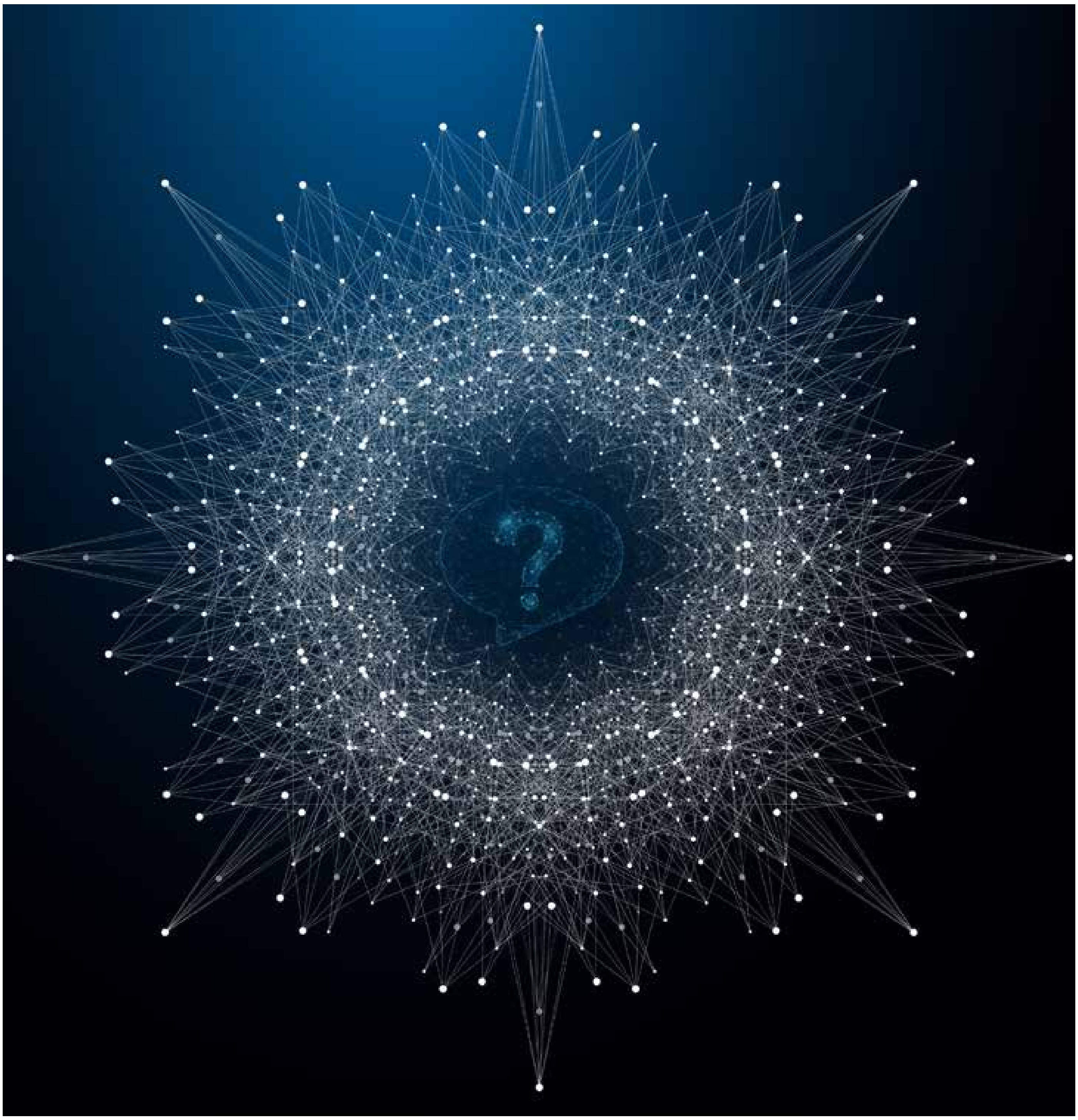
Improving Search Relevance in Question Answering Engine

Whitepaper by Abinaya Mahendiran, Assistant Manager, NEXT Labs;
Faustina Selvadeepa, Senior Software Engineer, NEXT Labs



Contents

1. Introduction

The Question Answering (QA) system is an information retrieval system. Unlike search engines, they automatically answer questions posted by users in natural language rather than just providing links or references to the answer. Traditionally, a structured database of information known as knowledge base is used to store questions and corresponding answers - and a computer program is used to query this knowledge base for relevant answers. Since the knowledge base contains humungous data, it takes a lot of time to search for relevant information.

