Crypto-Answer: a QA-System for cryptocurrency

Annika Köhler
University Regensburg
Regensburg, Germany
annika.koehler@stud.uni-regensburg.de

Markus Schmidbauer
University Regensburg
Regensburg, Germany
markus.schmidbauer@stud.uni-regensburg.de

ABSTRACT

Purpose
Cryptocurrencies as alternative investment forms are in high demand [19]. However, cryptocurrency websites are often overwhelming especially for newcomers [3][19]. In this study we want to create a Question-Answering (QA) system on cryptocurrency that is simple and effective for the user. Therefore our paper focuses on analysing different settings of our QA-system to create the best performance as well as including real time data on cryptocurrencies.
Methodology
For a comparison of different settings we created multiple QA-system pipelines with Haystack [7] in Google Colab [16]. These included the fine-tuned and the pre-trained default version of the reader [14] paired with an Elasticsearch Retriever, a Dense Passage Retriever as well as a combination of both [11]. We used a webcrawler [10] to create a document collection with cryptocurrency websites. Consequently these documents were pre-processed and stored in the Elasticsearch document-store [6]. Furthermore, we created a website connecting to our Google Colab notebook for running queries as well as updating the documents with real time data. For the evaluation of our QA-system probands created a ground truth with 100 gold standard question-answer pairs and assessed the answers returned by our QA-system [23].
Results
Comparing the three retrievers, in combination with the fine-tuned version the Joined Retriever has the highest top-3-accuracy with 82%, the second best is the Elasticsearch-Retriever with 70% and the Dense-Passage-Retriever reaches a value of 68%. By comparing the accuracy of the fine-tuned and pre-trained default reader, the fine-tuned version paired with the Joined Retriever provides higher values with a top-3-accuracy of 82% than the default version with 66%. The fine-tuned reader combined with the Joined Retriever has a lower probability of 18% to return no answer, while the default reader has a probability of 34%. The biggest difference is seen considering the exact match value of the Joined Retriever as it has tripled from 21% (default reader) to 62% (fine-tuned reader).

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Information systems** → *Users and interactive retrieval*; • **Human-centered computing** → *Empirical studies in HCI*.

KEYWORDS

cryptocurrency, bitcoin, question answering, haystack, fine-tuning, webcrawler, pipeline

ACM Reference Format:
Annika Köhler and Markus Schmidbauer. 2021. Crypto-Answer: a QA-System for cryptocurrency. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cryptocurrencies are digital currencies, which represent a demanded alternative to conventional investment forms like shares, real estate, bank books or gold. One reason for the high demand is the decentralization and limitation of cryptocurrencies preventing State interference and inflation losses [19]. Since 2009 the most demanded cryptocurrency is Bitcoin. It ran through an increase in value from 0.0008 US-Dollar to around 65.000 US-Dollar in April 2021. Since then there are thousands of cryptocurrencies of which it is easy to loose track. Strong exchange rate fluctuations require a timely analysis of the cryptocurrency market [19]. Cyptocurrency-websites which display data including information about the price or the volume of each of the thousands currencies are confusing and overwhelming especially for newcomers [3][19]. A Question-Answering (QA) system for cryptocurrency can fix this problem. The purpose of our QA-System is to be simple and effective for a user to recieve an answer to a question on the topic of cryptocurrency. Studies have shown that it is important to users that questions can be phrased in a natural manner [17].

There are multiple libraries in order to support the creation of QA-Systems including Haystack [7] and HuggingFace [13]. Haystack is a library enabling the build of pipelines for search use cases, including QA-Systems. For the key components of Haystack's pipelines, including Indexing, file-conversion, pre-processing, document-storing, retrieving and reading, Haystack has multiple parameters that can be optimized for individual use cases [7]. Besides that Huggingface enables the building, training and deploying of state of the art models for search systems [13].
Various Question-Answering systems have been implemented in the past, including e.g. a Covid-19 system called COBERT [24], as well as an B2C eCommerce QA-system [15] using BERT. However there is still little known about which methods, tools and parameters are best used for each individual use case.

This study will explicitly go into further detail about which settings for our cryptocurrency QA-architecture returns the best results as well as how to include the use of real time data. Therefore, using Haystack [7] as a basis, we built multiple versions of our QA-System upon different options for key components and designed our own architecture. First we applied a webcrawler [10] to create a document collection. As starting pages we used websites with real time data of cryptocurrencies [3], frequently asked questions [2] as well as Wikipedia articles [29]. We fine-tuned our reader, a pre-trained Huggingface RoBERTa QA model [14], with our own question-answer pairs to get better results. Following that, we implemented three versions for our retriever, a sparse-model, a dense-model and a combined retriever for our pipeline. After implementing the QA-systems, we created a ground truth with gold standard question-answer pairs by recruiting probands. Consequently, we could evaluate if fine-tuning improved our reader and which of the three retrievers achieved the best results.

