ClarQ: A large-scale and diverse dataset for Clarification Question Generation

Vaibhav Kumar
Language Technologies Institute
Carnegie Mellon University
vaibhav2@cs.cmu.eu

Alan W Black
Language Technologies Institute
Carnegie Mellon University
awb@cs.cmu.edu

Abstract

Question answering and conversational systems are often baffled and need help clarifying certain ambiguities. However, limitations of existing datasets hinder the development of large-scale models capable of generating and utilising clarification questions. In order to overcome these limitations, we devise a novel bootstrapping framework (based on self-supervision) that assists in the creation of a diverse, large-scale dataset of clarification questions based on post-comment tuples extracted from stackexchange. The framework utilises a neural network based architecture for classifying clarification questions. It is a two-step method where the first aims to increase the precision of the classifier and second aims to increase its recall. We quantitatively demonstrate the utility of the newly created dataset by applying it to the downstream task of question-answering. The final dataset, ClarQ, consists of ∼2M examples distributed across 173 domains of stackexchange. We release this dataset[1] in order to foster research into the field of clarification question generation with the larger goal of enhancing dialog and question answering systems.

1 Introduction

The ubiquitous nature of conversations has led to the identification of various interesting problems (Clark et al., 2019). One of these problems is the ability of a system to ask for clarifications (Rao and Daumé III, 2018; Aliannejadi et al., 2019) to a natural language question.

A user's complex information need is often lost due to the brevity of the posed question. This leads to an under-specified question containing information gaps which lowers the probability of providing the correct answer. Thus, it would be an improvement if a conversational or a question answering system had a mechanism for refining user questions with follow-ups (De Boni and Manandhar, 2003). In literature, such questions have been termed ***Clarification Questions*** (De Boni and Manandhar, 2003; Rao and Daumé III, 2018, 2019).

In the domain of question-answering, the major advantages of a clarification question are its ability to resolve ambiguities (Wang et al., 2018; Aliannejadi et al., 2019) and to improve the probability of finding the most relevant answer. For conversational systems, asking such questions help in driving the conversation deeper along with better engagement of the user (Li et al., 2016; Yu et al., 2016).

Recently, Rao and Daumé III (2018, 2019) have provided a dataset based on stackexchange and used it for clarification question retrieval as well as generation. They also modify a dataset based on Amazon Question-Answering and Product Reviews (McAuley et al., 2015; McAuley and Yang, 2016) to make it suitable for the same task. On the other hand, Aliannejadi et al. (2019) created a dataset (Qulac) built on top of TREC web collections.

However, there are several shortcomings to these datasets, which limit the development of generalizable and large-scale models aimed to tackle the problem of clarification question generation. The stackexchange dataset (Rao and Daumé III, 2018) is created by utilising simple heuristics. This adds a lot of noise, thereby reducing the number of actual clarification questions. It also limits the inclusion of diverse types of questions as it is collected from three similar domains (askubuntu, superuser and unix). The question generation model of Rao and Daumé III (2019) achieves a very low BLEU score when trained on this dataset. On the other hand, the dataset based on Amazon reviews is a poor proxy for clarification questions because product descriptions are not actual questions demanding an answer and

[1] https://github.com/vaibhav4595/ClarQ

there is no information gap that needs to be addressed.

To overcome the shortcomings of existing datasets, we devise a novel bootstrapping framework based on self-supervision to obtain a dataset of clarification questions from various domains of stackexchange. The framework utilises a neural network based architecture to classify clarification questions. In a two step procedure, the framework first increases the precision of the classifier and then increases its recall. The first step is called down-sampling, where the classifier is iteratively trained on the most confident predictions (carried forward over from the previous iteration). The second step is the up-sampling procedure, where the classifier is iteratively trained by successively adding more positively classified examples. This step provides a boost in recall while restricting the drop in precision to a minimum. The classifier trained on the final iteration is then used for identification of clarification questions.

The overall process ensures that the final dataset is less noisy and, at the same time, consists of a large and diverse number of examples. We must emphasize that, given the large amount of data available on stackexchange, a classifier with moderate recall still serves our purpose. However, it is imperative that precision of the classifier be reasonably high.

2 Methodology

Stackexchange is a network of online question answering websites. On these websites, users may comment on the original post with content such as third party URLs, clarifying questions, etc. We only want to select comments which act as clarifying questions and remove the rest as noise. To this end, we devise a bootstrapping framework for training a classifier capable of identifying clarifying questions.

The bootstrapping method utilises a neural network based classifier $\mathcal{L}$ which is posed with the task of clarification question detection. Formally, given a tuple (p, q), where $p \in P$ is a post and $q \in q_p$ is a comment made on p, the task is to predict whether q is an actual clarification question for p. This makes it a binary classification problem, where a label 1 indicates q being an an actual clarification question and 0 indicates otherwise.

2.1 Data Collection

We first utilise the stackexhange data dump available at `https://archive.org/details/stackexchange`. We extract the posts and the comments made by users on those posts from 173 different domains. We remove all posts which did not have a provided answer. The comments made on the posts act as a potential candidate for clarifying question. This leads to 6,186,934 tuples of (p, q).

2.2 Bootstrapping

First, we initialise a seed dataset that is used to train $\mathcal{L}$ using the process of iterative refinement as described later. Iterative-refinement itself is subdivided into two parts: (1) Down-Sampling (2) Up-Sampling.

