

MARRS: Multimodal Reference Resolution System

Halim Cagri Ates, Shruti Bhargava¹, Site Li, Jiarui Lu, Siddhardha Maddula, Joel Ruben Antony Moniz², Anil Kumar Nalamalapu, Roman Hoang Nguyen, Melis Ozyildirim, Alkesh Patel, Dhivya Piraviperumal³, Vincent Renkens, Ankit Samal, Thy Tran, Bo-Hsiang Tseng⁴, Hong Yu⁵, Yuan Zhang, Rong Zou*
{¹shruti_bhargava, ²joelmoniz, ³dhivyaprp, ⁴bohsiang_tseng, ⁵hong_yu}@apple.com
Apple

Abstract

Successfully handling context is essential for any dialog understanding task. This context maybe be conversational (relying on previous user queries or system responses), visual (relying on what the user sees, for example, on their screen), or background (based on signals such as a ringing alarm or playing music). In this work, we present an overview of MARRS, or Multimodal Reference Resolution System, an on-device framework within a Natural Language Understanding system, responsible for handling conversational, visual and background context. In particular, we present different machine learning models to enable handling contextual queries; specifically, one to enable reference resolution, and one to handle context via query rewriting. We also describe how these models complement each other to form a unified, coherent, lightweight system that can understand context while preserving user privacy.

1 Introduction

Fast-paced advancements across modalities have presented exciting opportunities and daunting challenges for dialogue agents. The ability to seamlessly integrate and interpret different types of information is crucial to achieve human-like understanding. One fundamental aspect of dialogue agents, therefore, is their ability to understand references to context, which is essential to enable them carry out coherent conversations. Traditional reference resolution systems (Yang et al., 2019) are not sufficient for multiple modalities in a dialogue agent.

In this work, we introduce Multimodal Reference Resolution System (MARRS), targeted to understand and resolve diverse context understanding use cases. MARRS leverages multiple types of context to understand a request, while completely running on-device, keeping memory and privacy as key design factors. The key objective of MARRS

* Authors listed in alphabetical order

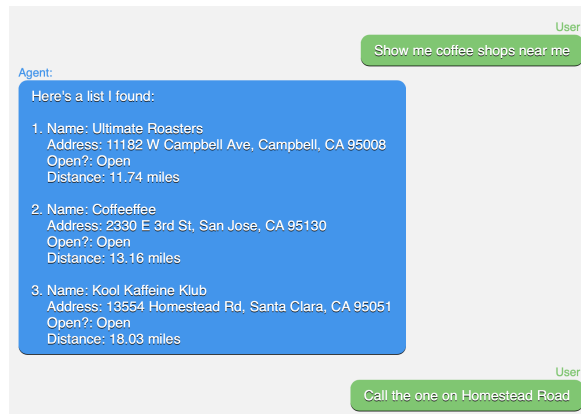


Figure 1: An example of Conversational Entity Resolution. All coffee shop names shown are author-created.

is two-fold: first, to track and maintain coherence during multiple turns of a conversation, and second, to leverage visual context to enhance context understanding. It thus aims to provide a centralized domain agnostic solution to diverse discourse and referencing tasks including, but not limited to¹:

Anaphora Resolution

User: What is Ohio's capital?
Agent: Columbus is the capital of Ohio.
User: How far away is it?

Ellipsis Resolution

User: What is the currency of France?
Agent: The Euro is the currency of France.
User: What about United States?

Screen Entity Resolution

User: Share this number with John.

Conversational Entity Resolution Note that here, the entity may be a part of the interaction without being explicitly mentioned.

User: Show me pharmacies near me.

¹Examples shown are author-created queries based on anonymized and randomly sampled virtual assistant logs.

Agent: Here are some near you: <list>

User: Call the second one

Background Entity Resolution

alarm starts ringing

User: Switch it off

Correction by Repetition

User: What is the population of Australia

Agent: The population of Australia is ...

