Smart Chatbot with Document Retrieval and Extractive Question Answering

Mr. Virendra Bagade[1] Dr. S. G. Godse[2]

[1 & 2] Ph.D. Scholar, Computer Engineering, VIIT Research Center, Pune

Abstract:
Support engineers on various projects deal with many problems while carrying out their daily work. When they face an issue in a project, they have to read a large amount of relevant documentation to figure out what should be the fix to the issue at hand. They often have to read many of the documents that can be too technical and time consuming to read through every section. In the age of artificial intelligence and automation, we can leverage natural language processing methods to ease this process.

Index Terms - Chabot, information retrieval, natural language processing, neural search, document.

Introduction:
To solve this problem faced by support engineers, we propose an interactive smart dialogue system that can converse with a support engineer and provide them with the appropriate responses for their queries. To make this dialogue system more natural, we make it so that the Chabot can not only provide info on the required topic but also talk naturally with the support engineer interacting with it. This means that the Chabot would reply promptly to queries such as "How are you?", "What is your name", "Bye" with replies such as "I am great! How are you?", "I am the support Chabot for your help." and "Have a good day!" respectively. This would make it much easier for the engineer to interact with the software. Along with the natural responses to questions we ask in daily life, we train our Chabot to detect queries that need their answers fetched from documentation. Once we detect a query such as "What is Ubuntu?" we then pass this question onto a neural search module that uses Elastic Search to find out which document has the highest probability of containing the answer. We achieve this by indexing all the documents with Elastic Search and BM25. We use the haystack framework to implement this search module. We first integrate the Chabot and the neural search module together such that if we detect the query, we can send the query to our neural search module to find out what document will contain our answer. To achieve the final step, that is to fetch the answer from the document and present it to the user. We do this with a RoBERTA model that is fine-tuned on a question answering dataset SQuAD 2.0. Extractive question answering works in a way such that we need to pass the question as well as the document as context and get the answer as an output. This step can take a significant amount of time depending on the size of the model and the availability of a Graphics Processing Unit for extractive question answering. Once we fetch the answer or the top 3 answers, we integrate this extractive question answering module with our Chabot that after sending the query to neural search is waiting for a response API. Once we receive the answer, we present it to the user in the Chabot user interface.

II. RELATED WORK

1] L. Yang, H. Zamani, Y. Zhang, J. Guo, and W. B. Croft used neural matching models and per- formed question retrieval and next question prediction on the Quora questions dataset and Ubuntu chat logs. Information Retrieval methods like BM25, TRLM and LSTM-CNN-Match are used. Out of these, the LSTM-CNN-Match gives the best results for question retrieval as well as next question prediction.

2]W. Wu, G. Liu, H. Ye, C. Zhang, T. Wu, D. Xiao, W. Lin, and X. Zhu,"EENMF have come up with a neural matching framework for ecommerce ads. The authors have proposed a two-stage deep matching framework for vector-based advertisement retrievals and pre-ranking these ads by using neural networks.

3] Contains a full-fledged literature regarding neural methods for information retrieval. B. Mitra and N. Craswell have shed light on the origins of information retrieval, text representations, term embeddings, deep neural networks and deep neural models for Information Retrieval.

4] J. Guo, Y. Fan, X. Ji, and X. Cheng, "Matchzoo suggested a system for neural matching tasks, through a user interface where researchers and users can apply a host of techniques like Data Preparation pipeline, Automatic ML, Model Construction, Model Designing and Model practicing.

5] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol have proposed a transformer-based architecture for intent classification and entity recognition, which is better than BERT and also faster to train.

6] In this paper T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol proposed the original Rasa chat-bot framework, which contains the Rasa NLU and various NLP techniques and ML libraries.

7], a new model called Hybrid Co-Attention network is proposed by J. Rao, L. Liu, Y. Tay, W. Yang, P. Shi, and J. Lin. A study related to semantic and relevance matching is also carried out.
Graph Neural Networks are leveraged by X. Ling, L. Wu, S. Wang, G. Pan, T. Ma, F. Xu, A. X. Liu, C.Wu, and S. Ji

8], to carry out semantic code retrieval. The model called DGMS was tested on two public code retrieval datasets of Java and Python language.
A new algorithm which implements semantic matching called S-Match was proposed by F. Giunchiglia, P. Shvaiko, and M. Yatskevich

9]. Mapping between nodes of two semantically-related graphs is done in this method.
Y. Fan, J. Guo, X. Ma, R. Zhang, Y. Lan, and X. Cheng present a comprehensive study on relevance modeling in

10] and put forth ideas related to comparing differences with respect to the Natural Language Understanding and how it can be improved.

11]Z. Zeng, D. Ma, H. Yang, Z. Gou, and J. Shen propose a domain-independent tool for automatically doing intent-slot induction. This tool resulted in a better performance in State-Of-The-Art results by 76
An all-inclusive model for knowledge relevance, topical relatedness and semantic similarity is proposed by X. Li, J. Mao, W. Ma, Y. Liu, M. Zhang, S. Ma, Z. Wang, and X. He in 12] which helps estimate the relatedness between query and document.

III. METHODOLOGY

The proposed work consists of three major modules namely

1) Chatbot module
2) Search Module
3) Extractive Question Answering

A. Chatbot Module

The function of chatbot is to make use of an intent classifier to classify messages given by the users into various predefined intents. It has to then decide the response to be given based on these intents according to the rules set by the programmer. e.g.) User : "Hello" this message will be classified as "greet" intent and the program/bot will output a response as "Hi, how may I help you". Similarly, if the user inputs a message saying "Who are you?" or "Are you a bot?" The intent classifier will classify it as "bot challenge intent" and will output a predetermined response as set by the programmer.

If the intent is not one of the predefined ones then it will be classified as a query and will be sent to the neural search module as a search string. The neural search module will then look for the query in documents present on the intranet and will return the search results. The intents used are listed as follows:

1) Knowledge question: This intent is used for questions asked by the user which require access to data. e.g. What is the capital of France?