Analysing the QA-architectures and their corresponding tools might help information scientists to better understand the selection of tools for the implementation of an individual use case, the issues that can arise and how they can be fixed. Furthermore, we demonstrate the use of real time data from the World Wide Web in our QA-architecture design. Precisely, we built a QA-system that focuses on satisfying the information need of the user by retrieving corresponding answers to cryptocurrency-related questions.

2 RELATED WORK

In Information Science Question-Answering, as retrieving answers to natural phrased questions, is an important topic of research [28]. The main purpose of QA-systems is to fulfill human information needs. There are multiple types of implementations, including knowledge based systems, by querying structured databases as well as systems based on unstructured collections of natural language documents [4]. As described by Krishnamoorthy [28], Natural Language Processing (NLP) made a huge leap forward in recent years. One key system that benefits from this is Question-Answering. While traditionally QA-systems used rule-based and statistical methodology, the use of new developments like neural networks and deep learning achieve far better results [28]. Before addressing diverse methods of implementing QA-systems, another relevant topic are the frameworks and models, which build the basis for QA-systems.

A language representation model in particular is the Bidirectional Encoder Representations from Transformers (BERT) [21], which introduces pre-training on unlabeled data as well as fine-tuning on labeled data. BERT as a pre-trained model provides fine-tuning on individual data to create NLP systems like Question-Answering. With this technique it claims to be "the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures"[21]. The introduction of BERT lead to the innovation of various other libraries.

Among those Huggingface [13] has a collection of datasets and models that can be used to build, train and deploy BERT-based systems. In our implementation we use the roberta-base-squad2 model to train our system [14].

Another library based on BERT is Haystack [7]. "Haystack is an open-source framework"[8] and an end-to-end tool for building usable architectures for search systems. A basic pipeline of Haystack includes a document-store, containing the collection of relevant text files, as well as a retriever that filters these documents by the highest probability to include the answer to a question. Following that, the reader calculates which text sections are most likely to contain the answer. Reader and retriever can be modified and fine-tuned to achieve better results for an individual system.

Chen et al. demonstrated in their study that training QA-systems on Wikipedia articles can deliver great results [18]. SQuAD 1.1 [20] is a dataset trained on Wikipedia articles with more than 100,000 question-answer pairs. The more recent version 2.0 [27] extends SQuAD 1.1 by adding 50,000 unanswerable questions. With the obtained knowledge SQuAD 2.0 is even able to determine if no answer can be given to a question.

On the other hand studies have shown that working with data from the World Wide Web could lead to complex challenges. One of the reasons is that websites do not follow a common pattern. Therefore, extracting data and especially filtering the important data from the unstructured format of websites is an issue that people try to prevent by adding pre-processing steps. Another challenge in that regard is the ambiguity of words, the content of websites as well as the user questions [22][1].

Xie et al. proposed a Chatbot and QA system for cryptocurrency based on BERT. As underlying structure, BERT's Transformer architecture has an encoder and decoder for pre-processing text inputs and generating predicted results. Xie et al. loaded the question-answer pairs into an embedding process to fine-tune with the pre-trained BERT model. While training, the model learns a specific embedding to be able to distinguish between these sentences. The Chatbot was developed to support the crypto-market by conversing and answering questions of the user regarding cryptocurrency [26].

Another conversational agent by Xie et al., a "Chatbot Application on Cryptocurrency" [25] has been implemented for interested parties and investors to get information about cryptocurrency and their capabilities. The question-answer pairs were generated using community Question-Answering websites. This dataset was also trained by using the BERT model. In addition Xie et al. implemented coin-market API calls to download the real time prices from the first 25 cryptocurrencies. This data was additionally used to answer the users' questions. The Chatbot showed great results answering questions about the price trend of a cryptocurrency [25]. It is a convenient implementation for the user to aquire information about a cryptocurrency.

Chatbots have been implemented [25][26] to help providing reliable knowledge about cryptocurrency, since a lot of terms are still unfamiliar to the general public [26]. With our QA-system we do not only want to focus on answering questions to gain new knowledge about the field, we also want to create an everyday helper. Therefore we use real time information on cryptocurrency,

like the price or the volume, which is needed for investors who want to exchange and buy cryptocurrencies.