User: I meant Austria

MARRS consumes the transcribed request, and as a part of the language understanding block, aims to allow for fluent conversations spanning multiple modalities. It takes screen entities, conversation history, and other contextual entities as input alongside the latest transcribed request; and outputs a context independent rewritten request as well as spans that link references to entities. In some cases, like *conversational referencing* above, a span with entity id may be preferred by downstream components, while a rewritten query may be preferred in *ellipsis resolution* for transparent low-effort adoption downstream. Central to the success of MARRS are its two components: the query rewriter and the reference resolver (comprising, in turn, of the mention detector and the mention resolver). The query rewriter aims to rewrite a user query to make it context independent, thereby making it self-contained. The mention detector and resolver on the other hand aim to generate reference spans.

In this paper, we delve into the system design of MARRS, its components, the reasoning behind them and how they integrate together for efficient context understanding. Note that while we find our system highly efficient and performant, detailed benchmarks and results are outside the scope of this paper. We believe this work will foster an understanding of multimodal context understanding systems and pave the way for more sophisticated and contextually-aware agents.

2 System Design

The context carryover problem is usually tackled with coreference resolution (Ng and Cardie, 2002). Traditional coreference resolution systems often identify mentions and link the mention to entities in the previous context (Lee et al., 2017). Another approach to address the problem is to rewrite the user request into a version which can be executed in a context independent way (Nguyen et al., 2021;

Quan et al., 2019; Yu et al., 2020; Tseng et al., 2021). There are pros and cons for each of the two approaches.

On one hand, coreference resolution provides spans with entities, which downstream systems can consume. This removes the need to perform entity linking again, which may add latency and/or errors, and also supports references to complex entities (like calendar events) where rewriting to a natural language query could be hard. On the other hand, the rewrite approach can handle not only the coreference resolution problem, but also other discourse phenomena such as intent carryover, corrections and disfluencies. Further, a coreference resolution system generates spans that need to be adopted by downstream systems, while a rewriting system reformulates the query itself, requiring no explicit adoption. In MARRS, we generate both reference spans and query rewrites in order to take advantage of both approaches. See Figure 2 for the design of MARRS.

There have been multiple prior works as shown in Table 1, trying to solve different aspects of reference resolution. A real-world dialogue system, however, requires the ability to simultaneously handle all of these aspects. In the MARRS system we do this using 2 major components, the Query Rewriter and the Reference Resolution System. The query rewrite component rewrites the current utterance with previous context, solving problems like anaphora and ellipses. Our reference resolution (or MDMR) component takes in contextual and screen entities and decorates the current utterance with entity information. This helps solve use cases related to screen, background and conversational references. Note that both the Query Rewriter and the Reference Resolution System are independent of each other; consequently, for efficiency, they can be run in parallel. Overall, this system consumes dialog context, the current utterance, and entities as input, and produces a rewritten utterance and reference spans as output. Furthermore, the system has been designed to run on the (relatively low-power) device to preserve the privacy of the users.

Within our coreference resolution system, our system runs on *all* user queries, since we do not know a priori if a user query requires resolution. Consequently, while end-to-end approaches have been proposed (Lee et al., 2017), we find it extremely beneficial for system performance to have a 2-stage pipeline: a light-weight Mention Detec-

Previous work on Reference Resolution	Resolu- tion	Anaphora	Ellipses	Correction by Repeti- tion	Screen Entity Reso- lution	Conversational Entity Resolu- tion
Bohnet et al. (2023)	✓	✗	✗	✗	✗	✗
Bhargava et al. (2023)	✗	✗	✗	✗	✓	✗
Nguyen et al. (2021)	✗	✗	✗	✓	✗	✗
Tseng et al. (2021)	✓	✓	✓	✗	✗	✗

Table 1: Comparison of previous work on reference resolution covering various use cases

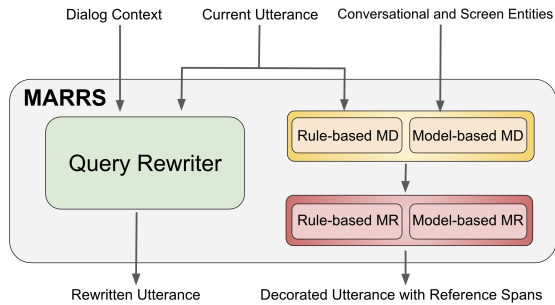


Figure 2: High level diagram show-casing how MARRS models interface with each other

tor (MD), followed by a more expensive Mention Resolver (MR) which is only run if MD detects a mention. We discuss each component in the following sections and detailed model architectures are provided in Section A.1.

