Asking Clarifying Questions

Narsupalli Yaswanth 19AE3AI02 yasshu.yaswanth@gmail.com

M.R Sreevatsa 19AE30013

Ayush Pandey 19AE10001

Ashish Sudhir Gokarnkar 18IM30027

Abstract

This research paper explores the problem of clarifying questions in open domain dialogue systems. The paper presents two subtasks that must be solved to enable the system to obtain additional information and context for a conversation. The first subtask is determining when to ask a clarifying question - a classification model, and the second subtask is determining which question to ask - a retriever model. The paper proposes a solution to these subtasks via implementing pre-trained language learning models, which enables open domain dialogue systems to ask relevant clarifying questions and improve the quality of conversations. Also, an effort to curate an ensemble model is pursued to club both the classification and retriever based tasks. The effectiveness of the proposed solution is evaluated through experiments, and the results indicate its potential to enhance the performance of open domain dialogue systems.

1 Introduction

The concept of asking clarifying questions in NLP (Natural Language Processing) involves using various techniques and methods to obtain additional information or clarification on a specific topic or issue. It involves asking questions to further understand a person's intent, meaning, or perspective when communicating through written or spoken language. It is an important step in natural language understanding, as it helps to reduce ambiguity and provide more accurate interpretations of text or speech. This can be done through various methods such as keyword spotting, named entity recognition, sentiment analysis, and topic modeling, among others. The following can be solved using 2 sub-tasks:

1. **When to ask a clarifying question?**: Given an instruction, does it require for a clarifying question to be asked? This can be seen as a binary classification problem. The label '1' would indicate a need for asking a clarifying question while '0' would indicate the opposite.

2. **Which clarification question to ask?**: Given an instruction, the model has to rank a list of clarifying questions according to their relevance.

2 Overall Methodologies

2.1 Dataset (Aliannejadi et al., 2021), (Aliannejadi et al., 2020)

Dataset used is the ClariQ dataset. It is an extension on the Qulac (Aliannejadi et al., 2019) dataset the benchmark is mostly on the training data that Qulac provides. Summary of ClariQ is as follows:

Feature	Value
# train (dev) topics	187 (50)
# faceted topics	141
# ambiguous topics	57
# single topics	39
# facets	891
# total questions	3,929
# single-turn conversations	11,489
# multi-turn conversations	1 million
# documents	2 million

Table 1: Summary of ClariQ

train.tsv and *dev.tsv* have the same format. They contain the topics, facets, questions, answers, and clarification need labels. These are considered to be the main files, containing the labels of the training set. Regarding the question relevance labels for each topic, these labels can be extracted indirectly: each row only contains the questions that are considered to be relevant to a topic. Therefore, any other question is deemed irrelevant while computing Recall@k. In the *train.tsv* and *dev.tsv* files, these fields exist

- **topic_id**: the ID of the topic (initial_request)
- **initial_request**: the query (text) that initiates the conversation
- **topic_desc**: a full description of the topic as it appears in the TREC Web Track data
- **clarification_need**: a label from 1 to 4, indicating how much it is needed to clarify a topic. If an *initial_request* is self-contained and would not need any clarification, the label would be 1. While if a *initial_request* is absolutely ambiguous, making it impossible for a search engine to guess the user's right intent before clarification, the label would be 4. Labels 2 and 3 represent other levels of clarification need, where clarification is still needed but not as much as label 4.
- **facet_id**: the ID of the facet
- **facet_desc**: a full description of the facet (information need) as it appears in the TREC Web Track data
- **question_id**: the ID of the question as it appears in question_bank.tsv
- **question**: a clarifying question that the system can pose to the user for the current topic and facet. answer: an answer to the clarifying question, assuming that the user is in the context of the current row (i.e., the user's initial query is initial_request, their information need is facet_desc, and question has been posed to the user)

2.2 Metrics

2.2.1 Macro Average F1 Score (Mic)

It is used for classification (task 1): "When to ask a clarifying question?".

Macro Average F1 Score measures the performance of a model in multi-class classification problems. It is the average F1 score calculated for each class, where F1 score is the harmonic mean of precision (model's accuracy) and recall (completeness). Precision measures the proportion of correctly predicted positive instances out of all predicted positive instances, while recall measures the proportion of correctly predicted positive instances out of all true positive instances.

It is a useful metric when the dataset is imbalanced, i.e., some classes have fewer samples than others, as it gives equal weightage to each class. A high Macro Average F1 Score indicates that the model has good overall performance in predicting all classes, while a low score indicates poor performance in one or more classes.