2) greet : This intent is used for simple greet dialogues. e.g. Hi, Hello, good evening, etc

3) goodbye: The goodbye intent is used for exit dialogues. e.g. Bye, see you later etc.

4) affirm : This intent is used for affirmative responses. e.g. yes, indeed, etc.

5) bot challenge: This intent is used for questions like "who are you?" which asks for the identity of the chatbot. e.g. Who are you?Are you human?

6) Nlu fallback: This intent is used when the response cannot be classified into any of the above intents.

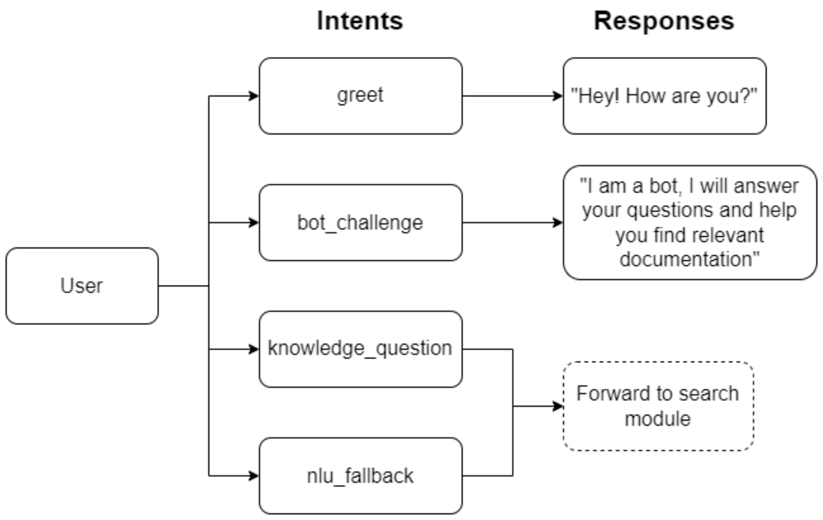


Fig. 1. Chatbot flow Diagram

B. Search Module

The search module is used to fetch the documents which are most relevant to the search query. For this we experimented with various algorithms such as Levenshtein Distance and BM25. The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ & \textit{otherwise} \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1 \end{cases} & \end{cases} \quad .. \textit{ Equation-01}$$

Fig 2. Levenshtein Distance Formula equation-01

The BM25 algorithm is a vector-based string searching method. It's the successor to TF-IDF and is the result of optimizing TF-IDF primarily to normalize results based on document length. It saturates tf after a set number of occurrences of the given term in the document and it normalizes by document length so that short documents are favored over long documents if they have the same amount of word overlap with the query.

These are sparse methods, that is they operate by looking for shared keywords between the document and query. We use Elasticsearch to implement the BM25 algorithm and for indexing the documents.

$$BM25(D,q) = \frac{f(q,D)*(k+1)}{f(t,D)+k*(1-b+b*\frac{D}{d_{eq}})} * \log\left(\frac{N-N(q)+0.5}{N(q)+0.5}+1\right) \quad .. \textit{ Equation-02}$$

Fig. 3. BM-25 distance formula

Using elasticsearch helps us to achieve fast results as it's written in Java and use its features such as distributed nature, ability to access using a REST api and to later deploy it as a hosted or managed service.

C. Extractive question answering module

The function of this module is to extract the answers of the given question using the context fed to it. The context is retrieved by the search module. For this we use the roberta- base QA model trained on the SQuAD 2.0 dataset. The model was trained on the SQUAD data-set that is a reading comprehension data-set, consisting of questions posed by crowdworkers on a set of Wikipedia articles. Here, we are able to extract the answer given the question and the context from which to extract the answer. The documents retrieved using the retriever module act as the context to the Question Answering model.

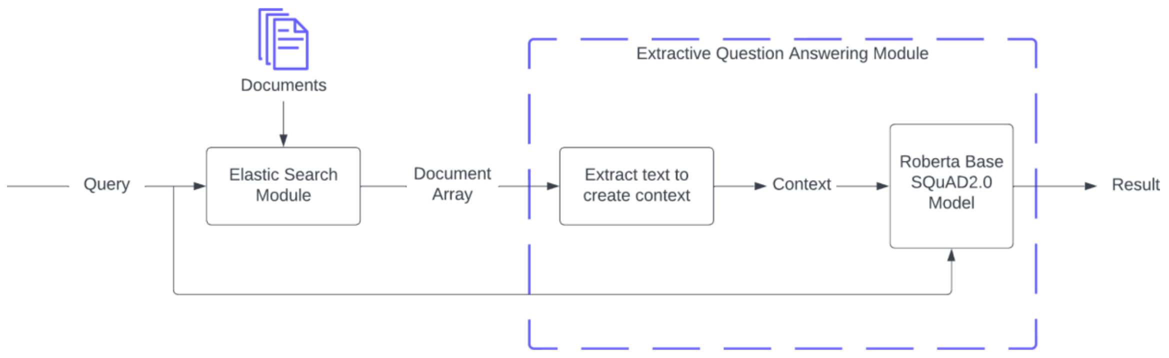

Documents
Extractive Question Answering Module
Query
Elastic Search Module
Document Array
Extract text to create context
Context
Roberta Base SQuAD2.0 Model
Result

Fig. 4. Extractive Question Answering

D. Knowledge Graph construction module

Knowledge Graphs were first proposed by the Google Search team. Data is structured and represented in a graphical format. Knowledge Graphs, as the name suggests is a graph which contains entities as nodes and actions as edges. They help in linking and unifying the data. These are mostly directed graphs, with nodes and links containing specific domain-related information. This data structure represents a network of entities, and the relationships that exist amongst them.