3 METHODOLOGY

The goal of our study is to create a Question-Answering system for cryptocurrency that has the best performing pipeline and additionally answers questions using real time data. Therefore we used Haystack [7] as a basis for our QA-system. Haystack provides tools for creating an individual QA-Pipeline. The tools that can be modified and used are document-stores, retriever, reader, pre-processor and a webcrawler.

The websites we use for answering the questions are Wikipedia articles [29], official support websites from Blockchain [2] as well as cryptocurrency websites like CoinMarketCap [3] to recieve the real time data of cryptocurrencies.

To obtain the relevant articles, we modified Haystack's webcrawler in order to work on our integrated development environment Google Colab [16]. We used two starting websites, Blockchain [2] and CoinMarketCap [3] and let the webcrawler obtain their sub-links to recieve all relevant documents linked to the first one. This method is shown in Figure 1. We crawled CoinMarketCap [3] for the first 100 most demanded cryptocurrencies. Blockchain [2] contains websites regarding frequently asked questions. For information from Wikipedia we started with the cryptocurrency Wikipedia article [29]. Crawling the article included too many irrelevant sub-websites. Consequently we only used a collection of the relevant articles as starting pages. These sites were not further crawled. While crawling the websites we collected all the articles' links to be able to reload the websites for real time data at a later point.