2.2.2 Mean Reciprocal Rank (Mea)

Retrieval task (task 1): "When to ask a clarifying question?" is evaluated using this metric.

MRR is a metric used to evaluate the performance of information retrieval systems. It measures the effectiveness of a system in ranking relevant results higher than non-relevant results. For each query, the reciprocal rank of the first relevant result is calculated by taking the inverse of the rank of that result in the list of retrieved results. MRR is calculated as the average of the reciprocal ranks of the first relevant result for each query in the evaluation dataset.

MRR is a useful metric for evaluating the overall performance of a retrieval system because it considers not only the presence of relevant results but also their rank in the list of retrieved results. A higher MRR score indicates a better performing system, as it is able to retrieve more relevant results and rank them higher.

2.3 Models used for tasks

For classification (task 1): "When to ask a clarifying question?", the following models have been used:

- GPT, GPT-2, OPT, BERT, BERT, RoBERTa

With respect to retrieval (task 2): "Which clarification question to ask?", these models have been implemented:

- BERT, RoBERTa, ALBERT, ELECTRA

2.4 Ensemble Model

Combining a classifier model and retriever model for asking clarifying questions in open-domain NLP can be achieved through a pipeline architecture. The pipeline architecture consists of two main components: a retriever model and a classifier model. Once the classifier model has classified each document, the pipeline architecture can select the most relevant document and extract the answer to the question from that document. If the classifier model determines that none of the candidate documents are relevant to the question, the pipeline architecture can prompt the user to provide more information or rephrase the question. A pipeline architecture can improve the accuracy by leveraging the strengths of each model. The retriever model is effective at finding relevant information, while the classifier model can accurately classify whether the information is relevant to the question. Pipeline architecture that is implemented:

1. Train the classifier model using the datasets given
2. Question is inputted to the classifier model (e.g. BERT)
3. If the classification is:
    - 1 or 2: Question doesn't require clarity. Output this directly.
    - 3 or 4: Question requires clarity. Proceed further.
4. Send the question to the retriever model (e.g. Electra)
5. Electra ranks the top 3 clarifying questions as per the contextual semantics
6. Choose a clarifying question out of the 3 as per a probability function

3 Model Implementations

3.1 GPT (Radford et al., 2018)

GPT is a deep learning model developed by OpenAI for NLP tasks. It is based on the Transformer architecture and uses unsupervised learning to pretrain the model on large amounts of text data. This allows the model to learn the patterns and structures of language, making it effective on a wide range of NLP tasks without extensive fine-tuning.

GPT can be used for classification tasks by fine-tuning the pre-trained model on a labeled dataset and adding a classification layer on top of the model's output. GPT-based models have achieved state-of-the-art performance on various classification tasks. One advantage of using GPT for classification tasks is that it can leverage the pre-trained language models' knowledge of the underlying structure and patterns of language, which can lead to improved performance on tasks with limited labeled data or noisy input. However, specialized models designed specifically for a task may outperform GPT-based models on that task.

3.2 GPT-2 (Radford et al., 2019)

GPT-2 is a large-scale language model developed by OpenAI that can generate human-like text in a variety of styles and formats. It is based on the Transformer architecture and is pre-trained on a large corpus of text data, allowing it to learn the underlying patterns and structures of language. GPT-2 can perform well on a wide range of NLP tasks without the need for extensive fine-tuning and has been used in applications such as chatbots, language translation, and content generation.

Similar to GPT, GPT-2 can be fine-tuned for classification tasks by adding a classification layer on top of the pre-trained model. It can achieve state-of-the-art performance on various classification tasks due to its ability to capture complex patterns and dependencies in the input data. However, GPT-2 may not always perform as well as specialized models designed for the specific task. It is important to evaluate the performance of GPT-2 on a particular task before using it for classification.

3.3 OPT (Zhang et al., 2022)

The OPT model was proposed in Open Pre-trained Transformer Language Models by Meta AI. OPT is a series of open-sourced large causal language models which perform similar in performance to GPT3. Key features of OPT are:

- OPT has the same architecture as BartDecoder
- Contrary to GPT2, OPT adds the EOS token </s> to the beginning of every prompt

3.4 Electra (Clark et al., 2020)

Electra is a pre-training method for NLP tasks that uses a generative adversarial network (GAN). The generator model is an encoder that produces a sequence of contextualized embeddings, while the discriminator model is another encoder that classifies whether each token is the original token from the input sequence or a masked token replaced by the generator model. The discriminator is trained to identify the original tokens, while the generator is trained to generate realistic replacements for the masked tokens, with the goal of minimizing the discriminator's accuracy. This approach is designed to improve the efficiency and effectiveness of pre-training models for NLP tasks.

