Improving Human Annotation Effectiveness for Fact Collection by Identifying the Most Relevant Answers

Pranav Kamath[1*], **Yiwen Sun**[2], **Thomas Semere**[2], **Adam Green**[2],
Scott Manley[2], **Xiaoguang Qi**[2], **Kun Qian**[2], **Yunyao Li**[2], **and Mina Farid**[2]
University of Washington[1], Apple[2]
pranavpk@uw.edu,
{yiwen_sun, tsemere, adam_green2}@apple.com
{scott_manley, xiaoguang_qi, kunqian, yunyaoli, m_farid}@apple.com

Abstract

Identifying and integrating missing facts is a crucial task for knowledge graph completion to ensure robustness towards downstream applications such as question answering. Adding new facts to a knowledge graph in real world system often involves human verification effort, where candidate facts are verified for accuracy by human annotators. This process is labor-intensive, time-consuming, and inefficient since only a small number of missing facts can be identified. This paper proposes a simple but effective human-in-the-loop framework for fact collection that searches for a diverse set of highly relevant candidate facts for human annotation. Empirical results presented in this work demonstrate that the proposed solution leads to both improvements in i) the quality of the candidate facts as well as ii) the ability of discovering more facts to grow the knowledge graph without requiring additional human effort.

1 Introduction

A knowledge graph (KG) is an efficient way of storing information and relations between different types of entities, and is an integral part of many real-world applications such as question answering (Chen et al., 2020). However, incompleteness is a well-known issue that inherently exists in a KG caused by missing facts and entities, and it significantly limits the capability of the downstream applications (Socher et al., 2013). Since richness of the facts in a KG can directly impact its quality, identifying and collecting missing facts is a crucial task. Additionally, it is essential to ensure that the facts being added to KG are verified to be highly accurate.

There is research work published on how to automatically identify missing facts using machine learning models (Ji et al., 2021). Many of them use embedding models over the entities and relations in a KG to train link prediction models to predict missing facts within the KG (Nguyen et al., 2017; Wang et al., 2015), while some use a search-based question-answering approach to get candidate answers from the web and then identify the correct missing fact by aggregating the found answers (West et al., 2014). These approaches can automatically identify a large number of missing facts, but there was usually no human verification (which is also infeasible given the volume of the identified facts) to make sure that the newly discovered facts meet real-world-system-level accuracy.

Collecting high-quality new facts for a live KG is a very complicated process that often requires human effort. For example, the success of finding the birth date of a person from the web heavily depends on the popularity of the person and the uniqueness of their name. On the other hand, open access to the web makes it hard to protect the accuracy of its information; even Wikipedia suffers from vandalism (Šarūnė Bar). This influenced our initial design of a fact collection framework with a human-in-the-loop component. The system starts with an unanswerable query, leverages state-of-the-art technologies to identify the intent and entity of the input query, retrieves a few candidate answers via web search using a natural language based question-answering (QA) system, then instead of using a machine learning model to aggregate the results, e.g., (West et al., 2014), we direct the candidate facts to human annotators for review.

While this pipeline is successfully deployed in a real world system, there is an opportunity for improvements to be made. We observed that the natural language-based QA system sometimes returns no candidate answer for a query, leading to a low coverage problem. On the other hand, even if the QA system returns some candidate answers for certain queries, many of them do not contain the correct answers. In such cases, the human annotations ended up with negative responses, indicating a waste of human labor. To address this problem,

[*]Work done during an internship at Apple.

Proceedings of the Fourth Workshop on Data Science with Human-in-the-Loop (Language Advances), pages 74 - 80

one option is to train a better QA system either by using more data or adopting more complicated models, which needs expensive engineering efforts. Another option is to increase the number of candidates returned by the QA system and involve more human annotators for fact verification, but because of the low-correct-ratio nature of the candidate answers, this approach would mean an even more severe waste of human effort.

In this paper a new human-in-the-loop fact collection framework is proposed that addresses the aforementioned problems without training new QA systems nor adding additional human annotators. To solve the low coverage problem of the QA system, the input query is perturbed by generating semantically equivalent variations of it. These variations are then input to the QA system to get diverse answer sets (similar to what has been done in (West et al., 2014)). Then, to address the low-correct-answer-ratio problem, a candidate answer selector is introduced, inspired by the well-known uncertainty-based active sampling strategy Query-by-Committee (QbC) (Seung et al., 1992; Freund et al., 1997; Gurajada et al., 2019), to choose the most relevant answers for human annotation. To summarize, the main contributions of this work are:

- A simple but effective human-in-the-loop fact collection framework that uses a natural language-based QA system as a black-box model to collect diverse candidate answers from the open web.
- By employing a QbC-based selector, the proposed framework can identify more relevant candidate answers and filter out less relevant ones to effectively identify more missing facts with limited human annotation budget.
- Empirically it is observed that this approach can significantly improve both recall and relevancy of the fact collection process in real world settings. Moreover, although only two intent use cases were reported in the experiments, the framework is generic enough to be immediately extended to many more scenarios.

