Intelligent Question Answering Module for Product Manuals

Abinaya Govindan, Gyan Ranjan, and Amit Verma

Neuron7.ai, USA

Abstract. Question Answering (QA) has been a well-researched NLP problem over the past few years. The ability for users to query through information content that is available in a range of formats - organized and unstructured - has become a requirement. This paper proposes to untangle factoid question answering targeting the Hi-Tech domain. This paper addresses issues faced during document question answering, such as document parsing, indexing and retrieval (identifying the relevant documents) as well as machine comprehension (extract spans of correct answers from the context). Our suggested solution provides a comprehensive pipeline comprised of document ingestion modules that handle a wide range of unstructured data across various sections of the document, such as textual, images, and tabular content. Our studies on a variety of "real-world" and domain-specific datasets show how current fine-tuned models are insufficient for this challenging task, and how our proposed pipeline is an effective alternative.

Keywords: machine comprehension, document parser, question answering, information retrieval

1 Introduction

This study examines the challenge of factoid question-answering in a constrained situation, such as the Hi-tech domain, with several product manuals as the data source, where an agent attempts to discover a technical response to a client query by perusing the manuals.With multiple editions of a product, product manuals can be regarded as a constantly evolving source of information. Unlike a knowledge base with a structured source of information like FAQs, which are easier for computers to comprehend and process but may not be exhaustive and require time and manual effort to create and validate, manuals provide a more reliable and up-to-date source that becomes a perfect solution for a long-term and scalable system. These manuals, on the other hand, are intended for humans to decipher rather than machines, making automatic parsing more challenging.

This system intends to reduce the amount of time and effort required to find the most relevant manual that answers the user's question and locate the suitable section with the most appropriate answer by manual intervention.As a result, for any user query, the system returns a set of relevant sections from numerous manuals, rather than just the top one as standard question answering systems do. This decision is based on a business inference that multiple manuals may include information relevant to a user query. A typical case is when a user asks, ***What is the expected***

David C. Wyld et al. (Eds): NATL, NETWORKS, Fuzzy - 2021
pp. 19-32, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.112002

time for my battery to be fully charged? and the answer can be found in the manuals for numerous devices, so all of those sections must be recommended to the user. The system also needs to be able to comprehend 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 should recognise that ***device A*** is an additional context and should be able to only look through manuals for ***device A***.

The use of product 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 of instructions among hundreds of acquired manuals 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 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.
- **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**, 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 multilayer 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 An-

swering system outperforms traditional question answering systems on standard business-specific datasets.

1.1 Related work

The first type of question answering that the research teams concentrated on was factoid questions, which are questions for which the answers can be retrieved with certainty from the specified text source.However, in real-world scenarios, there may be a few exceptions to this assumption that can be handled by various classifier modules. Factoid questions such as ***"Where was X born?"***, ***"Which year did Y take place?"*** were the target area for these teams. Now, the focus is on answering complicated problems like ***"How can Y be done?"***, ***"A was moved from B to C and later to D. Where is A now?"***. In some circumstances, these questions require complicated comprehension and inference of contexts, as well as information flow between sentences. Simple comprehension models or named entity models can no longer answer these problems. Starting in 1999, an annual evaluation track of question answering systems has been held at the Text Retrieval Conference (TREC) (Voorhees 2001, 2003b). Following the success of TREC, in 2002 both CLEF and NTCIR workshops started multilingual and cross-lingual QA tracks, focusing on European languages and Asian languages respectively (Magnini et al. 2006; Yutaka Sasaki and Lin 2005 [8]). Other datasets that focused on question answering such as P. Rajpurkar,et al. 2016 [11] and P. Rajpurkar, et al. 2018 [10] were also released. The amount of literature in the general field of QA has grown to the point where there are numerous models with significant performance that reliably address the QA domain. The majority of these models and techniques, on the other hand, concentrate on academic data sources, which have been curated by humans and adhere to excellent grammar and linguistic patterns. In real business world, data is frequently dispersed over pages or portions of pages, causing parsing and further inference of this type of data to perform far worse than in the academic environment. There are also a number of 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 focuses on "data redundancy rather than sophisticated language analyses of either questions or potential responses," in other words, it doesn't focus on machine comprehension like we do. Few approaches attempt to address both unstructured and structured information, such as text segments and documents, as well as knowledge bases and databases.One such example is DeepQA. Other systems such as YodaQA which are based after DeepQA combines information extractiom from unstructured sources such as websites, text and Wikipedia. This task is challenging because researchers have to face issues in both scalability and accuracy. In the last few years, rapid

progress has been made and the performance of factoid and open-domain QA systems has been improved significantly (Chen et al., 2017;S.Schwager et al., 2019;K. Jiang et al., 2019). Several different approaches were proposed, including twostage ranker-reader systems such as DrQA (Chen et al., 2017) [2], end-to-end transformer based models (S.Schwager et al., 2019) [4] and unified framework based models to solve all text based language problems (Raffel et al., 2020) [7].

