QuAX: Mining the Web for High-utility FAQ

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

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

Vagelis Hristidis
University of California, Riverside
Riverside, California
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 questions-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

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3482289

ACM Reference Format:
Muhammad Shihab Rashid, Fuad Jamour, and Vagelis Hristidis. 2021. QuAX: Mining the Web for High-utility FAQ. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482289

1 INTRODUCTION

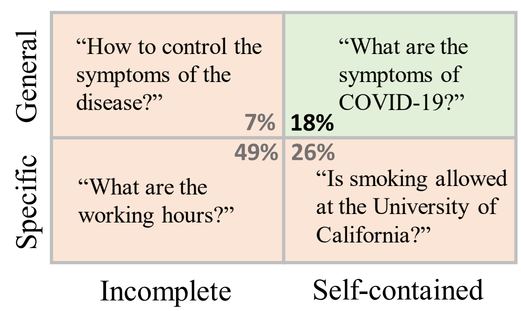

Figure 1: Only 18%[1] of FAQ on the Web are general and self-contained; these are the only questions that are suitable for building general-purpose knowledge-bases.

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

[1]Based on a study of 1,176 questions extracted from 170 FAQ web pages in the mental counseling and dental health domains.

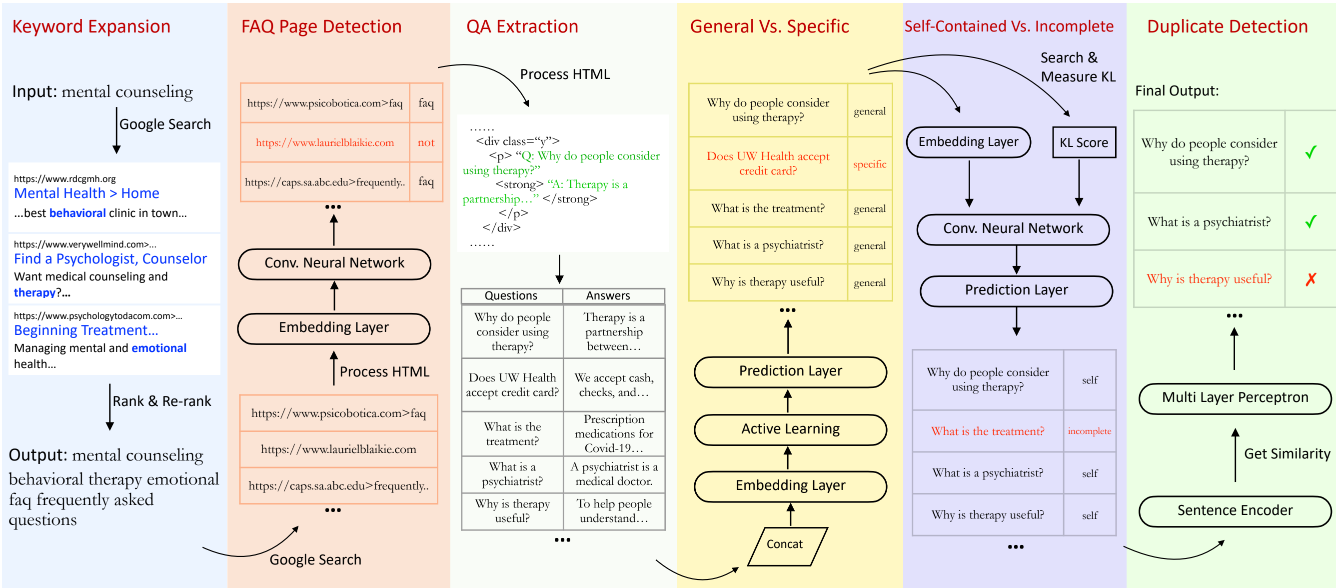

Figure 2: Overview of QuAX.