The rest of this paper is organized as follows: Sections 2 describes our human-in-the-loop fact collection framework. Section 3 demonstrates experimental results. Section 4 reviews related work. Finally, concluding remarks and future works are included in Section 5.

2 Methodology

Architecture of the proposed system is illustrated in Figure 1. User query answering is generally carried out by a two-stage model that first extracts the relevant information relating to the intent of the query from a web page, then ranks and selects the most relevant articles that match the entity and the intent of the query. This process can be improved by fine tuning the model used, though it requires quality data and is time-consuming. The proposed framework does not rely on fine tuning but makes changes to the input of the model. The system leverages the sensitivity of language based models to the input and combines it with a selector that works on the principle of Query-by-Committee. The rest of the section shall illustrate each component of the proposed pipeline via a concrete example.

2.1 Query Annotator

The input to Query Annotator is a user query that is unanswerable by the current KG. Assuming the KG does not currently contain the height of Michael Jordan, the following query is an example of an unanswerable query

q_1: `how tall is Michael Jordan?`

Given such a query, the main task for Query Annotator is to identify the intent of the query (e.g., '`height`') as well as the key named entities (e.g., '`Michael Jordan`') in the query. All these rich annotations will be added to the original query and sent to the next component Query Synthesizer to generate query variations. As a concrete example, for the unanswerable query mentioned earlier, the output of Query Annotator would be:

q_1': `how tall is Michael Jordan?`
- `entity:` $\langle$ '`Michael Jordan`', $kgid$ $\rangle$
- `intent: 'height'`,

where $kgid$ is an identifier of the entity in the KG.

2.2 Query Synthesizer

Given a query q' enriched with key named entities and intent, the Query Synthesizer tries to generate semantically equivalent variations and perturbations of q'. A template-based approach is used to create new synthetic queries by replacing the entities in the sentence (e.g., '`how tall is [PERSON ENTITY]`'). Additional information about the detected entities from the KG, such as *occupation* or *aliases* are also used to generate new variations

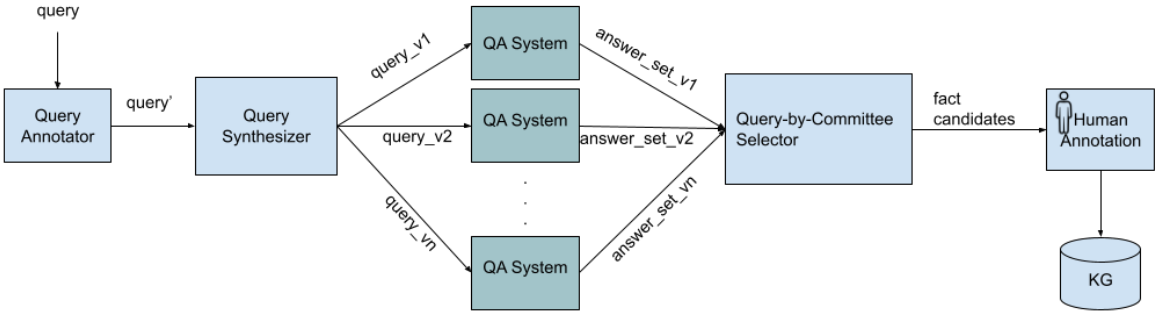

Figure 1: Proposed System Architecture using query by committee. Note that the QA System is kept fixed.

(e.g., 'Michael Jordan' can be replaced with one of the aliases 'Michael Jeffrey Jordan').

There are two main requirements for the Query Synthesizer: (1) only semantically equivalent queries should be created; (2) the generated queries should be natural and realistic. It is evident that semantically nonequivalent queries lead to undesirable results, which would hurt precision. Moreover, it is also crucial that the newly generated queries remain natural to humans, so the QA system which was trained to understand natural language questions can return meaningful answers. To meet these requirements, a randomly sampled anonymous query log is analyzed in order to understand user behaviour and how users interact with web search. Based on that, multiple templates to ask single question are determined. The question templates depend heavily on the intent of the query, and therefore different templates for each intent and entity pair were generated. For example, the query 'how tall is Michael Jordan?' might be translated to

q_1^1: what's the height of Michael Jordan?
q_1^2: tell me Michael Jordan's height.
q_1^3: how tall is Michael Jeffrey Jordan?
...

