### 4.1 Query resolution as term classification in the conversation history

Previous work has modeled query resolution as a sequence to sequence task [15, 21], where the source sequence is $q_{1:i}$ and the target sequence is $q_i^*$, where $q_i^*$ is a gold standard resolution of the current turn query $q_i$. For instance, the gold standard resolution of turn #4 in Table 1 is: "When was Saosin's first album released?"

However, since (i) the initial retrieval step of our pipeline (Section 3.2.1) is a term-based model that treats queries as bag of words, and (ii) the supervised neural ranker we use in the re-ranking step (Section 3.2.2) is robust to queries that are not well-formed natural language texts [48], our query resolution model does not necessarily need to output a well-formed natural language query but rather a set of terms to expand the query. Besides, sequence to sequence based models generally need a massive amount of data for training in order to get reasonable performance due to their generation objective [16]. Therefore, we model query resolution as a term classification task: given the conversation history $q_{1:i-1}$ and the current turn query $q_i$, output a binary label (relevant or non-relevant) for each term in $q_{1:i-1}$. Terms in the conversation history $q_{1:i-1}$ that are tagged as relevant are appended to the current turn query $q_i$ to form the predicted current turn query resolution $\hat{q}_i$.

We define the set of relevant resolution terms $E^*(q_i)$ as:

$$E^*_{q_i} = terms(q_i^*) \cap terms(q_{1:i-1}) \setminus terms(q_i), \tag{2}$$

where $q_i^*$ is a gold standard resolution of the current turn query $q_i$. Under this formulation, the set of relevant terms $E^*_{q_i}$ represents the missing context from the conversation history $q_{1:i-1}$. For instance, the set of gold standard resolution terms $E^*_{q_i}$ for turn #4 in Table 1 is {Saosin, first}. Note that $E^*_{q_i}$ can be empty if $q_i = q_i^*$, i.e., the current turn query does not need to be resolved, or if $terms(q_i^*) \cap terms(q_{1:i-1})$ is empty. In our experiments $terms(q_i^*) \cap terms(q_{1:i-1}) \approx terms(q_i^*)$, and therefore almost all the gold standard resolution terms can be found in the conversation history.

### 4.2 Query resolution model

In this section, we describe our query resolution model, QuReTeC. Figure 2a shows the model architecture of QuReTeC. Each term in the input sequence is first encoded using bidirectional transformers [43] – more specifically BERT [13]. Then, a term classification layer takes each encoded term as input and outputs a score for each term. We use BERT as the encoder since it has been successfully applied in tasks similar to ours, such as named entity recognition and coreference resolution [13, 20, 24]. Next we describe the main parts of QuReTeC in detail, i.e., input sequence, BERT encoder and Term classification layer.

(1) *Input sequence.* The input sequence consists of all the terms in the queries of the previous turns $q_{1:i-1}$ and the current turn $q_i$. It is formed as: [<CLS>, $terms(q_1)$, ..., $terms(q_{i-1})$, <SEP>, $terms(q_i)$], where <CLS> and <SEP> are special tokens. We add a special separator token <SEP> between the previous turn $q_{i-1}$ and the current turn $q_i$ in order to inform the model where the current turn begins. Figure 2b shows an example input sequence and the gold standard term labels.

(2) *BERT encoder.* BERT first represents the input terms with WordPiece embeddings using a 30K vocabulary. After applying multiple transformer blocks, BERT outputs an encoding for each term. We refer the interested reader to the original paper for a detailed description of BERT [13].

(3) *Term classification layer.* The term classification layer is applied on top of the representation of the first sub-token of each term [13]. It consists of a dropout layer, a linear layer and a sigmoid function and outputs a scalar for each term. We mask out the output of <CLS> and the current turn terms, since we are not interested in predicting a label for those (see Equation (2) for the definition and Figure 2b for an example).

In order to train QuReTeC we need a dataset containing gold standard resolution terms $E^*_{q_i}$ for each $q_i$. The terms in $E^*_{q_i}$ are labeled as relevant and the rest of the terms ($terms(q_{1:i-1}) \setminus E^*_{q_i}$) as non-relevant. Assuming there exists a gold standard resolution $q_i^*$ for each $q_i$, we can derive $E^*_{q_i}$ using Equation (2). We use standard binary cross entropy as the loss function.

