QuAX: Mining the Web for High-utility FAQ

Muhammad Shihab Rashid  
University of California, Riverside  
mrash013@ucr.edu

Fuad Jamour  
University of California, Riverside  
fuadj@ucr.edu

Vagelis Hristidis  
University of California, Riverside  
vagelis@cs.ucr.edu

ABSTRACT

Frequently Asked Questions (FAQ) are a form of semi-structured data that provides users with commonly requested information and enables several natural language processing tasks. Given the plethora of such question-answer pairs on the Web, there is an opportunity to automatically build large FAQ collections for any domain, such as COVID-19 or Plastic Surgery. These collections can be used by several information-seeking portals and applications, such as AI chatbots. Automatically identifying and extracting such high-utility question-answer pairs is a challenging endeavor, which has been tackled by little research work. For a question-answer pair to be useful to a broad audience, it must (i) provide general information – not be specific to the Web site or Web page where it is hosted – and (ii) must be self-contained – not have references to other entities in the page or missing terms (ellipses) that render the question-answer pair ambiguous. Although identifying general, self-contained questions may seem like a straightforward binary classification problem, the limited availability of training data for this task and the countless domains make building machine learning models challenging. Existing efforts in extracting FAQs from the Web typically focus on FAQ retrieval without much regard to the utility of the extracted FAQ. We propose QuAX: a framework for extracting high-utility (i.e., general and self-contained) domain-specific FAQ lists from the Web. QuAX receives a set of keywords from a user, and works in a pipelined fashion to find relevant web pages and extract general and self-contained question-answer pairs. We experimentally show how QuAX generates high-utility FAQ collections with little and domain-agnostic training data, and how the individual stages of the pipeline improve on the corresponding state-of-the-art.

CCS CONCEPTS

• Information systems → Data extraction and integration; Information retrieval.

KEYWORDS

information retrieval; faq retrieval; faq extraction; question answering; web mining

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CIKM ’21, November 1–5, 2021, Virtual Event, QLD, Australia  
© 2021 Copyright held by the owner/author(s).  
https://doi.org/10.1145/3459637.3482289

1 INTRODUCTION

Frequently Asked Questions (FAQ) lists provide users with frequently requested information on a given topic. For example, many healthcare providers offer a FAQ list on COVID-19, which allows users to obtain relevant information with ease. More importantly, FAQ lists also facilitate many important tasks such as retrieval-based question answering [26], training generative question answering models [32], augmenting chatbot knowledge bases [16], and allowing search engines to provide a short, relevant list of question-answer pairs when given a search query [27].

Reliable FAQ lists are typically created and maintained manually by domain experts, which is laborious and time consuming. The Web offers a plethora of FAQ lists on almost every topic; thus, mining the Web for FAQ lists provides a scalable way of acquiring and curating FAQs.

Automatically mining and curating FAQ lists from the Web is a challenging task due to the different ways of presenting FAQ lists on the Web, and the inherent noisiness of the available FAQs. There have been many works on mining FAQ from the Web; for example, the authors in [23] propose extracting FAQ lists from web pages by identifying HTML list constructs in web pages and the authors in [8] mine online forums for question-answer pairs using sequential pattern features and graph-based ranking. Existing FAQ mining

Figure 1: Only 18%1 of FAQ on the Web are general and self-contained; these are the only questions that are suitable for building general-purpose knowledge-bases.

1Based on a study of 1,176 questions extracted from 170 FAQ web pages in the mental counseling and dental health domains.
works [1] focus on retrieving FAQ lists, and they neglect filtering out noisy question-answer pairs. Although the mere retrieval of large-scale FAQ lists is useful on its own, low utility FAQ lists may provide incomplete or misleading information. Specifically, for question-answer pairs to be useful to a wide audience, outside the Web site where the question is hosted, the questions must be general and self-contained.

General questions are those whose utility is universal. For example, the question "What are the symptoms of COVID-19?" asks for information that is universally applicable, and thus is of high utility. On the other hand, questions such as "Is smoking allowed at the University of California?" appear in contexts where users ask for information that pertain to a certain entity; such questions have limited utility and thus should not be included in general-purpose FAQ lists (of course this question is useful for students of the University of California, but our goal is to extract questions with much wider scope). Self-contained questions are ones that are complete on their own, in the sense that they do not contain references or ellipses. In contrast, questions such as "How to control the symptoms of the disease?" require access to a larger context than the question-answer pair to be useful; consequently, such questions should also be discarded when building universal FAQ lists.

It turns out that only a small percentage of FAQ on the web is general and self-contained and thus are useful for general-purpose knowledge-bases (see statistics in Figure 1). Identifying these questions is challenging because they do not follow a specific pattern; hence, using static rules to identify such questions is not feasible, and the lack of high-utility labeled training data makes it difficult to train a classification model. Furthermore, if machine learning models are to be used, such models should be domain-oblivious. A key requirement of QuAX is that no domain-specific training data should be required.

