QUESTION ANSWERING MODULE LEVERAGING HETEROGENEOUS DATASETS

Abinaya Govindan, Gyan Ranjan and Amit Verma

Neuron7.ai, USA

ABSTRACT

Question Answering has been a well-researched NLP area over recent years. It has become necessary for users to be able to query through the variety of information available - be it structured or unstructured. In this paper, we propose a Question Answering module which a) can consume a variety of data formats - a heterogeneous data pipeline, which ingests data from product manuals, technical data forums, internal discussion forums, groups, etc. b) addresses practical challenges faced in real-life situations by pointing to the exact segment of the manual or chat threads which can solve a user query c) provides segments of texts when deemed relevant, based on user query and business context. Our solution provides a comprehensive and detailed pipeline that is composed of elaborate data ingestion, data parsing, indexing, and querying modules. Our solution is capable of handling a plethora of data sources such as text, images, tables, community forums, and flow charts. Our studies performed on a variety of business-specific datasets represent the necessity of custom pipelines like the proposed one to solve several real-world document question-answering.

KEYWORDS

Machine comprehension, document parser, question answering, information retrieval, heterogeneous data

1. INTRODUCTION

In this study, we examine factoid question-answering when a data source is restricted to a constrained domain, such as the Hi-Tech one. The agent peruses several product manuals as the data source to determine a technical answer to a client query. Product manuals can be considered a source of information that is continuously evolving when there are multiple versions and releases of a product. Manuals offer a more reliable and up-to-date source than a knowledge base with a structured source of information like FAQs, which may not be comprehensive and require time and manual effort to create and validate, but are easier for computers to digest and process. Hence manuals become the perfect candidate for a long-term and scalable system. However, these manuals are intended for humans to read rather than for machines to parse, making them more difficult to parse automatically.

In businesses, compiling product manuals requires time, effort, and coordination by several teams, which often results in the lack of manuals for a very long period. We could solve this issue if we leveraged real-time interactive data sources such as community forums and technical discussion forums where users post problems they face and community members respond with their solutions and useful links. The extraction of data from these forums can be very useful to businesses in creating a crowd sourced knowledge platform. The data, however, cannot be consumed in the raw form and has noise associated with it that needs to be cleaned and processed for our system to use this effectively. So, the community conversation threads are also an integral component of our system for extracting relevant information.

The goal of this system is to reduce the amount of time and effort required to locate relevant articles in the manual or community conversation forums, then manually navigate to the appropriate section with the most appropriate answer, as a result of manual intervention. As a result, for every user request, the system returns a set of related topics from numerous guides and conversations, as well as the parent, just like a standard question-and-answer system. This decision is based on the business rationale that several manuals may contain information relevant to your request. When a user asks a question like *What is the expected time for my battery to be fully charged?* and the solution may be found in the manuals for a variety of devices, all of those sections must be recommended to the user. The system must also be able to interpret any additional context provided by the user that aids in narrowing down the manuals - for example, if the user asks *What is the expected time for device A's battery to be charged?*, the system must recognize that *device A* is an additional context and should only be able to search manuals for *device A*.

The use of hundreds of conversation threads and user manuals for question answering involves the inclusion of a document indexer engine in the question answering system, which should be executed at scale because the questions should be addressed in real-time. As a result, the system should be able to retrieve relevant sections among hundreds of acquired manuals and conversations for each user query. Because it can process both textual (paragraphs, summaries, etc.) and non-textual (tables, images, flow diagrams, etc.) information, this system can be extended to any domain as long as manuals, documents, or even books and articles are available.

In traditional question answering scenarios, a small chunk of text can be regarded as an answer to the question asked. We cannot, however, make the same argument for our business use case. If a user inquires about *What are the steps for me to log in to a device?*, the response cannot be provided in a short segment and must be replied using an entire section titled *How to use and set up?*. As a result, the system should be able to decide whether the answer should be returned as a short sequence or as a portion of text in real-time.

In this paper, we show how various existing systems try to solve this domain based question answering by comparing their performances on a standard business dataset. We also introduce Intelligent Question Answering system which is composed of

- **Document parser**, a transformer-based deep learning model that can parse and manage a wide range of unstructured data, including images, tables, textual content etc. The parser is mainly comprised of two sub-modules to parse user manual documents and extract conversation threads from the technical forums.
- **Document indexer**, a module that uses indexed databases to index documents with essential information to keep all different types of data in a single collection, such as images, tables, and so on.
- **Document Retriever**, a natural-language-based query processor that handles several business-specific preparation processes, recognizes if the query has any "context," and gets the top relevant chunks of text from the indexed database.
- **Document Reader**, a multi-layer transformer-based model which has been fine-tuned for the task of specific domain-based question answering. The document reader also has a classifier that has been trained to decide if the answer should be a small segment or section of text.

In this paper, we study the application of several deep learning models to the question answering task. Our experiments show that the Intelligent Question Answering system outperforms traditional question answering systems on standard business-specific datasets.

1.1. Related work

The study teams first focused on factoid questions, which are queries for which answers can be extracted with certainty from a defined text source. However, there may be a few deviations to this assumption in real-world scenarios, which can be handled by other classifier modules. These teams were mostly interested in factoid questions, such as *"Where was X born?"* and *"Which year did Y take place?"*. Now, the emphasis is on solving difficult problems which one encounters regularly in real-world like *"How can Y be done?"*, *"A was moved from B to C and later to D. Where is A now?"*. These issues, in some cases, necessitate complex comprehension and context inference, as well as information flow between sentences. These challenges can no longer be solved with simple comprehension models or named entity models. The Text Retrieval Conference (TREC) has featured an annual evaluation track of question answering systems since 1999. (Voorhees 2001, 2003b). Following TREC's success, both the CLEF and NTCIR workshops began multilingual and cross-lingual QA tracks in 2002, with a focus on European and Asian languages, respectively as done by Magnini et al. 2006; Yutaka Sasaki and Lin 2005 in [8]. Other datasets, such as P. Rajpurkar, et al. 2016 [11] and P. Rajpurkar, et al. 2018 [10], that focused on question answering, were also released in the past few years.