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 the 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 paged are ranked again based on their entropy and the 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:

$$e_{\mathrm{w}} = -\sum_{s} \frac{f_{\mathrm{s}}(w)+\lambda}{\sum_{\mathrm{s}} f_{\mathrm{s}}(w)+|S|\lambda} log_2 \frac{f_{\mathrm{s}}(w)+\lambda}{\sum_{\mathrm{s}} f_{\mathrm{s}}(w)+|S|\lambda}, \quad (1)$$

where s ∈ S = {SS, RS}, *SS* being the Snippets Set (Snippets returned from searching Google), *RS* being Random Set. λ 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 [2] 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 (`<h1-4>, <p>`) are replaced with `TAG2`, and questions marks are replaced with the tag `QUESTION MARK`. As we 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>`.

[2] https://boilerpipe-web.appspot.com/

Algorithm 1 QA Extractor

```
Input: FW: List of FAQ Websites
Output: UQA: Unclassified QA Pairs
1: Q ← [], A ← [], UQA ← {}
2: for each website w ∈ FW do
3:    D_w ← ExtractHTML(w) {Extract DOM structure of web page}
4:    if Check1Catg(D_w) is true {Web page falls into category 1} then
5:       for each element e ∈ D_w do
6:          if element e ∈ {h1, h2, h3, h4} then
7:             NH_e ← ExtractNextElement(e) {Parse next element in DOM tree}
8:             if NH_e ∈ {p, div} and length(e) > min_Qlength and length(NH_e > min_Alength) then
9:                Q ← Q ⋃ Text(e)
10:               A ← A ⋃ Text(NH_e)
11:   else
12:      if Check2Catg(D_w) is true {Web page falls into category 2} then
13:         Repeat 5-10 for <p> or <div> pair
14:      else
15:         for each element e ∈ D_w do
16:            C_e ← GetAllChildElements(e)
17:            for each child element c ∈ C_e do
18:               if e ∈ {p, div} and c ∈ {strong, br, a} then
19:                  Q ← Q ⋃ Text(e)
20:                  A ← A ⋃ Text(c)
21:                  UQA ← UQA ⋃{Q, A}
22: 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 following 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 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 its 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 contain 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>,
 etc. So we check all the child elements (line 18) and if child element contains
 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, 40] and we select our 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 re-trained. 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 θ to select the labels with the highest levels of confidence. We use the equation below to calculate posterior probabilities:

$$\phi_{\mathrm{LC}}(x) = \underset{x}{argmax}(1 - P_{\theta}(\hat{y}|x)) \tag{2}$$

where x is the instance to be predicted and $\hat{y}$ 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 co-reference). We collect such data by extracting random sentences from Mayoclinic articles [3]. We collect Specific sentences by selecting responses to specific user issues from Twitter customer support dataset [4]. 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

[3] https://www.mayoclinic.org/

[4] https://www.kaggle.com/thoughtvector/customer-support-on-twitter

Table 1: Datasets summary (used in our trainable modules)

Module	Num. of examples	Avg. tokens/instance	Description
FAQ Page detection*	250 websites	1,862	HTML of webpages
General vs. Specific*	19,442 sentences	16	Mayo Clinic & Twitter
Self-cont. vs. Incomplete*	1,002 questions	8	FAQ from the web
Duplicate Detection†	404,291 question pairs	23	Labeled question pairs

*Original datasets; Available at: https://github.com/shihabrashid-ucr/quax-dataset

†Available 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:

$$D_{\mathrm{KL}}(P||Q) = -\sum_{x\in\chi} P(x)log(\frac{Q(x)}{P(x)}), \tag{3}$$

where P and Q are defined over the same probability space χ.

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 χ 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

Table 2: Keywords precision

Domain	Without KE	DR	Updated DR
Mental Counseling	0.7000	0.7200	0.8000