We propose QuAX: a framework for retrieving general and self-contained FAQ lists from the Web for a given domain. Figure 2 shows an overview of QuAX, which receives a list of keywords that describe a certain domain (e.g. COVID-19 or Plastic Surgery), and works in six pipelined steps to produce a high-utility list of FAQ on the given domain as follows. First, QuAX augments the given list of keywords to include extra terms that would help retrieve comprehensive yet relevant FAQ pages. Then, the expanded list of keywords is used to make a Google search to retrieve web pages with relevant information to the given domain. The retrieved web pages are fed into our FAQ Page Detection module, which first pre-processes the HTML content of the pages and then uses a CNN based classifier to identify pages that contain FAQ lists. After that, our QA Extraction module uses HTML tags to extract the actual question-answer pairs from the given FAQ pages. Our General vs. Specific module filters out specific questions using a CNN classifier and an active learning strategy to mitigate the scarcity of training data. Our Self-contained vs. Incomplete module then filters out incomplete questions using a CNN classifier coupled with a KL-divergence based feature generator. Finally, our Duplicate Detection module filters out redundant questions using a multilayer perceptron. To train our classifiers, we propose a strategy to collect and annotate reliable training examples.

We evaluate each module in our pipeline and compare its performance against that of strong baselines and show that each of our individual modules outperforms the baselines. Furthermore, we perform case studies with five domains: mental counseling, dental health, plastic surgery, medical marijuana, and COVID-19. For each domain, we generate a set of descriptive keywords and pass them through our pipeline, and we show that the resulting FAQ lists are relevant, general, and self-contained.

In summary, we claim the following contributions in this paper:
• We propose the first complete pipeline for retrieving comprehensive yet high-utility, general and self-contained, FAQ lists.
• We collect and annotate training data for training the classifiers in our modules, published on a public repository.
• We show that an active learning strategy and a KL-divergence based feature extraction method help mitigate the scarcity of training data in our most critical modules (the General vs. Specific, and Self-contained vs. Incomplete modules).
• We experimentally evaluate our pipeline and show that our resulting FAQ lists are of high utility.

The rest of this paper is organised as follows. We present the details of QuAX in Section 2. We present the results of our experimental evaluation in Section 3. We discuss the related work in Section 4 and finally conclude in Section 5.

2 OUR SYSTEM
In this section we describe each module of the QuAX pipeline shown in Figure 2.

2.1 Keyword Expansion Module
To extract high-utility FAQs, the mined web pages must contain question-answer pairs that are relevant to the input keywords. Due to the high volume of content on the web, selecting the right set of keywords can be challenging, this problem is known as search keyword mining [39]. Using such keywords to search the web for relevant pages may result in retrieving pages that may not contain question-answer pairs, or that are irrelevant to the input keywords due to users not having deeper domain knowledge when selecting initial keywords. Existing works on keyword expansion [5] enable producing extra keywords that can improve the relevance of the retrieved pages (such as retrieving relevant twitter posts). However, such works do not particularly produce keyword expansions that result in retrieving pages with question-answer pairs. In our keyword expansion module, we extend the work in [39] to produce keyword expansions that not only facilitate retrieving more relevant pages but also contain question-answer pairs.

The authors in [39] present a keyword expansion algorithm that retrieves more relevant twitter posts by using a double ranking approach. However, it does not take into consideration the type of content extracted (i.e. faqs). We extend their work as follows: First, it retrieves web pages using the domain input keywords and ranks the words in the first k retrieved pages based on their entropy. Then, our module uses the l top-ranking words in the resulting vocabulary to do another search. The words in the newly retrieved pages are ranked again based on their entropy and the l top-ranking ones in the re-ranked list are returned as keywords expansions. We repeat these steps for the word “faq” separately and concatenate the resulting words with the initial result. The system may return very common words (e.g., the and of) from the English vocabulary. To mitigate this, similarly to [39], we use a Random Words Set (RS), consisting of 400k words, where the words are taken from 300 random Wikipedia articles. If any word appears frequently in the random set, it gets a lower entropy score. We use the following formula to compute entropy:

\[ e_w = -\sum \frac{f_s(w) + \lambda}{|S|} \log \frac{f_s(w) + \lambda}{|S|} \]

where \( s \in S = \{SS, RS\} \), SS being the Snippets Set (Snippets returned from searching Google), RS being Random Set. \( \lambda \) and \(|S|\) are smoothing parameters in case any word does not appear in any snippet. They are set to 0.005. Snippet frequency (how many times a word appears in a snippet) is denoted by \( f(w) \).

2.2 FAQ Page Detection Module
Since most modern web pages are dynamic, their complex DOM structure makes classification and information extraction challenging. To overcome this, our FAQ Page Detection Module first converts dynamic pages into static ones by flattening interactive elements in a page into a single-layer DOM tree using the boiler-pipe APIs [5] article extractor. Our module then uses a Convolutional Neural Network (CNN) model with HTML pre-processing to classify the resulting static HTML pages into FAQ or NOT-FAQ pages. We chose a CNN due to their reported success in text classification tasks [20] and having the best performance among baseline deep learning techniques (Section 3).