The collection of knowledge in the field of quality assurance has expanded to the point where various models reliably address the QA domain. On the other hand, the majority of these models and strategies focus on academic data sources that have been selected by humans and follow great grammar and linguistic patterns.

Data is commonly distributed across pages or portions of pages in the real world, making parsing and further inference of this type of data significantly more difficult than in the academic setting.

There are also several advanced complete pipeline QA systems that leverage either the Web, as does QuASE (Sun et al., 2015) [13], or Wikipedia as a resource, as do Microsoft's AskMSR (Brill et al., 2002) [14], IBM's DeepQA (Ferrucci et al., 2010) [20] and YodaQA (Baudi s, 2015; Baudi s and Sediv'y, 2015) [15]. AskMSR is a search-engine-based QA system that prioritizes "data redundancy over complex linguistic analyses of either queries or probable responses," in other words, it doesn't prioritize machine understanding as we do. Few methods attempt to deal with both unstructured and structured data, such as text segments and documents, as well as knowledge bases and databases. DeepQA is one such instance. Other systems based on DeepQA, such as YodaQA, incorporate information extraction from unstructured sources such as websites, text, and Wikipedia.

This task is difficult since researchers must deal with scalability and accuracy issues. Rapid development has been made in recent years, and the performance of factoid and open-domain QA systems has greatly improved (Sasaki et al. [8], 2017; S. Schwager et al., 2019 [4] ;K. Jiang et al., 2019, [1]). Several alternatives have been offered, notably DrQA's two-stage ranker-reader system (Chen et al., 2017 [9]) end-to-end transformer-based models (S.Schwager et al., 2019) [4] and unified framework-based models to handle all text-based language difficulties (Raffel et al., 2020) [7].

QAConv, as developed by C.S Wu et al., [23] is a new dataset that uses conversations as knowledge source. The data is mainly composed of business emails, panel discussions, technical conversations and work channels. Unlike open-domain and task-oriented dialogues, these conversations are usually long, complex, asynchronous, and involve strong domain knowledge. Traditional question answering models have proven to be inefficient on this kind of data, proving the need for a custom and enhanced pipeline to address such data sources.

2. Our Proposal

Our system, Intelligent Question Answering Pipeline, is described in the following sections, and it is made up of four parts: Document Parser (1), Document Indexer (2), Document Retriever (3), and Document Reader (4) modules. 5 is a diagram of the entire architecture.

2.1. Document parser

The document parser is the input processing block that reads the data and converts it into a format that can be handled by ensuing modules. The document parser is composed of two main building blocks a) The deep learning based framework that is used to convert conversational threads from technical forums and internal chat forums to our desired format so that they can be further processed for question answering. b) The Mask RCNN based fine tuned instance segmentation model, which is fine adjusted to recognise tables and images with text contained in the document, is the core part of the document parser. In addition to the existing branch for classification and bounding box regression, the Mask R-CNN extends Faster RCNN by adding a branch for predicting segmentation masks on each Region of
Interest (RoI).

Deep learning based conversational thread parser The deep learning based conversation parser module is depicted in fig 1.

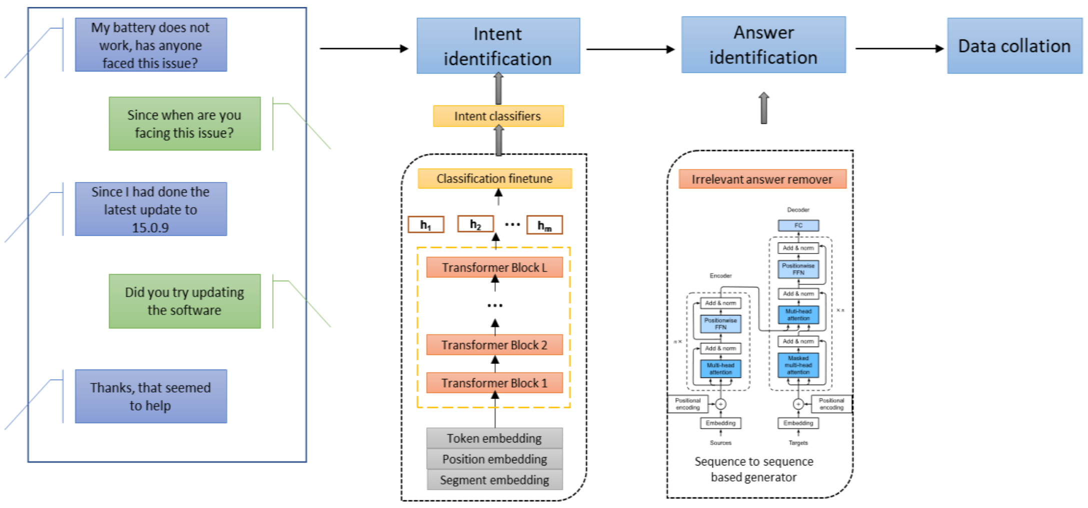

My battery does not work, has anyone faced this issue?
Since when are you facing this issue?
Since I had done the latest update to 15.0.9
Did you try updating the software
Thanks, that seemed to help
Intent identification
Intent classifiers
Classification finetune
h_1 h_2 ... h_m
Transformer Block L
Transformer Block 2
Transformer Block 1
Token embedding
Position embedding
Segment embedding
Answer identification
Irrelevant answer remover
Sequence to sequence based generator
Data collation

Fig 1. Detailed overview of Intelligent Question Answering for Product manuals

We built a specialized pipeline to process information from technical forums and community websites. The pipeline has the following characteristics :

- uses natural language inference models to identify the key sections of a conversation
- from the conversation threads, extracts the most relevant response
- compiles the information into a usable format that can be put in an indexed database with information collected from manuals.

