Conversation Disentanglement As-a-Service

Edoardo Riggio, Marco Raglianti, Michele Lanza
REVEAL @ Software Institute – Università della Svizzera italiana, Lugano, Switzerland

***Abstract*—Modern instant messaging applications (*e.g.,* Gitter, Slack, Discord) provide users with real-time communication means. Developers use them for collaborative development, to ask for code reviews, and to have software-related discussions. In short, a (potential) treasure trove for program comprehension. However, as with any high-throughput "chat application", messages interleave, leading to concurrent conversations. Associating messages to conversations is called *conversation disentanglement*, a useful and necessary pre-processing step to analyze datasets of instant messages. Although various conversation disentanglement algorithms have been proposed, it is cumbersome to set up proper execution environments and hard to ensure input data format consistency, calling for better practices and tool support.**

We present CODI, a RESTful API micro-service and web interface for conversation disentanglement. It provides an easy way to disentangle conversation transcripts with pre-trained models or to train new ones on custom datasets, features, and hyper-parameters. CODI achieves state-of-the-art performances on transcripts of IRC, Slack, and Discord conversations. We show how CODI can provide a significant improvement to reusability (and replicability) of research results, while reducing the efforts and potential mistakes due to configuration, setup, and execution.

CODI's source code: https://github.com/USIREVEAL/CODI

***Index Terms*—CoDi, conversation disentanglement, instant messaging, micro-services**

I. Introduction

Instant Messaging (IM) applications, such as WhatsApp, Slack, and Discord, are ubiquitous, supplanting asynchronous communication means (*e.g.,* emails) for professional and personal use. Developers are no exception: Software communities are born, evolve, thrive, and sometimes die on such virtual communication hubs. This paradigm shift also induces concerns about information persistence and accessibility. IM platforms from private companies (*e.g.,* Slack, Discord) usually provide an API to access information on public servers. However, this access is often restricted (*e.g.,* Slack's free tier plan provides access only to messages of the last 90 days) without guarantees on future availability of public content.

When a community has a wide scope or simply grows in popularity, scalability problems arise. Many public servers feature channels where questions and answers about a specific topic can be sent. High-traffic channels experience very soon an interleaving of messages pertaining to different conversations (interleaving colors, Figure 1 left), with multiple messages being sent almost simultaneously. A human reader is moderately capable of reconstructing in real-time the conversation flow. Aids like replies indication, temporal intervals, cue words and explicit Q&A structures can be used for disambiguation. However, *reliable automatic reconstruction of conversations remains an open challenge.*

Elsner and Charniak proposed a disentanglement algorithm for Internet Relay Chats (IRC) [1]. While IRC provides only a textual representation for messages, modern platforms support features (*e.g.,* multimedia sharing, explicit replies) that can be leveraged for disentanglement. Their algorithm was adapted by Chatterjee *et al.* [2] to disentangle developer conversations in Slack, while Subash *et al.* [3] used it for Discord.

No approach for conversation disentanglement is available out-of-the-box, reducing its usefulness. Having conversations as higher-order constructs provides richer semantics than simple sequences of messages, and is key to improve the quality of inputs to software engineering research on developers' chats.

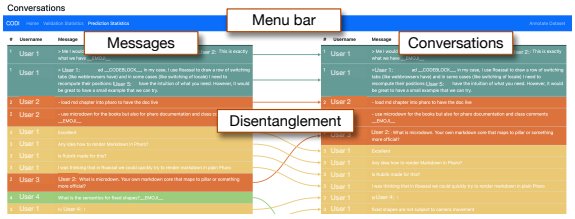

Fig. 1: Conversation Disentanglement Interface in CODI.

We present CODI, an extensible service-based API and web interface (Figure 1) for disentangling developer chats. It can be integrated in preprocessing pipelines to clean up collected data. CODI's input/output formats improve interchangeability of disentanglement algorithms. This is a step towards easier comparison of alternative approaches. The CODI web interface helps in exploratory phases with fast iteration cycles. The proposed visualization of disentangled conversations provides qualitative (*e.g.,* message grouping) and quantitative information (*e.g.,* accuracy, F-score, computation time) about the disentanglement process with the selected model. The tool and the approach we propose reduce inconsistencies in configuration, simplify the setup, and improve reliability of results.