First, our module pre-processes input HTML pages such that HTML tags which do not usually contain question answer pairs are replaced with uniform labels (For example, \(<div>\) and \(<a>\) are replaced with \(<TAG1>\) and tags which contain the question answer strings (\(<h1>..., <p>\) are replaced with \(<TAG2>\), and questions marks are replaced with the tag QUESTION MARK. As we show in our experimental evaluation in Section 3.2, this pre-processing results in improved classification accuracy. Given the pre-processed pages, our model generates an embedding for each word in an input HTML page using a pre-trained word2vec [28] and then combines these embeddings into a feature matrix. Our embedding layer is connected to 5 parallel one dimensional convolutional layers with a filter size of 200 and ReLU activation. We use a global max pooling layer to reduce the size of the feature map and a Sigmoid activation function (to accommodate our classification task) at our last dense layer.

We train our model using training data generated as follows. To generate positive examples, we use the Google search results for keywords from different domains concatenated with the word ‘FAQ’ and use the first 25 pages. We generate negative examples similarly but with without adding the ‘FAQ’ keyword. We train our model using the Adam optimizer with binary cross-entropy loss function.

2.3 QA Extract Module
Although there are tools for extracting question-answer pairs from FAQ pages, some of these tools are proprietary [1] and the others require lots of labeled training data [17]. Therefore, we built our own algorithm for this task. Our question-answer extraction module is based on Algorithm 1. Our algorithm utilizes the HTML tree structure of pages; the main insight is that question-answer pairs are usually nested within certain HTML tags such as \(<h>, <p>, \) and \(<div>\).

\[\text{https://boilerpipe-web.appspot.com/}\]
Algorithm 1 QA Extractor

Input: FW: List of FAQ Websites
Output: UQA: Unclassified QA Pairs

1: $Q \leftarrow []$, $A \leftarrow []$, $UQA \leftarrow []$
2: for each website $w$ in FW do
3: $D_w \leftarrow$ ExtractHTML($w$) [Extract DOM structure of webpage]
4: if Check1Catg($D_w$) is true [Web page falls into category 1] then
5: for each element $e$ in $D_w$ do
6: if $e \in \{h1, h2, h3, h4\}$ then
7: $NH_e \leftarrow$ ExtractNextElement($e$) [Parse next element in DOM tree]
8: if $NH_e \in \{p, div\}$ and length($e$) $> min\_Q\_length$ and
9: length($NH_e > min\_A\_length$) then
10: $Q \leftarrow Q \cup Text(e)$
11: $A \leftarrow A \cup Text(NH_e)$
12: else
13: if Check2Catg($D_w$) is true [Web page falls into category 2] then
14: Repeat 5-10 for $p$ or $div$ pair
15: else
16: for each element $e$ in $D_w$ do
17: $C_e \leftarrow$ GetAllChildElements($e$)
18: for each child element $c$ in $C_e$ do
19: if $e \in \{p, div\}$ and $c \in \{strong, br, a\}$ then
20: $Q \leftarrow Q \cup Text(e)$
21: $A \leftarrow A \cup Text(c)$
22: $UQA \leftarrow UQA \cup \{Q, A\}$
23: return $UQA$

Algorithm 1 receives a list of FAQ pages as input, and it produces a list of question-answer pairs. For each page, Algorithm 1 works as follows.

Variable $Q$ and $A$ stores questions and answers extracted from the webpage (line 1). We extract the HTML tree of the webpage using Jsoup and store in $D_w$ (line 3). The HTML is cleaned using boiler-pipe. Each static cleaned FAQ webpage is usually divided into three categories. 1) The questions are in $<h>$ tag and answers are in either $<p>$ or $<div>$. 2) Questions and answers are in different $<p>$ or $<div>$. 3) Questions and answers are both in the same $<p>$ or $<div>$. Line 4 checks whether the webpage falls in category 1. If yes, then Line 5-8 checks each element in HTML tree, if it’s a $<h>$ tag then it looks at the next element of tree (line 7). If the next element is a $<p>$ or $<div>$ and the text length of the tag is greater than certain threshold, we add the textual content of the tags in corresponding $Q$ and $A$ (line 9-10). The intuition being, if a question resides in a $<h>$ block, the subsequent block should contain textual content which may be considered as the answer to that question. If the webpage falls in category 2 (line 12), which means the question does not reside in $<h>$ block, then we repeat the same process as before for $<p>$ and $<div>$. The extraction process becomes trickier when both the question and answer is situated under the same tag. Usually the question is separated from the answer using tags like $<strong>$, $<br>$ etc. So we check all the child elements (line 18) and if child element contains $<br>$ or $<strong>$ we put the textual content of parent tag in $Q$ and content of child tag in $A$.