For a collection of text documents, we have first transformed the text documents into csv files. Each sentence in the document is stored as an individual data entry in the csv file or in the form of a list. We construct the knowledge graph for a document by first splitting the sentence into its subject, object and predicate. The subject here refers to who the information is about, the object refers to what and the predicate is generally the verb, which tells the connection between the subject and object. The knowledge graph is thus represented as having subject and object as the nodes, with the edges going from subject to object, and the edge contains the verb or action extracted from the sentence. Representing information in the form of Knowledge Graph will help in retrieval and also produce relevant results accord- ing to the query passed by the users.

E. Document Similarity Module

Document similarity module facilitates comparing different documents with help of a common mathematical metric. In this module the text is encoded into a 512 dimension vector, irrespective of the size of the document. In order to compare two documents each document is encoded to a 512 dimension vector. Cosine distance between these vectors is then calculated. The cosine similarity is then calculated from this distance.

$$cosine\ distance = 1 - cosine\ similarity.. \quad \text{..Equation-03}$$

$$cosine\ similarity = \cos\theta = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

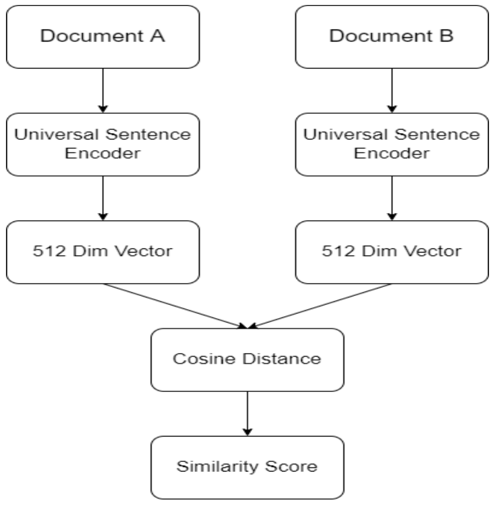


Fig 5. Document Similarity

F. Table Question Answering Module

Given a query, we will search a large number of indexed tables and find the respective record or group of records that can answer the query. Haystack provides models and APIs to fetch tables and then use various models such as TAPAS to encode tables for further retrieval. We also use a tri-encoder model that has separate encoders for table, text and query. The models are based upon BERT architecture. The indexing will require a GPU to speed up the process significantly. For this we use the bert-small-mm-retrieval-question-encoder model.

IV. ALGORITHMS OR MODELS USED

1) RoBERTa:

RoBERTa is a pretraining approach to BERT that fully leverages the capabilities of BERT with the right hyperparameters and training size. They use the same model and achieve state-of-the-art accuracy on various tasks. The model achieves these accuracies on GLUE,RACE and SQuAD datasets. For pretraining, they use tasks like Masked Language Modelling and Next Sentence Prediction.

Proposed Model Implementation Pseudo Code:

```
procedure PreprocessData [P]:
    initialize tokens[]
    for each para in paragraphs:
        para = removeNonAscii(para)
        para = removeFormat(para)
        para = removeStopWord(para)
        tokens[i] = tokenize(para)
    return tokens
end procedure
procedure AnswerQuestion[P]:
    initialize tokens[] = use procedure ProcessData[p]
    initialize maskedTokens[] = applyMask(tokens[])
    initialize start_logits, end_logits = predictLogits(maskedTokens[])
    initialize normTokens = applyNorm(start_logits, end_logits)
```

```
        return deTokenize(normTokens)
end procedure
```

2) TAPAS:

TAPAS stands for Table Parsing. This model is trained to understand tables and answer questions based on the data we have in the form of tables. This model overcomes the need to generate logical forms. It uses weak supervision rather than a fully supervised approach. It predicts a bunch of cells as the answer and then optionally applies an aggregation function that can give answers based on more than one row. TAPAS adds a table as an input to the core model of BERT and is then trained fully. It performs on par with WIKISQL and WIKITQ datasets, with a very simple approach. It also surpasses the state- of-the-art when using transfer learning from one dataset to another.

A. Transformers

Transformers were developed to solve the problem of sequence transduction, or neural machine translation. That means any task that transforms an input sequence to an output sequence. This includes things like speech recognition and text-to-speech conversion. Prior to transformers, most state- of-the-art NLP systems depended on gated RNNs with extra attention mechanisms, such as LSTM and gated recurrent units (GRUs). Transformers are created utilizing these attention technologies without the use of an RNN structure, demonstrat- ing that attention mechanisms can match the performance of RNNs with attention. Because token computations are depen- dent on the results of prior token computations, parallelization on present deep learning hardware is difficult. As a result, RNN training may become inefficient. Attention methods were used to solve these issues. Attention mechanisms allow a model to draw from any previous state in the sequence. The attention layer can access all previous states and weigh them according to a learned measure of relevancy, providing relevant information about far-away tokens.

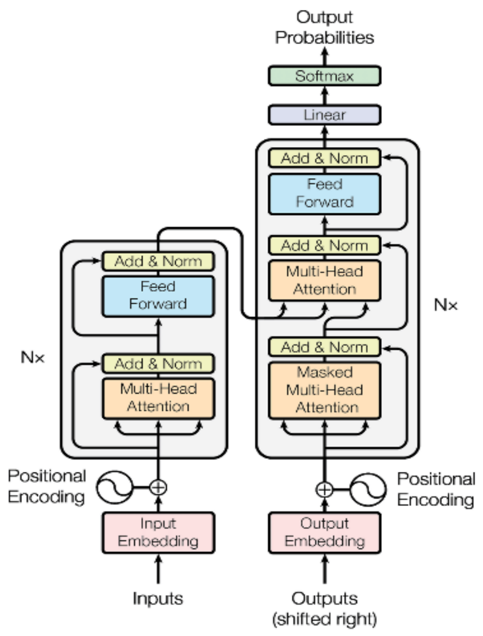


Fig 6. Transformer Model Architecture

B. The BM-25 Algorithm

Okapi BM25 (BM stands for best matching) is a ranking formula used by search engines to estimate the relevance of content to a certain search query. The BM25 retrieval function scores a set of documents based on the query phrases that exist in each document, independent of their closeness within the text.