II. CODI

CODI is implemented in *python* and it is composed of three main modules (Figure 2): The conversation *disentangler*, the *REST*ful *API*, and the web-based frontend *client*.

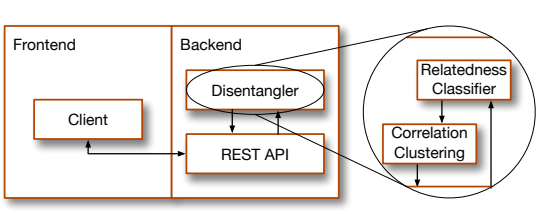

Fig. 2: CODI Architecture Overview

 DOI: 10.1109/ICPC58990.2023.00018

The disentangler module is a reimplementation of the model proposed by Elsner and Charniak [1] and later modified by Chatterjee *et al.* [2]. Message pairs are evaluated with respect to their relatedness, according to features like time interval and term similarity, and then clustered in conversations based on their relatedness scores. We extend some of the features used by the latest version of the model (*e.g.,* by adding variations of possible greetings, with different types of mention features) and allow to use two different *relatedness classifiers*: Random forest [2] and logistic regression [1].

The RESTful API provides an interface to access the disentangler by defining endpoints for training, validation, and prediction (Table I). Two endpoints provide utilities to convert input files between different formats and retrieve performance statistics of the model computations (*e.g.,* accuracy, F-score).

TABLE I: REST API Endpoints

Path	Type	Description
api/train	POST	Sends an input dataset to the disentangler to train the classifier
api/validate	POST	Sends an input annotated dataset to the disentangler. It will predict the conversations and compare them to the provided annotations
api/predict	POST	Sends an input dataset to the disentangler to predict the conversations
api/statistics	GET	Retrieves the disentangled conversations and performance statistics (after *validate* or *predict*)
api/convert	POST	Converts a dataset from *ANNOT* to *JSON* format

A. Towards Research Code as Infrastructure

CoDi is a step towards *Research Code as Infrastructure*. This approach aims to improve reliability of results, reusability of implementations, and comparability of approaches in research prototypes for conversation disentanglement.

Trying to replicate previous studies [1]–[4], we set up an environment with different python versions, complying with needs of older and newer scripts; we had to include a precompiled version of the MEGAM[1] max entropy classifier– a dependency whose latest version dates back to October 2007. These details took significantly more effort than a file drag-and-drop in a web page and browsing results with performance information of the algorithm. We also question the reliability of the obtained output with respect to inconsistencies in python interpreter versions and outdated libraries.

Inspecting the intermediate format used as input for the disentanglement algorithm, we identified an inconsistency in the interaction with the pseudonymization script.

The version used to disentangle Discord messages by Subash *et al.* [3] is the same used by Chatterjee *et al.* for Slack [5]. To partially comply with user anonymization needs, user names of message authors in the ANNOT format have been randomly substituted with common first names. In this pseudonymization process, references to the authors in the text of messages have not been translated accordingly. This minor detail impacts the reply feature of the relatedness classifier, as reply links are completely lost. An ablation study could confirm this in the original algorithm but such a study is beyond the scope of this work.

[1]See **`https://tinyurl.com/mr3537ae`** [acc. June 1, 2023]

B. Input JSON format

The JavaScript Object Notation (JSON) input format of CoDi (Figure 3) is richer than the *ANNOT* representation used as exchange format in the reference model [1], [2]. We provide an endpoint to convert between the two for compatibility and to cross-validate results from previous studies. The representation we propose better suits the features of modern IM platforms (*e.g.,* replies, quotes, complex mentions, attachments), and allows to more easily verify input consistency.

```
"platform": "Discord",
"id": "b4138f14-af37",
"name": "Agile Everything",
"members": [ {
    "id": "d1ff9c5b-f1fc",
    "name": "Jermaine Fontaine"
  } , ...
],
"channels": [ {
  "id": "c65d238f-d987",
  "name": "visualization",
  "path": "agile/visualization",
  "topics": [ {
      "keywords": [ "Visualization" , ... ],
      "description": "Agile Visualization is cool!"
    }
  ],
  "messages": [ {
      "id": "34ce13f1-6577",
      "authorId": "d1ff9c5b-f1fc",
      "content": "Jermaine: $(date) is in large format",
      "conversation": "T35",
      "timestamp": "2022-02-06T19:24:23.777+00:00",
      "mentions" : [ { "authorId" : "d1ff9c5b-f1fc" } ],
      "repliesTo" : [ { "messageId" : "c1dr4c2r-z7at" } ],
      "attachments" :
        [ { "url" : "https://cdn.discordapp.com/..." } ]
    }, ...
```