Electra is a pre-training method for NLP tasks that uses a discriminative approach to distinguish between real and fake input sequences. This method, called "discriminative pre-training, has been shown to produce more accurate and efficient models than traditional pre-training methods. When used for classification, the pre-trained Electra model is fine-tuned on a labeled dataset by adding a classification layer on top of the model's output. Electra's modular architecture allows for easier customization and fine-tuning for specific tasks, and it can leverage discriminative pre-training to learn more efficiently from limited labeled data.

3.5 BERT (Devlin et al., 2018)

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model introduced by Google in 2018. It is based on the Transformer architecture and is pre-trained on a large corpus of text data using a masked language modeling task and a next sentence prediction task. These pre-trained representations can then be fine-tuned on specific NLP tasks, such as text classification, question answering, and named entity recognition, by adding a task-specific layer on top of the pre-trained model. One of the advantages of BERT is its ability to capture complex patterns and relationships between different words in a sentence, leading to improved performance on a wide range of NLP tasks.

One advantage of using BERT for classification tasks is that it is a bidirectional model, meaning that it can leverage the context of both the preceding and following words in a sentence when generating representations for each word. This allows BERT to capture complex patterns and dependencies in the input data, which can lead to improved performance on tasks with limited labeled data or noisy input. Additionally, BERT can be fine-tuned on a wide range of classification tasks with relatively little additional training data, making it a versatile tool for NLP applications.

BERT can also be used for retrieval tasks, such as question answering and document retrieval. In this case, BERT is fine-tuned on a dataset where the task is to retrieve relevant documents or answers given a query or question. The fine-tuned BERT model can then be used to generate embeddings for the query and each document in a collection. The similarity between the query and each document can be computed based on the cosine similarity between their embeddings, and the top-k most similar documents can be returned as the retrieval results.

3.6 RoBERTa (Liu et al., 2019)

RoBERTa (Robustly Optimized BERT Pre-training Approach) is a natural language processing (NLP) model developed by Facebook AI Research. It is based on the BERT architecture and is trained using a similar methodology, but with several improvements to the pre-training process. These improvements include longer training times, larger batch sizes, and dynamic masking during training. One of the key advantages of RoBERTa is its ability to capture more subtle nuances in language, such as sarcasm and irony, due to its longer training and improved masking strategy.

When used for classification tasks, RoBERTa can be fine-tuned on a labeled dataset by adding a classification layer on top of the model's output. The pre-trained RoBERTa model generates contextualized embeddings for each input text, which are then fed into the classification layer for prediction. RoBERTa-based models have demonstrated competitive performance on various classification tasks, particularly in domains with limited labeled data or complex language patterns.

To use RoBERTa for retrieval tasks, a fine-tuning approach similar to classification tasks can be used. Specifically, a passage or a document can be encoded by the RoBERTa model, and the query can be encoded similarly. Then, a similarity score can be calculated between the encoded query and each encoded passage/document, and the passages/documents with the highest similarity score can be returned as the top results.

3.7 ALBERT (Lan et al., 2019)

ALBERT was developed to improve the efficiency and scalability of BERT by reducing the number of parameters while maintaining or even improving the model's performance. ALBERT achieves this by using two parameter-reduction techniques called "factorized embedding parameterization" and "cross-layer parameter sharing." Factorized embedding parameterization reduces the number of parameters required to represent the vocabulary by factorizing the embedding matrix into two smaller matrices. Cross-layer parameter sharing reduces the number of parameters required by sharing parameters across the layers of the model. These techniques allow ALBERT to use fewer parameters while still achieving state-of-the-art performance on various NLP tasks.

ALBERT can be used for retrieval tasks by encoding the input query and the candidate documents using the pre-trained model and computing the similarity between them using a similarity function such as dot product or cosine similarity. The candidate documents can be ranked based on their similarity scores to the query, and the top-k documents can be returned as the retrieval results.

4 Results

4.1 Classification Task

1. GPT performed the best among the models tested for the classification task with an F1 score of 0.5761

2. Other models tested, including OPT, Electra, GPT2, BERT, and ROBERTA had lower F1 scores ranging from 0.54779 to 0.47569

3. The differences in F1 scores between the models are relatively small, with the highest and lowest scores differing by only about 10

4. The specific pre-trained models tested may not be representative of all models within their respective architectures

5. Fine-tuning and hyperparameter tuning can also have a significant impact on the models' performance The small dataset used for training may make it difficult for even complex models like GPT and OPT to perform well.