2.4 General vs. Specific Classification

This module identifies general questions given a list of question-answer pairs that include both general and specific questions. Although this is a straightforward binary text classification task, the scarcity of training data makes it challenging. Furthermore, the training data for such a text classifier shall be domain-oblivious for the module to accommodate any domain. We mitigate the training data scarcity issue using active-learning [3, 4] and we select the training data carefully in such a way that our active-learning classifier is kept domain-oblivious. We describe our active-learning classifier followed by our data collection methodology in the rest of this section.

Active-learning has shown good results in text classification tasks where training data is scarce [3, 4]. It is a form of semi-supervised learning that uses self-learning feature. This technique first learns from a standard automated labeled training data then continues to learn labels from domain specific unlabeled data that it infers with high confidence. The predicted unlabeled instances with high confidence are added to the standard model and are retrained. The intuition is that such labels resemble human-labeled data and thus allow the classifier to provide better predictions for data points whose inferred labels’ confidence is not conclusive. We select the training data points among the unlabeled instances using uncertainty sampling using modAL framework [10]. This sampling technique uses the posterior probabilities of the resulting labels produced by a model $\theta$ to select the labels with the highest levels of confidence. We use the equation below to calculate posterior probabilities:

$$
\phi_{LC}(x) = \arg\max_x (1 - P_\theta(y|x))
$$

where $x$ is the instance to be predicted and $y$ is the most likely prediction.

To train our base model in this module, we collect training sentences that are general (i.e., have few instances of ellipsis and coreference). We collect such data by extracting random sentences from MayoClinic articles. We collect Specific sentences by selecting responses to specific user issues from Twitter customer support dataset. Such sentences are specific because they address issues that pertain to specific entities.

2.5 Self-contained vs. Incomplete Classification

Given a list of general questions, this module extracts the ones that are self-contained. We use a CNN classifier and we propose a novel multi-feature extraction method with KL that is designed to improve our classifier’s ability to distinguish self-contained questions. We use the intuition that self-contained questions (i.e., “Can I get COVID-19 from my pets?”) can be answered without knowing any context, which means that the answers to such questions have a high degree of similarity. We retrieve answers to a given question using the Google search engine: we issue the question as a query and consider the first ten snippets as answers, given that these

---

3https://www.mayoclinic.org/
4https://www.kaggle.com/thoughtvector/customer-support-on-twitter
Table 1: Datasets summary (used in our trainable modules)

<table>
<thead>
<tr>
<th>Module</th>
<th>Num. of examples</th>
<th>Avg. tokens/instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAQ Page detection⁷</td>
<td>250 websites</td>
<td>1,862</td>
<td>HTML of webpages</td>
</tr>
<tr>
<td>General vs. Specific*</td>
<td>19,442 sentences</td>
<td>16</td>
<td>Mayo Clinic &amp; Twitter</td>
</tr>
<tr>
<td>Self-cont. vs. Incomplete*</td>
<td>1,002 questions</td>
<td>8</td>
<td>FAQ from the web</td>
</tr>
<tr>
<td>Duplicate Detection¹</td>
<td>404,291 question pairs</td>
<td>23</td>
<td>Labeled question pairs</td>
</tr>
</tbody>
</table>

*Original datasets; Available at: https://github.com/shihabrashid-ucr/quax-dataset
¹Available at: www.kaggle.com/c/quora-question-pairs.

typically provide direct or closely related answers. We quantify the similarity across answers by calculating the Kullback-Leibler divergence score [22] (KL) using the following equation:

\[ D_{KL}(P||Q) = - \sum_{x \in \chi} P(x) \log \left( \frac{Q(x)}{P(x)} \right), \quad (3) \]

where P and Q are defined over the same probability space \( \chi \).

KL divergence is a statistic used to measure the similarity between two probability distributions and it is typically used in information retrieval to measure similarity across documents. Here, the probability space \( \chi \) represents all words occurring in the union of two lists of snippets. We use term frequencies of each word to calculate the probability of a word \( (P(x)) \) given a snippet. We compute the average pair-wise KL divergence score for the snippets that answer a question and pass the floating point KL score to our classifier as a feature in addition to the word embeddings of the input query. We train our model using manually labeled datasets from five domains. We labeled 200 questions from each domain, resulting in a total of 1000 training examples. While training, we do not use instances from the same domain (i.e. we use 800 examples for training for a particular domain). We use domain-specific data in inference time, ensuring fairness and domain independence.

2.6 Duplicate Detection

Questions such as “What is rhinoplasty?” and “How do you define rhinoplasty?” are equivalent despite being expressed differently. To produce a higher utility list of question-answer pairs, we eliminate duplicate questions, where duplicates include questions that are semantically very similar. Since we need to identify semantically similar questions even if they have a large string-based distance, record linkage [11] and other string-based methods are inadequate. Instead, we build a similarity classifier which we train using a question similarity dataset from a Kaggle duplicate detection competition (dataset details are in Section 3). We use the Google universal sentence encoder to encode our questions before passing them to a sequential multi layer neural network. The input to the neural network model are question pairs, which are put through Google sentence embedder of dimension 512. The encodings are concatenated and batch normalized to avoid overfitting. ReLU activation function and “adam” optimizer are used. For each pair of questions, the output is a binary 0 or 1 which indicates whether the pair is a duplicate or not.