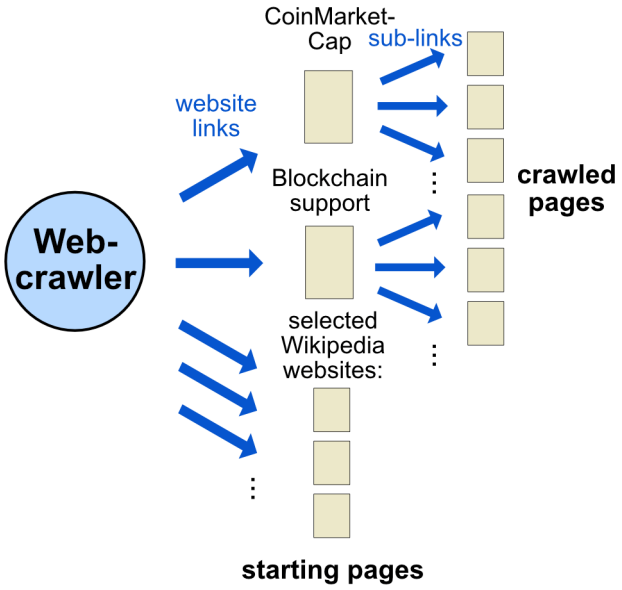

Figure 1: Web-crawler

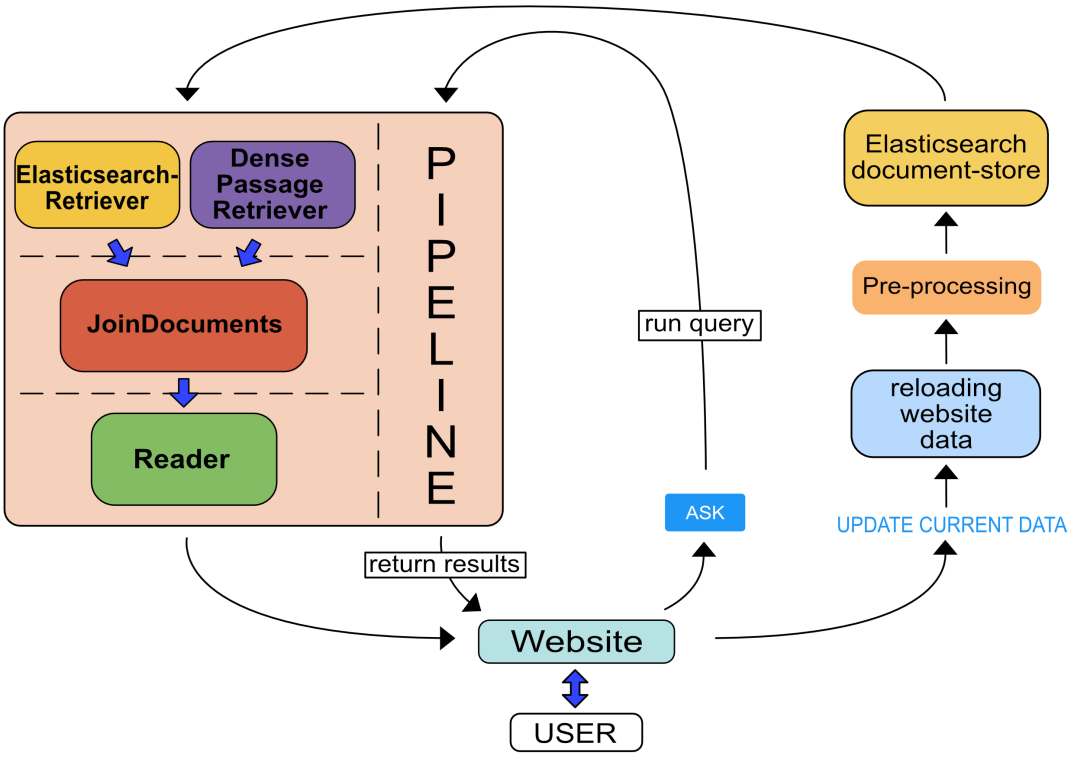

Figure 2: Architecture for our QA-System

Following, we pre-processed the articles' data for optimizing reading and retrieving. The pre-processing steps we used were converting the data into a dictionary, cleaning unnecessary escape-sequences as well as splitting longer texts into multiple smaller paragraphs [10].

Haystack also provides document-stores like e.g. the Elasticsearch or In Memory document-store to save the pre-processed dictionary of the articles' data. For our QA-system we used the Elasticsearch document-store that supports all retrievers including BM25, the Elasticsearch sparse retriever, as well as the Dense Passage Retriever [6].

Furthermore, Haystack's framework supports the building of Retriever-Reader Pipeline systems. The retriever determines which documents are relevant. Following, the reader looks through the relevant documents to extract possible answers [9].

For our QA-system we compared three different retrievers to get the best result. We implemented one system which uses a sparse retrieval method with Haystack's Elasticsearch-Retriever, one system that uses a dense retrieval method with Haystack's Dense-Passage-Retriever and one system that combines both retrievers by joining their results [9]. These retrievers are connected to the Elasticsearch document-store, to return the dictionary entries of the websites which have the highest probability of containing the right answer [6].

For our pipeline we fine-tuned the reader to optimize the answers on our document collection. As a base for the reader we used the roberta-base-squad2 pre-trained model from Huggingface [14]. This model is built upon the SQuaD2.0 dataset, which is trained on Wikipedia articles [14][27]. Therefore it is well suited for our document set. For the fine-tuning we used the Annotation tool provided by Haystack from Deepset-ai [5]. With this tool it is possible to create questions and mark their corresponding answers in uploaded documents. In total we created over 350 gold standard question-answer pairs to fine-tune our reader. To determine if fine-tuning creates better results, we analysed the performance of the reader with and without fine-tuning of the roberta-base-squad2 model.

To complete the pipeline we added the retriever component, in case of multiple retriever, the document-joiner, as well as the reader component. Running a query on the pipeline leads to the retrieval of the highest probable corresponding answers [9].

Our Google Colab notebook [23] is connected to a website, which is used for the user interaction. The website displays an input-field to type in a question. When clicking on the "ASK" button the question is handed over to our QA-system to run the query on our pipeline. Our QA-system returns the results to the website to display in the user interface. Our website also includes the button "UPDATE CURRENT DATA", which reloads the information by using the articles' links we saved while crawling the websites for real time data. Consequently the pre-processing steps have to be repeated, as well as the update of the document-store and creation of the pipeline based on the new document-store's content, as seen in Figure 2. We display the first three most probable answers (corresponding to rank 1-3) returned by our QA-system, as studies have shown that the top 3 results of a search engine are more relevant for users than lower rank results [12].

For the evaluation of our QA-system we created a Ground truth of 100 gold standard question-answer pairs. To minimize confirmation-bias, we recruited five probands for a set of questions and their corresponding answers regarding cryptocurrencies. The probands are experts as well as newcomers on the topic of cryptocurrency thereby attaining a variety in the collection of questions. The gold standard QA-pairs were used to analyse the exact match value of all the systems we evaluated. This includes all three retriever models on our fine-tuned reader and on the default reader. To calculate the accuracy and compare the answer distribution we let the probands analyse the results of our QA-system to all their questions. They had to assess if and at which rank one of the answers satisfied their information need .

4 RESULTS

Probands were asked for a set of questions and their corresponding gold standard answers to create a ground truth. Even though Haystack provides tools for the calculation of the metrics, like e.g. the exact match or the top-n-accuracy values, we calculated these values with the help of our probands. This is due to the fact that our QA-system includes the use of real time data like the price of cryptocurrency. An exact match metric would need a gold standard question-answer dataset of real time information to enable the evaluation. Therefore the probands were asked to assess if the answers to their questions returned by the different settings of our QA-system satisfied their information need, at which rank and if the question was answered as an exact match. An answer is considered an exact match, if it precisely included the ground truth gold standard answer.

To determine which system works best, we tested our fine-tuned as well as our default reader with our three different retriever models. For the evaluation of our systems we analysed the answer distribution, the top-n-accuracy, the Mean Reciprocal Rank and the exact match values.