The main benefit of creating such variations of the input query is to provide the QA system with more diverse signals, leading to more potential answers that might not be found by asking the original query alone.

2.3 QA System

The QA system is responsible for searching and retrieving answers from the internet that are relevant to the input question. In the proposed system, a state-of-the-art QA system is used as a black-box to get candidate answers for a given query. The input to the QA system is a natural question, and the output is a ranked list of answer tuples, where each tuple is of the format (passage, fact, score). Passage is the passage from a web page that may contain the fact that can answer the question, fact is an extracted and normalized answer to the question, and score is a confidence score that the QA system calculated that indicates the quality of the answer found. As a running example, one answer tuple could be (...a height he himself claimed in 1994 "I'm 6-foot-6"... , '6-foot-6', 0.49). Although the QA system may return a lot of answers for a given query, only the top-k tuples by score are kept, where k is a hyperparameter of the system usually set to a small number such as 3 or 5.

One thing to emphasize is that the QA system is very sensitive to the input. Changing the language or ordering of the input query may lead to obtaining different types of answers as the output. This also motivated us to use semantically equivalent variations of the original query to get diverse answer sets to improve the recall. While, one potential pitfall is that the answer sets returned from the QA system can be noisy. For example, for the query How tall is Michael Jordan?, some of the passages the QA system returns are extracted from the following webpages:

- anonymous-url-1 - a webpage that contains multiple different answers
- anonymous-url-2 - a webpage that contains the word height and Michael Jordan, but not the answer

These issues can come up with any entity and intent. It is very difficult to identify. This is a major motivation for us to adopt multiple variations of the query. Additionally, the noisy results indicate that blindly trusting the results returned by the QA system may hurt precision of the system, therefore, the next component Query-by-Committee selector

is adopted to prune the answer sets.

2.4 Query-by-Committee Selector

While the previous Query Synthesizer and QA system components focus on improving the recall by creating semantically equivalent variations, the Query-by-Committee Selector aims to improve precision by selecting the most relevant candidate answers. After receiving the likely noisy answer sets from the QA system in previous step, an immediate question is how to distinguish relevant and irrelevant answer sets. Towards this, a selection approach motivated by the well-known sampling strategy Query-by-Committee is introduced.

QbC is an uncertainty-based modal agnostic active learning algorithm. The core idea of QbC is to form a voting committee consisting of multiple classifiers that were trained in slightly different ways. The degree of agreement among the committee is used as a proxy for uncertainty. If the committee highly disagrees on the label of a data point, then it is an uncertain example, which should have higher priority to be manually verified than those examples where the committee highly agree. Active learning uses this approach to choose the most uncertain and informative examples for human annotation.

Motivated by the idea of using degree of agreement to measure uncertainty, the proposed solution uses QbC in an opposite way, that is, instead of identifying uncertain examples, the goal is to identify highly certain examples. To simulate the QbC-based sampling, a voting committee has to be formed. One option is to perturb the models, i.e., train multiple variations of the QA systems, and another option is to perturb the input query. The proposed solution chooses the second one because it requires much less computational resources, and it is easier to maintain. Therefore, semantically equivalent variations of the input query are generated, which will be sent to the same QA system to get multiple, but likely different answer sets. Next, to identify the most relevant answer among these answer sets, all the answer tuples are ordered based on the number of occurrences in the returned answer sets, finally the top-p answer tuples are chosen as candidates for human annotation.

Note that, the voting mechanism provides a good way to overrule the confidence score returned by the QA system, because an answer tuple with lower confidence score might rank higher than a higher-scoring answer tuple if it gets more votes, which is a way to mitigate the potential bias existed in the QA system. The confidence scores calculated by the QA system is only used as a secondary metric to rank answer tuples.

In this example illustrated here, assuming that it is required of the Selector to select only the top 2 candidate answers that are relevant to {`height`, `Michael Jordan`}. The first answer set contains `anonymous-url-1` and `anonymous-url-2`. The second answer set contains `anonymous-url-1`, `Wikipedia`, and `anonymous-url-3`. Assuming that the order of score for these answer sets is: `Wikipedia` > `anonymous-url-1` > `anonymous-url-3` > `anonymous-url-2`. Since `anonymous-url-1` appeared in both runs of the model, it is ranked the highest. The other three candidates all appear only once. Then, based on the tie-breaker condition, i.e., ordered by confidence scores returned by the QA system, `Wikipedia`, which has the highest score, is selected as the second candidate answer. These selected answers move on to the next step human annotation.