3 EXPERIMENTAL EVALUATION

We evaluated our framework on five domains in the healthcare area: mental counseling, dental health, plastic surgery, medical marijuana, and COVID-19. We start with simple keywords that describe each respective domain (i.e., the domain name succeeded by the word “faq”), for example “mental counseling faq”, and obtain a list of FAQs using our framework. Along the way, we evaluate each component independently. We present our experimental results in the subsequent sections and show that our framework produces a list of high-utility FAQs (i.e., general and self-contained question-answer pairs) and that its modules provide more accurate results than strong baselines that could have been used in our modules’ place. Since the training data and the evaluation metrics for each module are different, we present these in each respective subsection.

Datasets. To fully automate high-utility FAQ extraction and tackle the training data scarcity in the context of our trainable modules, we have created three training datasets for the modules FAQ page detection, General vs. Specific classification, and Self-contained vs. Incomplete classification, respectively. For the rest of our trainable modules, we have used publicly available datasets. Table 1 summarizes the datasets we used to train and evaluate our trainable modules. All examples in the datasets of the FAQ Page Detection and Self-contained vs. Incomplete modules are labeled manually by annotators. For the General vs. Specific dataset, the examples are automatically labeled. The original datasets we have collected and annotated can facilitate further research on high-utility FAQ extraction; we explain the procedure of collecting our original datasets in each respective subsection.

3.1 Keyword Expansion

Competitors. We compare our module to a simple baseline where we use the input keywords as is, and we also compare it against Double Rank [39] where the authors presented a re-ranking algorithm of keywords expansion based on entropy.

Evaluation Methodology. We use Precision, which is a standard evaluation metric in Information Retrieval, to evaluate our keyword expansion module. Precision is computed by dividing the total number of “correct” webpages returned by searching Google using the keywords by the total number of webpages returned. We first search Google with expanded keywords and take into consideration the top 50 returned results from Google. If any returned webpage is a FAQ page and is on topic then it is a correct result. By “on-topic” we mean that the content of the webpage is related to the domain. We produce ground truth by manually evaluating whether each webpage is precise or not. Similarly to [39], for the values of \( k \) and \( l \), we used 10 and 8, respectively.

Results. We present our results in Table 2. In the final column we show results of our module. We see that, in most cases, our method
We also see in Table 4 that our method achieves an average increase of 5% in accuracy and 6% increase in F1 scores.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Without KE</th>
<th>DR</th>
<th>Updated DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mental Counseling</td>
<td>0.7000</td>
<td>0.7200</td>
<td>0.8000</td>
</tr>
<tr>
<td>Dental Health</td>
<td>0.8200</td>
<td>0.8600</td>
<td>0.7800</td>
</tr>
<tr>
<td>Plastic Surgery</td>
<td>0.8400</td>
<td>0.8000</td>
<td>0.9600</td>
</tr>
<tr>
<td>Medical Marijuana</td>
<td>0.6400</td>
<td>0.8400</td>
<td>0.7800</td>
</tr>
<tr>
<td>Covid 19</td>
<td>0.5200</td>
<td>0.6800</td>
<td>0.7800</td>
</tr>
<tr>
<td>Average</td>
<td>0.7000</td>
<td>0.7800</td>
<td>0.8200</td>
</tr>
</tbody>
</table>

We see in Table 3 that CNN provides significantly better scores in this evaluation.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Baseline (CNN)</th>
<th>Baseline (RNN)</th>
<th>Baseline (LSTM)</th>
<th>Baseline (BiLSTM)</th>
<th>Baseline (Self-BiLSTM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>0.8400</td>
<td>0.6400</td>
<td>0.6200</td>
<td>0.6200</td>
<td>0.6200</td>
</tr>
<tr>
<td>DH</td>
<td>0.8800</td>
<td>0.6200</td>
<td>0.5800</td>
<td>0.6000</td>
<td>0.6400</td>
</tr>
<tr>
<td>PS</td>
<td>0.8400</td>
<td>0.6000</td>
<td>0.5800</td>
<td>0.6000</td>
<td>0.6200</td>
</tr>
<tr>
<td>MM</td>
<td>0.8800</td>
<td>0.6600</td>
<td>0.6000</td>
<td>0.6000</td>
<td>0.6000</td>
</tr>
<tr>
<td>CO</td>
<td>0.8600</td>
<td>0.6000</td>
<td>0.5800</td>
<td>0.6200</td>
<td>0.6000</td>
</tr>
<tr>
<td>Average</td>
<td><strong>0.8600</strong></td>
<td>0.6240</td>
<td>0.5920</td>
<td>0.6040</td>
<td>0.6160</td>
</tr>
</tbody>
</table>