2 Our proposal

In the following, we describe our system - Intelligent Question Answering Pipeline which consists of four components: (1) Document Parser module (2) Document Indexer module (3) Document Retriever (4) Document Reader. The whole architecture is depicted in 3.

2.1 Document parser

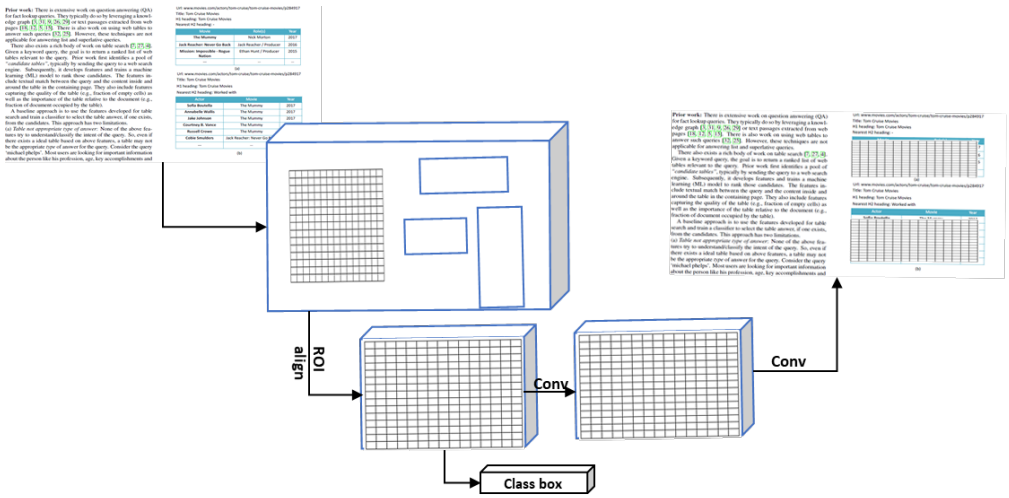

Fig. 1. Mask RCNN framework for instance segmentation

The document parser is the input processing block which is responsible for reading the data and processing it into a format that can be handled by the subsequent modules. The main parts of the document parser are the Mask RCNN based fine tuned instance segmentation model which is fine tuned to identify tables and images with text present in the document. The Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression . The architecture of a Mask RCNN is as mentioned in 1 and 2

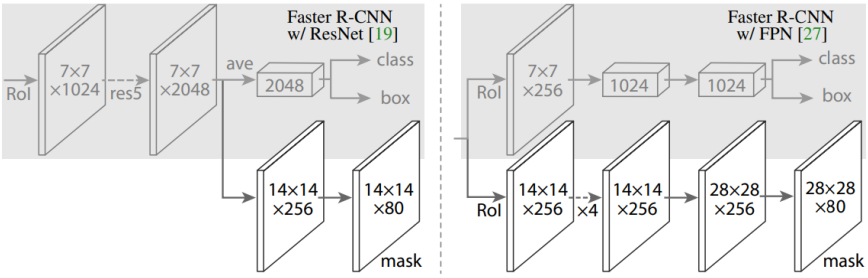

Fig. 2. Head architecture of Faster R-CNN

Mask R-CNN is conceptually simple: Faster R-CNN [17] has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask. Mask R-CNN is thus a natural and intuitive idea. But the additional mask output is distinct from the class and box outputs, requiring extraction of much finer spatial layout of an object. Mask RCNN adopts image centric training and hence the images are resized such that their scale is 800 pixels. Formally, during training, a multi-task loss on each sampled RoI is defined as

$$L = L_{cls} + L_{box} + L_{mask}$$

The classification loss L_{cls} is defined as

$$L_{cls}(p, u) = -logp_u$$

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

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

The mask branch has a Km^2- dimensional output for each RoI, which encodes K binary masks of resolution $m \times m$, one for each of the K classes. To this we apply a per-pixel sigmoid, and define L_{mask} as the average binary cross-entropy loss. For an RoI associated with ground-truth class k, L_{mask} is only defined on the k^{th} mask (other mask outputs do not contribute to the loss).

