

Table 3: Cosine Similarity Scores — Topics: Diabetes, Treatment

findings of the first case; all of the articles have the highest similarity scores on the topic "depression", which is understandable given that they are all about depression. Except for the third article, "MedlinePlus - Medical Encyclopedia" obtained the lowest scores for all topics, including negative scores for the topic "causes"; the remaining articles received similar scores within the 0.05 to 0.2 range.

Table 3 shows the results of the second case, in which a pre-defined topic list was ["diabetes", "treatment"], and four articles discussing diabetes were collected. All of the articles earned around 0.4 similarity scores on the topic "diabetes," and the second and fourth articles both scored 0.25 on the topic "treatment," but the third article only scored 0.08. In other words, BERTopic was able to determine that all of the articles were diabetes-related because they all scored the same on this topic. The second and fourth articles discussed treatments more than the first, whereas the third article barely talked about them. However, the cosine similarity scores are still uncertain because the system was not evaluated.

5. Evaluation

This section discusses hypothetical evaluation metrics that have not yet been conducted.

5.1 Intrinsic Evaluation

According to Hasan and Ng (2014), the traditional approach to evaluating a keyphrase extraction system is to build a mapping between the keyphrases in the gold standard and those in the system output using exact match, and then score the output using evaluation metrics such as precision, recall, and F-score. To put this technique into action, we need to first recruit annotators to

manually extract keywords from the prompts and then score their relevance. After analyzing the works of all annotators, a gold standard will be established. Use this human-annotated gold standard as a reference to perform an exact match with the system-extracted keywords. The formulas for the common metric are as follows.

- Precision = matched keywords retrieved / total keywords retrieved
- Recall = matched keywords retrieved / total gold standard keywords
- F1 = harmonic mean of precision and recall

5.2 Topic Word Expansion and Matching

Zhang, Jin, and Zhou (2010) presented a method known as word expansion to describe topics more precisely; this could be a viable strategy for evaluating the cosine similarity score from BERTopic. To ensure that the article is related to a specific topic from the topic list with a high score, word expansion is suggested, including more relevant words from the topics. For example, the keyword "depression" is commonly associated with words such as "sad", "mental health", "emotional", and so on. If these words are repeated in context in the article, it may provide evidence to support the high similarity score. This strengthens the similarity and enables a fuzzier, more semantically aware matching.

6. Conclusion

This report proposes two XAI methods to explain the RAG retrieval step: prompt key token extraction and fine-grained explanation of retrieved articles. Prompt key token extraction applies a tool called KeyBert to extract keywords from prompts and rank them by relevance score. This way allows RAG to better understand the prompt and more easily retrieve related documents using the keywords. Fine-grained explanation uses BERTopic to display the similarity score between the article and keywords; the scores indicate the keywords or content the article focuses on and how the article relates to the prompt.
There are two evaluation metrics provided for evaluating the XAI methods. The first metric uses human annotation to provide a gold standard because the system does not have one, and then calculates common metric scores like as precision, recall, and F1 scores. The second metric utilized on BERTopic is word expansion, which provides more terms related to the topic word to determine whether any of the words in the article are repeated. This measure ensures that the score is supported by giving additional robustness.

7. Limitations and Future Work

All of the evaluation metrics are just envisioned on paper; due to time constraints, none of them are implemented. As a result, one of the next steps will be to implement the evaluation metrics to ensure that all of the XAI methods and tools are valid for use. I also performed a sloppy retrieval perturbation to test whether the generation would alter based on the articles I gave. The experiment was conducted using Deepseek, a recently announced open-source LLM. However, the outcome was not entirely accurate, as LLMs have a tendency to mix and randomly select source articles to generate answers. For example, one statement was removed when I removed the first of the five reference articles; however, when I removed both the first and second reference articles, the missing statement from the last case reappeared. Furthermore, after I eliminated all of the reference articles, the LLM just used the final reference article as a resource to generate the answers. As a result, the next step in this research will be to improve the perturbation method through utilizing confidence scores, as well as to ask LLM to determine which of the statements is most linked to a reference article, and then to obtain the confidence score. And then begin the perturbation method from there.