In a development environment, most of the developers' time goes into finding the right solution for their technical queries. QA systems help in reducing search times by providing the most relevant solution to the users' query, with the help of various Natural Language Processing (NLP) techniques and Machine Learning (ML) algorithms. These algorithms collectively form the central part of the QA system's scoring mechanism that is described in this paper.

2. System architecture

Fig. 1 gives the overall system architecture of the QA system. When a user enters a query in the search engine, a subset of questions that match the query are extracted and fed into the system. The Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW) modules are feature engineering models that extract the relevant features from the data for multi-label classification module and Latent Dirichlet Allocation (LDA) module respectively.

The multi-label classification module predicts tags for each question and generates a score. Likewise, the LDA module extracts topics for each question and generates a score. A generic module generates an aggregated score from measures like upvotes/downvotes to an answer, favorite count, etc., - if available in the data. All the scores mentioned above are passed into a Principal Component Analysis (PCA) module, which rank orders the questions based on relevance.

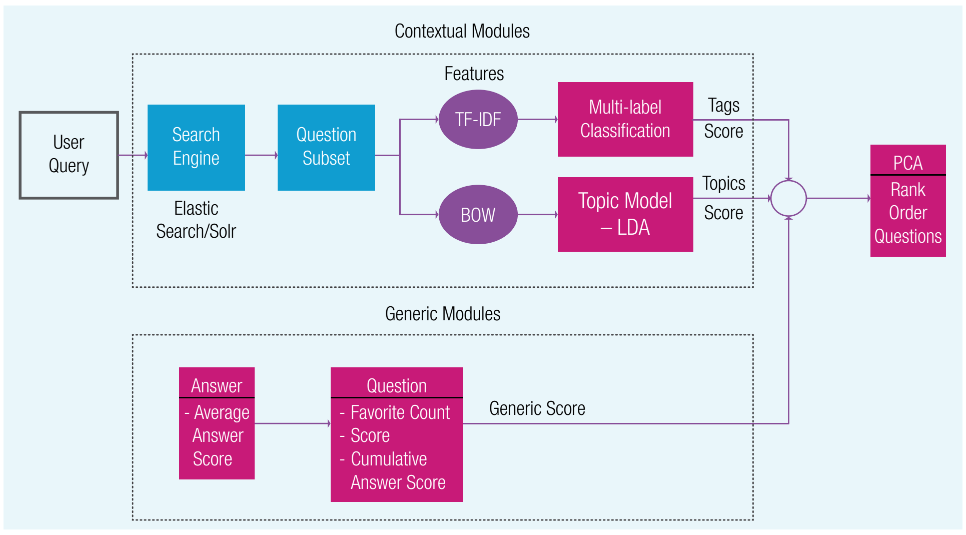


Fig. 1 - System architecture

3.
Data

The QA system's knowledge base draws data from various Community Question and Answer (CQA) sites. Data must contain questions and their corresponding answers, along with tags. Tags provide a useful way to group a related question-answer together and each question-answer pair can have multiple tags. The QA system can thus adapt to any 'question-answer' data, irrespective of the domain.

The following sections describe the different modules in detail.

a) Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a scoring measure used to evaluate how important a word is to a document in a collection or corpus. This importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Thus, lesser weightage is given to the most commonly occurring words - and a higher weightage to the rare and most informative word.

