INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

# 5IDER: Unified Query Rewriting for Steering, Intent Carryover, Disfluencies, Entity Carryover and Repair

*Jiarui Lu*, Bo-Hsiang Tseng*, Joel Ruben Antony Moniz*,*
*Site Li, Xueyun Zhu, Hong Yu, Murat Akbacak*

Apple
{jiarui_lu,bohsiang_tseng,joelrubenantony_moniz,
site_li,xueyun_zhu,hong_yu,makbacak}@apple.com

## Abstract

Providing voice assistants the ability to navigate multi-turn conversations is a challenging problem. Handling multi-turn interactions requires the system to understand various conversational use-cases, such as steering, intent carryover, disfluencies, entity carryover, and repair. The complexity of this problem is compounded by the fact that these use-cases mix with each other, often appearing simultaneously in natural language. This work proposes a non-autoregressive query rewriting architecture that can handle not only the five aforementioned tasks, but also complex compositions of these use-cases. We show that our proposed model has competitive single task performance compared to the baseline approach, and even outperforms a fine-tuned T5 model in use-case compositions, despite being 15 times smaller in parameters and 25 times faster in latency.

**Index Terms**: voice assistants, steering, intent carryover, disfluency, entity carryover, repair, reference resolution, multitask learning

## 1. Introduction

With voice assistants making large improvements recently, natural conversations with virtual assistants has become a more common expectation. Users tend to interact with virtual assistant in an increasingly contextual fashion, expecting them to behave more like a human agent [1]. This brings out a growing challenge: the understanding system behind a voice assistant needs to be robust to several aspects of natural, conversational language that have been traditionally difficult to deal with.

One line of challenges involves the ability to use context from one turn to complete the understanding of another in multi-turn conversations. This includes use-cases like **steering** (where a follow-up turn is used to provide clarifying information to a previous turn), **intent carryover** (where a follow-up turn implicitly has the same intent as a previous turn, but for a different entity) [2] and **entity carryover** (where a follow-up turn refers to an entity in a previous turn, often through anaphora or nominal ellipses) [3, 4, 5, 6]. Another set of challenges involves artifacts in human speech that appear by virtue of humans changing their mind during a conversation, or beginning a conversation without having fully decided their intent or how they would like to convey it. This yields **disfluencies** (i.e., artifacts in speech such as filler words that don't contribute to the intent of a query and can be removed) and **repair** (where a user corrects the intent and/or entity previously referenced) [7]. This problem of understanding spoken language is further complicated by the ability of these artifacts to mix with each other, with these phenomena often appearing simultaneously in natural language. We illustrate these problems with examples in Table 1. The 5 aforementioned challenges are shown at the top, with a pair of context and follow-up queries. We also present 2 compositional challenges at the bottom, where multiple conversational challenges are involved.

*Equal contribution

Traditional approaches handle each of these challenges with dedicated systems: entity carryover is often tackled with coreference [8, 9] and ellipsis [10] resolution; intent carryover can be treated as a form of slot carryover [11, 12]; disfluencies can be solved as a sequence tagging problem [13]. While each dedicated system offers competitive performance for each challenge, a solution for all challenges would involve cascading these systems, which introduces error propagation, extra latency and disk space, and management complexities. A unified problem formulation could yield a single, joint modeling solution.

One such formulation is query rewriting [14, 15, 16, 17], which reformulates contextual queries into their context-independent counterparts. Examples of target rewritten queries for all use cases and their combinations are shown in the *Rewrite* column in Table 1. For example, for the composition of Entity Carryover and Repair, given the context query *"Who is Homer Simpson's eldest doctor"* and the follow-up query *"I said his eldest daughter"*, the reference to the context entity *Homer Simpson* needs to be resolved along with a repair of the ASR error; solving both then yields a self-contained query: *"Who is Homer Simpson's eldest daughter"*. While entity carryover [15, 17], intent carryover [2, 16] (and their combinations), disfluency [18] and repair [14] have been studied as query rewriting problems individually, to the best of our knowledge, this is the first work to look at all 5 challenges (including the under-studied challenge of handling steering) jointly, and to investigate compositional challenges.