### 4.3 Generating distant supervision for query resolution

Recall that the gold standard resolution $q_i^*$ includes the information in $q_i$ and the missing context of $q_i$ that exists in the conversation history $q_{1:i-1}$. As described above, we can train QuReTeC if we have a gold standard resolution $q_i^*$ for each $q_i$. Obtaining such special-purpose gold standard resolutions is cumbersome compared to almost readily available general-purpose passage relevance labels for $q_i$. We propose a distant supervision method to generate labels to train QuReTeC. Specifically, we simply replace $q_i^*$ with a relevant passage $p^*_{q_i}$ in Equation (2) to extract the set of relevant resolution terms $E^*_{q_i}$. Table 1 illustrates this idea with an example dialogue and the relevant passage to the current turn query. The gold standard resolution terms extracted with this distant supervision procedure for this example are {Saosin, first, band}.

Intuitively, the above procedure is noisy and can result in adding terms to $E^*_{q_i}$ that are non-relevant, or adding too few relevant terms to $E^*_{q_i}$. Nevertheless, we experimentally show in Section 6.2 that this distant supervision signal can be used to substantially reduce the number of human-curated gold standard resolutions required for training QuReTeC.

The distant supervision method we describe here makes QuReTeC more generally applicable than other supervised methods such as the method in Elgohary et al. [15] that can only be trained with gold standard query resolutions. This is because, apart from manual annotation, query-passage relevance labels can be potentially obtained at scale by using click logs [19], or weak supervision [12].

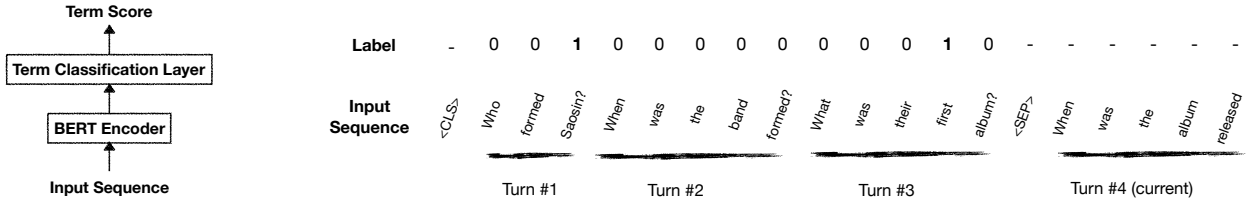
## 5 EXPERIMENTAL SETUP

### 5.1 Research questions

We aim to answer the following research questions:

(RQ1) How does the QuReTeC model perform compared to other state-of-the-art methods?

(RQ2) Can we use distant supervision to reduce the amount of human-curated training data required to train QuReTeC?




**(a) QuReTeC model architecture.**

**(b) Example input sequence and gold standard term labels (1: relevant, 0: non-relevant) for QuReTeC.**

**Figure 2**

(RQ3) How does QuReTeC's performance vary depending on the turn of the conversation?

For all the research questions listed above we measure performance in both an intrinsic and an extrinsic sense. *Intrinsic* evaluation measures query resolution performance on term classification. *Extrinsic* evaluation measures retrieval performance at both the initial retrieval and the reranking steps.

## 5.2 Datasets

*5.2.1 Extrinsic evaluation – retrieval.* The TREC CAsT dataset is a multi-turn passage retrieval dataset [11]. It is the only such dataset that is publicly available. Each topic consists of a sequence of queries. The topics are open-domain and diverse in terms of their information need. The topics are curated manually to reflect information seeking conversational structure patterns. Later turn queries in a topic depend only on the previous turn queries, and not on the returned passages of the previous turns, which is a limitation of this dataset. Nonetheless, the dataset is sufficiently challenging for comparing automatic systems, as we will show in Section 6.1.3. Table 3 shows statistics of the dataset. The original dataset consists of 30 training and 50 evaluation topics. 20 of 50 topics in the evaluation set were annotated for relevance by NIST assessors on a 5-point relevance scale. We use this set as the TREC CAsT test set. The organizers also provided a small set of judgements for the training set, however we do not use it in our pipeline. The passage collection is the union of two passage corpora, the MS MARCO [28] (Bing), and the TREC CAR [14] (Wikipedia passages).[4]