Dental Health	0.8200	0.8600	0.7800
Plastic Surgery	0.8400	0.8000	0.9600
Medical Marijuana	0.6400	0.8400	0.7800
Covid 19	0.5200	0.6800	0.7800
Average	0.7000	0.7800	**0.8200**

Table 3: FAQ page detection accuracy (baselines)

Domain	Baseline (CNN)	Baseline (RNN)	Baseline (LSTM)	Baseline (BiL-STM)	Baseline (Self-BiL-STM)
MC	0.8400	0.6400	0.6200	0.6200	0.6200
DH	0.8800	0.6200	0.5800	0.6000	0.6400
PS	0.8400	0.6000	0.5800	0.5800	0.6200
MM	0.8800	0.6600	0.6000	0.6000	0.6000
CO	0.8600	0.6000	0.5800	0.6200	0.6000
Average	**0.8600**	0.6240	0.5920	0.6040	0.6160

performs better than the baseline (Without Keyword Expansion (KE)) and *Double Ranking (DR)*. Only for dental health, our system performs worse than both the baseline and *Double Ranking*. For this domain, the expanded keywords are "dental health hygiene teeth gums decay". We see that some of the keywords here are general and apply to many different domains related to healthcare like "hygiene" and "decay". Because of this, some of the returned results were articles regarding hygiene and not FAQ pages. Overall our updated algorithm performs better than both baseline and *Double Ranking*; our module achieves 17% higher precision on average.

3.2 FAQ Page Detection

Competitors. We show the performance of many deep learning based classifiers including RNN [29], LSTM [15], BiLSTM [41], and Self-BiLSTM [43]. We also show the improved performance of the best performing classifier after integrating our HTML preprocessing algorithm to show the efficacy of our algorithm.
Evaluation Methodology. For each domain, the test set includes the top 25 returned FAQ websites and the top 25 returned not-FAQ websites by searching Google with the expanded keywords. The training set is our proposed dataset which includes 250 labeled HTML pages. While training for a domain, the data points from that domain are omitted for fair results. We use accuracy and F1 scores in this evaluation.
Results. We see in Table 3 that CNN provides significantly better performance than the rest of the classifiers. The textual content of an HTML page is complex and large; therefore, convolutional networks can take the best advantage of such complex structure. We also see from Table 4 that our method achieves an average increase of 5% in accuracy and 6% increase in F1 scores.

Table 4: FAQ page detection comparison

Domain	CNN		QuAX Page Detector	
	Accuracy	F1	Accuracy	F1
MC	0.8400	0.8400	0.9000	0.8900
DH	0.8800	0.8650	0.9400	0.9350
PS	0.8400	0.8350	0.8600	0.8500
MM	0.8800	0.8700	0.9200	0.9150
CO	0.8600	0.8500	0.9200	0.9150
Average	0.8600	0.8500	**0.9080**	**0.9010**

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

Domain	Baseline (CNN)	Baseline (RNN)	Baseline (LSTM)	Baseline (BiL-STM)	Baseline (Self-BiL-STM)
MC	0.6670	0.6150	0.5567	0.5690	0.6280
DH	0.7343	0.6640	0.6610	0.6754	0.6670
PS	0.7230	0.6410	0.5830	0.5830	0.6230
MM	0.6338	0.5778	0.5319	0.5322	0.5715
CO	0.7215	0.6730	0.6020	0.6410	0.6678
Average	**0.6959**	0.6341	0.5869	0.6001	0.6314

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 6: General vs. Specific classification comparison

Domain	CNN		Active Learning + CNN	
	Accuracy	F1	Accuracy	F1
MC	0.6670	0.6560	0.7118	0.7008
DH	0.7343	0.7128	0.7656	0.7474
PS	0.7230	0.7210	0.7830	0.7800
MM	0.6338	0.6042	0.7042	0.6913
CO	0.7215	0.6823	0.7974	0.7567
Average	0.6959	0.6752	**0.7524**	**0.7352**

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

Domain	Baseline (CNN)	Baseline (RNN)	Baseline (LSTM)	Baseline (BiL-STM)	Baseline (Self-BiL-STM)
MC	0.6500	0.5400	0.6000	0.5600	0.5900
DH	0.6230	0.6385	0.5901	0.5081	0.4918
PS	0.5990	0.5990	0.5545	0.6000	0.5545
MM	0.6640	0.6150	0.5983	0.5664	0.5664
CO	0.5324	0.5040	0.4748	0.5100	0.5539
Average	**0.6136**	0.5793	0.5635	0.5489	0.5513

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.

Table 8: Self-contained vs. Incomplete classification comparison

Domain	CNN		KL + CNN	
	Accuracy	F1	Accuracy	F1
MC	0.6500	0.6485	0.6853	0.6746
DH	0.6230	0.6012	0.6400	0.6370
PS	0.5990	0.5920	0.6363	0.6300
MM	0.6640	0.6520	0.6923	0.6811
CO	0.5325	0.5240	0.5600	0.5410
