

Enhance E-Commerce Recommendations with RAG

In an e-commerce website, the difference between a search feature and a recommendation feature is that the search feature is just a hard filter on the products that might retrieve thousands of results, while the recommendation feature is meant to be smarter and retrieve a few products that are the most relevant to the user.

The Nuclia RAG offers all the tools to build an end-to-end recommendation feature that will help your users finding the products they are looking for.

Indexing the primary AND the secondary information

Products are usually described by a set of attributes plus a description. That is what a classical e-commerce search engine will index. That way, you can search for: “red shoes” and get all the red shoes.

But to provide a good recommendation, you need to integrate other criteria that can go beyond the product description and the hard criteria the user is entering in the search bar.

For example, if you want red shoes to go out on the town and dance, or at the contrary if you want red shoes to walk in a park, the best recommendation will not be the same, but the product attributes might not be good enough to make the difference.

Typically, by entering: “I love dancing and I am looking for red shoes”, you would obtain a long list of red shoes, but maybe not necessarily the best ones for dancing. (And, yes, it is unlikely a user will enter such a query, but that is

exactly the query you *need* to produce a good recommendation, and you will see below how you will obtain it.)

What could be interesting would be to index the photos of the shoes. If the photos show people dancing, then the user might be more convinced that these shoes are good for dancing, and will trust the recommendation more.

The photos are secondary data in your product catalog, but they can give a lot of context to the recommendation process.

Nuclia is able to index your e-commerce website pages directly, just by providing their URL:

```
nuclia kb upload link --uri=https://www.my-ecommerce-  
website.com/red-shoes
```

Note: this example (and the following ones) uses the Nuclia CLI, but you can also index your web pages with our Python SDK, our JS/TS SDK, or even the API directly.

Nuclia will automatically extract all the text and metadata from each page, but it will also analyse the images and produce a short description of each image. This description will be indexed and can then be used by the Nuclia RAG engine to produce good recommendations.

Now you can search for: “I love dancing and I am looking for red shoes” and get the best red shoes for dancing.

Enrich the query with user information

Of course, you cannot expect the user to enter a full sentence like: “I love dancing and I am looking for red shoes”. The user will probably just enter “red shoes”.

To get a good recommendation, you need to combine this short query with the user's profile and turn it into a complete sentence that can be effectively used by the RAG engine to perform an appropriate semantic search.

That's what the `/predict/rephrase` endpoint allows you to do. It will take a short query plus a list of strings as context, and turn it into a full sentence that can be used by the RAG engine:

```
nuclia nua predict rephrase --question="red shoes"
```

```
--user_context='["the question is a user query on an e-commerce website, turn it into a suitable product search taking into account the user preferences", {"user": {"preferences": ["dance", "restaurant", "party"]}]']'
```

And you obtain:

What type of shoes would be suitable for a user with preferences for dance, restaurants, and parties?

With this full sentence, you can now run a semantic search that will take into account the user preferences and provide the best recommendations.

As you can see, the user preferences are passed as a JSON object in the `user_context` parameter, as it accept both plain natural language and structured data. This JSON object can contain any information you want to inject into the RAG process to provide the best recommendations.

Leverage metadata

The generated answer will be based on the text content of the indexed pages, describing the product and explaining why it is a good fit for the user.

But instead of a bare response, it can be interesting to provide a rich card with the product image, the price, and a link to the product page.

To do so, you need to inject metadata associated with the product page into the generated answer.

That's what the `/ask` endpoint's "Metadata" RAG strategy allows you to do. It will append to each matching result different metadata that can be used to generate a rich card. You can then indicate in your system prompt how you want to display this metadata.

```
{  
  
  "query": "What type of shoes would be suitable for a user with  
preferences for dance, restaurants, and parties?",  
  
  "prompt": {  
  
    "system": "Use the origin url to provide a markdown link to  
the corresponding result."  
  
  },  
  
  "rag_strategies": [  
  
    {  
  
      "name": "metadata_extension",  
  
      "types": ["origin"]  
  
    }  
  
  ]  
}
```

```
}
```

It will return something like:

```
These shoes are great for dancing, [buy them  
now!](https://www.my-ecommerce-website.com/red-shoes)
```

Here you are using the origin metadata, but there are several types of metadata that can be used, one of them is the extra metadata, which you can use to store in your resources any additional information you might need, like the price, the stock, the delivery time, discount codes, etc.

If you need to render a more complex UI than a simple link, you might not want to rely on markdown code generated by an LLM. In that case, you can use the `answer_json_schema` parameter to retrieve the information in a structured format that can be easily processed by your front-end (see the [Use JSON output](#) how-to).