An intent categorization model is used in the initial section of the conversational thread parser. This model is built on a BERT-based classifier that has been fine-tuned using the natural language inference dataset. As described in [26], the classifier was fine-tuned for the terms *entailment, non-entailment, and neutral*. As a zero-shot textual entailment issue, we employ this fine-tuned model to classify intents in our data. The classes for which the model was trained as "intents" are *"application related, feature update, software error" etc* . These intents were noun chunks that were the most prominent in the data and were prioritised using domain expertise provided by agents.

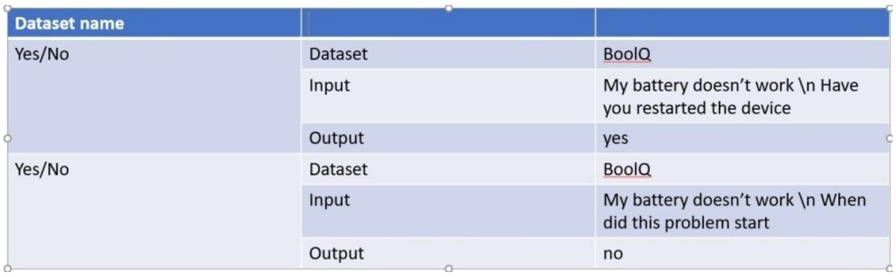
The intent identification model determines the question's intent from a list of pre-defined intents I_k and preserves messages in the conversation thread that are also part of the same intent as I_k. This removes non-relevant messages, which is the first degree of noise reduction. For example, if the user asks, *"My battery does not function, has anyone had this issue,"* the intent classifier predicts *"software error"* as the intent for the inquiry. The model then predicts intents for all individual messages in the threads, keeping just those that contain the expected intent of *"software error."*

Since these forums are typically free-text based, there is a risk of users adding non-relevant inputs to the threads. This degree of intent-based-filtering helps the model remove all the unnecessary messages such as *"Thanks"* or other random remarks in the threads.

The Answer identification module makes up the second half of the conversational thread parser. Since we do not have that information of which answer in a thread actually solved a user question, the answer identification module plays a crucial role in generating actual question-answer pairs which can be considered "completely error free" and can be indexed with the other the data sources. Once the first level of noisy messages has been removed from a thread using the intent identification module, it is critical to determine if a proposed response truly answers the user's inquiry and to keep only that.

This is performed by our question-answering system which is built on modern text-to-text framework such as T5 [27]. The data we utilised to fine-tune our model was curated in a way similar to the BoolQA dataset, in which we extracted question and answer pairs from community forums. Around 700 discussion threads were retrieved, with a total of 2500 question-answer pairs annotated from these threads. The data structure is as mentioned in 2 :

To encode the question and answer together into a single input into a text in - text out format, we first establish an uniform encoding format. The query is encoded first, followed by knowledge context - the candidate responses for each of the dialogue threads. The delimiter "\n" is used to connect these inputs into a single sequence. We don't use any prefixes, data, or task specific identifiers in the encoding or model, unlike other sequence to sequence models fine-tuned on T5. The model is trained in such a way that the answers are inferred from the context and question, which is far superior to standard question answering systems that extract



Dataset name		
Yes/No	Dataset	BoolQ
	Input	My battery doesn't work \n Have you restarted the device
	Output	yes
Yes/No	Dataset	BoolQ
	Input	My battery doesn't work \n When did this problem start
	Output	no

Fig 2. Sample data format used to fine tune T5 model

or abstract answer blocks from the provided context. The model has been trained to minimise the categorical cross entropy loss defined as

$$CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{N_c} y_i \log(\hat{y}_i) \tag{1}$$

Where y ∈ R5 is a one-hot label vector and N_c is the number of classes.

Once the model has been fine tuned to identify if an answer is really the technical solution which helps solves a question, we use this model to fully clean our data and extract question and corresponding answer pairs from conversations. This is done by passing questions and all the noise-removed candidate answers, in our desired encoded format to the fine tuned model. Since the model has been trained to return "yes" if a context actually contains the technical solution to the question, we retain only the answers in a thread which has output "yes" as the model's output. If there are more than one answers which has been predicted relevant (i.e. model output is yes), we use the prediction probabilities from the output softmax layer and keep the answer which has the maximum prediction probability. This finally helps us narrow down to one particular answer from the entire thread, removing the other noisy responses which can further be passed to the question answering modules.

Once we've extracted the correct response to each question given in the user forum, we'll need to encode it in the same format as the other data sources so it can be indexed in the same indexed databases as the others. From the entire data of the technical forums and community forums, the data collator extracts structured pairs of questions and related responses. The collator then adds suitable metadata fields to act as data source identifiers before pushing the collected data into the indexed databases.