Average	0.6137	0.6035	**0.6427**	**0.6327**

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

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

Table 10: Top 10 Extracted Questions

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

not as many individual organizations that have FAQs about COVID-19 compared to other domains.

Table 10 shows examples of the generated questions for the mental counseling and the COVID-19 domains. From Table 10, in the mental counseling domain, we see that there are some questions that are not properly classified. For example, Question 6 *"What behavioral health concerns does UW health treat?"*. Although this is a self-contained question, it talks about information regarding University of Washingtons health system. This is a specific question but this type of question is difficult for the system to detect. The 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 the intersection of question answering and FAQ retrieval which is similar to [12, 31]. In their FAQ retrieval task, they rely on Google search's *"intitle:faq"* which is ineffective compared to our respective module because it misses pages which do not have "FAQ" in title. The tasks of question generation [25, 34] where questions are generated from an input passage and question answering with FAQ retrieval [19, 26, 33] where appropriate FAQ are retrieved from a knowledge base of FAQ both resemble our problem but differ in various ways. The authors in [33] argue that the number of QA 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 but, to the best of our knowledge, most existing research has been on malicious or phishing website detection [2, 24]. A work that is relevant to Module 3 is QnA Maker by Microsoft that does similar 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. Whereever 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

[1] Parag Agrawal, Tulasi Menon, Aya Kam, Michel Naim, Chaikesh Chouragade, Gurvinder Singh, Rohan Kulkarni, Anshuman Suri, Sahithi Katakam, Vineet Pratik, et al. 2020. QnAMaker: Data to Bot in 2 Minutes. In *Companion Proceedings of the Web Conference 2020*. 131–134.

[2] Betul Altay, Tansel Dokeroglu, and Ahmet Cosar. 2019. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection. *Soft Computing* 23, 12 (2019), 4177–4191.

[3] Bang An, Wenjun Wu, and Huimin Han. 2018. Deep Active Learning for Text Classification. In *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*. 1–6.

[4] Garrett Beatty, Ethan Kochis, and Michael Bloodgood. 2019. The use of unlabeled data versus labeled data for stopping active learning for text classification. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*. IEEE, 287–294.

[5] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 795–804.

[6] Ankush Chatterjee, Manish Gupta, and Puneet Agrawal. 2020. FAQaugmenter: suggesting questions for enterprise FAQ pages. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 829–832.

[7] Jiangning Chen, Heinrich Matzinger, Haoyan Zhai, and Mi Zhou. 2018. Centroid estimation based on symmetric kl divergence for multinomial text classification problem. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1174–1177.

[8] Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 467–474.

[9] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781* (2016).

[10] Tivadar Danka and Peter Horvath. [n.d.]. modAL: A modular active learning framework for Python. ([n. d.]). https://github.com/cosmic-cortex/modAL available on arXiv at https://arxiv.org/abs/1805.00979.

[11] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2006), 1–16.

[12] Sparsh Gupta and Vitor R Carvalho. 2019. FAQ retrieval using attentive matching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 929–932.

[13] Kristian Hammond, Robin Burke, Charles Martin, and Steven Lytinen. 1995. FAQ finder: a case-based approach to knowledge navigation. In *Proceedings the 11th Conference on Artificial Intelligence for Applications*. IEEE, 80–86.