2.1 Mention Detector

The Mention Detector (MD) identifies sub-strings in the user utterance that can be grounded to one or more contextual entity. These are also known as referring expressions or mentions. Some examples of referring expressions are:

How big is [this house]
Where does [he] live

2.1.1 Model-based MD

This predicts which sequences of tokens (spans) need to be resolved to an entity. The model takes in token embeddings and enumerates all spans consisting of start and end token indices. For each span, the first and last token embeddings are concatenated and fed into a feed forward network that performs binary classification. We opt for this approach instead of an LSTM or self-attention based sequence tagger because using a classifier that classifies all spans allows the model the flexibility to

identify multiple (possibly overlapping) mentions, while also allowing all spans to be classified independently in parallel (as opposed to sequentially or auto-regressively). Our approach is very similar to that of Lee et al. (2017), except that we empirically observe very little impact by removing the self-attention layer in their mention detector (primarily because our coreference dependencies tend to be much shorter than theirs), while observing very large improvements in both latency and memory. The model architecture is shown in A.1 Figure 3.

2.1.2 Rule-based MD

While the model detects referring expressions (which often include marker words like "this" and "that"), there are cases when the user refers to a contextual entity by name only (omitting the referring expression). In a user request like "Call customer support", "customer support" might refer to a support number on the user's screen. To keep the model light-weight, MD model does not consume entities; consequently, it is unable to detect that "customer support" is a referring expression. The Rule-based MD component bridges this gap by matching the gathered contextual entities to the utterance through smart string matching. If a contextual entity is found in the utterance, this sub-component outputs the span and the corresponding entity as a potential reference.

2.2 Mention Resolver

The Mention Resolver (MR) resolves references in user queries to contextual entities like phone numbers and email addresses. As with the overall system, the focus is on a low memory footprint and reusing the existing components in the pipeline. MR operates on the text and location of screen or conversational entities recognized by upstream component and the metadata of the background entities. It consumes the possible mentions identified

by MD and matches each mention to zero, one or more entities, providing a relevance score for each. It includes a mixture of a rule-based system and a machine learned model. The rule-based system is high precision and extremely fast. Consequently, if it outputs a resolution, the model is not run, which yields a substantial latency reduction.

2.2.1 Rule-based MR

Rule-based MR utilizes a set of pre-defined rules and keywords to match references to the correct category, location and text. For example, references like ordinals are matched with regex patterns sorted by the longest match; likewise, music and movie entities can be matched by relying on the presence of verbs like ‘play’.

2.2.2 Model-based MR

We also design a modular reference resolution model, inspired by Yu et al. (2018). This is trained to score how well an entity matches with the detected mention. The entities for which score crosses a threshold are then predicted. The model contains 3 modules: 1. the category module, which matches the mention with the entity’s category; 2. the location module, which matches the mention with the entity’s location; 3. the text module, which matches the mention with the text within entities, like screen texts and alarm names. Weights are computed using the request tokens to determine the aggregation of the the three module scores. We refer interested readers to Bhargava et al. (2023) for a more in-depth understanding of the model; we show the model architecture in Appendix Figure 4.

Screen-based The entities on screen are the candidate referents here. Each entity has a category like phone number and address, a bounding box representing its location on the screen and associated text values. Each of the three modules thus receive input for screen entities, and play a key role in understanding diverse references.

Conversational Here, a user’s previous conversational interaction and the VA’s responses are considered as referents. In such cases, descriptive references made by a user, such as when referring to addresses (Eg: "Show me coffee shops near me" -> "Call the one on Homestead Road") are to be handled by the text module. The location module is critical in resolving ordinal and spatial references (Eg: "Show me coffee shops near me" -> "Call the bottom one" or "Call the last one").

Background In this case, entities relevant to background tasks are potential referents. These tasks may include user-initiated tasks, such as music that’s playing in the background, or system-triggered tasks, such as a ringing alarm or a new notification. The category module is particularly important here, since a user’s references tend to be related to the type of the referent (Eg: “pause it” likely refers to music or a movie, while “stop that” could also refer to an alarm or a timer).