Mask RCNN based instance segmentation model The architecture of a Mask RCNN is as depicted in 3 and 4

The premise of Mask R-CNN is simple: For each candidate object, the faster R-CNN [17] has two outputs: a class label and a bounding-box offset; to this, we

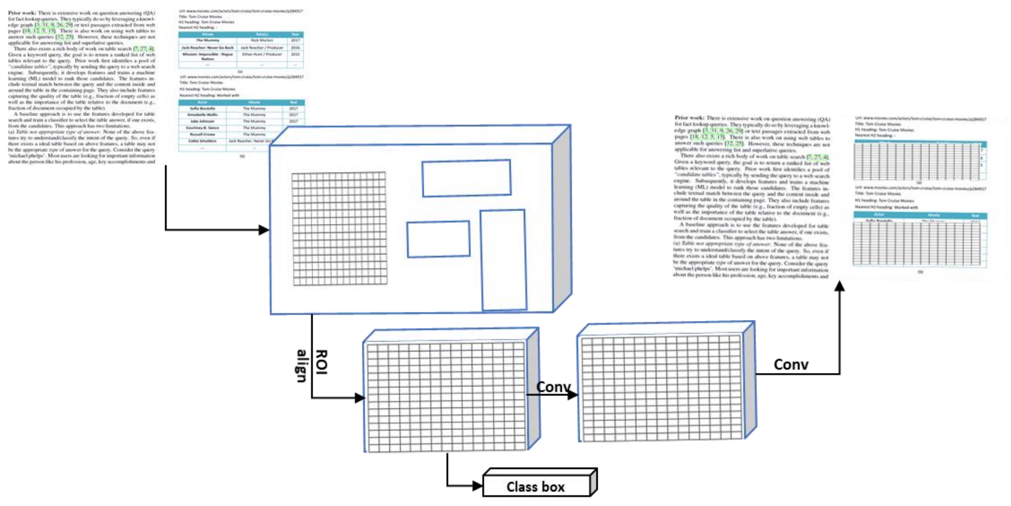


Fig 3. Mask RCNN framework for instance segmentation

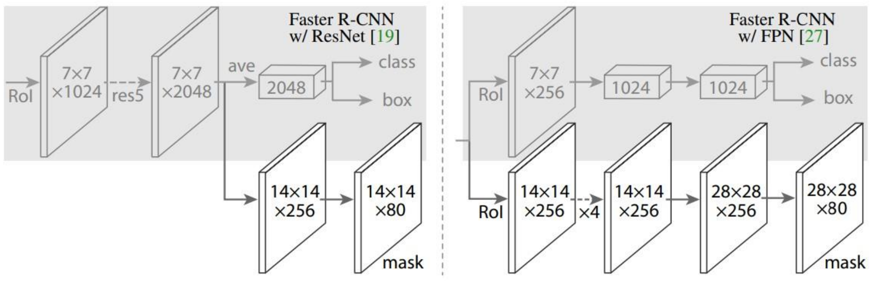


Fig 4. Head architecture of Faster R-CNN

add a third branch that produces the object mask. The additional mask output, on the other hand, is separate from the class and box outputs, necessitating the extraction of a much finer spatial arrangement of an item. Because Mask RCNN uses picture centric training, the images are shrunk to a scale of 800 pixels. Formally, a multi-task loss on each sampled RoI that is used during training is defined as

$$L = Lcls + Lbox + Lmask$$

The classification loss L_{cls} is defined as

$$L_{cls}(p,u) = -\log p_u$$

which is the log loss for the true class u and bounding-box loss L_{box} is defined as

$$L_{box}(t^u, v) = \sum_{i\epsilon\{x,y,w,h\}} smooth_{L1}(t_i^u - v_i)$$

where

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

For each RoI, the mask branch produces a Km^2- dimensional output, which encodes K binary masks of resolution m m, one for each of the K classes. A perpixel sigmoid is applied, and L_{mask} is defined as the average binary cross-entropy loss. L_{mask} is only specified on the k^{th} mask for a RoI associated with ground-truth class k. (other mask outputs do not contribute to the loss).

The mask RCNN [18] based object detector is fine tuned as depicted in 3. The following modules make up the Document Parser's primary stages:

- The RCNN model has been fine-tuned to recognise two primary objects: tables and photos with captions.
- The Document Parser then uses the fine-tuned model to divide the incoming document into three categories: tables, images with captions, and paragraph sections.
- Based on the identified object, all three sections of the document are then stored in suitable databases.

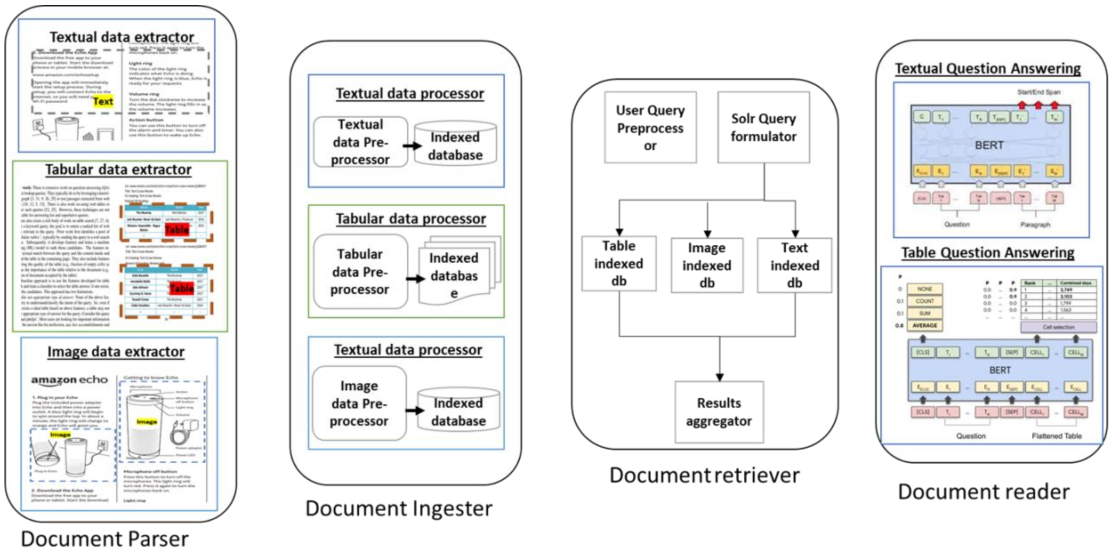


Fig 5. Detailed overview of Intelligent Question Answering for Product manuals

2.2. Document indexer

The indexer is responsible for indexing and storing the parsed data in structured databases and indexed databases. These sections are indexed with relevant indicator/metadata characteristics during index time, which can then be mapped to the object class such as table, text, and so on. We chose a Lucene-based indexer for indexed databases after examining several business characteristics such as data volume, indexing speed, and retrieval speed during query time. The document parsing and indexing is done in batches, with triggers configured to start the process when new documents are added to the source repository.