The mask RCNN [18] based object detector is fine tuned as depicted in 1. The following modules form the main stages in the Document Parser :

– The RCNN model has been fine tuned to identify two main objects - tables and images with text caption.
– The Document parser then makes use of the fine tuned model to categorize input document into three classes - tables, images with text caption and paragraph sections
– All the three sections of the document are then stored in appropriate databases based on the detected object.

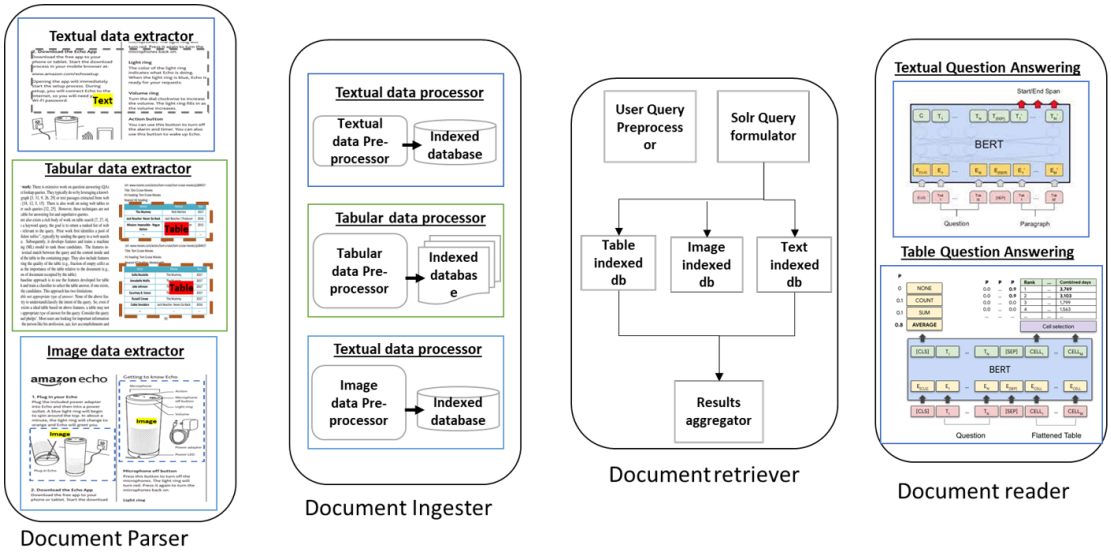

Fig. 3. Detailed overview of Intelligent Question Answering for Product manuals

2.2 Document indexer

The indexer is used to index and store the parsed data into indexed databases and structured tables. During index time, these sections are indexed with necessary indicator/metadata attributes which can later be mapped to the object class such as table, text etc. For the sake of indexed databases, we chose a Lucene based indexer after considering various business parameters such as volume of data, speed of indexing and speed of retrieval during query time. The document parsing and indexing happen at a batch level and triggers have been set to initiate the process if and when new documents get added to the source repository.

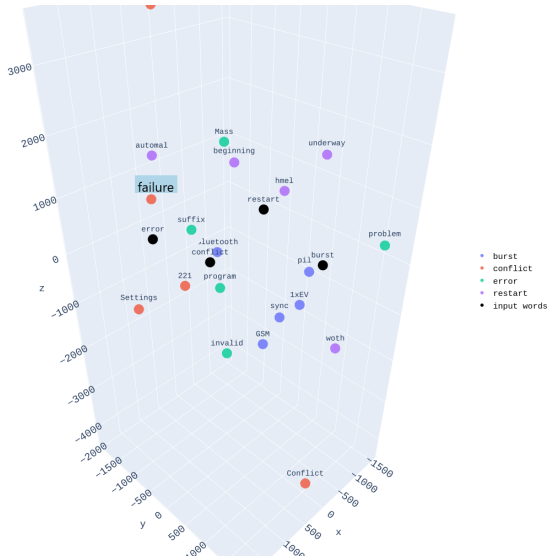

Fig. 4. Fine tuned word embedding vector space

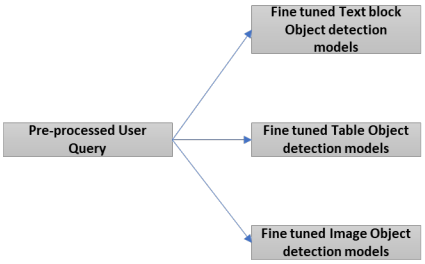

Fig. 5. Fine tuned models to parse user query

2.3 Document retriever

Once the documents have been indexed in a batched fashion, the retriever and reader take care of the real time query process. Despite having thousands of documents, which span across hundreds of pages, we have the ability to query through them real-time due to the fast information retrieval that is enabled by the document retriever. The retriever is composed of the following modules -

Contextualised synonyms extractor For the retriever to have semantic abilities during query stage, we leverage GloVe word embeddings [21] which have been fine tuned on our business data. With brevity of this paper in mind, most of the details of these models have not been discussed, aside from the the fact that they attempt to maximize the log probability as a context window scans over the corpus. Training proceeds in an online, stochastic fashion, but the implied global objective function can be written as,

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

This results in word embedding that look like 4 in the higher dimensional vector space.