Table 3: FAQ page detection accuracy (baselines)

<table>
<thead>
<tr>
<th>Domain</th>
<th>CNN</th>
<th>QuAX Page Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>0.8400</td>
<td>0.8400</td>
</tr>
<tr>
<td>DH</td>
<td>0.8800</td>
<td>0.8650</td>
</tr>
<tr>
<td>PS</td>
<td>0.8400</td>
<td>0.8350</td>
</tr>
<tr>
<td>MM</td>
<td>0.8800</td>
<td>0.8700</td>
</tr>
<tr>
<td>CO</td>
<td>0.8600</td>
<td>0.8500</td>
</tr>
<tr>
<td>Average</td>
<td>0.8600</td>
<td>0.8500</td>
</tr>
</tbody>
</table>

Table 3: Keywords precision

Table 4: FAQ page detection comparison

Table 5: General vs. Specific classification accuracy (baselines)

3.3 QA Extraction

To evaluate this module, we collected 100 different FAQ websites using Google search and fed them to our algorithm to see whether it is able to extract QA pairs. The 100 webpages were taken from a mixture of the domains. Our algorithm was able to successfully extract 71 webpages out of 100.

3.4 General vs. Specific Classification

Competitors. We show the results for several deep learning based text classifiers (Table 5) and then show the effect of integrating our active learning approach in Table 6.

Evaluation Methodology. We use our 19,000 training data points to train our classifier in this module. For general training data set, we need textual content that are general in nature, meaning there are less number of ellipsis and co-reference and free from context. We propose Mayoclinic websites articles as the source for our training data for class: general. We extract 9,325 random sentences from articles chosen at random about different diseases, medicines from their website and label them as "general". Our training datasets are domain independent due to the nature of the sentences being used as data points. Finding "specific" sentences was challenging because most documents on the web focus on general textual content. We chose the Twitter customer support dataset from Kaggle and extracted the tweets sent by customer service agents of any specific organization and users filing any complaint. We took 10,117 tweets and labeled them as specific. We use an embedding layer which uses word2vec and projects each sentence into 300 dimensional vector. We test the model with our extracted QA pairs. We concatenate a question and its corresponding answer into one string and then test.
<table>
<thead>
<tr>
<th>Domain</th>
<th>CNN</th>
<th>Active Learning + CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>F1</td>
</tr>
<tr>
<td>MC</td>
<td>0.6670</td>
<td>0.6560</td>
</tr>
<tr>
<td>DH</td>
<td>0.7343</td>
<td>0.7128</td>
</tr>
<tr>
<td>PS</td>
<td>0.7230</td>
<td>0.7210</td>
</tr>
<tr>
<td>MM</td>
<td>0.6383</td>
<td>0.6042</td>
</tr>
<tr>
<td>CO</td>
<td>0.7215</td>
<td>0.6823</td>
</tr>
<tr>
<td>Average</td>
<td>0.6959</td>
<td>0.6752</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Baseline (CNN)</th>
<th>Baseline (RNN)</th>
<th>Baseline (LSTM)</th>
<th>Baseline (BiLSTM)</th>
<th>Baseline (Self-BiLSTM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>0.6500</td>
<td>0.5400</td>
<td>0.6000</td>
<td>0.5600</td>
<td>0.5900</td>
</tr>
<tr>
<td>DH</td>
<td>0.6230</td>
<td>0.6385</td>
<td>0.5901</td>
<td>0.5081</td>
<td>0.4918</td>
</tr>
<tr>
<td>PS</td>
<td>0.5990</td>
<td>0.5990</td>
<td>0.5545</td>
<td>0.6000</td>
<td>0.5545</td>
</tr>
<tr>
<td>MM</td>
<td>0.6640</td>
<td>0.6150</td>
<td>0.5983</td>
<td>0.5664</td>
<td>0.5664</td>
</tr>
<tr>
<td>CO</td>
<td>0.5324</td>
<td>0.5040</td>
<td>0.4748</td>
<td>0.5100</td>
<td>0.5539</td>
</tr>
<tr>
<td>Average</td>
<td>0.6136</td>
<td>0.5793</td>
<td>0.5635</td>
<td>0.5489</td>
<td>0.5513</td>
</tr>
</tbody>
</table>

Table 7: Self-contained vs. Incomplete classification accuracy (baselines)

Results. We see from Table 6 that our active learning approach improves the performance of CNN (the best performing classifier). The main reason is that, in baseline, the training dataset does not hold too much information. Because the data points were automatically extracted and labeled, not all labeled data points can be accurate. Using a semi-supervised approach like active-learning mimics human-labeled data and thus produces better results. We observed better results if questions and answers are merged into a single strings (each pair). Our test set contains around 200 question-answer pairs for each domain.