2.3. Document retriever

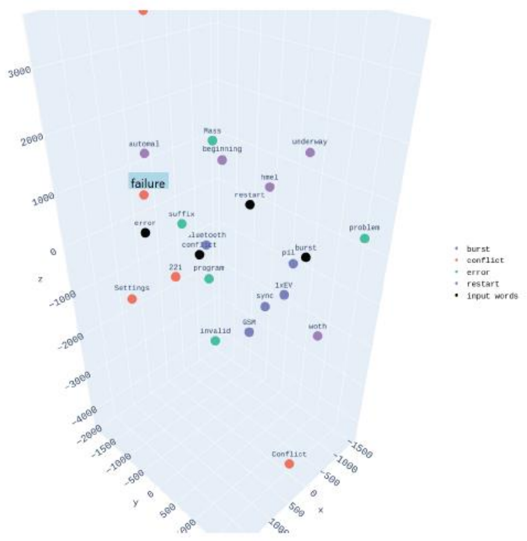


Fig 6. Fine tuned word embedding vector space

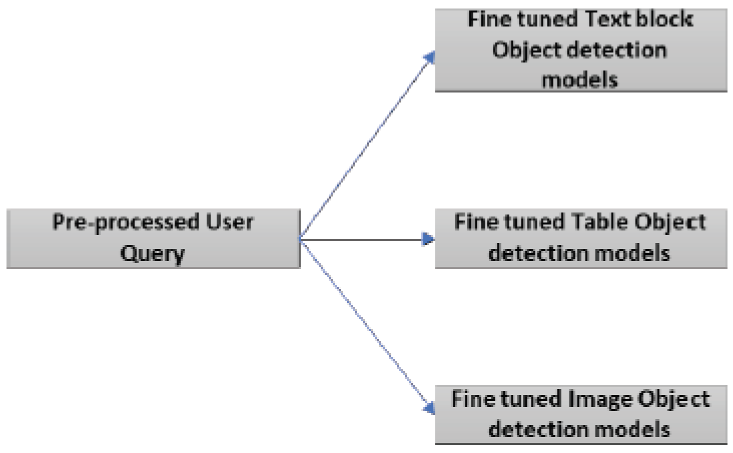

Fine tuned Text block Object detection models
Pre-processed User Query
Fine tuned Table Object detection models
Fine tuned Image Object detection models

Fig. 7. Fine tuned models to parse user query

The real-time query process is handled by the retriever and reader after the documents have been batched indexed. Despite the fact that we have thousands of documents spanning hundreds of pages, we can query through them in real time because to the document retriever's quick information retrieval. The modules that make up the retriever are as follows:

Contextualised synonyms extractor We use GloVe word embeddings [21], which have been fine-tuned on our business data, to give the retriever semantic abilities during the query stage. Aside from the fact that they seek to maximise the log probability while a context window scans across the corpus, much of the intricacies of these models have not been explored due to the brevity of this study. Although the training is done in an online, stochastic manner, the inferred global objective function can be expressed as,

$$J = -\sum_{\substack{i \,\epsilon\, corpus \\ j \,\epsilon\, corpus_i}} logQ_{ij}$$

This results in word embedding that look like 6 in the higher dimensional vector space.

Using fine-tuned word embedding, we cluster words based on their embedding to group semantically similar words together, resulting in a set of contextual synonyms that aid in semantic and contextual query retrieval.

Metadata and Context extractor A context extractor, which is a named entity model that has been trained on metadata such as product family, product line, and model name, is the next stage of document retrieval. For the objective of single phrase tagging, this named entity model was trained using a variation of BERT cite5. This model's construction is depicted in 8. After extracting the entities from the user query, the metadata information is used to enhance the retrieval by pulling information only from documents that have relevant metadata.