As can be seen by comparing Figures 3 and 4 fine-tuning the reader has a noticeable impact on the answer distribution. The probability concerning questions where no correct answer was returned is reduced and in case of the Joined Retriever combined with the fine-tuned reader nearly halved. While for both readers the rank 2 and 3 results are nearly similar, the rank 1 values increase for the fine-tuned reader. In detail the value for the Elasticsearch Retriever (ESR) rises by nearly 20% and for the Dense Passage Retriever (DPR) as well as the Joined Retriever by more than 30%. Comparing the three retrievers the answer probability at rank 1 for both readers is highest for the Joined Retriever. Especially for the fine-tuned reader the probabilities of the DPR and ESR are nearly similar (55% and 54%) and by ~17% lower than the Joined Retriever. A remarkable difference is seen in the answer probabilities of rank 1 compared to rank 2 and 3 for both readers. For example the percentage of correctly answered questions for the fine-tuned reader paired with the Joined Retriever at rank 1 is larger by 80%.

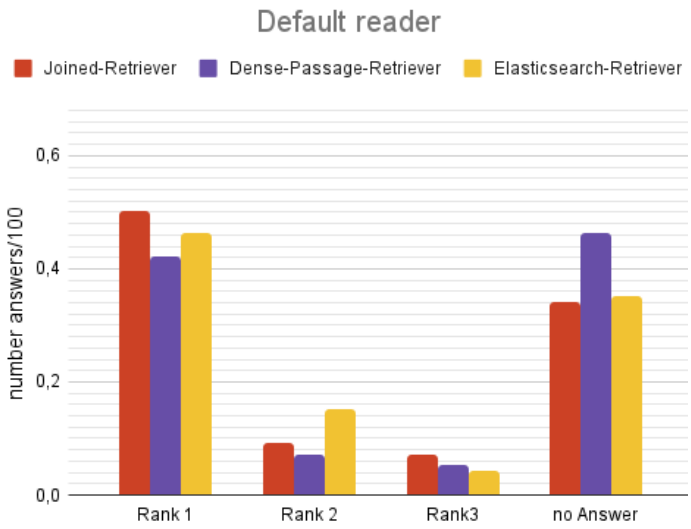

Figure 3: answer distribution for the default reader

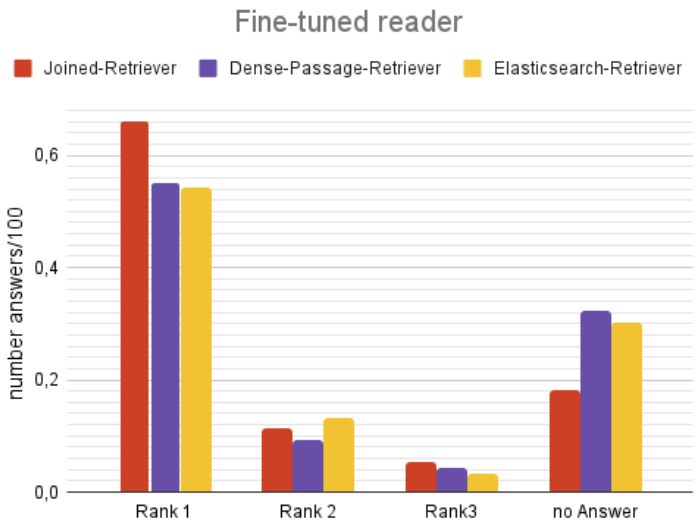

Figure 4: answer distribution for the fine-tuned reader

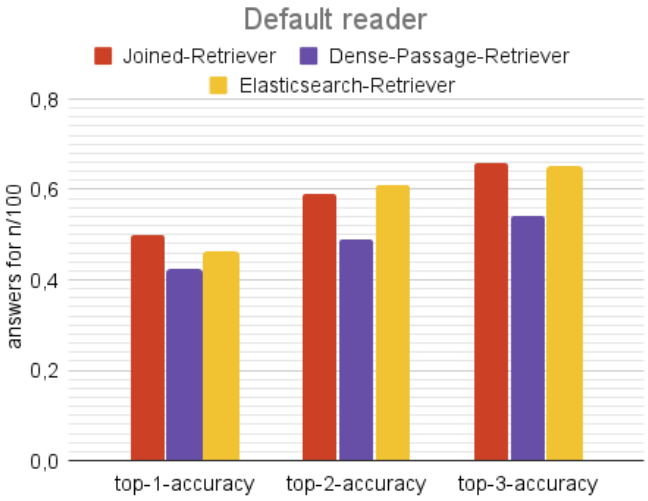

Figure 5: top-n-accuracies for the default reader

Figure 6: top-n-accuracies for the fine-tuned reader

In Figures 5 and 6 the top-n-accuracy for the first three ranks is shown. Comparing both readers, the top-n-accuracies of all retrievers increase when using the fine-tuned reader. While the ESR rises only slightly, the DPR and Joined Retriever system increase by ~30%. For the fine-tuned reader all top-n-accuracies of the DPR and the ESR are nearly similar at each n, while all top-n-accuracies of the Joined Retriever system are remarkably higher by ~20%. All top-n-accuracies for the fine-tuned reader at each n are noticeably high. The accuracy increases from 66% at n=1 to 82% at n=3 in combination with the Joined Retriever.