Reference

Datta, T., & Dickerson, J. (2023). *Who's Thinking? A Push for Human-Centered Evaluation of LLMs using the XAI Playbook*.

Devansh Guttikonda, Deepika Indran, Narayanan, L., Tanishka Pasarad, & J, S. B. (2025). Explainable AI: A Retrieval-Augmented Generation Based Framework for Model Interpretability. *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 948–955. https://doi.org/10.5220/0013241300003890

Liu, Y.-K., & Tsai, Y.-C. (2024). Explainable AI for Trustworthy Clinical Decision Support: A Case-Based Reasoning System for Nursing Assistants. *2024 IEEE International Conference on Big Data (BigData)*, 6502–6509. https://doi.org/10.1109/bigdata62323.2024.10825008

Kazi Saidul Hasan, & Ng, V. (2014). Automatic Keyphrase Extraction: A Survey of the State of the Art. *Meeting of the Association for Computational Linguistics*. https://doi.org/10.3115/v1/p14-1119

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1-4), 43–52. https://doi.org/10.1007/s13042-010-0001-0

Appendix

A. **KeyBERT** https://maartengr.github.io/KeyBERT/

Required Package Installation: pip install keybert sentence-transformers

```python
from keybert import KeyBERT

kw_model = KeyBERT()

sentence = "What are the treatments for flu symptoms?"

keywords = kw_model.extract_keywords(sentence)

print(keywords)
```

B. BERTopic

Required Package Installation: pip install bertopic lxml[html_clean] newspaper3k sentence-transformers torch

```python
from bertopic import BERTopic
from newspaper import Article
from sentence_transformers import SentenceTransformer, util
# List of URLs (your document links)
urls = [
    "https://www.cdc.gov/reproductive-health/depression/",
    "https://medlineplus.gov/depression.html",
    "https://medlineplus.gov/ency/article/000945.htm",

"https://www.cdc.gov/tobacco/campaign/tips/diseases/depression-anxiety.h
tml",
    "https://pubmed.ncbi.nlm.nih.gov/10318745/"
    "https://www.ncbi.nlm.nih.gov/books/NBK279282/"
]
documents = []  # This will store the article texts
for url in urls:
    try:
```

```
        article = Article(url)
        article.download()
        article.parse()
        documents.append(article.text)
    except Exception as e:
        print(f"Failed to process {url}: {e}")
topics = ["symptoms", "depression", "causes"]
# Load the SentenceTransformer model
model = SentenceTransformer('all-MiniLM-L6-v2')
# Encode documents and topics
doc_embeddings = model.encode(documents, convert_to_tensor=True)
topic_embeddings = model.encode(topics, convert_to_tensor=True)
# Compute similarity
cosine_scores = util.cos_sim(doc_embeddings, topic_embeddings)
# Define how many top topics you want to return
top_n = len(topics)

# Display all matched topics for each document
for idx, doc in enumerate(documents):
    print(f"\nDocument {idx+1}:")
    topic_scores = cosine_scores[idx]

    # Sort topics based on cosine similarity (descending order)
    top_indices = topic_scores.argsort(descending=True)

    # Display the top N topics for each document
    for i in range(top_n):
        top_idx = top_indices[i]
        print(f"Topic {i+1}: '{topics[top_idx]}' (Score:
{topic_scores[top_idx]:.2f})")
```

C. Keyword Color Coding

Prompt	Keyword Extraction	What to retrieve
“ What are the symptoms of flu?”	What, symptoms, flu	The articles related to the flu symptoms

“ What are the causes of flu symptoms?”	What, causes, flu, symptoms	The articles related to the cause of the flu
“ What are the treatments for flu symptoms?”	What, treatment, flu, symptoms	The articles related to treatment for the flu
“How to cure the flu symptoms?”	How, cure, flu, symptoms	The articles related to the treatment of the flu