*TF-IDF(t) = TF (t in a document) * IDF(t)*

- *TF(t)* = # of times the term appears in document/total # of terms in document
- *IDF(t)* = log (total # of documents/# of documents with term in it)

	Terms											
Documents	ability	amazons	come	ec2	image	instances	know	looking	new	running	save	server
doc1	0.274259	0.151581	0.226533	0.076613	0.147465	0.147025	0.102992	0.148132	0.130282	0.123239	0.218816	0.09427
doc2	0	0	0	0	0	0	0	0	0	0	0	0
doc3	0	0.060632	0	0.061291	0	0	0.082393	0	0	0	0	0
doc4	0	0	0	0	0	0	0	0	0	0	0	0
doc5	0	0	0	0	0	0	0	0	0	0	0	0
doc6	0	0	0	0	0	0	0	0	0	0.079939	0	0
doc7	0	0	0	0	0	0	0	0	0	0	0	0
doc8	0	0	0	0	0	0	0.030144	0	0	0.03607	0	0
doc9	0	0	0	0	0	0	0	0	0	0	0	0
doc10	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2 - A snapshot of tf-idf scores for each term in a document

Every question passed into this module is reduced to a vector of real numbers, each of which represents ratios of counts. The result is a term-by-document matrix X whose columns contain the tf-idf values for each of the documents in the corpus.

b) Bag of Words

Bag of Words (BoW) is a way of extracting features from text for use in ML algorithm.
A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words
- A measure of the presence of known words

D1 - ""I am feeling very happy today"
D2 - " I am sick today"
D3 - "I wish I could dance like my sister"

	I	am	feeling	very	happy	today	sick	wish	could	dance	like	my	sister
Doc1	1	1	1	1	1	1	0	0	0	0	0	0	0
Doc2	1	1	0	0	0	1	1	0	0	0	0	0	0
Doc3	2	0	0	0	0	0	0	1	1	1	1	1	1

Fig. 3 - A snapshot of BoW approach

c) Multi-label Classification

Multi-label classification is the problem of finding a model that maps inputs x to binary vectors y (assigning a value of 0 or 1 for each element (label) in y).

Fig. 4 shows that each entity can be associated with more than one class.

As question-answer pair in our dataset belongs to multiple tags, a multi-label classifier assigns more than one label/tag to each question.

Instance	Classes
1	A,B
2	A
3	A,B
4	C
5	B
6	A

Fig. 4 - Multi-label classification

d) Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus. It is used to classify text in a document into latent topics. Given a query, LDA groups it into a cluster of similar topics.

At the end of Realization stage, it is mandate to finalize the data migration strategy so that it becomes easy to migrate from any database to another one without any hassle. The migration planning is fractionalized as per these following steps:

The boxes are ‘plates’. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document. There are three levels to LDA representations. The parameters **α** and **β** are corpus level parameters, assumed to be sampled once in the process of generating a corpus. The variables $\boldsymbol{\theta_d}$ are document-level variables, sampled once per document. Finally, the variables $\mathbf{z_n}$ and $\mathbf{w_n}$ are word-level variables and are sampled once for each word in each document.

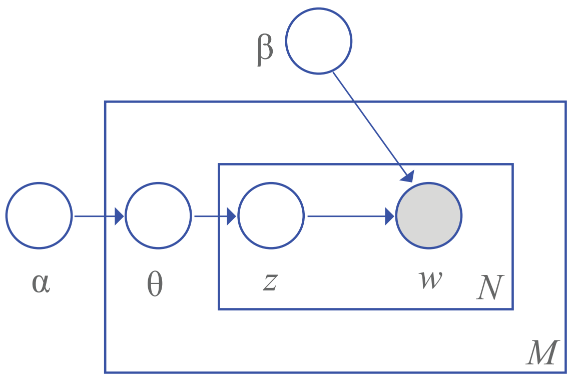


Fig. 5 - Latent Dirichlet Allocation (LDA)

LDA assumes the following generative process for each document w in a corpus D:

- Choose N _ Poisson(ζ)
- Choose $\mathbf{q}$ _ Dir(α)
- For each of the N words W_n:
  - o Choose a topic Z_n _ Multinomial($\mathbf{q}$)
  - o Choose a word W_n from $p(W_n \mid Z_n, \beta)$, a multinomial probability conditioned on the topic Z_n
- Parameterize probabilities by a k × V matrix β where $\beta ij = p(W_j = 1|Z_i = 1)$, which for now we treat as a fixed quantity that is to be estimated
- A k-dimensional Dirichlet random variable θ can take values in the (k −1)-simplex (a k-vector θ lies in the (k–1)-simplex if $\theta_i \geq 0, \sum^k{}_i=1\ \theta_i = 1$)
- Probability density on this simplex: $p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha i\right)}{\prod_{i=1}^{k} \Gamma(\alpha i)} \theta_1{}^{\alpha i\text{-}1} \cdots \theta_k{}^{\alpha k\text{-}1}$, where the parameter α is a k-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function
- Find Joint Distribution $p(\theta, z, w \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p\ (Z_n \mid \theta) p(W_n \mid Z_n, \beta)$, where $p(Z_n|\theta)$ is simply θ_i for the unique i such that $z_n^i = 1$.
- Calculate marginal distribution of a document
- $p(w \mid \alpha, \beta) = \int p(w \mid \alpha) \left(\prod_{n=1}^{N} \sum_{Z_n} p\ (Z_n \mid \theta) p(W_n \mid Z_n, \beta) d\Theta\right.$
- Calculate product of marginal probabilities of, we obtain the probability of a corpus: $p(D \mid \alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{Z_{dn}} p(Z_{dn}|\theta_d) p(W_{dn}|Z_{dn}, \beta) \right) d\theta_d$
- Calculate Z = Calculate $p(D|\alpha, \beta)$
- Output the significant z values greater than a given threshold

e) Generic Score

Generic score is calculated by aggregating the feedback scores available in the data.
These measures could include upvotes/downvotes for each question and answer, favorite count, cumulative answer score, etc.

f) Principal Component Analysis

The main idea of Principal Component Analysis (PCA) is to reduce the dimensionality of a d -dimensional data set consisting of many variables correlated with each other by projecting it onto a ***(k)***-dimensional subspace (where ***k<d***) to increase the computational efficiency while retaining most of the information.

A summary of the PCA approach:

- Standardize the data
- Obtain eigenvectors and eigenvalues from the covariance matrix or correlation matrix – or, perform Singular Vector Decomposition. The covariance between two features is calculated as follows:

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

We can summarize the calculation of the covariance matrix via the following matrix equation:

$\sum = \frac{1}{n-1}((x-\bar{x})^T(x-\bar{x}))$ where x¯ is the mean vector $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

- Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace (k≤d)
- Construct the projection matrix W from the selected $\boldsymbol{k}$ eigenvectors
- Transform the original dataset X via W to obtain a k-dimensional feature subspace Y

Doc_ID	PCA_score
27267	1.261710149
5545438	1.133725711
3980968	1.124644981
3140779	1.102559411
2499132	1.066664645

Doc_ID	PCA_score
7136337	1.04800105
6525270	1.031511975
3578430	1.02635061
701545	1.01282737

Fig. 6 - Score from the PCA module

4.

Results

The question-answer pair from the CQA sites is fed into the hybrid model and ranked, based on the various scores given by the modules.

Query: How to filter data based on a column value in sql?

Search Engine Results	**Scoring Module Results**
Title	Title
SQL Query to split data based on column value	SQL Filter based on results from SQL query
SQL Query where clause filter based on another column value	Filter based on column value match in SQL
SQL Query to order data based on other column value	SQL Query to get column result conditionally based on column value
Selective Filter rows based on column Value SQL	SQL select query based on a column value
Filter based on column value match in SQL	SQL Query to display a name based on column value
SQL PIVOT Query, column based on value	Split a column into two columns based on filter value in query

Fig. 7 - Results from scoring module

The snapshot above compares the results of scoring module with the traditional search engine. The scoring module provides more relevant results when compared to the search engine results.

Authors

Abinaya Mahendiran

Assistant Manager at Mphasis NEXT Labs

Abinaya Mahendiran is an Assistant Manager at Mphasis NEXT Labs. She holds a Master's degree in Computer Science with a specialization in Machine Learning and Deep Learning from International Institute of Information Technology Bangalore (IIIT-B). Her research areas include Natural Language Understanding/Processing, Machine Learning, Deep Learning and MLOps. She has an extensive software engineering and data science experience. At NEXT Labs, she has been building and productionizing NLU/NLP solutions for various clients both on premise and on cloud.

Faustina Selvadeepa

Senior Software Engineer at Mphasis NEXT Labs

Faustina Selvadeepa is a Senior Software Engineer at Mphasis NEXT Labs. She holds a Master's degree in Computer Application. Her research areas include Natural Language Processing and Machine Learning. She has worked on various client PoCs and helped in building one of the Mphasis IP, InfraGraf®.