2.5 Human Annotation

The candidate answers received from the QbC Selector will be verified by human annotators before integrated with the KG. Considering the given example, the human annotator would open the url of the website (e.g., `anonymous-url-1`) in a candidate answer tuple, check if the passage indeed comes from the webpage, then make a judgment call on whether or not the extracted fact (e.g., `6-foot-6`) is the correct answer.

3 Experiments

Implementation and Dataset. We implemented the system in python and used PySpark[1] for distributed computation tasks. A randomly sampled anonymous set from query logs is used to evaluate the proposed framework. It contains queries (entity-intent pairs) that did not have answers available in the knowledge graph. The dataset consists of 3648 unique entities for 2 intents.

Experiment setting. The proposed solution is compared against a baseline system, i.e., the system without the Query Synthesizer and QbC Selector. Concretely, the baseline model simply takes an unanswerable query, sends it to the QA system to

[1]https://spark.apache.org/docs/latest/api/python/

	Baseline	Proposed framework (ours)
Failed Answer Ratio	6.69%	**0.10%**
Recall	1102/1441 = 76.5%	1418/1441 = **98%**
Avg. Answer Confidence Score	0.5734	**0.5944**

Table 1: Results for intent 1.

	Baseline	Proposed framework(ours)
Failed Answer Ratio	5.69%	**0.07%**
Recall	1719/2207 = 77.9%	2183/2207 = **98.9%**
Avg. Answer Confidence Score	0.4297	**0.4492**

Table 2: Results for intent 2.

get the top-p answer candidates, then request human annotation. For both the baseline system and the proposed solution, the top-5 (i.e., $p = 5$) answers from the QA system are requested. In this comparison study, the following evaluation metrics are considered:

- ***Recall*** - fraction of input queries that can be answered.
- ***Failed Answer Ratio*** - fraction of input queries that result in an empty candidate answer set from the QA system.
- ***Answer Confidence*** - the average confidence scores of candidate answers provided by the QA system.
- ***Inter-annotator agreement (IAA)*** - given a query q and a candidate answer set A, the IAA of q and A is, the fraction of annotators found the correct answer to the query q from the answer set A, i.e.,

$$\text{IAA}_A^q = \frac{\text{\# annotators found the answer for } q \text{ in} A}{\text{\# of annotators reviewed } q},$$

These metrics evaluate the system in different aspects. Recall evaluates that the proposed method can indeed find more missing facts and increase the number of answerable queries. The failed answer ratio is different from recall in the sense that the former acts as a coverage metric to measure how well the system can find potential answers (not necessarily correct answers) for an input query. Answer confidence serves as a guardrail metric for the QbC Selector. Candidate answers with more votes should have higher confidence scores calculated by the QA system. Finally, IAA is a proxy for evaluating the relevancy of the answer set. Higher IAA values indicate that the candidate answers have higher relevancy and quality, hence easier to reach a consensus among annotators, though not directly related to recall.

Results. Tables 1 and 2 show that our proposed framework performs better across all measurements when compared to the baseline.

For both intents, we observed significant recall boosts indicating that our method is much more effective than the baseline model at finding the correct answers for broader input queries. To make the baseline system reach the parity, an intuitive approach would be to retrieve more candidate answers, so instead of the top-5, top-10 answers from the QA systems could be sent to annotators for review. This would substantially increase the amount of human annotation effort, though a good portion of it will be used to verify incorrect answers, which is actually saved by our system. Another observation is that our approach significantly decreased the Failed Answer Ratio, which can be explained by the fact that introducing Query Synthesizer to generate variations of the input query triggered the QA system to retrieve candidate answers that might have been overlooked by the baseline which runs the QA system only once per query. The average Answer Confidence score returned by the QA system is higher for our framework, which indicates that when using QbC method more relevant candidate answers can be identified. Meanwhile, less relevant answers with lower confidence scores are filtered out by the committee vote. Hence, higher scores indicate higher precision in finding quality answers to questions.