Utilising the fine tuned word embedding, we cluster the words using their embedding to group semantically similar words together and hence leading to a set of contextual synonyms which helps in making the query retrieval semantic and contextualised.

Metadata and Context extractor The next stage of document retrieval is a context extractor which is a named entity model which has been trained on metadata such as product family, product line and model name. This named entity model is trained using a variant of BERT [**?**] for the task of single sentence tagging. The architecture of this model is depicted in 6. Once the entities has been extracted from the user query, the metadata information is used to further refine the retrieval by extracting information only from documents with corresponding metadata.

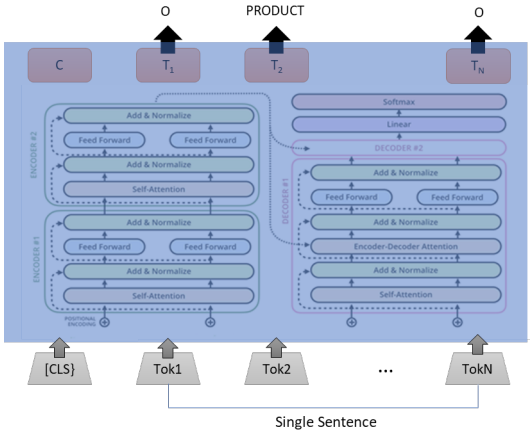

Fig. 6. Architecture for Named entity model fine tuning

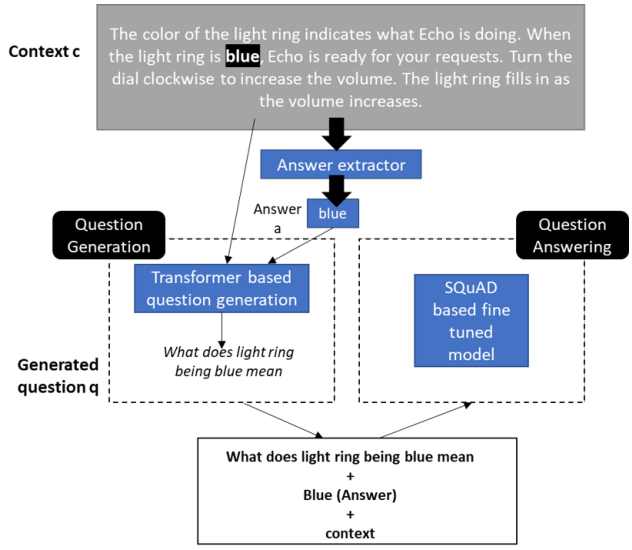

Fig. 7. Training data preparation for Question Answering

Tf-idf based retriever and collator The pre-processed query is then processed using a Lucene based indexed Query processor which encodes the query into a Lucene index specific format and retrieves n most relevant documents with their IDs, Tf-Idf relevance score and metadata information. The user query is passed in parallel to all the object classes as mentioned in 5.

The collated document results are then passed through our BM25 based similarity scorer that has been used to re-prioritise 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

Once the results have been retrieved and aggregated, the document reader is responsible to product the final consumable user results using the following models:

Fine tuned textual and Image question answering model Despite the vast availability of pre-trained question answering models that can answer generic questions, real-life questions often tend to yield less than good performance on these models. For this use case, we created question answering training dataset in an unsupervised fashion (like Cloze translation [16]) which was later than used to fine-tune pre-trained BERT [5] models as depcited in 7.

One difference from standard SQuAD scenario and our business use case is that the questions we might encounter need more descriptive answers. The questions relevant to our business case might be a *What* or *Why* or *How* question such as ***How shall I switch my phone off?*** or ***What is the meaning of error X?***. The first case might need a paragraph as answer whereas the latter needs a short segment of the paragraph to be returned as the answer. A standard Naive Bayes based classifier is used to classify incoming question into one of these classes and decision is taken accordingly if the answer to the question should be a short answer or a long answer.