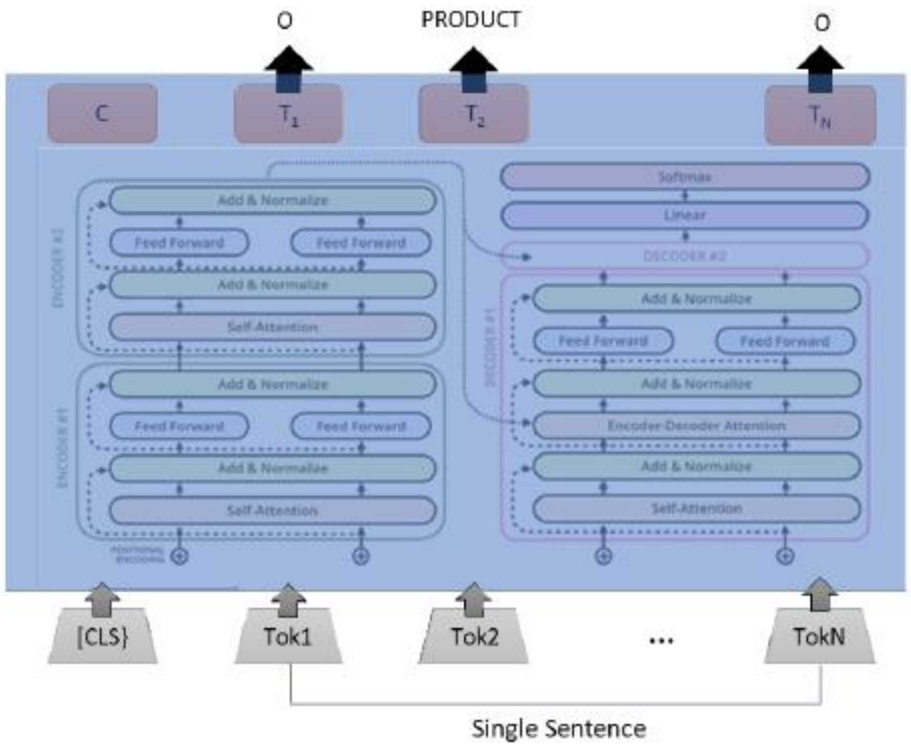


Fig. 8. Architecture for Named entity model fine Tuning

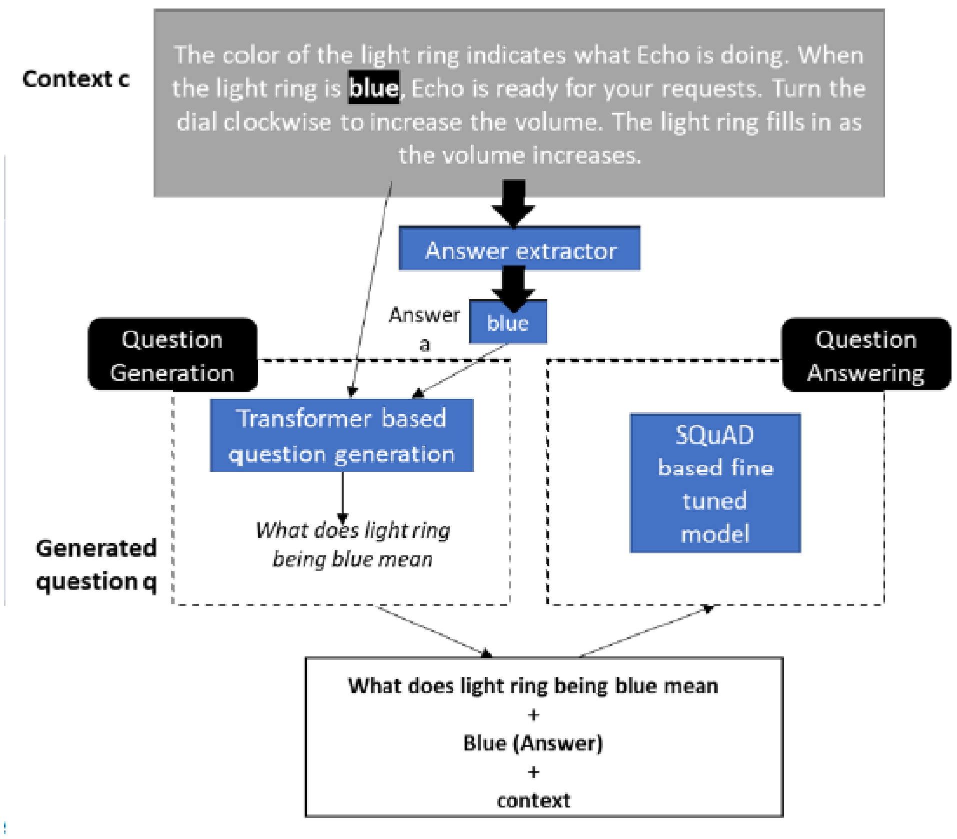

Context c
The color of the light ring indicates what Echo is doing. When the light ring is blue, Echo is ready for your requests. Turn the dial clockwise to increase the volume. The light ring fills in as the volume increases.
Answer extractor
Answer a
blue
Question Generation
Question Answering
Transformer based question generation
What does light ring being blue mean
Generated question q
SQuAD based fine tuned model
What does light ring being blue mean
+
Blue (Answer)
+
context

Fig. 9. Training data preparation for Question Answering

Tf-idf based retriever and collator The pre-processed query is then run through a Lucene-based indexed Query processor, which converts the query into a Lucene index-specific format and returns the *n* most relevant documents, along with their IDs, Tf-Idf relevance scores, and metadata. As shown in 7, the user query is delivered in parallel to all object classes.

The gathered document results are then sent into our BM25-based similarity scorer, which re-prioritizes the retrieved results as follows:

$$fScore(D,Q) = \frac{w1 * (Tf(D,Q) * Idf(D,Q)) + w2 * score(D,Q)}{w1 \ + \ w2}$$

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,D).(k1+1)}{f(q_i,D) + k1.(1-b+b.\frac{|D|}{avgdl})}$$

where

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

2.4. Document Reader

The document reader is responsible for producing the final consumable user results using the following models once the results have been retrieved and aggregated:

Fine tuned textual and Image question answering model Despite the widespread availability of pre-trained question-answering models that can answer generic queries, real-world questions typically provide less than satisfactory results. For this use case, we constructed an unsupervised question answering training dataset (similar to Cloze translation [16]), which was then utilised to fine-tune pre-trained BERT [5] models, as shown in 9.

One difference between the usual SQuAD scenario and our commercial use case is that the questions we might be asked require more detailed responses. It’s possible that the questions related to our business case are *What* or *Why* or *How* inquiries, such as *How shall I switch my phone off?* or *What is the meaning of error X?*. In the first situation, a paragraph may be required as an answer, whereas in the second, only a small portion of the paragraph may be required. To categorise incoming questions into one of these classes, a conventional Naive Bayes based classifier is employed, and a decision is made based on whether the answer to the question should be a short answer or a long one.