In addition the Mean Reciprocal Rank (MRR) is illustrated in Figures 7 and 8. The default reader shows nearly similar values for the different retrievers. Values are increasing using the fine-tuned reader, with the highest value given paired with the Joined Retriever (73%).

Next to the Mean Reciprocal Rank, Figures 9 and 10 display the exact match values. When using the fine-tuned reader the values of the DPR and the ESR double, while the value of the Joined Retriever triples and exceeds the other retriever systems by 0.22.

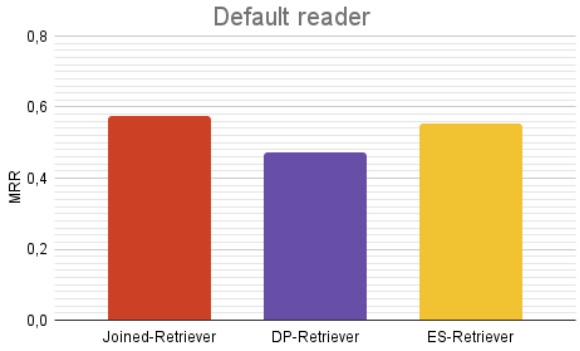

Figure 7: Mean Reciprocal Rank for the default reader

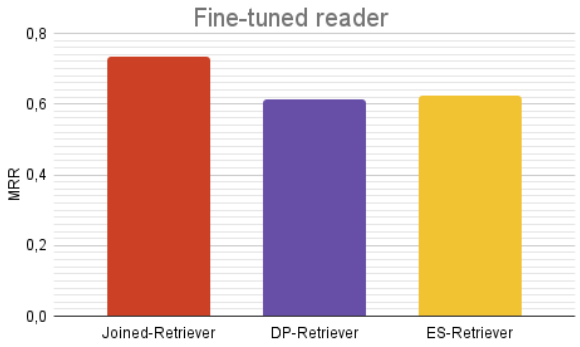

Figure 8: Mean Reciprocal Rank for the fine-tuned reader

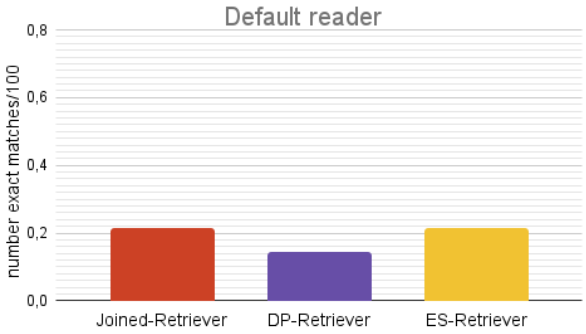

Figure 9: Exact match values for the default reader

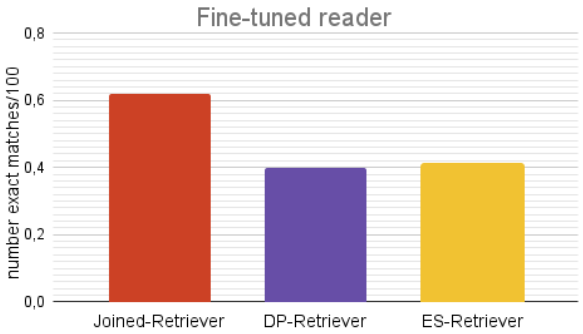

Figure 10: Exact match values for the fine-tuned reader

5 DISCUSSION

Our evaluation allows us to determine the most effective settings for our QA-system. It turns out that the fine-tuned reader in combination with the Joined-Retriever provides the best performance.

Fine-tuning our reader has a great impact on the results of our QA-system, as the number of questions, where no correct answer was returned, has a significant decrease. In case of the Joined Retriever the number of unanswered questions even halve from 34% to 18%. In addition, the rank 1 values especially for the Joined Retriever system increases remarkably. In conclusion the system with the Joined Retriever delivers the best results for satisfying the users' information need.