Many prior query rewriting works consider it a summarization task [2, 15, 16, 17, 19, 20]. While these models usually offer competitive performance, they suffer from heavy latency costs brought by the decoding loop, which would have a significant impact on the responsiveness of a virtual assistant. To mitigate this, text-editing rewrite models were proposed, which output a sequence of edit actions (like deletion, insertion, swap and reordering) instead of generating tokens. Applying these edits on the source tokens yields the rewritten query, drastically reducing the output search space. In LaserTagger [21], the authors use a single layer Transformer Decoder paired with a BERT encoder [22] to reduce decoding latency. FELIX [23] goes one step further using a non-autoregressive model with tagging, reordering and inserting, eliminating the decoding loop completely.

In this paper, we aim to address this latency issue with a non-autoregressive edit-based model using composable edit-actions. In addition, we evaluate the query rewriting problem in

Table 1: *Examples of conversational use cases and their compositions tackled in this work. Examples shown are author-created queries based on anonymized and randomly sampled virtual assistant logs. Tokens marked in red indicate those marked for deletion. Tokens marked in green or with a green underline are those marked to be replaced, while those marked in blue or with a blue underline are the detected replacements. Note that colorization indicates a first step substitution, and underlines indicate a second substitution step (which only ever occurs in compositional use cases).*

| Use Case | Context | | Follow-up | Rewrite |
|---|---|---|---|---|
| **Steering** | Play Sweeny Todd | [SEP] | In my living room | Play Sweeny Todd in my living room |
| **Intent Carryover** | How old is Homer Simpson | [SEP] | What about Bart Simpson | How old is Bart Simpson |
| **Disfluency** | - | [SEP] | Take me to Suki Sushi no I said Fuki Sushi | Take me to Fuki Sushi |
| **Entity Carryover** | When does Rocket Sushi close | [SEP] | How long does it take to drive there | How long does it take to drive to Rocket Sushi |
| **Repair** | How far is San Jose by car | [SEP] | I meant San Francisco | How far is San Francisco by car |
| **Entity Carryover + Intent Carryover** | How tall is Homer Simpson | [SEP] | What about his wife | How tall is Homer Simpson's wife |
| **Entity Carryover + Repair** | Who is Homer Simpson's eldest doctor | [SEP] | I said his eldest daughter | Who is Homer Simpson's eldest daughter |

a multitask setting, including compositional use cases, which, to the best of our knowledge, is a first. Through experimental evaluation, we show that our proposed approach has competitive single- and multi-task performance compared to baseline approaches on the target use cases, and can even surpass a fine-tuned T5-small [24] model in use-case combinations, despite being 15x smaller in parameters. Our main contributions are:

1. We propose a joint edit-based query rewrite model to handle all five conversational understanding use cases and their compositions.
2. We show that our non-autoregressive model achieves performance close to or even better than strong baselines, with 25x faster latency, 15x smaller disk size and 20x better compositional data efficiency.

## 2. Data

Collecting data for conversational use cases that are not currently supported by the virtual assistant is challenging because of the cold-start problem: users' behavior attunes to the known capability of the virtual assistant, and interactions targeting unsupported use cases rarely exist. As a result, collecting such data requires clever strategies and annotations, or partial synthesis. Our data collection process starts by randomly sampling virtual assistant logs from anonymized opt-in users, which we use to create 6 datasets in total, 5 for each of the tasks we focus on, and 1 containing compositions of the tasks, all of which are either human annotated or synthetically created.

Our entity and intent carryover datasets start with identifying consecutive query pairs with common entities or intents between them. Given these pairs, annotators provide a contextual query that simplifies the follow-up through entity or intent carryover, similar to [16].

For our repair dataset, when users correct themselves by manually editing ASR transcriptions, we extract the slot where the correction was performed. We then use the slot to populate synthetic templates such as (original turn, correction phrase ("No I meant", etc.) + slot, correction turn), where the first element represents the context turn, the second element represents the followup-turn, and the third element represents the rewrite.

We formulate our disfluency dataset similarly to our repair dataset above, but use synthetic templates to stitch together the correction slot inside the query, randomly adding interregnum to simulate the disfluency.

For steering, we look at consecutive queries in which the first query is an exact prefix of the second. The non-prefix part of the second query helps us simulate what a user who wishes to use a follow-up in a steering fashion might say.