2.3 Query Rewriter

The Query Rewriter (QR) is the component that rewrites the last user utterance in a conversation between the user and the VA into a context-free utterance such that it can be fully interpreted and understood without the dialog context. Three use cases mentioned in Section 1 can be tackled through rewriting: Anaphora, Ellipses, and Corrections by Repetition. The output rewritten utterance is provided as an alternative to downstream components together with the original utterance, to provide them with the flexibility of choice.

Again, for the sake of latency and privacy, the QR model is run on device along with MD and MR. Unlike the more complex components in MD and MR, QR is merely an LSTM-based seq2seq model with a copy mechanism (Gu et al., 2016). It takes as input both conversational context (i.e., a sequence of interactions) and the last user query, and generates the rewritten utterance. On top of the encoder, there is a classifier that consumes the input embeddings and predicts the type of use case (‘Anaphora and Ellipsis’, ‘Correction by Repetition’ or ‘None’). ‘None’ means no rewriting is required, in which case, to further reduce latency, no decoder inference needs to be run, and the module can simply pass-through the input utterance as the output. This classification signal is also sent as part of the output to downstream systems for their use. The model architecture can be referred to Figure 5 in Appendix.

3 Datasets

Since the system handles varied kinds of references, different datasets are used for training the different components. We briefly describe here the datasets used by the system.

For Screen Entity Resolution, we collect requests referring to entities on screens by showing screenshots containing entities to annotators. One

entity is highlighted as the target entity. Annotators are asked to provide requests that refer uniquely to the marked entity. The collected requests are sent through another round of annotation for getting the mentions, in order to train MD. Interested readers can refer to [Bhargava et al. \(2023\)](#) for more details on the data collection. The requests and mentions collected alongside the entities are used to train the model-based MD and MR, as well as to evaluate the overall system.

For Conversational Entity Resolution, we show annotators a list of entities similar to the Agent turn in Figure 1. These lists are synthetically generated based on the domain. Annotators are asked to provide a request referring to any one entity in the list (similar to the second User turn in Figure 1), along with the the mention (to train MD) and the list index of the entity being referred (to train MR).

For Entity Resolution, we additionally have a synthetic data pipeline. Requests are generated through templates like ‘play [this]’ or ‘share [that address] with John’, with the marked mention used to train MD. A synthetic list of targeted entities is part of the templates, and synthetic negative entities are added while training MR.

For Query Rewriting, we use mined data from the anonymized opt-in usage data. We first identify the opportunities where user experience can be significantly improved if the desired features are enabled. In particular, for the use case of anaphora and ellipsis, we identify user queries discussing the same entity in two consecutive turns without the use of any referring expressions (context-free query). We then ask annotators to simplify these complex queries to provide queries in a more natural way (context-dependent query). For the use case of correction by repetition, we recognize queries where the user tapped on the transcribed prompt to edit the query into something else. The resulting utterance serves as a complete context-free query. We then prepend the edit parts with common prefixes such as ‘*I said*’ to synthesize the context-dependent query. By doing so, we simulate the pair of original queries and their rewrites for the two desired features. More detailed statistics and examples of both use cases are shown in Appendix A.2 and A.3.

4 Experimental Results

4.1 Metrics

We compute a bag of words token level F1 metric on the subset of tokens that are present in the target

Model	Dataset/Task	F1	EM
QR	AER	91.44	87.83
	CbR	88.12	71.44
MDMR	Screen	83.39	80.8
	Conversational	89.85	91.50
	Synthetic	97.56	96.90

Table 2: Experimental results for QR and MDMR. Here, the Synthetic MDMR dataset tests performance of both Conversational and Background Entity Resolution use-cases.

rewrite, but not in the corresponding context dependent query. This metric reflects the model’s ability to carry over tokens from previous context. We also calculate an exact string match accuracy (EM) between the model prediction and the target rewrite as a more strict comparison. Metrics for anaphora and ellipsis resolution (AER) and correction by repetition (CbR) are measured separately.