Given a query Q, containing keywords *q1, ..., qn*, the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=i}^{n} IDF(q_i) * \frac{f(q_i,\ D) * (k_i + 1)}{f(q_i,\ D) +\ k_1 * (1 - b + b * \frac{|D|}{avgdl})}$$

..Equation-04

Where,

- *f(qi , D)* is qi's term frequency in the document D.
- |*D*| is the length of the document D in words.
- avgdl is the average document length in the text collection from which documents are drawn.
- *k1* and *b* are free parameters, usually chosen, in absence of an advanced optimization, as $k1 \in [1.2,\ 2.0]$ and *b = 0.75*.
- *IDF(qi)* is the *IDF (inverse document frequency)* weight of the query term *qi*.

IDF(qi) is usually computed as:

$$IDF(q_i) =\ \ln\left(\frac{N - N(q) + 0.5}{N(q) + 0.5} + 1\right)$$

Equation-05

Where,

- N is the total number of documents in the collection.
- *n(qi)* is the number of documents containing *qi*.

The formula's IDF component counts how many times a term appears in all of the documents and "penalizes' ' terms that appear frequently.

Because the multiplier for inquiries containing these more uncommon terms is higher, they contribute more to the final score. In practically every English document, the word "the" will appear. As a result, when a user searches for "the elephant," "elephant" is likely to be more essential — and we want it to contribute more to the score.

The score for the document lowers as terms not matching the query in the document increase. If a 400-page document only mentions a word once, It's less likely to play a role in it than a brief sentence that mentions it once.

The effectiveness of BM25 is the major feature that makes it popular. It performs very well in many ad-hoc retrieval tasks.BM25 is the current state-of-the-art TF-IDF-like retrieval model. However, there are some approaches for normalizing document length and satisfying the term frequency's concavity criterion (e.g., considering the logarithmic TF, instead of the raw TF).

Based on these heuristic techniques, BM25 often achieves better performance compared to TF-IDF.

C. The Levenshtein Algorithm

Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single- character edits (insertions, deletions or substitutions) required to change one word into the other.

$$lev_{a,b}(i,j) = \begin{cases} |a| & if\ |b| = 0 \\ |b| & if\ |a| = 0 \\ lev\big(tail(a), tail(b)\big) & if\ a[0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & otherwise, \end{cases}$$

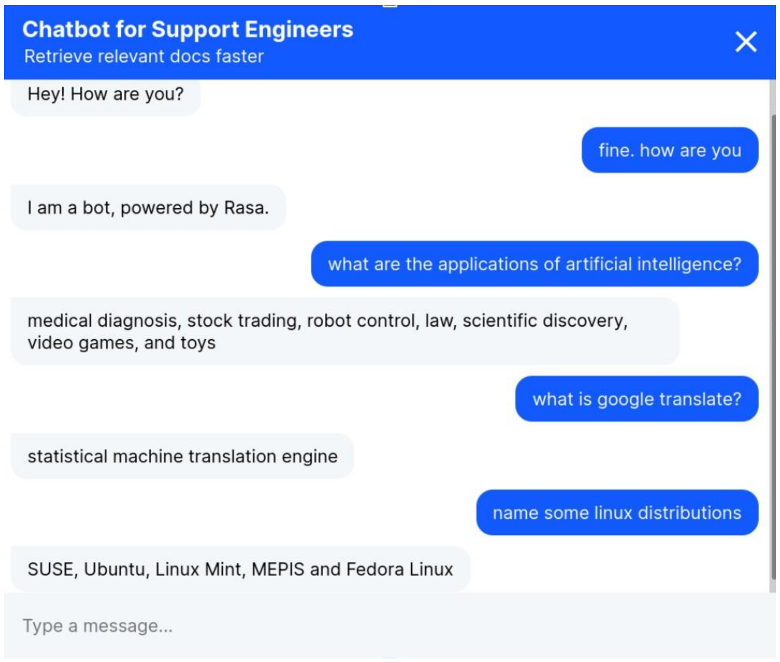
..Equation-06

The most common way of calculating this is by the dynamic programming approach.
Spell checkers, correction systems for optical character recognition, and software to help natural language translation based on translation memory are just a few examples.
Between two longer strings, the Levenshtein distance can be calculated. But the cost to compute it. This is impracticable because it is roughly proportional to the product of the two string lengths.

V. PIPELINE WALKTHROUGH

Initially the documents are indexed using the elastic search indexer. Then the relevant documents to the search query are retrieved using elastic search. Then using these documents as contexts, the extractive QA model gives us the relevant answer to our search query.

VI. USER INTERFACE


Chatbot for Support Engineers
Retrieve relevant docs faster

Hey! How are you?

fine. how are you

I am a bot, powered by Rasa.

what are the applications of artificial intelligence?

medical diagnosis, stock trading, robot control, law, scientific discovery, video games, and toys

what is google translate?

statistical machine translation engine

name some linux distributions

SUSE, Ubuntu, Linux Mint, MEPIS and Fedora Linux

Type a message...