The system including the Joined Retriever as a combination of the Dense Passage Retriever (DPR) and the Elasticsearch Retriever (ESR) had the highest values for all metrics. That is due to the fact that the documents retrieved from both systems were combined. This means, if questions were answered correctly by only one of the systems, the system with the Joined Retriever also included these answers. As a result this leads to an extended collection of correct answers from both the DPR and the ESR systems. This is also shown by calculating the top-n-accuracy of the systems including the different retrievers. In combination with the fine-tuned reader, the Joined retriever reaches the highest top-3-accuracy of 82%.

With 66% for the top-1-accuracy, it can be seen that the greatest proportion of questions is already answered at rank 1. In addition the Mean Reciprocal Rank (MRR) demonstrates the shifting of the answers to higher ranks, caused by the fine-tuning of the reader. Especially the Joined Retriever system rises from 57% to 73%, as more questions are answered at rank 1. A study shows that users expect the answer at rank 1 to be most relevant and correct. This is due to a trust bias in search engines to return the most relevant answer at rank 1 as users view website results from top to bottom [12].

Furthermore, the exact match values display the biggest difference between the fine-tuned and the default reader. While the default reader in combination with the Joined Retriever has a score of 21%, using the fine-tuned reader results in a triplication up to 62%. For the exact match values we compared the gold standard answers to the answers our QA-system produced. If the answer from the gold standard was included precisely in the answer of our QA-system, the answer was classified as an exact match. One of the reasons why the exact match values are less for the default reader than for the fine-tuned reader is due to the fact that the default version generally kept answers short and compact. However, some of our gold standard questions required a detailed answer that needed a longer, more complex explanation. Consequently they could not be counted as exact matches. Fine-tuning the reader on our own document collection resulted in longer and more detailed answers.

In-depth questions benefited from this adaptation, as complex answers to these questions were returned. As a result of that, exact match values for the fine-tuned reader were comparably high.

With our QA-system we want to focus on simplicity and efficiency for the end-user. This is why we include only necessary elements on our website. For the effectiveness the system should also update the website data every time a user asks a question. However, updating the data for real time information by reloading the websites, pre-processing, overwriting the document-store and initialising the pipeline on the new data has not been optimized yet, as it currently takes a few minutes. For that reason reloading the data every time before asking a question is not efficient. With the separation of both functions, asking questions and reloading real time data, the user needs to only update the data once for each search session. Asking a question is therefore more efficient as it takes our QA-system only about two seconds to return the results. Consequently updating the data efficiently is of great importance. It is possible to improve the performance with enhanced computing power and faster internet connections. At the same time libraries, procedures and tools could be refined or developed to significantly increase the performance for loading real time data.

Another aspect for improving our QA-system is adding further gold standard question-answer pairs to the already existing set of 350 to fine-tune the reader. As a result the probability of 18% to return no correct answer could be reduced even further.

Additionally the Pre-Processor could also be enhanced. While the Pre-Processor already structures the text, other unnecessary input, like e.g. advertisements, could additionally be removed.

Another possibility enhancing the accuracy is including more websites using the webcrawler in order to have a wider range of documents to choose and agree on the answers. However, with the extension of the website collection, the update time will also increase correspondingly. Contrary the performance needs to be improved as more data needs to be processed.

6 CONCLUSION/FUTURE WORK

The main goals of our paper were to create a QA-system for cryptocurrency, which includes real time data as well as analyze which settings deliver the best performance. The comparison between all the systems with different settings demonstrated which one returned the best results. We conclude that for our QA-system the Joined Retriever in combination with the fine-tuned reader has the best performance. It delivered a top-3-accuracy of 82% and is therefore the best QA-system for our use case. Even though all settings produced great results, combining two retrievers extends the general collection of correct answers. Fine-tuning the reader also improved the performance of our QA-system. This is due to the fact that our system is based on a precise document collection and one specific topic, cryptocurrency.

The tools provided by Haystack [7] enabled us to create our QA-system. With the modification and customization of the provided pipeline elements, it was possible to adjust the tools in accordance to our own use case. A complex challenge was loading the real time data before each question, as it still takes too long to be applicable with our QA-system. Reducing the time it takes to reload the data as well as the corresponding pipeline elements is therefore an important topic for future studies.

With our paper, we hope that researchers as well as data scientist can obtain more insight about the performance of the different settings for QA-architectures and use our study as a reference for further research. In addition our application can be used by cryptocurrency investors or newcomers as an everyday helper, to obtain real time information on cryptocurrencies and answer questions about the topic.