Fig. 3: JSON Input Format Example

C. Web User Interface

CoDi provides a web-based user interface (Frontend – Client in Figure 2). In the *Home* page (Figure 4) it is possible to select the operation to perform ① (*e.g.,* train, validate, predict), the type of platform ② (*e.g.,* IRC, Slack, Discord), parameters of the model ③ (*i.e.,* features to use in the classifier), and to drop a file for elaboration ④.

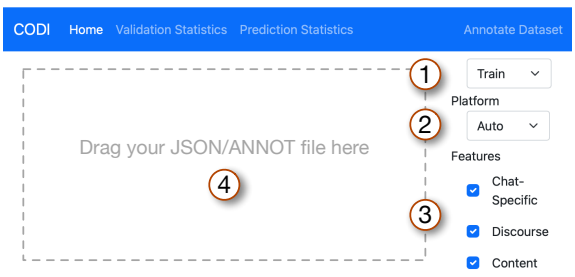

Fig. 4: CoDi Home Page

In the *Validation Statistics* page it is possible to compare the output of a disentanglement with a provided ground truth (Figure 5). Color coding helps in identifying conversation blocks. For example, the message from Khylon is misclassified and assigned to a new conversation (left) instead of being the last message of conversation 1 (ground truth, right).

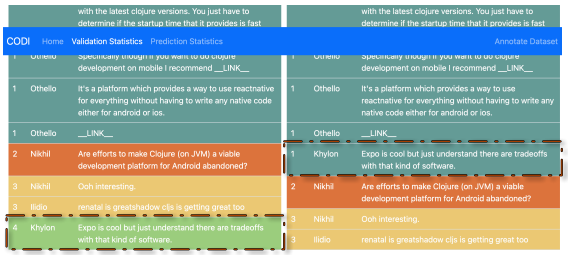

Fig. 5: CODI Validation Page

In the *Prediction Statistics* page (already shown in Figure 1) it is possible to see the output of the disentanglement. Messages in chronological order (left) are mapped, with arrows, to their counterpart in disentangled conversation blocks (right).

Prediction and validation statistics pages have an expandable statistics section at the top (Figure 6). It provides detailed information on performance and other metrics for the output conversations as well as for internal components (*e.g.,* classifier accuracy, computation time for each step).

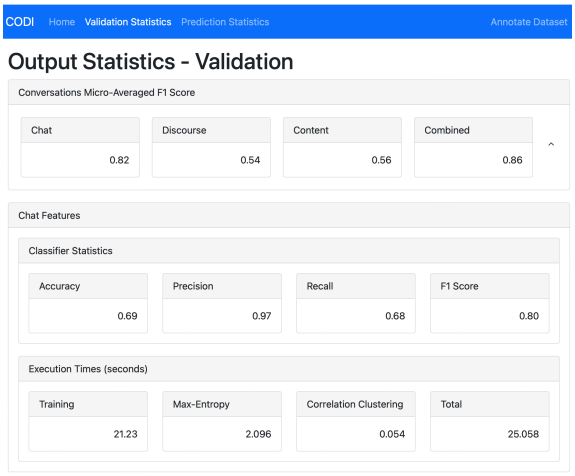

Fig. 6: CODI Statistics Expanded

III. Evaluation

To assess correctness of our implementation, we compared disentanglement annotations from CODI (logistic regression relatedness classifier, all features) with the reference model [2]. We trained both models with the same ground truth: 3,544 manually disentangled messages from Chatterjee *et al.* [2]. We compared their performances on two previously published datasets [3] and a new one we extracted and partially manually annotated (see Section III-A and Table II). We transformed datasets from the ANNOT format into the CODI JSON format and vice-versa, as needed to allow full cross comparisons.

A. Datasets

Literature datasets: We used two subsets of messages from two Discord datasets published by Subash *et al.* [3]. The smaller one is the *clojure* Discord community dump (Feb – Apr 2020: 464 messages), the larger one is the *python* Discord community dump (Mar 2020: 56,763 messages).