3.5 Self-contained vs. Incomplete Classification

Competitors. We use supervised deep learning approaches as baselines and show their results in Table 7. We pick the best performing method and integrate our KL method to show its efficacy.

Evaluation Methodology. We use our proposed training dataset of 1000 manually labeled questions. To ensure fairness, while training for a domain, we do not include data points from that domain. This shows, our method is domain oblivious. For testing, we used ~150 questions from each domain with equal class sizes.

Results. As seen from Table 8, integrating our KL method into the best performing classifier CNN improves its performance across all domains. Note that even the best performing classifier struggles to achieve impressive accuracies and F1 scores because of the limited amount of training data used. However, our contribution shows that using a multi-feature approach can improve the performance of strong baselines while being domain independent.

3.6 Duplicate Detection

For duplicate detection, we use the 400,000 quora QA pairs dataset to train. Our duplicate detection module shows 80% accuracy while testing on with Quora dataset.

3.7 Entire Framework Evaluation

We present in this subsection the average precision and average recall for our end-to-end pipeline. To calculate precision for each domain, we divide the number of general and self-contained questions by the total number of QA pairs generated by the system. To calculate recall for each domain, we divide the total count of general and self-contained questions generated by duplicate detection module by the number of all general and self-contained questions generated by QA extraction module. Our system achieves an average precision of 78.6% and an average recall of 60.20%. Note that, the goal of a high utility FAQ extractor should be to not generate false positives. However, missing out on some FAQs, which results in a relatively low recall, should not be an issue considering huge number of FAQ websites on the Web. These results show that our framework manages to obtain reasonable percentage of high-utility question-answer pairs despite training data scarcity.

3.8 Case Studies

We present in this subsection statistics and qualitative analysis of the results of our framework in the domains we have selected, and we further discuss a sample from the results in the mental counseling and the COVID-19 domains. Table 9 shows the counts of the results of each module in our framework. We used the top 100 results from Google and pushed them through our framework. We chose 100 results because Google API has a limit of 100 results per request. From this table we can make the following observations:

- The total number of general questions is low compared to the total number of QAs on the web. This shows the importance of classification of general questions to build knowledge base. It is not enough to extract QAs and use them as knowledge bases.
- In each module, low-utility question-answer pairs are filtered out incrementally until reaching the last step where a list of high-utility question-answer pairs are produced.
- For the COVID-19 domain, the number of general questions detected is very high. This is because, most questions regarding COVID-19 are general as this is a recent topic. There are
Table 9: Performance of QuAX

<table>
<thead>
<tr>
<th>Domain</th>
<th>No of FAQ Pages</th>
<th>Total QA</th>
<th>General Questions Detected</th>
<th>Self-contained Detected</th>
<th>Without Duplicates</th>
<th>% of High-utility Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>83</td>
<td>776</td>
<td>363</td>
<td>223</td>
<td>219</td>
<td>70</td>
</tr>
<tr>
<td>DH</td>
<td>94</td>
<td>559</td>
<td>446</td>
<td>256</td>
<td>251</td>
<td>88</td>
</tr>
<tr>
<td>PS</td>
<td>92</td>
<td>783</td>
<td>518</td>
<td>284</td>
<td>277</td>
<td>79</td>
</tr>
<tr>
<td>MM</td>
<td>91</td>
<td>617</td>
<td>327</td>
<td>127</td>
<td>127</td>
<td>82</td>
</tr>
<tr>
<td>CO</td>
<td>82</td>
<td>855</td>
<td>679</td>
<td>285</td>
<td>282</td>
<td>74</td>
</tr>
</tbody>
</table>

Table 10: Top 10 Extracted Questions

<table>
<thead>
<tr>
<th>Domain</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mental Counseling</td>
<td>Why do people consider using therapy?</td>
</tr>
<tr>
<td></td>
<td>For what concerns do students seek personal counseling?</td>
</tr>
<tr>
<td></td>
<td>What are the different types of mental health professionals?</td>
</tr>
<tr>
<td></td>
<td>How long are therapy sessions themselves?</td>
</tr>
<tr>
<td></td>
<td>What is psychotherapy?</td>
</tr>
<tr>
<td></td>
<td>What behavioral health concerns does UW Health treat?</td>
</tr>
<tr>
<td></td>
<td>How does a student know if s/he needs counseling?</td>
</tr>
<tr>
<td></td>
<td>What are the Benefits of Telemental Health?</td>
</tr>
<tr>
<td></td>
<td>What is the purpose of this website?</td>
</tr>
<tr>
<td></td>
<td>What is genetic counseling?</td>
</tr>
<tr>
<td>Covid 19</td>
<td>How can you tell the difference between the novel coronavirus and a cold?</td>
</tr>
<tr>
<td></td>
<td>What are the symptoms in children?</td>
</tr>
<tr>
<td></td>
<td>What is social distancing?</td>
</tr>
<tr>
<td></td>
<td>What does it mean that covid-19 is a global pandemic?</td>
</tr>
<tr>
<td></td>
<td>What is the state recommending for social distancing?</td>
</tr>
<tr>
<td></td>
<td>When are you open for vaccines?</td>
</tr>
<tr>
<td></td>
<td>What are the treatments for covid-19?</td>
</tr>
<tr>
<td></td>
<td>What is quarantine?</td>
</tr>
<tr>
<td></td>
<td>I have been around someone else who was exposed to a person with covid-19.</td>
</tr>
<tr>
<td></td>
<td>What should I do?</td>
</tr>
<tr>
<td></td>
<td>Does health insurance cover covid-19 testing and care?</td>
</tr>
</tbody>
</table>