Table 3 shows the IAA comparison between both frameworks for the two intents respectively. As seen in the table, the IAA for our framework is higher than the baseline, which suggests that the candidate answers from our framework are more relevant to the queries, so different annotators come to an agreement more often. Although the IAA number increased only by a small margin, keep

	Baseline	Ours
IAA(intent 1)	64.81%	65%
IAA(intent 2)	68.89%	70.83%

Table 3: Inter-annotator agreement on an average for two intents between baseline and our framework.

in mind that our framework has significantly increased the recall than the baseline. This indicates that our method can find many more answers than the baseline without degrading the quality of the answers.

4 Related Work

In (West et al., 2014), the authors adopted a very similar idea of perturbing the input query, then they used a search-engine with open web access to identify diverse answers, and finally aggregated the answers using scoring model to identify the most likely answers. Although this work is in same spirit, the proposed approach is different from theirs in two ways: first, a much simpler aggregation mechanism without the need to train a scoring model is used, thus the proposed approach is easier to implement and can work with any QA system; second, human-in-the-loop verification is included to ensure the high quality of identified facts.

In addition to improving the human annotation effectiveness, reducing human annotation efforts is another related research topic. A lot of the work try to solve this problem by introducing a deep learning or machine learning model(Varga and Lőrincz, 2020). Certain papers also aim to reduce the annotation work by selecting only samples that are optimal to training the model(Zesch et al., 2015). The proposed solution is different from these works as this solution does not train new machine learning models to reduce annotation efforts rather uses selective sampling.

The use of QbC to identify relevant answers is also related to the ensemble learning paradigm, where the goal is to improve the predictive performance of a single model by training multiple variations and combining their predictions (Sagi and Rokach, 2018; Dietterich et al., 2002). In fact, QbC can be viewed as a way to create the ensemble with the specific purpose of using it to identify uncertain examples for active learning (Melville and Mooney, 2004; Krogh and Vedelsby, 1994). Both QbC and ensemble learning require creating a group of slightly different models so that a diverse predictions or decisions can be later combined or integrated to produce more accurate prediction. The proposed approach was inspired by this idea, and unlike QbC whose primary focus is to identify uncertain examples for human annotation to decrease the uncertainty of the model (Gilad-bachrach et al., 2005), the goal here is to identify certain examples.

Ranking web search results is an important topic in elevating user experience. There are various approaches that try to solve this problem. Aggregated search results and re-ranking based on similarity is one of the solutions(Kumar and Nath, 2013). While this work takes inspiration from the aggregated search results, the proposed solution differs in the re-ranking aspect. Similarity of web pages is not checked for, instead the voting mechanism in the QbC selector to find the best potential answers.

5 Conclusion & Future Work

This paper proposes a new framework that aims to reduce the efforts of human annotators in the fact collection process for enriching a knowledge graph. Empirical observations of the proposed solution when compared against a baseline framework demonstrate that the proposed framework has better recall, and can identify much more relevant facts. This framework is easy to implement, provides an additional automated and scalable layer of enrichment to web answer retrieval, and gives a significant boost to the fact collection process in terms of quality and coverage. Future work shall include testing the proposed framework within other domains such as open domain question answering. Moreover, this framework can significantly boost the recall, which can help in collecting more labeled data to train a model to predict the reliability of a website given a query's intent. In this way, the QA system can be enhanced to provide more relevant candidate answers.

References

Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948.

Thomas G Dietterich et al. 2002. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2):133–168.

Ran Gilad-bachrach, Amir Navot, and Naftali Tishby. 2005. Query by committee made real. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Sairam Gurajada, Lucian Popa, Kun Qian, and Prithviraj Sen. 2019. Learning-based methods with human-in-the-loop for entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2969–2970.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Anders Krogh and Jesper Vedelsby. 1994. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7.

Naresh Kumar and Rajender Nath. 2013. A meta search engine approach for organizing web search results using ranking and clustering.

Prem Melville and Raymond J Mooney. 2004. Diverse ensembles for active learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 74.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2017. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 287–294, New York, NY, USA. Association for Computing Machinery.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26.

Viktor Varga and András Lőrincz. 2020. Reducing human efforts in video segmentation annotation with reinforcement learning. *Neurocomputing*, 405:247–258.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Twenty-fourth international joint conference on artificial intelligence*.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526.

Torsten Zesch, Michael Heilman, and Aoife Cahill. 2015. Reducing annotation efforts in supervised short answer scoring. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–132, Denver, Colorado. Association for Computational Linguistics.

Šarūnė Bar. 64 of the funniest wikipedia edits by internet vandals. https://www.boredpanda.com/funny-wikipedia-edits/?utm_source=google&utm_medium=organic&utm_campaign=organic. Accessed: 2022-09-15.