Our compositional dataset is created through synthetic templates because their rarity makes collecting real-world data exceedingly difficult. Each template requires two of the aforementioned tasks to be resolved. We identified 5 valid challenge pairs, from which we created 20 unique templates. Templates used to create training, validation and test sets are disjoint to keep this task challenging. Random entities are then filled into the templates to create the data.

Each of the 5 datasets contains 60k datapoints, and the compositional dataset contains 20k datapoints. Each dataset has an 8:1:1 training:validation:test split. On average, context turns have 5.6 tokens, current turns have 5.2 tokens, rewritten turn have 6.0 tokens, and 47% of tokens in the rewritten turn can only be found in the context turn.

## 3. 5IDER

Our proposed method 5IDER (shown in Figure 1) is an encoder-based model that learns to predict the defined *edit operations* to accomplish rewriting across all use cases. In this section, we first define these edit operations, then explain how our model architecture and training objective help to learn these operations.

### 3.1. Rewriting with edit operations

To accomplish rewriting across all different use cases, it is essential to capture the relationship between text spans within the input sequence. For instance, in the example of *entity carryover* shown in Table 1, the model needs to resolve the pronoun *it* in the follow-up turn to the named entity *Rocket Sushi* mentioned in the context. To achieve this goal, we define two edit operations, substitution and deletion, which are made possible by three model components:

- Replacement detection: this component detects the replacement, a text span which will substitute another text span to make a self-contained rewrite.
- Replacement resolution: this component detects the replaced, a text span to be substituted by the replacement.
- Deletion: this component detects tokens that need to be deleted to complete a rewrite.

These edit operations act on the concatenated dialog history, in the form of *context query + separator token ([SEP]) + follow-up query*. For single use case data, we apply the following post-processing steps to create the rewritten query: first, apply deletion; second, apply substitution by deleting the replacement, and substituting the replaced with the replacement; finally, extract the token sequence starting from the last [SEP] to the end of the sequence (or the whole token sequence if no [SEP]s are left). As an example, the replacement, replaced and deletion for each of the 5 use cases are color coded in blue, green and red in Table 1. This thus implies that, as in [2], the output vocabulary of this design is limited to the input tokens.

This edit operation design differs from [21, 23]: instead of general purpose edit operations, each use case has an inde-

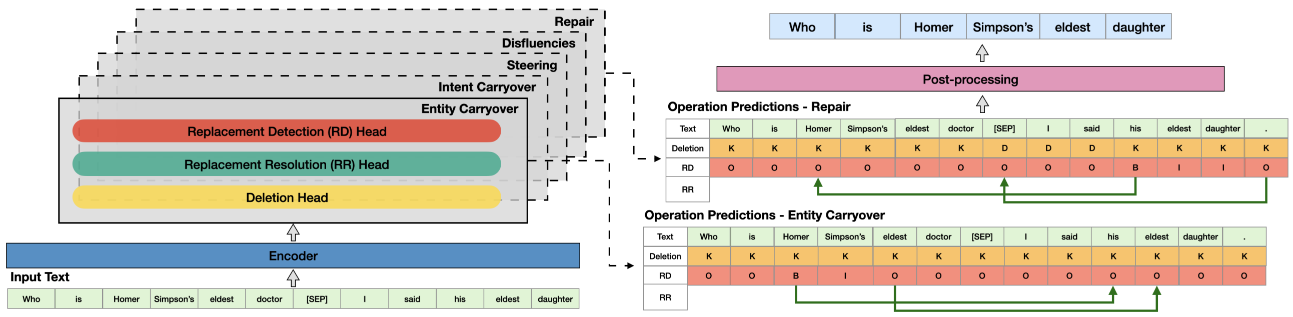
Repair
Disfluencies
Steering
Intent Carryover
Entity Carryover
Replacement Detection (RD) Head
Replacement Resolution (RR) Head
Deletion Head
Encoder
Input Text
Who is Homer Simpson's eldest doctor [SEP] I said his eldest daughter
Who is Homer Simpson's eldest daughter
Post-processing
Operation Predictions - Repair
Text
Deletion
RD
RR
Operation Predictions - Entity Carryover