For reference resolution, metrics are computed by comparing the true target entities with the predicted entities (for which scores cross the threshold). Similar to above, we report F1 and exact match metrics. Here, exact match is 1 if the predicted entities over all predicted references exactly match the true target entities, and is 0 if any additional or missing predicted entities exist.

4.2 Performance

We present an overview of our system performance as measured on the datasets described in Section 3 in Table 2, with additional results in Appendix A.4. We find that our models afford excellent performance despite being extremely small, lightweight enough with respect to both model size and runtime inference latency to potentially deploy them to a low-power device. In particular, the reported results use a QR model with just 4.5M parameters with 1-layer 128-dim LSTMs as encoder and decoder; the MD and MR models are even lighter, with just 116k and 196k parameters respectively.

5 Conclusions

In this paper, we propose and provide a system-level overview of MARRS, a low memory system that combines multiple models to solve context understanding. Our design choices offer an interpretable and agile system. This system can improve user experience in a multi-turn dialogue agent in a fast, efficient, on-device and privacy-preserved manner.

References

- Shruti Bhargava, Anand Dhoot, Ing-marie Jonsson, Hoang Long Nguyen, Alkesh Patel, Hong Yu, and Vincent Renkens. 2023. [Referring to screen texts with voice assistants](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 752–762, Toronto, Canada. Association for Computational Linguistics.
- Bernd Bohnet, Chris Alberti, and Michael Collins. 2023. Coreference resolution through a seq2seq transition-based system. *Transactions of the Association for Computational Linguistics*, 11:212–226.
- Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.
- Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.
- Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 104–111.
- Hoang Long Nguyen, Vincent Renkens, Joris Pelemans, Srividya Pranavi Potharaju, Anil Kumar Nalamalapu, and Murat Akbacak. 2021. User-initiated repetition-based recovery in multi-utterance dialogue systems. *arXiv preprint arXiv:2108.01208*.
- Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. Gecor: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4547–4557.
- Bo-Hsiang Tseng, Shruti Bhargava, Jiarui Lu, Joel Ruben Antony Moniz, Dhivya Piraviperumal, Lin Li, and Hong Yu. 2021. Cread: Combined resolution of ellipses and anaphora in dialogues. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3390–3406.
- Wei Yang, Rui Qiao, Haocheng Qin, Amy Sun, Luchen Tan, Kun Xiong, and Ming Li. 2019. [End-to-end neural context reconstruction in Chinese dialogue](#). In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 68–76, Florence, Italy. Association for Computational Linguistics.
- Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. [Mattnet: Modular attention network for referring expression comprehension](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1307–1315.
- Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. [Few-shot generative conversational query rewriting](#). In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20*, page 1933–1936, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 Detailed Model Architectures

This section provides the model architectures of MD (Figure 3), MR (Figure 4) and QR (Figure 5) adopted in the MARRS system.