This model is also used to handle images with text captions. The Image based question answering model includes a OCR parser that extracts textual information from the image segments of the document. We then use the fine tuned question answering model to extract relevant answer segments.

Tabular question answering model The second part of the Data Reader utilises fine-tuned models for table question answering based on the BERT architecture. This approach is based on TableQnA by K. Chakrabarti et al. [2] This module includes identification of the right table from set of tables and using the TableQnA model to identify the cell which can be the answer. The model also has a set of table handling and parsing algorithms which transform various tabular formats to the format desired by TableQnA and a post processor that shall return the answer in a format consumable by the user.

3 Experiments and data

For the scope of comparison, we had considered a comparative analysis between two standard approaches and our pipeline.

3.1 Standard Tf-Idf based retrieval based approach

Information retrieval as a domain has been considered as a search engine based task where retrieval from indexed databases takes us to the final solution. A simple inverted index lookup followed by term vector model scoring performs quite well on this task for many question types. We indexed all the subsections of the manuals into Lucene based indexed database with minimal contextualisation. We then passed the test queries as database queries to the database and extracted the top relevant section of manual as the answer.

3.2 SQuAD based retrieval approaches

The second approach that we wanted to compare was traditional SQuAD based models [4] to see how they fare in real life use cases. The SQuAD model has been always tested on very small text block and hence when we use this for larger pieces of text such as Wikipedia or documents, they often fail in terms of both run time and performance by failing to capture the right section of the manual.

3.3 Comparison of Average run time

The average run time taken by the methods on a test size of 50 user questions are as follows :

Table 1. Average run time (ms)

Tf-Idf based	SQuAD based	Our approach
100	300000	150

As we can see here, since the problem is to return the response in real-time, the SQuAD approach makes it impractical for the user to wait for nearly 5 minutes for each question for an answer to be produced.

3.4 Comparison of Performance metrics

For this exercise, we decided on metrics that would indicate both lexical and semantic closeness of the predicted answer to the actual answer. The first metric used was Rouge score which 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 second metric is the Overlap similarity which aims to capture the closeness between the expected and predicted answer. This is defined 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 represents the fraction of tokens in expected answer S1 which is semantically similar with 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 numbers that we see here is when only the first prediction was considered. The numbers increase significantly when even top 3 results were considered for the comparison and was accepted for the business case.

3.5 Inference

While the Tf-IDF approach worked well for straightforward cases, the returned answers were too long to be consumed by the end user. In most of these cases, the answers were present only in a small subsection of these manuals and hence going through long passages to locate the answer seems to be impractical in a business

scenario. Also, since there were many sections with same keywords, the answer was most often not present in the first returned result and was present somewhere deep down. The SQuAD based results had two main disadvantages - run time and quality of results. The model often failed to return the right answer both for straight forward and complex questions. This proves that even though traditional methods can perform well on a certain scenario, the smaller business specific intricacies that has been fed to our pipeline has proven effective in capturing the right answer in the shortest possible time.

4 Conclusion

In this paper, we proposed a novel pipeline of question answering based on structured and unstructured documents such as manuals, images and product user guides. We have also demonstrated with conclusive evidence, how to overlay bespoke domain knowledge on top of current and traditional systems to deliver contextualised outputs for a certain business domain. Our system has also been applied in a number of business fields, assisting users in quickly identifying appropriate solutions to their problems.

One limitation we discovered is that the existing approach does not allow for the inclusion of human feedback for various sub modules. We intend to do so as part of our pipeline enhancement efforts, as outlined in Future Work.

5 Future Work

There is a simple layer of feedback in the existing system that is utilised to improve the final answer generated by the automated pipeline.

As part of our future work, user signals such as feedback and additional annotated data for new labels will be incorporated. These signals will be plugged into various pipeline submodules, resulting in better performance of individual components. Designing a framework capable of consuming such feedback, as proposed by G. Abinaya et al. citeb22, is one area of improvement. The aforementioned structure will comprise dedicated modules that determine which section of the pipeline the feedback should flow into, the cadence with which the feedback should be reflected in the module, and the weights that should be assigned to feedback based on its eminence, user previliges, among other things.

We prioritised the question answering module, which was trained for the Hi-Tech domain, for the purposes of this paper. The incorporation of domain knowledge and terminologies in the named entity recognizer module will be another area of focus, starting with the generation of domain specific data using unstructured approaches and then building named entity classifiers utilising this data.

References

1. K. Jiang,D. Wu, and H. Jiang, "FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style 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. Magnini, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Peñas, 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