Roassal dataset: We extracted the full history of the *roassal*[2] channel in the *Pharo*[3] Discord server (Apr 2017 – May 2022: 8,093 messages). To perform an independent testing on a new ground truth, two authors manually annotated four days of conversations in this dataset (*roassal_m*, 294 messages). The third author was involved in the discussion on conflicting assignments until consensus for all the messages was reached.

B. Result Analysis

In Table II, we show the results of comparing CODI's disentanglement with automatically disentangled conversations obtained from the reference algorithm (top-3 rows). We also report CODI's testing performance on the new dataset we manually annotated as ground truth (*roassal_m* row).

We report accuracy, precision, recall, and F1-score of the relatedness classifier and the micro-averaged F1-score (μAvg-F1) [6] of the final clustering of disentangled conversations. We used the μAvg-F1 measure to evaluate correspondence between conversation clusters for comparability with previous results [1], [2], [4]. Moreover, a multi-way average over conversation clusters is suitable to compare automatic disentanglement to a ground truth and correlates with the one-to-one accuracy metric [1]. For python, due to the large number of messages in the dataset, we computed μAvg-F1 on a small random sample of 654 consecutive messages.

TABLE II: CODI vs Reference Comparison and New Test

Dataset Comparison	Relatedness Accuracy	Relatedness Precision	Relatedness Recall	Relatedness F1	Clustering μAvg-F1
clojure	67	94	68	79	**88**
python	60	86	62	72	**78**
roassal	77	95	78	86	**80**
New Test					
roassal_m	68	68	77	79	**63**

Correctness: High values of the μAvg-F1 score indicate disentangled conversations similar to those in the reference model. Our implementation suffers from lower recall of message relatedness. This indicates an overestimation of related utterances in the classifier. Lower μAvg-F1 score for the python and roassal datasets indicates that although conversations are similar, there are still significant inconsistencies.

Problematic Cases: By manual inspection we found two main cases: *single point* and *split/merge* inconsistencies. In single point inconsistencies (Figure 7) one message is assigned to the wrong conversation. It can be the previous one, the next one, or a new conversation. Split/merge inconsistencies (Figure 8) happen when a conversation is split into two or two conversations are merged into one.

[2]See **tinyurl.com/roassal** [acc. June 1, 2023]
[3]See **https://pharo.org** [acc. June 1, 2023]

Fig. 7: Example of Single Point Inconsistency

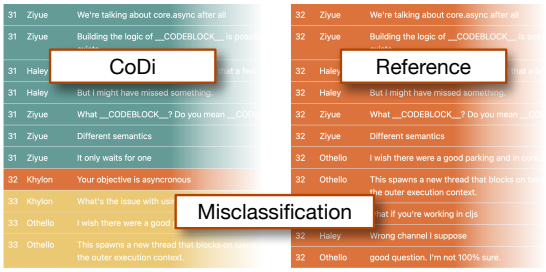

Fig. 8: Example of Split/Merge Inconsistency (Split)

Single point inconsistencies preserve most of the features of extracted conversations. Split/merge inconsistencies have a higher impact on μAvg-F1 but are often limited to a single conversation mismatch. Manual inspection confirmed that there is an abundance of split inconsistencies with the python and roassal datasets. This is the main cause of the lower μAvg-F1 score with respect to clojure.

Generalizability: Testing with the roassal_m dataset shows lower performance in terms of μAvg-F1 score (reference implementation μAvg-F1 $= 0.61$) with respect to the results reported by Subash *et al.* [3]. This might be caused by messages in the new dataset having different characteristics (*e.g.,* lower time dependence, different usage of mentions) and can potentially hint at low generalizability of the approach. Our study highlights the need to increase the amount of manually annotated disentanglement datasets to replicate studies.

C. Conversion and Formats

Chatterjee *et al.* used an Extensible Markup Language (XML) representation for the input [2]. The XML was converted to ANNOT format for compatibility with the original implementation. Due to the conversion, the XML format could not be exploited by the algorithm, thus remaining unexplored in terms of potential disentanglement improvements.

Format conversion also comes with limitations to cross-compatibility of input. For example, mentions in simple textual format are expensive for computing mention-related features. Providing them explicitly with a richer message representation, as already supported by CoDi's JSON input format, can improve performance and reliability of the disentanglement. In fact, the ANNOT format showed to be error prone in pseudonymized datasets due to inconsistent pre-processing steps (*i.e.,* name substitutions impairing mention traceability).

