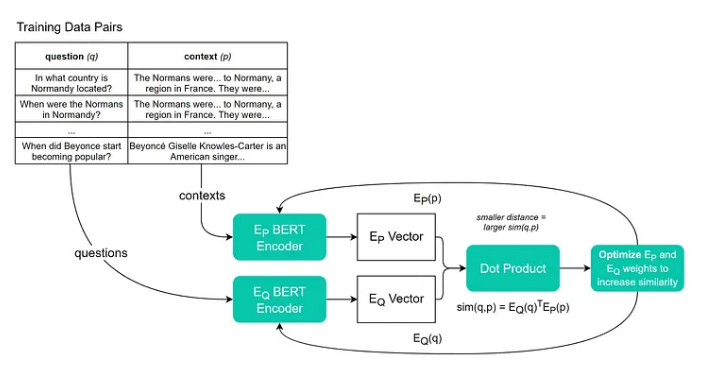
Haystack and ElasticSearch

After trying out a bunch of techniques we moved on to use Haystack and ElasticSearch.

[Haystack](https://www.deepset.ai/haystack-nlp) is a Python framework for natural language processing (NLP) that focuses on semantic search and question answering (QA). ElasticSearch is an open-source document store framework that is suitable for storing unstructured data to be used along with Haystack. We used ElasticSearch because it is very fast, highly scalable, and its indexing and reverse-indexing capabilities allows to query complex full-text searches in a very short time.

Here, we indexed the entire dataset into a pipeline which splits the data according to theme and appends all paragraphs into a single document and then loaded into a ElasticSearch Document Store for that particular theme. After that a question answering pipeline is created which is made up of 2 modules – Retriever and Reader. Retriever is a module in haystack where the embeddings of the documents and the embeddings of the question are generated using Hugging Face Transformer models/ self-generated model, and then they are the checked for similarity scores using cosine similarity. Using these cosine similarity scores, the Retriever module picks out the top K documents out of the document store where K is the amount of documents to be retrieved set by the user.



These K documents are then fed to the Reader module. Reader module is the place where the question-answering takes place. Here we use a FARMReader ( Framework for Adapting Representation Models” ). Here a model, either Hugging Face Transformer-based or self-generated, takes in a document from the set of documents retrieved and the question and gives out the logits tensor of the start position, end position and relevance score.

This is similar in architecture to the Retro Reader model with the difference that the Sketchy and the Intensive Reader module looks at all the documents and in that Sketchy module gives out the answer prediction and Intensive module gives the answer start logits and end logits whereas the Retriever module selects k relevant documents and passes those documents to the reader module. If there are no relevant documents then we can safely predict that answer is not possible and skip the Reader part. Hence the time taken is significantly less in the case of Haystack Retriever-Reader module.

The models we tried to use in the Retriever-Reader models are

1. Retriever
   1. Dense Passage Retrievers - DPR
      1. Facebook’s DPR Encoder
         1. facebook/dpr-question\_encoder\_single\_nq\_base
         2. facebook/dpr-ctx\_encoder\_single\_nq\_base
      2. voidful’s DPR Encoder
         1. voidful/dpr-question\_encoder-bert-base-multilingual
         2. voidful/dpr-ctx\_encoder-bert-base-multilingual
   2. BM25 Retriever
2. Reader –
   1. RoBERTa Tiny – SquAD v2 Finetuned
   2. //add more models which were used

However, we had to drop this method because

1. We tried to some models which were light-weight but unable to as they were not able to store the resultant embeddings back to document store.
2. Large Inference Times on better models .