Figure 1: *The proposed 5IDER architecture. The input text, the concatenation of two turns, is first encoded with TinyBert and an LSTM encoder. Then, 5 edit operation predictors (one for each task), perform Replacement Detection, Replacement Resolution and Deletion Detection for each task. Finally, all edit operations are applied in a post-processing step.*

pendent interpretation of the operations. For entity carryover, a substitution is akin to anaphora resolution, while for repair, a substitution somewhat resembles entity linking. The motivation behind this is to create a model architecture with the ability to apply edit operations independently for each use case, automatically solving use case composition by simply composing these operations. An example is shown in Figure 1. In *Who is Homer Simpson's eldest doctor [SEP] I said his eldest daughter*, entity carryover requires *his* being replaced by *Homer Simpson's*, repair requires *Homer Simpson's eldest doctor* being replaced by *his eldest daughter*, and *[SEP] I said* being deleted. To tackle this, we need to modify the second step of our aforementioned post-processing logic. We define the concept of **substitution dependency**. A substitution is dependent on another, if its replacement contains the replaced of the other. In our example, the repair substitution is dependent on the entity carryover substitution. We perform a topological sort for substitutions, applying those without dependencies and moving upwards. We thus first apply the entity carryover substitution to obtain *Who is eldest doctor [SEP] I said Homer Simpson's eldest daughter*. After this, the repair's replaced has to be adjusted from *Homer Simpson's eldest doctor* to *eldest doctor*. We then apply the repair's substitution to obtain *Who is Homer Simpson's eldest daughter [SEP] I said*. Finally, after deletion, we have the rewritten query *Who is Homer Simpson's eldest daughter*.

## 3.2. Model Architecture

As shown in Figure 1, 5IDER takes as input the concatenation of text from the context and the follow-up turn. The input sequence is then passed through a frozen TinyBert [25] model to obtain contextualized embeddings of input tokens, which is then further encoded with a trainable BiLSTM encoder.

To handle data with an arbitrary mixture of use cases, the model contains 5 copies of the 3 components mentioned in Section 3.1, one for each use case. First, the **Replacement Detection** (RD) head predicts which span in the input is a replacement. This is modeled as a BIO sequence tagging task. As shown in the running example, the entity *"Homer Simpson's"* is identified as the replacement in *entity carryover*, and *"his eldest daughter"* in the *repair* use case. When no valid replacements are predicted in a use case component (i.e., the output of the RD head is $O$ across the input sequence), the input sequence does not require any substitutions for that use case (e.g., use cases *intent carryover*, *steering* and *disfluencies* in the running example). Second, the **Replacement Resolution** (RR) head identifies the text that the replacement needs to substitute (i.e., the replaced text). Concretely, biaffine attention [26] is employed to perform self-attention to capture the relationship between input tokens. Similar to the mechanism of locating the answer span in machine comprehension, the attention distribution at the position of the beginning (end) of the replacement is supervised to attend to the beginning (end) of the corresponding replaced, as shown by the solid green arrows. Finally, the **Deletion** head, a binary classifier, is applied at each position to determine whether to delete ($D$) or keep ($K$) the token. The model predicts these edit operations separately for each use case, and they are then consolidated to form the final rewrite, as described in Section 3.1 above.

## 3.3. Optimization

When optimizing the model, supervision is provided for all three edit operation heads. For **Replacement Detection**, RD heads across all use cases are optimized with the loss $L^{RD}$, using cross-entropy (CE) between the ground-truth and the predicted RD sequence:

$$L^{RD} = \frac{1}{|U|}\sum_{u}^{|U|}\sum_{i}^{T} CE(p_{u,i}^{RD}, y_{u,i}^{RD}) \tag{1}$$

where $u$ and $i$ are respectively the indices for the use case and input position. $U$ is the total number of use cases; $T$ is the length of the input sequence; $p_{u,i}^{RD} \in \mathbb{R}^3$ is the BIO prediction at the position $i$ for the use case $u$.

For **Replacement Resolution**, the RR heads are supervised only within use cases where a replacement exists in the input sequence, in which case the loss is defined as the sum of the cross-entropy between the ground-truth and the predictions at the positions of the replacement boundary (i.e., start & end):

$$L_u^{RR} = \begin{cases} \sum_{i=\{start,end\}} CE(p_{u,i}^{RR}, y_{u,i}^{RR}), & \text{replacement exists} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $p_{u,i}^{RR} \in \mathbb{R}^T$ is the prediction over the input sequence. The overall RR loss is added up across use cases: $L^{RR} = \sum_u^{|U|} L_u^{RR}$.