REFERENCES

[1] Watanabe Y.; Dhingra B. and Salakhutdinov R. 2017. Question Answering from Unstructured Text by Retrieval and Comprehension.

[2] Blockchain. 2021. Cryptocurrency FAQs. Blockchain Ltd. Retrieved September 15, 2021 from https://support.blockchain.com/hc/en-us/categories/201149143-Cryptocurrency-FAQs

[3] CoinMarketCap. 2021. CoinMarketCap. CoinMarketCap OpCo. Retrieved September 15, 2021 from https://coinmarketcap.com/

[4] Jurafsky D. and Martin J. H. 2020. Speech and Language Processing.

[5] Deepset-ai. 2021. Annotation Tool. Retrieved September 15, 2021 from https://haystack.deepset.ai/guides/annotation

[6] Deepset-ai. 2021. DocumentStores. Deepset-ai. Retrieved September 15, 2021 from https://haystack.deepset.ai/components/document-store

[7] Deepset-ai. 2021. Haystack. Retrieved September 15, 2021 from https://github.com/deepset-ai/haystack

[8] Deepset-ai. 2021. Haystack. Retrieved September 15, 2021 from https://haystack.deepset.ai/overview/intro

[9] Deepset-ai. 2021. Pipelines. Deepset-ai. Retrieved September 15, 2021 from https://haystack.deepset.ai/components/pipelines

[10] Deepset-ai. 2021. Preprocessing. Deepset-ai. Retrieved September 15, 2021 from https://haystack.deepset.ai/components/preprocessing

[11] Deepset-ai. 2021. Retriever. Deepset-ai. Retrieved September 15, 2021 from https://haystack.deepset.ai/components/retriever

[12] Joachims T.; Granka L.; Pan B.; Hembrooke H.; Radlinski F. and Gay G. 2007. Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search. *ACM Trans. Inf. Syst.* 25, 2, 7–es. https://doi.org/10.1145/1229179.1229181

[13] Hugging face. 2021. Hugging face. Retrieved September 15, 2021 from https://huggingface.co/

[14] Hugging face. 2021. Roberta-base-squad2. Retrieved September 15, 2021 from https://huggingface.co/deepset/roberta-base-squad2

[15] Tapeh A. G. and Rahgozar M. 2008. A knowledge-based question answering system for B2C eCommerce.

[16] Google. 2021. Google Colab. Google. Retrieved September 15, 2021 from https://colab.research.google.com/

[17] Radev D.; Fan W.; Qi H.; Wu H. and Grewal A. 2005. Probabilistic Question Answering on the Web. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/511446.511500

[18] Chen D.; Fisch A.; Weston J. and Bordes A. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *ACL*.

[19] Hosp J. 2018. Blockchain 2.0. FinanzBuch Verlag.

[20] Rajpurkar P.; Zhang J.; Lopyrev K. and Liang P. 2018. SQuAD: 100,000+ Questions for Machine Comprehension of Text.

[21] Devlin J.; Chang M.; Kenton L. and Toutanova K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Association for Computational Linguistics, Minneapolis, Minnesota. https://doi.org/10.18653/v1/N19-1423

[22] Kodra L. and Meçe E. K. 2017. Question Answering Systems: A Review on Present Developments, Challenges and Trends.

[23] Schmidbauer M. and Köhler A. 2021. Crypto-Answer. Retrieved September 15, 2021 from https://github.com/00Markus0/Crypto-Answer

[24] Alzubi J.; Jain R.; Singh A.; Parwekar P and Gupta M. 2021. COBERT: COVID-19 Question Answering System Using BERT. *Arabian journal for science and engineering*, 1—11. https://doi.org/10.1007/s13369-021-05810-5

[25] Xie Q.; Zhang Q.; Tan D.; Zhu T.; Xiao S.; Li B.; Sun L.; Yi P. and Wang J. 2021. Chatbot Application on Cryptocurrency.

[26] Xie Q.; Zhang Q.; Zhang X.; Tian D.; Ruixuan W. R.; Zhu T.; Yi P. and Li X. 2021. A Context-Centric Chatbot for Cryptocurrency Using the Bidirectional Encoder Representations from Transformers Neural Networks.

[27] Rajpurkar P.; Jia R. and Liang P. 2020. Know What You Don't Know: Unanswerable Questions for SQuAD.

[28] Krishnamoorthy V. 2021. Evolution of Reading Comprehension and Question Answering Systems. *Procedia Computer Science* 185, 231–238. https://doi.org/10.1016/j.procs.2021.05.024

[29] Wikipedia. 2021. Cryptocurrency. Wikipedia Foundation Inc. Retrieved September 15, 2021 from https://en.wikipedia.org/wiki/Cryptocurrency