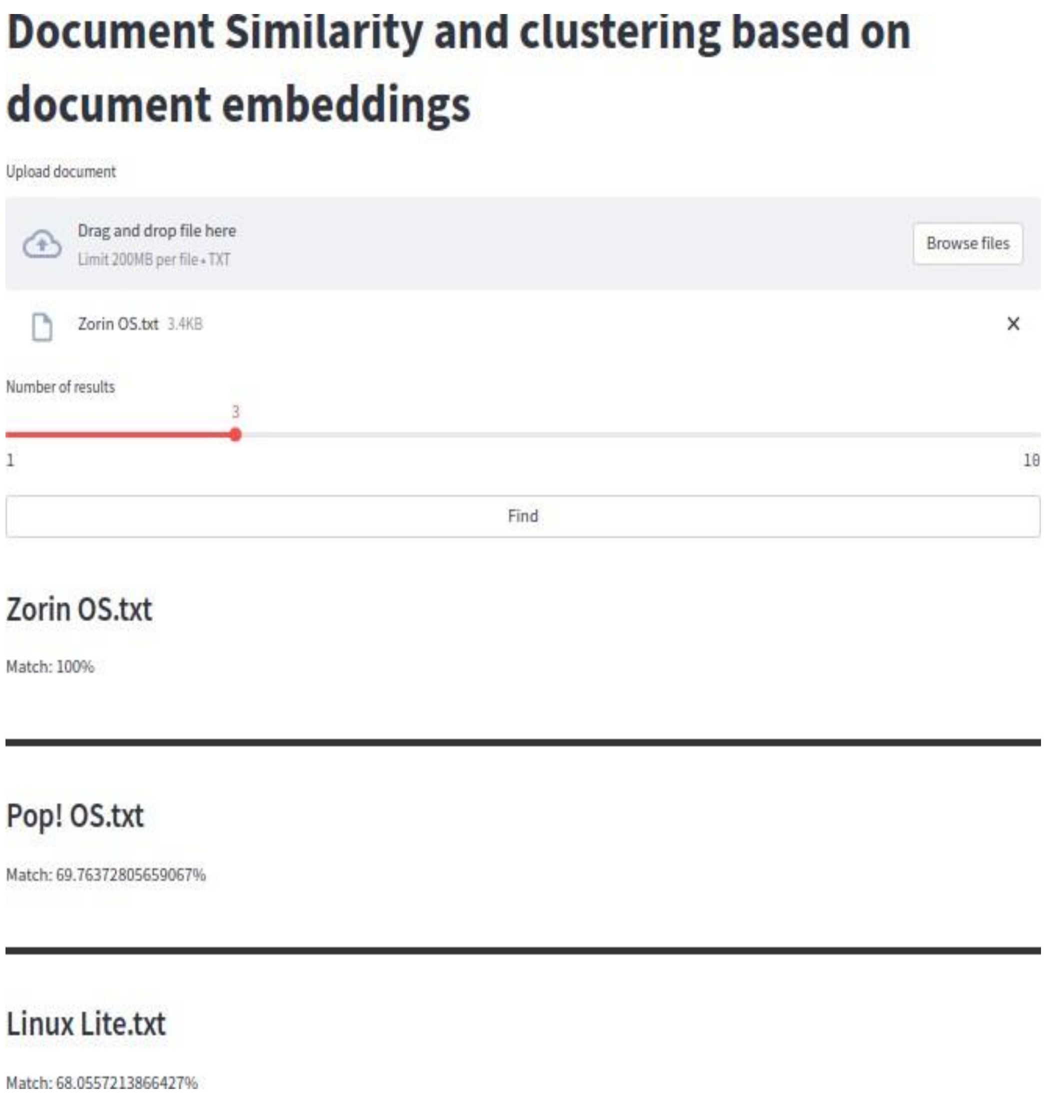
Fig 7. Chatbot User Interface

Output in Tabular Format:

Analysis	User Query	Result

Document similarity and clustering based on document embeddings	Uploaded file: Zorin Os.txt	Found files: Pop! Os.txt Match: 69.76372 % Linux Lite.txt Match: 68.05572 %
Extractive Question Answering	What are the applications of Machine Learning?	Applications of Machine Learning==Applications of Machine Learning Customer Relationship Management – Inverted Pendulum balance and equilibrium system. Natural Language Processing (NLP) Automatic Taxonomy Construction Relevance: 0.5293998 Source: Outline of machine learning.txt
Table Question Answering	Which year was zindagi ek safar released? Number of tables	0 1971 Zindagi Ek Safar Andaz Shankar Jaikishan Hasrat Jaipuri 1 1971 Yeh Jo Mohabbat Hai Kati Patang Rahul Dev Burman Anand Bakshi 2 1972 C hingari Koi Bhadke Amar Prem Rahul Dev Burman Anand Bakshi 3 1973 Mere Dil Mein Aaj Daag : A Poem of Love Laxmikant-Pyarelal Sahir Ludhianvi 4 1974 Gaadi Bula Rahi Hai Dost Laxmikant-Pyarelal Anand Bakshi 5 1974 Mera Jeevan Kora Kagaz Kora Kagaz Kalyanji Anandji M.G.Hashmat 6 1975 Main Pyaasa Tum Faraar Kalyanji Anandji Rajendra Krishan 7 1975 0 Manj hi Re Khushboo Rahul Dev Burman Gulzar 8 1977 Aap Ke Anurodh Anurodh Laxmikant-Pyarelal Anand Bakshi 91978 0 Saathi Re Muqaddar Ka Sikandar Kalyanji Anandji Anjaan 10 1978 Hum Bewafa Harghiz Shalimar Rahul Dev Burman Anand Bakshi 11 1979 Ek Rasta Hai Zindagi Kaala Patthar Rajesh Roshan Sahir Ludhianvi 12 1980 Om Shanti Om Karz Laxmikant-Pyarelal Anand Bakshi 13 1981 Hameh Tumse Pyar Kudrat Rahul Dev Burman Majrooh Sultanpuri 14 1981 Chhookar Mere Mann Ko Yaraana Rajesh Roshan Anjaan 15 1983 Shayad Men Shaadi Souten Usha Khanna Sawan Kumar Tak 16 1984 De De Pyar De Sharaabi Bappi Lahiri Anjaan 17 1984 I nteha Ho Gayi Sharaabi Bappi La hid Anjaan 18 1984 Log Kehete Hai (Mujhe Naulakha Manga De) Sharaabi Bappi Lahiri Anjaan elevance: 1.0