IV. Related Work

Most recent works address IM applications popular also among developer communities like Gitter [7]–[11], Slack [5], [11]–[13], and Discord [3], [14]–[16]. Few works properly account for the disentanglement problem in their studies.

Conversation disentanglement in online chats has been addressed both with unsupervised [6], [17], [18] and supervised [1], [2], [19], [20] approaches. According to Liu *et al.* [18], an orthogonal distinction in disentanglement techniques is between two-step approaches and end-to-end ones. The former [1], [2], [21] combine local relatedness (*e.g.,* message pair relatedness classifier) and conversation clustering, while the latter [22]–[24] try to capture global properties of conversations in one step.

Only a few published datasets have a manually annotated ground truth [1], [21], [25] and only one of them is considered large [25]. Due to this lack, three large datasets [2], [3], [7] of automatically disentangled developer conversations have also been proposed for Software Engineering research.

V. Future Work

We evaluated CoDi by comparing it with the reference model and on a dataset we manually annotated. We plan to extend the evaluation to the largest manually annotated dataset by Kummerfeld *et al.* [25] and explore improvements to platform-specific features (*e.g.,* quotes, replies). We are deploying CoDi for researchers to integrate it in their pipelines, which calls for an evaluation of the the user interface.

Configurability options, performance optimization, and robustness to malformed inputs need to be improved upon, with respect to the reference model and local approaches in general, especially on large datasets. CoDi can be deployed in pre-trained mode for multiple users. We will implement API-key access to support concurrent usage in segregated instances, allowing training custom models also on the publicly deployed service. Finally, we need to extend the model to another approach (*i.e.,* end-to-end) to demonstrate generalizability of our design choices for the JSON input format (Section II-B) and extensibility of the architecture.

VI. Conclusion

Conversation disentanglement is a key pre-processing step to improve knowledge extraction from instant messaging platforms. We need effective tools to stop such fast and volatile sources from being only an on-demand support for program comprehension, unleashing their full potential.

We presented CoDi, an extensible object-oriented micro-service architecture for conversation disentanglement, reporting also on a comparative evaluation. Our work aims at improving the reliability of conversation disentanglement results while reducing the technical barriers to reuse state-of-the-art models: CoDi is a first step towards making disentanglement models usable and accessible.

Acknowledgements. This work is supported by the Swiss National Science Foundation (SNSF) through the project "INSTINCT" (SNF Project No. 190113).

REFERENCES

[1] M. Elsner and E. Charniak, "Disentangling chat," *Computational Linguistics*, vol. 36, no. 3, pp. 389–409, 2010.

[2] P. Chatterjee, K. Damevski, N. A. Kraft, and L. Pollock, "Software-related Slack chats with disentangled conversations," in *Proceedings of MSR 2020 (International Conference on Mining Software Repositories)*. ACM, 2020, pp. 588–592.

[3] K. M. Subash, L. P. Kumar, S. L. Vadlamani, P. Chatterjee, and O. Baysal, "DISCO: A dataset of Discord chat conversations for software engineering research," in *Proceedings of MSR 2022 (International Conference on Mining Software Repositories)*. IEEE/ACM, 2022, pp. 227–231.

[4] M. Elsner and E. Charniak, "You talking to me? A corpus and algorithm for conversation disentanglement," in *Proceedings of ACL-HLT 2008 (Association for Computational Linguistics: Human Language Technologies)*. ACL, 2008, pp. 834–842.

[5] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N. A. Kraft, "Exploratory study of Slack Q&A chats as a mining source for software engineering tools," in *Proceedings of MSR 2019 (International Conference on Mining Software Repositories)*. IEEE/ACM, 2019, pp. 490–501.

[6] D. Shen, Q. Yang, J.-T. Sun, and Z. Chen, "Thread detection in dynamic text message streams," in *Proceedings of SIGIR 2006 (International Conference on Research and Development in Information Retrieval)*. ACM, 2006, pp. 35–42.

[7] E. Parra, A. Ellis, and S. Haiduc, "GitterCom: A dataset of Open Source developer communications in Gitter," in *Proceedings of MSR 2020 (International Conference on Mining Software Repositories)*. ACM, 2020, pp. 563–567.