training dataset does not have information regarding "UW" being a specific organization and thus it mistakenly classifies it to be a general word. We also see that Question 9 “What is the purpose of this website?” is not a self-contained question. This question can be general depending on what "this website" refers to. If this co-reference is resolved, it will be counted as a self-contained question. There are some ambiguous questions which can be self-contained and incomplete simultaneously. For example, Question 4 “How long are therapy sessions themselves?" can be a self-contained question if we consider therapy in general. It can also be an incomplete question because the user does not know which organization’s therapy session is this question talking about. Ambiguous questions are also harder to detect but they do not affect the performance of our framework.

Table 10 also lists questions from the COVID-19 domain. We see questions like “What are the symptoms in children?” and “When are you open for vaccines?”. We know that these questions are asking about COVID-19 because the context is known to us. However, these sentences themselves are not self-contained. If the sentence was “What are the symptoms in children for COVID-19?”, it would have been a self-contained question. Finally, consider Question 9, where we see a question with a given context. In FAQs, questions with context play an important role and our module was able to extract and correctly classify these.

4 RELATED WORK

The authors in [32] propose a pipeline for producing FAQ by crawling the web. Although the mentioned work addresses the same problem, the proposed approach is a semi-automated way that integrates users’ feedback and usage mining to improve FAQ lists, whereas our framework is completely automated. Many works use FAQ lists/knowledge base/files for the classic task of question answering [14, 38]. The authors in [13] proposed a system where a query matches a FAQ file first and then the answer to the query is matched with one of the FAQ from that file. The authors in [6] identify missing topics in a FAQ webpage of an enterprise and suggest additional FAQ by searching the web. Although this work extracts ranked FAQ for an enterprise, it does not address the task of extracting general and self-contained question-answer pairs. This work only suggests additional questions instead of extracting every FAQ and it does not classify the question-answer pairs to be high utility (general or self-contained). Both specific and incomplete questions are extracted and suggested. The authors in [17] search the whole web to extract FAQ and then answer users’ questions by retrieving the appropriate question-answer pairs. Their task is at
We have presented a framework for extracting high-utility (i.e., works in a modular fashion to produce a final list of question-answer pairs in a FAQ page is not enough and they leverage this issue by using a FAQ list. We cover next the existing work that is relevant to each individual module in our framework. For Module 1, the authors in [39] propose a similar algorithm to ours; however, their algorithm deals with Twitter data only. Keyword identification from unstructured text is a common task [5, 21] but while searching using retrieved keywords, they do not consider a specific type of results (e.g., FAQ) to be returned. In our work, the retrieved results are question-answer pairs. Module 2 focuses on a particular website detection task; however, this work is proprietary. In [8], the authors extract question and answers from online forums. They address the problem of finding QAs from unstructured content. They extract every kind of QA not only high utility. In [23], the authors present a list detection algorithm to detect FAQ questions inside a webpage. The limitation of this work is that the system would require some domain knowledge to differentiate between FAQ lists and undesirable lists such as product categories. Although Module 4’s task sounds like it falls under the classical problem of Question Classification [30] (i.e. classifying a question into factoid, hypothetical, etc.), it is very much different. In this work, we focus on classifying a frequently asked question into two categories: general and specific. There has been a profusion of research on text classification, from starting with bag of words to very deep convolutional networks [9, 18, 42]. Deep learning has also been used in other NLP tasks such as paraphrasing [37], slot filling [35], and intent detection [36]. Active Learning has been used in scenarios where labeled data are scarce [3, 4, 40]. We are tackling a completely new domain where the class labels for classifications are new and there is no available training data. Even though KL is a popular statistic as an input for classification problems [7], it has yet to be used as a feature for deep learning frameworks for text classification.

5 CONCLUSIONS

We have presented a framework for extracting high-utility (i.e., general and self-contained) questions from the Web. Our framework works in a modular fashion to produce a final list of question-answer pairs. Within each module, we show that existing machine learning models are insufficient, either because they assume that large training datasets are available or because training data for each task is not available altogether. Wherever needed, we collect and annotate datasets to train our models within each module. We present extensive experimental evaluation results that show that our models within each module perform better than strong baselines, and we show that our framework indeed produces general and self-contained questions.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation (NSF) under grants IIS-1838222 and IIS-1901379.

REFERENCES