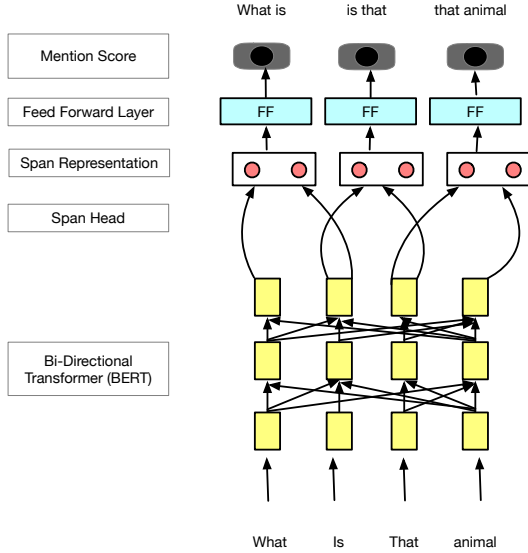


Figure 3: MD model overview

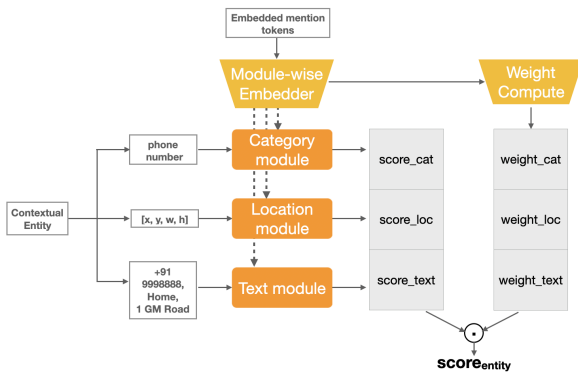


Figure 4: MR model overview

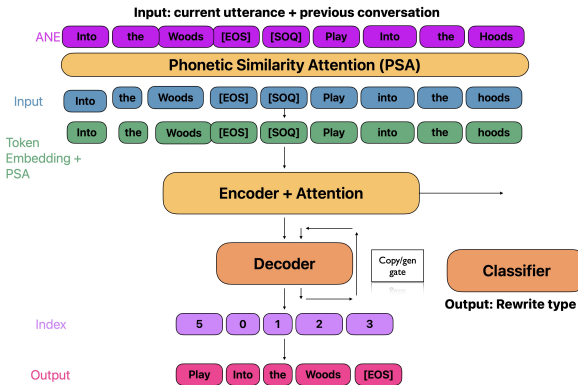


Figure 5: QR model overview

A.2 Data Statistics

Here, we present more detailed statistics around our datasets. In particular, we present the sizes of each dataset: we show how much data was used from each set for training, validation and testing. We present these numbers in Table 3.

Model	Dataset/Task	Train	Val	Test
MDMR	Screen	7.3k	0.7k	1.9k
	Conversational	2.3k	0.4k	1.2k
	Synthetic	3.9k	0.5k	1.1k
QR	AER	300.3k	37.3k	37.2k
	CbR	317.5k	39.7k	39.7k

Table 3: Dataset sizes used for the MDMR and QR models.

A.3 Data Collection

This section provides an example of anaphora, ellipsis and correction by repetition of our data mining methods, as shown in Figure 6.

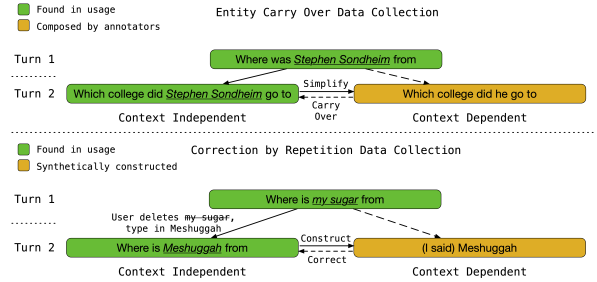


Figure 6: Illustration of data collection process for Anaphora, Ellipses and Correction by Repetition. Examples shown are author-created examples based on anonymized and randomly sampled virtual assistant logs. In both examples, utterances in green are improvement opportunities found in real-world usage, utterances in yellow are either annotated or synthetically generated. During data collection phase, we follow the solid lines. During model training, we follow the dotted lines.

A.4 Results

In this section, we present a deep dive of the results shown in Section 4.1. In particular, we present detailed precision, recall and F1 numbers for our QR, MD and MR models, as well as our joint MDMR performance. Note that in the case of QR, following Quan et al. (2019), we measure the F1 score by comparing generated rewrites and references for only the rewritten part of user utterances. This

Model	Dataset/Task	P	R	F1
MD	Screen	89.66	95.74	92.60
	Conversational	85.30	92.60	88.80
	Synthetic	99.00	99.70	99.30
MR	Screen	87.99	85.87	86.92
	Conversational	85.62	96.91	90.92
	Synthetic	98.09	97.53	97.81
MDMR	Screen	86.85	80.20	83.39
	Conversational	84.70	95.66	89.85
	Synthetic	97.92	97.21	97.56
QR	AER	92.48	90.42	91.44
	CbR	93.31	83.48	88.12

Table 4: Precision, recall and F1 scores for the MD, MR and QR models.

highlights the model’s ability to carry over essential information through rewriting. In case of MR, we consider the ground truth mentions and compute the metrics by comparing the predicted entities with the true target entities. For MDMR, we use the predicted mentioned from MD to run MR, and then compute metrics by comparing all the predicted entities with the true entities.