This approach is also applied to photographs that have text captions. The OCR parser in the image-based question-answering model extracts textual information from the document’s picture segments. The fine-tuned question-answering model is then used to extract relevant answer segments.

Tabular question answering model The Data Reader’s second section uses fine-tuned models based on the BERT architecture to answer table questions. This method is based on K. Chakrabarti et al TableQnA’s [2]. This module contains finding the correct table from a list of tables and utilising the TableQnA model to get the cell that could be the answer. The model also includes a collection of table handling and parsing algorithms that convert multiple tabular forms to the TableQnA format, as well as a post processor that returns the answer in a userfriendly format.

3. EXPERIMENTS AND DATA

We evaluated a comparison between two widely used standard approaches and our pipeline for the purposes of comparison.

3.1. Standard Tf-Idf based Retrieval Based Approach

As a domain, information retrieval has been thought of as a search engine-based task in which retrieval from indexed databases leads to the final solution. For many question types, a basic inverted index lookup followed by term vector model scoring works fairly well. With minimum contextualisation, we indexed all of the manuals' subsections into a Lucene-based indexed database. The test queries were then passed to the database as database queries, and the top relevant section of the manual was extracted as the answer.

3.2. SQuAD Based Retrieval Approaches

Traditional SQuAD-based models [4] were the second option we wanted to test to see how they performed in real-world use cases. Because the SQuAD model has only ever been evaluated on extremely small text blocks, when we apply it for larger chunks of text like Wikipedia or documents, it frequently fails in terms of both run time and performance by failing to capture the correct area of the product manual.

3.3. Comparison of Average Run Time

On a test size of 50 user questions, the methods' average execution time is as follows:

Table 1. Average run time (ms)

Tf-Idf based	SQuAD based	Our approach
100	300000	150

Because the objective is to return the response in real-time, the SQuAD approach makes it infeasible for the user to wait approximately 5 minutes for each question to arrive at the corresponding answer.

3.4. Comparison of Performance metrics

We chose measures for this experiment that would demonstrate the predicted answer's lexical and semantic similarity to the actual answer. The Rouge score was the first metric used, and it was defined as

$$ROUGE_n = \frac{\sum_{S \epsilon \{Refs\}} \sum_{ngram \varepsilon S} count_{match}(ngram)}{\sum_{S \epsilon \{Refs\}} \sum_{ngram \varepsilon S} count(ngram)}$$

where

$$count_{match}(ngram) = n(A \cap B)$$

A token t is considered to be common between A and B if the semantic similarity between t and at least one token in sequence B is greater than a pre-defined threshold. We use $ROUGE_1$ and $ROUGE_2$.

The Overlap similarity metric, on the other hand, seeks to capture the similarity between the expected and predicted answer. We define it as:

$$score(S1, S2) = \frac{\sum_{t1 \epsilon S1} \sum_{t2 \epsilon S2} \begin{cases} 1 \ if \ similarity(t1, t2) > t \\ 0 \ otherwise \end{cases}}{\sum_{t1 \epsilon S1} 1}$$

This metric denotes the percentage of tokens in the expected answer S1, which is semantically equivalent to S2. Semantic similarity between two vectors *A* and *B* is measured as

$$similarity(A, B) = \frac{\sum_{i=1}^{n} A_i . B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The performance numbers are as below:

Table 2. Performance comparison

Approach	Metric	Performance
Tf-Idf based approach	*Rouge1 - F score*	0.0838
	Rouge2 - F score	0.0514
	Overlap measure	0.2245
SQuAD based approach	*Rouge1 - F score*	0.0609
	Rouge2 - F score	0.0224
	Overlap measure	0.2041
Intelligent QnA Pipeline	*Rouge1 - F score*	0.2463
	Rouge2 - F score	0.2094
	Overlap measure	0.5918

The figures shown here are for when only the first prediction was taken into account for comparison. When even the top three results are accounted for comparison the numbers increase SIGNIFICANTLY AND WAS ACCEPTED FOR THE BUSINESS USE-CASE

3.5. Inference

The Tf-IDF approach worked well for simple scenarios, but the returned responses were too long for the end user to absorb. In majority of these cases, the solutions were only available in a subsection of the manuals, making it impractical in a business scenario to trawl through extensive passages to find the answer. Furthermore, because there were other sections with the same keywords, the answer was frequently not found in the first returned result, but rather somewhere further down. The SQuAD-based results had two major drawbacks: run time and result quality. For both simple and complicated problems, the model frequently failed to return the correct answer. This demonstrates that, while traditional methods can perform well in some situations, the smaller domain specific complexities provided into our pipeline have shown to be helpful in obtaining the right response in the least amount of time.

4. Conclusion

In this paper, we suggested a novel question-answering pipeline based on structured and unstructured data sources such as manuals, images, and product user guides in this paper. We have our main focus on extracting information from heterogeneous data sources which usually pose several practical challenges to parse and understand. We've also shown, with concrete data, how to layer bespoke domain knowledge on top of existing and traditional systems to produce contextualised outputs for a certain business domain. Our method has also been used in a variety of domain use-cases to help users find relevant answers to their problems fast.

This paper extends the work done on question answering on product manuals as studied in [24]. We apply several fine tuned models to handle heterogeneous data with significant effort and performance. One limitation of the existing approach is that the current pipeline does not allow the incorporation of human feedback for various sub modules. As described in Future Work, we expect to undertake that as part of our pipeline enhancement initiatives.