[8] O. Ehsan, S. Hassan, M. E. Mezouar, and Y. Zou, "An empirical study of developer discussions in the Gitter platform," *Transactions on Software Engineering and Methodology*, vol. 30, no. 1, pp. 1–39, 2020.

[9] L. Shi, X. Chen, Y. Yang, H. Jiang, Z. Jiang, N. Niu, and Q. Wang, "A first look at developers' live chat on Gitter," in *Proceedings of ESEC/FSE 2021 (European Software Engineering Conference and Symposium on the Foundations of Software Engineering)*. ACM, 2021, pp. 391–403.

[10] H. Sahar, A. Hindle, and C.-P. Bezemer, "How are issue reports discussed in Gitter chat rooms?" *Journal of Systems and Software*, vol. 172, p. 110852, 2021.

[11] E. Parra, M. Alahmadi, A. Ellis, and S. Haiduc, "A comparative study and analysis of developer communications on Slack and Gitter," *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–33, 2022.

[12] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why developers are slacking off: Understanding how software teams use Slack," in *Proceedings of CSCW/SCC 2016 (Conference on Computer Supported Cooperative Work and Social Computing Companion)*. ACM, 2016, pp. 333–336.

[13] V. Stray and N. B. Moe, "Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack," *Journal of Systems and Software*, vol. 170, p. 110717, 2020.

[14] M. Raglianti, R. Minelli, C. Nagy, and M. Lanza, "Visualizing Discord servers," in *Proceedings of VISSOFT 2021 (Working Conference on Software Visualization)*. IEEE, 2021, pp. 150–154.

[15] M. Raglianti, C. Nagy, R. Minelli, and M. Lanza, "Using Discord conversations as program comprehension aid," in *Proceedings of ICPC 2022 (International Conference on Program Comprehension)*. ACM, 2022, pp. 597–601.

[16] ——, "DiscOrDance: Visualizing software developers communities on Discord," in *Proceedings of ICSME 2022 (International Conference on Software Maintenance and Evolution)*. IEEE, 2022, pp. 474–478.

[17] P. H. Adams and C. H. Martell, "Topic detection and extraction in chat," in *Proceedings of ICSC 2008 (International Conference on Semantic Computing)*. IEEE, 2008, pp. 581–588.

[18] H. Liu, Z. Shi, and X. Zhu, "Unsupervised conversation disentanglement through co-training," in *Proceedings of EMNLP 2021 (Conference on Empirical Methods in Natural Language Processing)*. ACL, 2021, pp. 2345–2356.

[19] T. Li, J.-C. Gu, X. Zhu, Q. Liu, Z.-H. Ling, Z. Su, and S. Wei, "DialBERT: A hierarchical pre-trained model for conversation disentanglement," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2004.03760

[20] R. Zhu, J. H. Lau, and J. Qi, "Findings on conversation disentanglement," *arXiv preprint*, 2021. [Online]. Available: https://arxiv.org/abs/2112.05346

[21] S. Mehri and G. Carenini, "Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks," in *Proceedings of JCNLP 2017 (International Joint Conference on Natural Language Processing)*. ACL, 2017, pp. 615–623.

[22] J.-Y. Jiang, F. Chen, Y.-Y. Chen, and W. Wang, "Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking," in *Proceedings of ACL-HLT 2018 (Association for Computational Linguistics: Human Language Technologies)*, 2018, pp. 1812–1822.

[23] H. Zhu, F. Nan, Z. Wang, R. Nallapati, and B. Xiang, "Who did they respond to? Conversation structure modeling using masked hierarchical transformer," in *Proceedings of AAAI 2020 (Conference on Artificial Intelligence)*, vol. 34, no. 05, 2020, pp. 9741–9748.

[24] H. Liu, Z. Shi, J.-C. Gu, Q. Liu, S. Wei, and X. Zhu, "End-to-end transition-based online dialogue disentanglement," in *Proceedings of IJCAI (International Joint Conference on Artificial Intelligence)*, vol. 20, 2020, pp. 3868–3874.

[25] J. K. Kummerfeld, S. R. Gouravajhala, J. J. Peper, V. Athreya, C. Gunasekara, J. Ganhotra, S. S. Patel, L. C. Polymenakos, and W. Lasecki, "A large-scale corpus for conversation disentanglement," in *Proceedings of ACL 2019 (Annual Meeting of the Association for Computational Linguistics)*. ACL, 2019, pp. 3846–3856.