Output Screenshots



Document Similarity and clustering based on document embeddings

Upload document

Zorin OS.txt 3.4KB

Number of results

Zorin OS.txt

Match: 100%

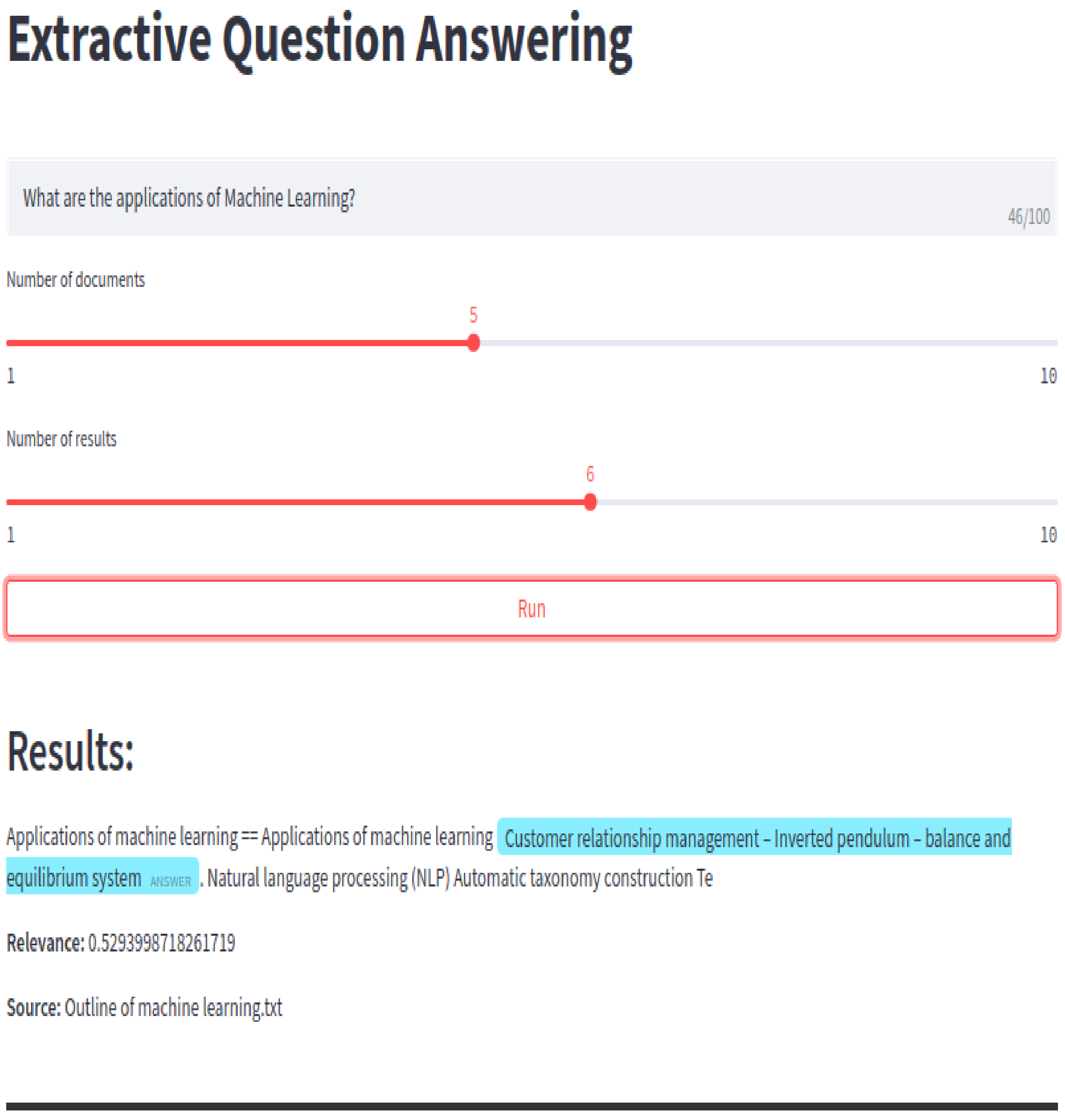
Pop! OS.txt

Match: 69.76372805659067%

Linux Lite.txt

Match: 68.0557213866427%

Fig. 8 Document Similarity



Extractive Question Answering

What are the applications of Machine Learning?

46/100

Number of documents: 5 (1–10)

Number of results: 6 (1–10)

Run

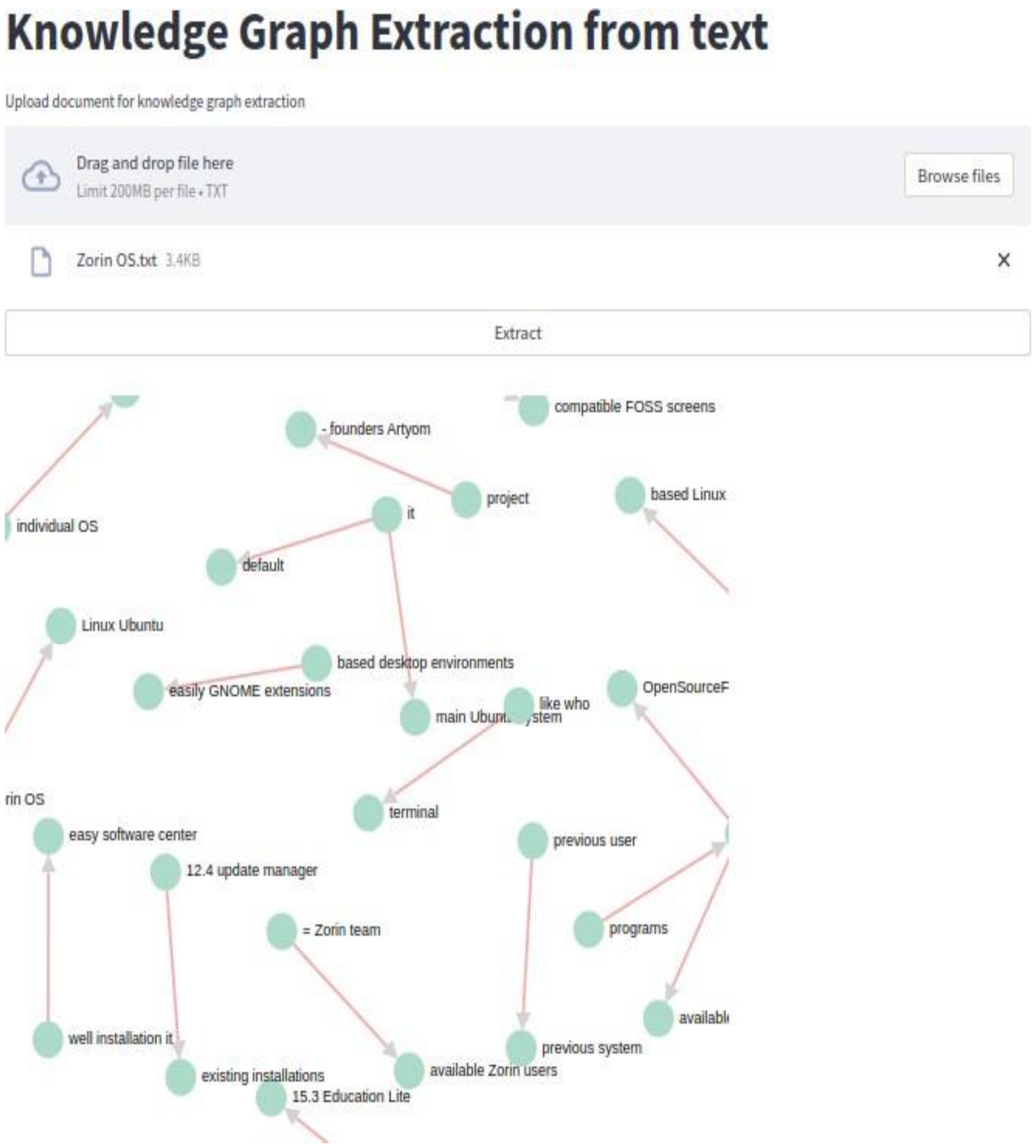
Results:

Applications of machine learning == Applications of machine learning Customer relationship management - Inverted pendulum - balance and equilibrium system ANSWER . Natural language processing (NLP) Automatic taxonomy construction Te

Relevance: 0.5293998718261719

Source: Outline of machine learning.txt

Fig 9. Extractive Question Answering


Knowledge Graph Extraction from text

Upload document for knowledge graph extraction

Drag and drop file here
Limit 200MB per file • TXT

Zorin OS.txt 3.4KB

Fig. 10. Knowledge Graph

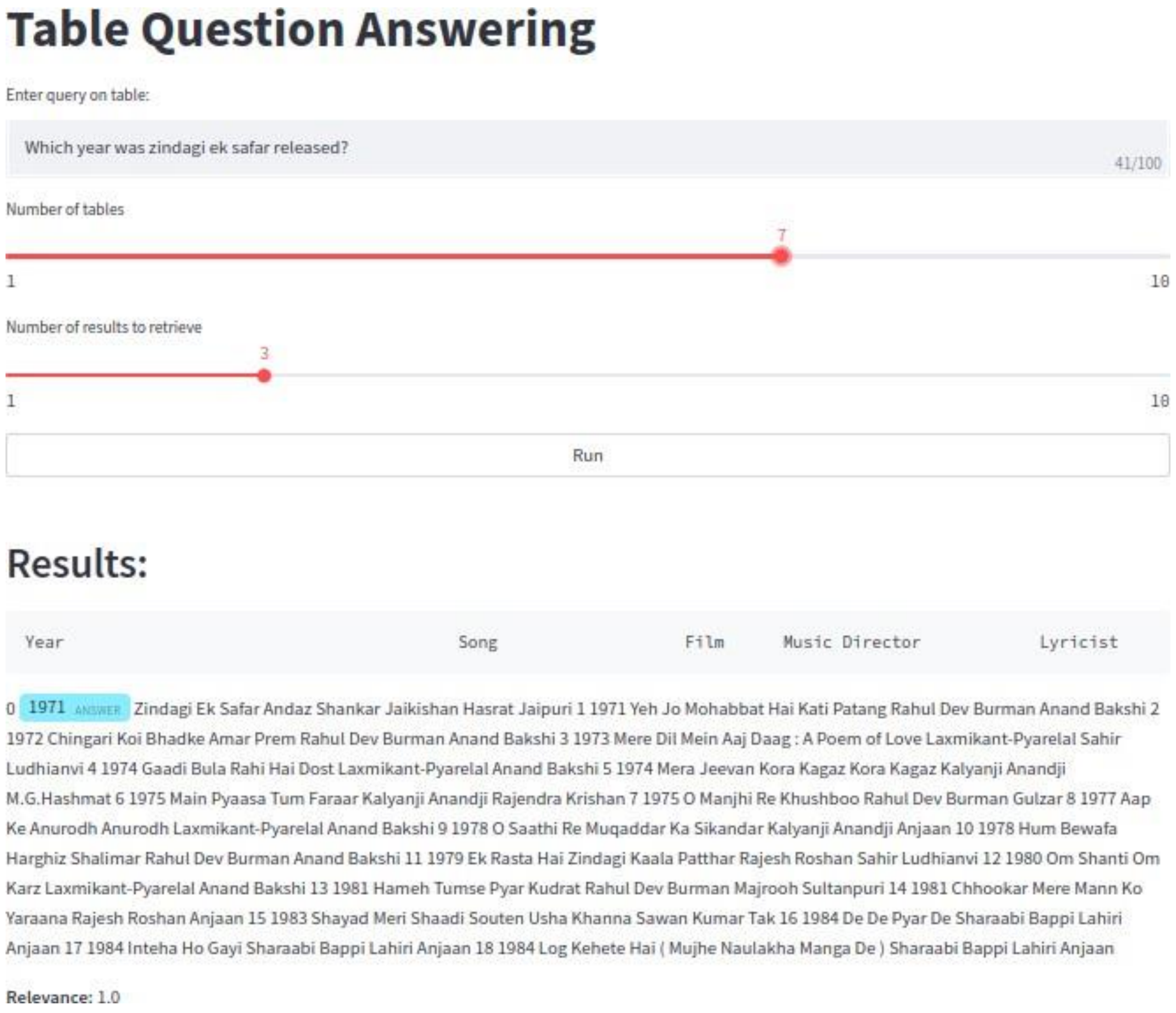


Table Question Answering

Enter query on table:

Which year was zindagi ek safar released? 41/100

Number of tables: 7 (1–10)

Number of results to retrieve: 3 (1–10)

Results:

Year Song Film Music Director Lyricist

0 1971 ANSWER Zindagi Ek Safar Andaz Shankar Jaikishan Hasrat Jaipuri 1 1971 Yeh Jo Mohabbat Hai Kati Patang Rahul Dev Burman Anand Bakshi 2 1972 Chingari Koi Bhadke Amar Prem Rahul Dev Burman Anand Bakshi 3 1973 Mere Dil Mein Aaj Daag : A Poem of Love Laxmikant-Pyarelal Sahir Ludhianvi 4 1974 Gaadi Bula Rahi Hai Dost Laxmikant-Pyarelal Anand Bakshi 5 1974 Mera Jeevan Kora Kagaz Kora Kagaz Kalyanji Anandji M.G.Hashmat 6 1975 Main Pyaasa Tum Faraar Kalyanji Anandji Rajendra Krishan 7 1975 O Manjhi Re Khushboo Rahul Dev Burman Gulzar 8 1977 Aap Ke Anurodh Anurodh Laxmikant-Pyarelal Anand Bakshi 9 1978 O Saathi Re Muqaddar Ka Sikandar Kalyanji Anandji Anjaan 10 1978 Hum Bewafa Harghiz Shalimar Rahul Dev Burman Anand Bakshi 11 1979 Ek Rasta Hai Zindagi Kaala Patthar Rajesh Roshan Sahir Ludhianvi 12 1980 Om Shanti Om Karz Laxmikant-Pyarelal Anand Bakshi 13 1981 Hameh Tumse Pyar Kudrat Rahul Dev Burman Majrooh Sultanpuri 14 1981 Chhookar Mere Mann Ko Yaraana Rajesh Roshan Anjaan 15 1983 Shayad Meri Shaadi Souten Usha Khanna Sawan Kumar Tak 16 1984 De De Pyar De Sharaabi Bappi Lahiri Anjaan 17 1984 Inteha Ho Gayi Sharaabi Bappi Lahiri Anjaan 18 1984 Log Kehete Hai (Mujhe Naulakha Manga De) Sharaabi Bappi Lahiri Anjaan

Relevance: 1.0

Fig 11. Tabular Question Answering

VII. CONCLUSION

The primary aim of the proposed work is to establish a system employing neural search techniques for information retrieval. The report outlines the steps intended for the development of this system and identifies key concepts and software tools to be utilized throughout the process. The intended expansion of the application aims to enhance clarity in ranking and accuracy of presented search results. The plan involves implementing an intranet search engine that encompasses not just text documents but also various forms of data, including images. The goal is to attain consistent outcomes across diverse data types while refining the chat-bot's capabilities to recognize casual conversation and search queries. Additionally, there is a plan to augment the chat-bot and dialogue system with contextual abilities for improved performance.

REFERENCES

[1] L. Yang, H. Zamani, Y. Zhang, J. Guo, and W. B. Croft, "Neural matching models for question retrieval and next question prediction inConversation", arXiv preprint arXiv:1707.05409, 2017.

[2] W. Wu, G. Liu, H. Ye, C. Zhang, T. Wu, D. Xiao, W. Lin, and X. Zhu,"EENMF: An end-to-end neural matching framework for e-commerce sponsored search", arXiv preprint arXiv:1812.01190, 2018.

[3] B. Mitra and N. Craswell, "Neural models for information retrieval",arXiv preprint arXiv:1705.01509, 2017.

[4] J. Guo, Y. Fan, X. Ji, and X. Cheng, "Matchzoo: A learning, practicing,and developing system for neural text matching", In Proceedings of the42nd International ACM SIGIR Conference on Research and Develop-ment in Information Retrieval, pages 1297–1300, 2019.

[5] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet:Lightweight language understanding for dialogue systems", arXivpreprint arXiv:2004.09936, 2020.

[6] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa:Open source language understanding and dialogue management", arXivpreprint arXiv:1712.05181, 2017.

[7] J. Rao, L. Liu, Y. Tay, W. Yang, P. Shi, and J. Lin, "Bridging the gap between relevance matching and semantic matching for short text sim-ilarity modeling", In Proceedings of the 2019 Conference on EmpiricalMethods in Natural Language Processing and the 9th International JointConference on Natural Language Processing (EMNLP-IJCNLP) , pages 5370–5381, 2019.

[8] X. Ling, L. Wu, S. Wang, G. Pan, T. Ma, F. Xu, A. X. Liu, C.Wu, and S. Ji, "Deep graph matching and searching for semantic code retrieval", ACM Transactions on Knowledge Discovery from Data(TKDD), 15(5):1–21, 2021.

[9] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-match: an algorithm and an implementation of semantic matching", In European semanticweb symposium, pages 61–75, Springer, 2004.

[10] Y. Fan, J. Guo, X. Ma, R. Zhang, Y. Lan, and X. Cheng, "A linguistic study on relevance modeling in information retrieval", In Proceedings Of the Web Conference 2021, pages 1053–1064, 2021.

[11] Z. Zeng, D. Ma, H. Yang, Z. Gou, and J. Shen, "Automatic intent-slot induction for dialogue systems", In Proceedings of the Web Conference 2021, pages 2578–2589, 2021.

[12] X. Li, J. Mao, W. Ma, Y. Liu, M. Zhang, S. Ma, Z. Wang, and X. He, "Topic-enhanced knowledge-aware retrieval model for diverse relevance estimation", In Proceedings of the Web Conference 2021, pages 756–767, 2021.