5. FUTURE WORK

In the current system, there is a simple layer of user feedback and named entities as mentioned in [25] that is used to improve the final answer generated by the automated pipeline. User signals such as feedback and more annotated data for new labels will be incorporated into our future work. Individual components of the current pipeline will perform better as a result of these signals being plugged into various pipeline sub-modules. One area of improvement, as proposed by G. Abinaya et al. [22], is designing a system capable of ingesting such feedback. The aforementioned structure will include dedicated modules that determine which section of the pipeline feedback should flow into, the cadence at which feedback should be reflected in the module, and the weights that should be assigned to feedback based on its importance, user preferences, and other factors.

For the scope of this paper, we prioritised the question answering module, which was trained for the Hi-Tech domain. Another area of consideration in the named entity recognizer module will be the incorporation of domain knowledge and domain specific jargons, starting with the generation of domain specific data using unstructured methodologies and then developing named entity classifiers utilising this data.

REFERENCES

[1] K. Jiang,D. Wu, and H. Jiang, "FreebaseQA: A New Factoid QA Data Set Matching TriviaStyle Question-Answer Pairs with Freebase," Proceedings of NAACL-HLT 2019 pages 318–323 Minneapolis, Minnesota, June 2 - June 7, 2019.

[2] K. Chakrabarti,Z. Chen, S. Shakeri, G. Cao, and S. Chaudhuri "TableQnA: Answering ListIntent Queries With Web Tables," Proceedings of the VLDB Endowment, Vol. 12, 2020.

[3] C. Deng, G. Zeng, Z. Cai, and X. Xiao, " A Survey of Knowledge Based Question Answering with Deep Learning," Journal on Artificial Intelligence 2020, No. 4 , pages 157-166

[4] S. Schwager and J. Solitario, "Question and Answering on SQuAD 2.0: BERT Is All You Need," ArXiv e-prints of 2019.

[5] J. Devlin , M.W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," ArXiv e-prints of 2018.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin "Attention Is All You Need," 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[7] C. Raffel , N. Shazeer, A. Roberts, K. Lee , S. Narang , M. Matena , Y. Zhou, W. Li and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," ArXiv e-prints of 2020

[8] Y. Sasaki, Y.Chen, K. Chen and C.J. Lin, "Overview of the NTCIR-5 Cross-Lingual Question Answering Task," Proceedings of NTCIR-5 Workshop Meeting, 2005, Tokyo, Japan

[9] D. A. Chen, J. W. Fisch, and A. Bordes. " Reading Wikipedia to Answer Open-Domain Questions" ArXiv e-prints, 2017.

[10] P. Rajpurkar, R. Jia, P. Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD." ArXiv:1806.038221v1, 2018.

[11] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang. " SQuAD: 100,000+ Questions for Machine Comprehension of Text," arXiv:1606.05250v3, 2016.

[12] W. T. Yih, X. D. He and C. Meek, "Semantic parsing for single-relation question answering," in Proc. of the 52nd Annual MTG of the Association for Computational Linguistics, Baltimore, Maryland, USA, pp. 643–648, 2014.

[13] H. Sun, H. Ma, W. Yih, C. Tsai, J. Liu, and M.i Chang. "Open domain question answering via semantic enrichment," Proceedings of the 24th International Conference on World Wide Web. ACM 2015, pages 1045–1055.

[14] E. Brill, S. Dumais, and M. Banko. "An analysis of the AskMSR question-answering system," Empirical Methods in Natural Language Processing (EMNLP). pages 257–264.'

[15] P. Baudi s and J. Sediv'y. "Modeling of the question answering task in the YodaQA system," International Conference of the Cross- Language Evaluation Forum for European Languages. Springer 2015, pages 222–228.

[16] P. Lewis, L. Denoyer and S. Riedel. "Unsupervised Question Answering by Cloze Translation," ArXiv e-prints of 2019

[17] R. Girshick. "Fast R-CNN," ArXiv e-prints of 2015

[18] K. He, G. Gkioxari , P. Doll'ar and R. Girshick. "Mask R-CNN," ArXiv e-prints of 2018 19. B.

[19] Magnini, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Pen˜as, M. Rijke, B. Sacaleanu, D. Santos and R. Sutcliffe "The Multilingual Question Answering Track at CLEF ," The Multilingual Question Answering Track at CLEF , 2006

[20] Ferrucci, David and Brown, Eric and Chu-Carroll, Jennifer and Fan, James and Gondek, David and Kalyanpur, Aditya and Lally, Adam and Murdock, J William and Nyberg, Eric and Prager, John and Schlaefer, Nico and Welty, Christopher "Building Watson: An Overview of the DeepQA Project,". AI Magazine, 2010. pages 59-79

[21] J. Pennington , R. Socher and C. Manning "GloVe: Global Vectors for Word Representation," Empirical Methods in Natural Language Processing, 2014

[22] G. Abinaya, G. Ranjan and P. Aswin Karthik, "Continuous learning mechanism of NLU-ML models boosted by human feedback," International Conference on Computational Intelligence in Data Science (ICCIDS), 2019, pages 1-6

[23] C-S Wu, A. Madotto, W. Liu, P. Fung, C. Xiong, "QAConv: Question Answering on Informative Conversations," CoRR 2021

[24] A. Govindan, G. Ranjan, A. Verma, "Intelligent Question Answering Module for Product Manuals," NATL - Nov, 2021

[25] A. Govindan, G. Ranjan, A. Verma, "Unsupervised Named Entity Recognition for Hi-Tech Domain," CNDC - Nov, 2021

[26] W. Yin, J. Hay and D. Roth, "Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach," CORR 2019

[27] C. Raffel, N. Shazeer, A. Roberts, K. Lee , S. Narang, M. Matena , Y. Zhou , W.Li and P. J. Liu , 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," CORR 2019.