4.2 Retrieval Task

1. ROBERTA may be the most effective model among those tested for the retrieval task with an MRR score of 0.35289

2. BERT and ELECTRA had similar performance with MRR scores of 0.32317 and 0.32161, respectively, while ALBERT scored 0.342369

3. The differences between the models may not be significant enough to make a definitive conclusion, and hyperparameter tuning can have a significant impact on the models' performance

5 Future Work

5.1 Implement an encoder decoder architecture like T5

1. Define the Encoder: It takes in the input text and produces a set of contextualized embeddings. T5 uses BERT as its encoder.

2. Define the Decoder: Takes in the contextualized embeddings produced by the encoder and generates the output text. T5 uses a Transformer-based decoder with a sequence-to-sequence model.

3. Fine-tune the Model: Once the encoder-decoder architecture is defined, the next step is to fine-tune the model on a specific task. This involves adding a task-specific head on top of the model and training it on a labeled dataset.

4. Tokenization: The input text and output text are tokenized into a sequence of subword units using a tokenizer. T5 uses the SentencePiece tokenizer.

5. Training: The model is trained on a large corpus of text data using unsupervised learning to pre-train the encoder. Once the encoder is pre-trained, the decoder is trained in a supervised manner on the task-specific dataset.

6. Inference: Once the model is trained, it can be used for inference on new input text to generate the corresponding output text.

5.2 Implement techniques to Improve retrieval task scores

1. Query expansion: Expands the original query with additional terms to retrieve more relevant documents. This can be done using methods like synonym expansion, word embedding-based expansion, or knowledge graph-based expansion.

2. Relevance feedback: Uses user feedback to improve the relevance of the retrieved documents. This can be done by allowing the user to provide feedback on the relevance of the retrieved documents and using that feedback to refine the search results.

3. Learning to rank: Training a machine learning model to rank the retrieved documents based on their relevance to the query. This can be done using methods like pointwise, pairwise, or listwise learning to rank.

4. Use of external knowledge: Incorporating external knowledge sources like knowledge graphs, dictionaries, or ontologies can improve the retrieval task scores by providing additional context and semantic information.

Rank	Creator	Model		Dev			Test	
			P	R	F1	P	R	F1
1	TAL ML	RoBERTa+++	0.6039	0.5600	0.5551	0.5981	0.6557	0.6070
2	CactusJam	RoBERTa+ Stats	0.6200	0.5800	0.5717	0.5963	0.5902	0.5416
3	TAL ML	RoBERTa++	0.5807	0.5400	0.5375	0.5290	0.5574	0.5253
4	Algis	RoBERTa+++ CatBoost	0.1402	0.2800	0.1854	0.5171	0.5246	0.5138
5	NTES_ALONG	cneed_add_prior_v2	0.6200	0.6000	0.5984	0.5007	0.5082	0.5018
6	NTES_ALONG	cneed_merge	0.5830	0.5200	0.5192	0.4847	0.5082	0.4960
7	NTES_ALONG	cneed_dist	0.5452	0.5200	0.5177	0.4852	0.4918	0.4868
8	Karl	RoBERTa-v2	0.4609	0.4600	0.4356	0.4465	0.5410	4871
9	NTES_ALONG	RoBERTa + prior	0.4554	0.4600	0.4567	0.4926	0.4754	0.4799
10	Algis	BartBoost	0.7008	0.7000	0.6976	0.4813	0.4754	0.4756

Table 2: ClariQ Official Scoreboard: Classification

Model	Validation F1 score	Test F1 score	Rank
GPT	0.40603	0.5761	2
OPT	0.456389	0.54779	3
Electra	0.48992	0.50482	5
GPT-2	0.42431	0.50263	6
BERT	0.42868	0.49313	8
RoBerta	0.50171	0.47569	10

Table 3: Classification

Rank	Creator	Model	Recall@5	Recall@10	Recall@20	Recall@30
1	NTES_ALONG	ReRanker-v4	0.3404	0.6329	0.8335	**0.8744**
2	NTES_ALONG	ReRanker-v3	0.3414	0.6351	0.8316	**0.8721**
3	NTES_ALONG	ReRanker-v2	0.3382	0.6242	0.8177	**0.8685**
4	CogIR	BERT fusion-topic-div-pass.-v2	0.3384	0.6314	0.8073	**0.8573**
5	TAL ML	RoBERTA++	0.3395	0.6251	0.8176	**0.8568**
6	Karl	RoBERTa-v2	0.3355	0.6237	0.7990	**0.8492**
7	CogIR	BERT fusion-topic-div-pass.-v2	0.3314	0.6149	0.8074	**0.8448**
8	Karl	Roberta	0.3406	0.6255	0.8006	**0.8436**
9	Soda	BERT + BM25	0.3272	0.6061	0.8013	**0.8433**
10	Soda	BERT + BM25-v2	0.3013	0.5866	0.8006	**0.8433**