For **Deletion**, the deletion heads across all use cases are trained with a binary signal for each input token:

$$L^{Del} = \frac{1}{|U|}\sum_{u}^{|U|}\sum_{i}^{T} CE(p_{u,i}^{Del}, y_{u,i}^{Del}) \tag{3}$$

where $p_{u,i}^{Del} \in \mathbb{R}^2$ is the binary prediction on whether to keep or delete the token at position $i$ for the use case $u$.

The overall loss, $L$, used for model training is the sum of all three losses with equal weight: $L = L^{RD} + L^{RR} + L^{Del}$.

Table 2: *Model comparison in model size, disk size and latency. Latency is measured on the same hardware and averaged across test sets. Seq2Seq latency is set as 1 unit for ease of comparison.*

| Model | No. Parameters | Disk Size | Relative Latency |
|---|---|---|---|
| Seq2Seq | 4.5 M | 17.9 Mb | 1 x |
| LaserTagger | 110 M | 430 Mb | 25.1 x |
| Felix | 220 M | 933 Mb | 22 x |
| T5-Small | 60.5 M | 242 Mb | 11.3 x |
| 5IDER | 4.2M | 16.9 Mb | 0.4 x |

Table 3: *Exact match accuracy (%) on different use cases and their average under two training setups.*

| Model | Intent | Entity | Repair | Disfluency | Steering | Avg. |
|---|---|---|---|---|---|---|
| | | Singe-task training | | | | |
| Seq2Seq | 94.9 | 84.4 | 78.0 | 73.1 | 98.2 | 85.7 |
| LaserTagger | 84.7 | 49.3 | 64.6 | 63.0 | 99.9 | 72.3 |
| Felix | 96.0 | 83.7 | 85.6 | 77.4 | 97.1 | 88.0 |
| T5-Small | 96.3 | 90.6 | 83.5 | 82.0 | 95.1 | 89.5 |
| 5IDER (ours) | 95.5 | 86.5 | 80.2 | 79.2 | 100.0 | 88.3 |
| | | Multi-task training | | | | |
| Seq2Seq | 95.4 | 84.5 | 78.9 | 75.6 | 97.1 | 85.3 |
| LaserTagger | 84.5 | 49.1 | 62.9 | 60.0 | 99.8 | 71.3 |
| Felix | 88.1 | 1.7 | 80.9 | 25.0 | 44.1 | 48.0 |
| T5-Small | 95.7 | 89.3 | 83.3 | 83.6 | 93.1 | 89.0 |
| 5IDER (ours) | 95.3 | 86.1 | 81.0 | 80.6 | 98.9 | 88.4 |

## 4. Experimental Setup

**Hyperparameters** For 5IDER, we use a 192-dim frozen TinyBERT embedding [25] and 128-dim for all LSTM hidden states, biaffine attention RR heads, classifiers in RD, and deletion heads. During training, the model is trained with batch size 64, learning rate 6e-4 with Adam and dropout of 0.2.

**Baseline Systems** We compare 5IDER with baseline models to show the efficacy of the designed editing mechanism in our model. The first baseline is a BiLSTM-based seq2seq model[1] with a copy mechanism [27]. This model is close to our system in terms of model capacity: with the same set of hyperparameters, its model size is approximately twice that of 5IDER due to the additional decoder. We also compare against generative (T5-Small [24]) and edit-based (LaserTagger [21] and FELIX [23]) transformer models to see if the rewriting performance can be boosted at the cost of run-time speed. The input to the two generation baselines are the same as 5IDER. All baselines are fine-tuned and optimized across all use cases.

## 5. Results

### 5.1. Inference Time

We compare the models in terms of model size, disk size and latency in Table 2. As shown, 5IDER has the best latency and disk size out of the five, occupying less than 20Mb of disk space and being over 25x faster than T5-Small, LaserTagger and Felix.

### 5.2. Single-Task and Multi-Task Experiments

We test the models' capacity by training and testing on the same use case (single-task training), as shown in the first part of Table 3. In general, all models perform similarly well across tasks, except for LaserTagger, which struggles on some use cases.[2]

[1]We also experimented with similar sized Transformer variants, and found that even with 3 times more parameters, Transformers still perform worse than an LSTM.