2.2.1 Classifier $\mathcal{L}$

We utilise a neural network based architecture for the classifier $\mathcal{L}$. Inspired by Lowe et al. (2015), $\mathcal{L}$ utilises a dual encoder mechanism i.e it uses two separate LSTMs (Hochreiter and Schmidhuber, 1997) for encoding a post p and a question q. The dual encoder generates hidden representations h_p and h_q for p and q respectively. The resulting element-wise product of h_p and h_q is further passed on to fully connected layers before making predictions via softmax. More formally, the entire process can be summarised as follows:

$$h_p = LSTM_P(p) \tag{1}$$

$$h_q = LSTM_Q(q) \tag{2}$$

$$h_{pq} = \phi(h_p \odot h_q) \tag{3}$$

$$\hat{y} = Softmax(h_pq) \tag{4}$$

where, $\odot$ represents the element-wise product, ϕ represents the non-linearity introduced by the fully connected layers and ψ represents the final classification layer.

2.2.2 Seed Selection

In order to select seeds for the bootstrapping procedure, we consider all the collected posts but only use the last comment made on these posts as clarifying questions. We make the assumption that the comments act as a proxy for a clarification question. Later, we remove all (p, q) tuples where q does not have a question mark. Intuitively, the last comment can be a better signal for identifying clarifying questions as it has more chances of capsulizing the requirements of the original post. It can also be more opinionated than others. We then randomly sample a question from the same domain as that of the post and treat it as an instance of a negative clarification question. Thus each question gets paired with a positive and a negative clarification question. We denote this seed dataset as D_0.

Algorithm 1 Iterative Refinement

```
1: $N \leftarrow 5$
2: $D_0 \leftarrow$ Seed Data
3: $\mathcal{T} \leftarrow$ Annotated Ground Truth
4: for $i = 1, 2, \ldots, N$ do                ▷ down-sampling
5:     $\mathcal{L} \leftarrow$ Classifier
6:     train $\mathcal{L}$ on $D_{i-1}$
7:     $D_{temp} \leftarrow []$
8:     for $(p, q) \in D_{i-1}$ do
9:         $y \leftarrow \mathcal{L}(p, q)$
10:        if $y$ is true positive then
11:            add $(p, q)$ to $D_{temp}$
12:        end if
13:    end for
14:    Sort $D_{temp}$ using prediction confidence
15:    $D_i \leftarrow$ top 40% of $D_{temp}$
16:    Randomly sample Negatives for $D_i$
17: end for
18: $S_N \leftarrow D_N$
19: for $i = N, N-1, \ldots 0$ do               ▷ up-sampling
20:    $\mathcal{L} \leftarrow$ Classifier
21:    train $\mathcal{L}$ on $S_N$
22:    $S_{temp} \leftarrow []$
23:    for $(p, q) \in D_{i-1}$ do
24:        $y \leftarrow \mathcal{L}(p, q)$
25:        if $y$ is true positive then
26:            add $(p, q)$ to $S_{temp}$
27:        end if
28:    end for
29:    $S_{i-1} \leftarrow S_{temp}$
30:    Randomly Sample Negatives for $S_{i-1}$
31: end for
32: $\mathcal{L}_{best} \leftarrow$ Classifier
33: $\mathcal{L}_{best}$ on $S_0$
34: Use $\mathcal{L}_{best}$ to classify remaining data
```

2.2.3 Iterative Refinement

The procedure is described in Algorithm 1. This entire process can be segmented into two parts.

Down-Sampling: The aim of this step is to increase the precision of the classifier. In the first iteration of this step, the classifier $\mathcal{L}$ is trained on the seed dataset D_0. After training is complete, $\mathcal{L}$ classifies D_0 and the most confident 40% of the positives are selected to train $\mathcal{L}$ in the next iteration. This process is continued for N iterations. Each iteration leads to a new dataset D_i (which is smaller in size than D_{i-1}. Intuitively, the precision of $\mathcal{L}$ on the task of selecting actual clarification question should increase at the end of each iteration as it is successively trained only on the examples which it was more confident about in the previous round.

Up-Sampling: This step is intended to improve the recall of $\mathcal{L}$ while restricting the loss of precision to a minimum. In the first iteration, $\mathcal{L}$ is trained on $S_N = D_N$ i.e the data obtained at the last iteration of the down-sampling procedure. After training is complete, $\mathcal{L}$ is used for classifying D_{N-1} (which is obtained during the second-list iteration of the down-sampling process). The tuples which get classified as positive are used for training $\mathcal{L}$ in the next round. This process continued for N iterations. Note that this procedure has two major differences to the iterative procedure of the down-sampling process. First, instead of using $\mathcal{L}$ for classifying the same dataset which it was trained on, it is used for classifying an up-sampled version of the current dataset. Second, it relaxes the condition of selecting 40% of the most confident examples. Intuitively, this relaxation should help in increasing the recall of the classifier and at the same time should not drastically hamper the precision (as it operates only on the examples which it classifies as positives).

Note that, in order to provide the classifier with examples of non-clarifying questions, we randomly sample negative examples at the end of each iteration (during both up and down-sampling). This is similar to the way in which the D_0 is created.

2.2.4 Classifying Remaining Data

At the end of the iterative refinement procedure, we obtain a dataset on which $\mathcal{L}$ can achieve a good precision and moderate recall on the task of classifying clarification questions. Thus, $\mathcal{L}$ is finally trained on S_0 and used for classifying the 6,186,934 tuples of (p, q) extracted from stackexchange. We again emphasize that it is more important to obtain better precision, as it reduces the amount of noise added to the dataset. Given that there are a large number of (p, q) tuples, a moder-