Table 4: ClariQ Official Scoreboard 1: Retrieval

5. Multi-modal retrieval: Retrieves documents that contain both textual and non-textual content, such as images or videos. This can be done using methods like cross-modal retrieval or joint embedding learning.

6. Data augmentation: Techniques like back-translation, paraphrasing, or sentence shuffling can be used to generate additional training data and improve the model's generalization ability, leading to better retrieval task scores.

References

Mean reciprocal rank - wikipedia. `https://en.wikipedia.org/wiki/Mean_reciprocal_rank`. (Accessed on 04/30/2023).

Micro, macro & weighted averages of f1 score, clearly explained | by kenneth leung | towards data science. `https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-expla` (Accessed on 04/30/2023).

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. Convai3: Generating clarifying questions for open-domain dialogue systems (clariq).

Rank	Creator	Model	MRR	P@1	nDCC@3	nDCC@5
-	ClariQ	oracle BestQuestion	0.4881	0.4275	**0.2107**	0.1759
1	Karl	RoBERTa	0.3190	0.2342	**0.1265**	0.1130
2	NTES_ALONG	ReRanker-v4	0.3140	0.2379	**0.1229**	0.1097
3	Soda	BERT + BM25-v2	0.3216	0.2453	**0.1196**	0.1097
4	NTES_ALONG	ReRanker-v2	0.3034	0.2119	**0.1171**	0.1033
5	ClariQ	BM25	0.3134	0.2193	**0.1151**	0.1061
6	Soda	BERT + BM25	0.3134	0.2193	**0.1151**	0.1061
7	ClariQ	NoQuestion	0.3223	0.2268	**0.1134**	0.1059
8	Pinta	BERT-v3	0.3044	0.2119	**0.1131**	0.1021
9	NTES_ALONG	BM25 + RoBERTa	0.3045	0.2156	**0.1108**	0.1025
10	CogIR	BERT fusion-topic-div-pass.	0.3025	0.2193	**0.1078**	0.0983

Table 5: ClariQ Official Scoreboard 2: Retrieval

Model	Recall@5	Recall@10	Recall@20	Recall@30
BERT	0.348657074	0.61764859	0.69128187	0.69128187
ROBERTA	0.329309392	0.583945796	0.689615203	0.69128187
ALBERT	0.346052032	0.617605983	0.689615203	0.69128187
ELECTRA	0.339292229	0.586681075	0.689853298	0.69128187

Table 6: Retrieval: Question Relevance

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10	NDCG@20
BERT	0.170833333	0.167429763	0.161111524	0.155963425	0.151207283
ROBERTA	0.209375	0.176425841	0.177347101	0.168699908	0.153682925
ALBERT	0.211979167	0.186889171	0.174268515	0.165212256	0.142191197
ELECTRA	0.18125	0.17231298	0.168288593	0.160605002	0.147239492

Table 7: Retrieval: Document Relevance 2

Model	P@1	P@3	P@5	P@10	P@20	MRR
BERT	0.21875	0.197916667	0.1825	0.166875	0.14	0.323175982
ROBERTA	0.2625	0.208333333	0.21	0.183125	0.141875	0.352893846
ALBERT	0.25625	0.2125	0.19625	0.178125	0.1253125	0.342369389
ELECTRA	0.225	0.2125	0.2025	0.176875	0.138125	0.321614638

Table 8: Retrieval: Document Relevance 3

Model	Recall@5	Recall@10	Recall@20	Recall@30
BERT	0.341289274	0.605563099	0.758474608	0.768167192
ROBERTA	0.321729956	0.572140384	0.751619168	0.768167192
ALBERT	0.340291296	0.612605306	0.756090107	0.768167192
ELECTRA	0.308047584	0.569919053	0.749075196	0.768167192

Table 9: Retrieval: Document Relevance 4

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2021. Building and evaluating open-domain dialogue corpora with clarifying questions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4473–4484, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd interna-*

tional acm sigir conference on research and development in information retrieval, pages 475–484.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.