*5.2.2 Intrinsic evaluation – query resolution.* The original QuAC dataset [7] contains dialogues on a single Wikipedia article section regarding people (e.g., early life of a singer). Each dialogue contains up to 12 questions and their corresponding answer spans in the section. It was constructed by asking two crowdworkers (a student and a teacher) to perform an interactive dialogue about a specific topic. Elgohary et al. [15] crowdsourced question resolutions for a subset of the original QuAC dataset [7]. All the questions in the *dev* and *test* splits of [15] have gold standard resolutions. We use the *dev* split for early stopping when training QuReTeC and evaluate on the *test* set. When training with gold supervision (gold standard query resolutions), we use the *train* split from [15], which is a subset of the train split of [7]; all the questions therein have gold standard resolutions. Since QuAC is not a passage retrieval collection, in order to obtain distant supervision labels (Section 4.3), we use a window of 50 characters around the answer span to extract passage-length texts, and we treat the extracted passage as the relevant passage. When training with distant labels, we use the part of the *train* split of [7] that does not have gold standard resolutions.

The TREC CAsT dataset [11] also contains gold standard query resolutions for its test set. However, it is too small to train a supervised query resolution model, and we only use it as a complementary *test* set.

The two query resolution datasets described above have three main differences. First, the conversations in QuAC are centered around a single Wikipedia article section about people whereas the conversations in CAsT are centered around an arbitrary topic. Second, the answers of the QuAC questions are spans in the Wikipedia section whereas the CAsT queries have relevant passages that originate from different Web resources besides Wikipedia. Third, later turns in QuAC do depend on the answers in previous turns, while in CAsT they do not (Section 3.1). Interestingly, in Section 6.1 we demonstrate that despite these differences, training QuReTeC on QuAC generalizes well to the CAsT dataset.

Table 4 provides statistics for the two datasets.[5] First, we observe that the QuAC dataset is much larger than CAsT. Also, QuAC has a larger number of terms on average than CAsT (~97 vs ~40) and a larger negative-positive ratio (~20:1 vs ~40:1). This is because in QuAC the answers to the previous turns are included in the conversation history whereas in CAsT they are not. For this reason, we expect query resolution on QuAC to be more challenging than on CAsT.

## 5.3 Evaluation metrics

*5.3.1 Extrinsic evaluation – retrieval.* We report NDCG@3 (the official TREC CAsT evaluation metric), Recall, MAP, and MRR at rank 1000. We also provide performance metrics averaged per turn to show how retrieval performance varies across turns.

We report on statistical significance with a paired two-tailed t-test. We depict a significant increase for $p < 0.01$ as $^{\blacktriangle}$.

*5.3.2 Intrinsic evaluation – query resolution.* We report on Micro-Precision (P), Micro-Recall (R) and Micro-F1 (F1), i.e., metrics calculated per query and then averaged across all turns and topics. We ignore queries that are the first turn of the conversation when calculating the mean, since we do not predict term labels for those.

[4] The Washington Post collection was also part of the original collection but it was excluded from the official TREC evaluation process and therefore we do not use it.

[5] Note that the first turn in each topic does not need query resolution because there is no conversation history at that point and thus the query resolution CAsT test has 20 (the number of topics) fewer queries than in Table 3.

**Table 3: TREC CAsT 2019 multi-turn passage retrieval dataset statistics.**

| Split | #Topics | #Queries | #Labelled passages per topic | #Relevant passages per topic | #Labelled passages per query | #Relevant passages per query |
|---|---|---|---|---|---|---|
| Test | 20 | 173 | 1,467.50 ± 252.86 | 406.00 ± 190.18 | 169.65 ± 36.69 | 46.94 ± 31.53 |

**Table 4: Query resolution datasets statistics. In the Split column, we indicate the where the positive term labels originate from: either gold (gold standard resolutions) or distant (Section 4.3).**

| Dataset | Split | #Queries | #Terms (per query) | |
|---|---|---|---|---|
| | | | Total | Positive |
| QuAC | Train (gold) | 20,181 | 97.96 ± 61.02 | 4.56 ± 3.88 |
| | Train (distant) | 31,538 | 99.78 ± 62.36 | 6.90 ± 5.59 |
| | Dev (gold) | 2,196 | 95.49 ± 58.79 | 4.49 ± 3.90 |
| | Test (gold) | 3,373 | 96.96 ± 59.24 | 4.30 ± 3.86 |
| CAsT | Test (gold) | 153 | 39.97 ± 17.97 | 1.89 ± 1.62 |

## 5.4 Baselines

We perform intrinsic and extrinsic evaluation by comparing against a number of query resolution baselines. Next, we provide a detailed description of each baseline:

- **Original** This method uses the original form of the query. We explore different variations for constructing $\hat{q}_i$: (1) current turn only (cur), (2) current turn expanded by the previous turn (cur+prev), (3) current turn expanded by the first turn (cur+first), and (4) all turns.
- **RM3 [1]** A state-of-the-art unsupervised pseudo-relevance feedback model.[6] RM3 first performs retrieval and treats the top-$n$ ranked passages as relevant. Then, it estimates a query language model based on the top-$n$ results, and finally adds the top-$k$ terms to the original query. As with Original, we report on different variations for constructing the query: cur, cur+prev, cur+first and all turns. In order to apply RM3 for query resolution we append the top-$k$ terms to the original query $q_i$ to obtain $\hat{q}_i$.
- **NeuralCoref**[7] A coreference resolution method designed for chatbots. It uses a rule-based system for mention detection and a feed-forward neural network that predicts coreference scores. We perform coreference resolution on the conversation history $q_{1:i-1}$ and the current turn query $q_i$. The output $\hat{q}_i$ consists of $q_i$ and the predicted terms in $q_{1:i-1}$ where terms in $q_i$ refer to.
- **BiLSTM-copy [15]** A neural sequence to sequence model for query resolution. It uses a BiLSTM encoder and decoder augmented with attention and copy mechanisms and also a coverage loss [40]. It initializes the input embeddings with pretrained GloVe embeddings.[8] Given $q_{1:i-1}$ and $q_i$, it outputs $\hat{q}_i$. It was optimized on the QuAC gold standard resolutions.

*5.4.1 Intrinsic evaluation – query resolution.* In order to perform intrinsic evaluation on the aforementioned baselines, we take the query resolution they output ($\hat{q}_i$) and apply Equation (2) by replacing $q_i^*$ with $\hat{q}_i$ to obtain the set of predicted resolution terms.

*5.4.2 Extrinsic evaluation – initial retrieval.* Here, apart from the aforementioned baselines, we also use the following baselines:

- **Nugget** [18]. Extracts substrings from the current and previous turn queries to build a new query for the current turn.[9]
- **QCM** [47]. Models the edits between consecutive queries and the results list returned by the previous turn query to construct a new query for the current turn.
- **Oracle** Performs initial retrieval using the gold standard resolution query. Released by the TREC CAsT organizers.

*5.4.3 Extrinsic evaluation – reranking.* Since developing specialized rerankers for multi-turn passage retrieval is not the focus of this paper, we evaluate the reranking step using ablation studies. For reference, we also report on the performance of the top-ranked TREC CAsT 2019 systems [11]:

- **TREC-top-auto** Uses an automatic system for query resolution and BERT-large for reranking.
- **TREC-top-manual** Uses the gold standard query resolution and BERT-large for reranking.

## 5.5 Implementation & hyperparameters

**Multi-turn passage retrieval** We index the TREC CAsT collections using Anserini with stopword removal and stemming.[10] In the initial retrieval step (section 3.2.1) we retrieve the top 1000 passages using QL with Dirichlet smoothing (we set $\mu = 2500$). We use the default value for the fusion parameter $k = 60$ [8] in Eq. (1). In the reranking step (section 3.2.2) we use a PyTorch implementation of BERT for retrieval [25]. We use the `bert-base-uncased` pretrained BERT model. We fine-tune the BERT reranker with MSMARCO passage ranking dataset [2]. We train on 100K randomly sampled training triples from its training set and evaluate on 100 randomly sampled queries of its development set. We use the Adam optimizer with a learning rate of 0.001 except for the BERT layers for which we use a learning rate of 3e−6. We apply dropout with a probability of 0.2 on the output linear layer. We apply early stopping on the development set with a patience of 2 epochs based on MRR.

**Query resolution** We use the `bert-large-uncased` model. We implement QuReTeC on top of HuggingFace's PyTorch implementation of BERT.[11] We use the Adam optimizer and tune the learning rate in the range {2e−5, 3e−5, 3e−6}. We use a batch size of 4 and do gradient clipping with the value of 1. We apply dropout on the term classification layer and the BERT layers in the range $\{0.1, 0.2, 0.3, 0.4\}$. We optimize for F1 on the QuAC dev (gold) set.

[6] Note that given the very small size of the TREC CAsT training set we do not compare to more sophisticated yet data-hungry pseudo-relevance feedback models such as [29].
[7] https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30
[8] https://nlp.stanford.edu/projects/glove/
[9] We use the nugget version that does not depend on anchors text since they are not available in our setting.
[10] https://github.com/castorini/anserini
[11] https://github.com/huggingface/transformers