**To build a search engine with questions and answers repository:**

**Machine learning**: NLP(Sentence Vectors, Elastic Search ,Semantic search), Euclidean distance, Universal sentence encoder, BERT, cosine similarity, KNN search

**Python Libraries**: elastic Search, pandas, TensorFlow hub

**Deployment**: FLASK API + AWSEC2 using Docker

Blog ref: <https://www.elastic.co/blog/text-similarity-search-with-vectors-in-elasticsearch>

**Problem definition:**

Type a question, you should get similar type of answers and order them based on relevance and should be fast.

Qi, with multiple answers A1, A2……Ak

We want high **precision** and **recall**, low server cos and quick to deploy.

If we are asking a question Q\* and if there is a Q10, relevant to Q\* then Q10 should be there on the top position

(Q\* is the question the user asks)

(Qi is the question that is part of our dataset)

**Dataset:**

Sample: <https://www.kaggle.com/stackoverflow/stacksample>

**How to handle this whole solution**.

**Design and Data Structure:**

For each Question Qi we will have Id, title and body as important fields

For each ans Aj we have id, body, parentid

Design Choices:

1. Whenever an user asks for a question, search will be question titles that are there in data store
2. Qi body
3. Aj body

Qi title will have seq of words w1, w2……wd

We will be using only title of each questions Qi to search for most similar title that somebody asks

**1)Keyword based similarity**

**Data Structure**:

A) Hash Table

Since hash table has o[1] search – fastest search

How to design a hashing system?

Hashing is used by dictionary in python

To define k,v is difficult so we will use **INVERTED INDEX**

We take a key as word and number of time that word exists in the questions related to that word

Each question can be thought of document

**B) For Score**: **TF-IDF(Term Frequency and Inverse doc frequency)**

Scoring function (Similarity Function)

Score(Q\* Qi) how closely both are related, how relevant qi to q\*

Elastic search uses default tf idf to rank. It implements inverted index does scoring using tfidf mechanism, is distributed, real-time queries

**2)Semantic Search**

Similarity between question can be represented as similarity between numeric vectors

Used Euclidean distance or cosine similarity to measure distance between sentences (similarity between questions)

Closer the distance between two points , more similarity.

Nearest Neighbor Search:

Kd tree, lsh , neighborhood graph

**Deployment**: FLASK API + AWSEC2 using Docker