[14] Stefan Henß, Martin Monperrus, and Mira Mezini. 2012. Semi-automatically extracting FAQs to improve accessibility of software development knowledge. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 793–803.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Jizhou Huang, Ming Zhou, and Dan Yang. 2007. Extracting Chatbot Knowledge from Online Discussion Forums.. In *IJCAI*, Vol. 7. 423–428.

[17] Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 76–83.

[18] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[19] Mladen Karan and Jan Šnajder. 2016. FAQIR–a frequently asked questions retrieval test collection. In *International Conference on Text, Speech, and Dialogue*. Springer, 74–81.

[20] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[21] Gary King, Patrick Lam, and Margaret E Roberts. 2017. Computer-Assisted Keyword and Document Set Discovery from Unstructured Text. *American Journal of Political Science* 61, 4 (2017), 971–988.

[22] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[23] Yu-Sheng Lai, Kuao-Ann Fung, and Chung-Hsien Wu. 2002. Faq mining via list detection. In *COLING-02: Multilingual Summarization and Question Answering*.

[24] Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, and Wenyin Liu. 2019. A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems* 94 (2019), 27–39.

[25] Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*. 1106–1118.

[26] Yosi Mass, Boaz Carmeli, Haggai Roitman, and David Konopnicki. 2020. Unsupervised FAQ Retrieval with Question Generation and BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 807–812.

[27] Yossi Matias, Dvir Keysar, Gal Chechik, Ziv Bar-Yossef, and Tomer Shmiel. 2017. Generating related questions for search queries. US Patent 9,679,027.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[29] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[30] Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. 2018. Question categorization and classification using grammar based approach. *Information Processing & Management* 54, 6 (2018), 1228–1243.

[31] Alejandro Moreo, Maria Navarro, Juan L Castro, and Jose Manuel Zurita. 2012. A high-performance FAQ retrieval method using minimal differentiator expressions. *Knowledge-based systems* 36 (2012), 9–20.

[32] Alejandro Moreo, M Romero, JL Castro, and Jose Manuel Zurita. 2012. FAQtory: A framework to provide high-quality FAQ retrieval systems. *Expert Systems with Applications* 39, 14 (2012), 11525–11534.

[33] Wataru Sakata, Tomohide Shibata, Ribeka Tanaka, and Sadao Kurohashi. 2019. FAQ retrieval using query-question similarity and BERT-based query-answer relevance. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1113–1116.

[34] Sheng Shen, Yaliang Li, Nan Du, Xian Wu, Yusheng Xie, Shen Ge, Tao Yang, Kai Wang, Xingzheng Liang, and Wei Fan. 2020. On the Generation of Medical Question-Answer Pairs.. In *AAAI*. 8822–8829.

[35] AB Siddique, Fuad Jamour, and Vagelis Hristidis. 2021. Linguistically-Enriched and Context-AwareZero-shot Slot Filling. In *Proceedings of the Web Conference 2021*. 3279–3290.

[36] AB Siddique, Fuad Jamour, Luxun Xu, and Vagelis Hristidis. 2021. Generalized Zero-shot Intent Detection via Commonsense Knowledge. *arXiv preprint arXiv:2102.02925* (2021).

[37] AB Siddique, Samet Oymak, and Vagelis Hristidis. 2020. Unsupervised paraphrasing via deep reinforcement learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1800–1809.

[38] Eriks Sneiders. 2009. Automated FAQ answering with question-specific knowledge representation for web self-service. In *2009 2nd Conference on Human System Interactions*. IEEE, 298–305.

[39] Shuai Wang, Zhiyuan Chen, Bing Liu, and Sherry Emery. 2016. Identifying Search Keywords for Finding Relevant Social Media Posts.. In *AAAI*. 3052–3058.

[40] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. 2009. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 917–926.

[41] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia conference on language, information and computation*. 73–78.

[42] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[43] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*. 207–212.