[2]LaserTagger's edit operations include swapping sentence order, deletion and insertion with a limited vocabulary. It thus cannot consistently support use cases like entity carryover, which require a text span from the context query to be copied to the middle of the follow-up.

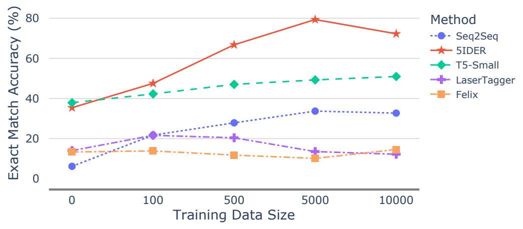

Figure 2: *Performance of various approaches on use-case composition. The x-axis shows training data size (in number of data points), while the y-axis shows Exact Match Accuracy (%).*

The general trend of good performance indicates that single-task training is straightforward for these models.

The more challenging but practical setup is the multi-task setting, where the same model needs to handle various linguistic patterns. Results are shown in the second part of Table 3. Although the task difficulty increases, 5IDER is still capable of tacking all 5 use cases, performing competitively with the strong T5 baseline (with only a 0.6% gap on average) in spite of its 25x latency and disk size benefit. However, the other two edit-based models perform much worse than 5IDER, despite using a BERT base encoder. Among them, Felix is unable to learn the different rewriting patterns in joint training (with an accuracy of 1.7% on the Entity use case). This indicates that our edit-based model is not only light-weight and low-latency, but also good at generalization in a multitask setting.

### 5.3. Few-shot Use Case Composition

We also experimented with challenging compositional use cases; our experimental results are shown in Figure 2. All models are trained with multitask single use case data, with a mixture of compositional data with varying dataset sizes. The x-axis shows the amount of compositional data used, while the y-axis shows the corresponding exact match accuracy on the test set. Note that the set of templates used to generate the train and test sets are disjoint, which ensures that the task cannot be solved through simple memorization.

Due to the aforementioned limitations, we find that LaserTagger and Felix perform relatively poorly irrespective of compositional data size. Interestingly, we find that 5IDER consistently outperforms the Seq2Seq model: in particular, 5IDER with no compositional training data outperforms the best Seq2Seq model trained with 5k data points. This indicates that 5IDER generalizes substantially better than the baseline systems on compositional use cases, without additional training data.

In addition, 5IDER's zero-shot use case composition performance is very close to that of the much larger T5 model. With 100 training data points, 5IDER achieves performance competitive to that of the best T5 model, and with just 500 training data points, 5IDER significantly outperforms the T5 model trained with 10k data points, using 1/20th the data, showcasing its great data efficiency on compositional use cases.

## 6. Conclusion

This work proposed a generalizable, multitasking, non-autoregressive query rewriting framework that handles 5 conversational use cases and their combinations. This model showed competitive performance in each use case, and significantly outperformed a fine-tuned T5-Small model in use case composition, while being 15 times smaller and 25 times faster.

## 7. References

[1] J. Kiseleva, K. Williams, J. Jiang, A. Hassan Awadallah, A. C. Crook, I. Zitouni, and T. Anastasakos, "Understanding user satisfaction with intelligent assistants," in Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval, 2016, pp. 121–130.

[2] H. Su, X. Shen, R. Zhang, F. Sun, P. Hu, C. Niu, and J. Zhou, "Improving multi-turn dialogue modelling with utterance ReWriter," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 22–31. [Online]. Available: https://aclanthology.org/P19-1003

[3] M. H. Maqbool, L. Xu, A. Siddique, N. Montazeri, V. Hristidis, and H. Foroosh, "Zero-label anaphora resolution for off-script user queries in goal-oriented dialog systems," in 2022 IEEE 16th International Conference on Semantic Computing (ICSC). IEEE, 2022, pp. 217–224.

[4] W. Yang, R. Qiao, H. Qin, A. Sun, L. Tan, K. Xiong, and M. Li, "End-to-end neural context reconstruction in Chinese dialogue," in Proceedings of the First Workshop on NLP for Conversational AI. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 68–76. [Online]. Available: https://aclanthology.org/W19-4108

[5] S. Vakulenko, S. Longpre, Z. Tu, and R. Anantha, "Question rewriting for conversational question answering," in Proceedings of the 14th ACM international conference on web search and data mining, 2021, pp. 355–363.

[6] P. Rastogi, A. Alexa, A. Gupta, and T. Chen, "Scaling multi-domain dialogue state tracking via query reformulation," NAACL HLT 2019, pp. 97–105, 2019.

[7] A. Chen, V. Zayats, D. Walker, and D. Padfield, "Teaching bert to wait: Balancing accuracy and latency for streaming disfluency detection," in NAACL HLT 2022, 2022, pp. 827–838.

[8] V. Ng and C. Cardie, "Improving machine learning approaches to coreference resolution," in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 104–111.

[9] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 188–197.

[10] J. G. Carbonell, "Discourse pragmatics and ellipsis resolution in task-oriented natural language interfaces," in 21st Annual Meeting of the Association for Computational Linguistics. Cambridge, Massachusetts, USA: Association for Computational Linguistics, Jun. 1983, pp. 164–168. [Online]. Available: https://aclanthology.org/P83-1025

[11] C. Naik, A. Gupta, H. Ge, M. Lambert, and R. Sarikaya, "Contextual slot carryover for disparate schemas," in Proc. Interspeech 2018, 2018, pp. 596–600. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1035

[12] T. Chen, C. Naik, H. He, P. Rastogi, and L. Mathias, "Improving long distance slot carryover in spoken dialogue systems," in Proceedings of the First Workshop on NLP for Conversational AI. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 96–105. [Online]. Available: https://aclanthology.org/W19-4111

[13] V. Zayats, M. Ostendorf, and H. Hajishirzi, "Disfluency Detection Using a Bidirectional LSTM," in Proc. Interspeech 2016, 2016, pp. 2523–2527.

[14] H. L. Nguyen, V. Renkens, J. Pelemans, S. P. Potharaju, A. K. Nalamalapu, and M. Akbacak, "User-initiated repetition-based recovery in multi-utterance dialogue systems," arXiv preprint arXiv:2108.01208, 2021.

[15] J. Quan, D. Xiong, B. Webber, and C. Hu, "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), 2019, pp. 4547–4557.

[16] S. Yu, J. Liu, J. Yang, C. Xiong, P. Bennett, J. Gao, and Z. Liu, "Few-shot generative conversational query rewriting," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1933–1936. [Online]. Available: https://doi.org/10.1145/3397271.3401323

[17] B.-H. Tseng, S. Bhargava, J. Lu, J. R. A. Moniz, D. Piraviperumal, L. Li, and H. Yu, "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, 2021, pp. 3390–3406.

[18] A. Gupta, J. Xu, S. Upadhyay, D. Yang, and M. Faruqui, "Disfl-QA: A benchmark dataset for understanding disfluencies in question answering," in Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Online: Association for Computational Linguistics, Aug. 2021, pp. 3309–3319. [Online]. Available: https://aclanthology.org/2021.findings-acl.293

[19] J. Mallinson, J. Adamek, E. Malmi, and A. Severyn, "EdiT5: Semi-autoregressive text editing with t5 warm-start," in Findings of the Association for Computational Linguistics: EMNLP 2022. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2126–2138. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.156

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., "Language models are unsupervised multitask learners," OpenAI blog, vol. 1, no. 8, p. 9, 2019.

[21] E. Malmi, S. Krause, S. Rothe, D. Mirylenka, and A. Severyn, "Encode, tag, realize: High-precision text editing," 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), 2019, pp. 5054–5065.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[23] J. Mallinson, A. Severyn, E. Malmi, and G. Garrido, "FELIX: Flexible text editing through tagging and insertion," in Findings of the Association for Computational Linguistics: EMNLP 2020. Online: Association for Computational Linguistics, Nov. 2020, pp. 1244–1255. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.111

[24] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu et al., "Exploring the limits of transfer learning with a unified text-to-text transformer." J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.

[25] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in Findings of the Association for Computational Linguistics: EMNLP 2020. Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.372

[26] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," International Conference on Learning Representations, 2017.

[27] J. Gu, Z. Lu, H. Li, and V. O. Li, "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), 2016, pp. 1631–1640.