IMPROVING LANGUAGE MODELS THROUGH CONTEXT

by

Dong-Ho Lee

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2025

Dedication

To my beloved wife, Soyoung,

who stood by me through every challenge and joy of this journey.

And to my parents, Jeonghwan and Heejin,

for their endless love and support.

Acknowledgements

First and foremost, I would like to thank my advisor, *Jay Pujara*, for his dedication and support in guiding me to become an independent researcher. In 2020, while working on the MACROSCORE project as a student researcher at USC, I was introduced to Jay. I sent him monthly updates and always received thoughtful, detailed feedback. I remember thinking, *this guy gives really sharp feedback.* Still, I didn't expect he would one day become my PhD advisor. My PhD journey with Jay has been both challenging and rewarding. His feedback remained as sharp as ever, constantly pushing me to think deeper and aim higher. His passion for research and endless curiosity helped me build a strong research foundation, one that allowed me to overcome challenges and pursue meaningful discoveries. Defending my thoughts against his insightful critiques motivated me to refine my arguments, sharpen my thinking, and ultimately produced research that I'm proud of. I truly believe that having the opportunity to begin and complete this journey under his advice was a gift I will always be grateful for. I hope to carry forward the same generosity, clarity, and vision that he showed me.

I would also like to sincerely thank all the committee members who supported me throughout my defense, proposal, and qualifying exam, as well as the USC faculty: Xiang Ren, Robin Jia, Fred Morstatter, Meisam Razaviyayn, Andreas Molisch, Morteza Dehghani, and Jonathan May. Their thoughtful feedback and constructive discussions were invaluable in shaping and refining my research. A special thanks to Xiang, my first advisor in academia. When I began my master's at USC, I had no prior research experience and couldn't imagine myself pursuing a research career. Being introduced to research under the guidance of such a brilliant advisor was a huge luck, and it made the extremely strong foundation for my entire research journey. I'm also deeply grateful to Robin. I still remember our first chat during my PhD interview. Even at that early stage, the experience encouraged me to reflect on how I needed to grow as a researcher. Since then, Robin has consistently provided thoughtful and insightful feedback, whether in class, during exams, or throughout this process. I truly appreciate the guidance and support.

Beyond the professors at USC, I would also like to thank the incredible mentors who have guided me along the way. Thank you to *Bill Yuchen Lin*, my first academic mentor, who taught me how to write papers, design experiments, and navigate the world of research. I miss the time we spent working together in the lab. Thank you to *Chungha Sung*, who has always been by my side, giving me advice on everything from research to personal life. Your support has meant so much to me, both professionally and personally. Thank you to *Woojeong Jin*, who consistently provided insightful feedback and valuable guidance. It was a huge fortunate to have a chance to work with these mentors.

I am also deeply grateful to my industry mentors: *Sujay Jauhar* and *Ryen White* at Microsoft Research, *Francesco Barbieri* at Snap, and *Adam Kraft*, *Long Jin*, and *Xinyang Yi* at Google. Thank you for your continued support, even long after the internships ended. Your guidance has meant a lot to me, and I truly appreciate your support until now.

I would also like to thank *Sungjoon Park* and *Jihyung Moon*, the founders of SoftlyAI. Thanks to you, I had the rare opportunity to gain invaluable startup experience during my PhD. Being deeply involved in every aspect of the company from prototyping to decision-making and team management was truly unforgettable. I learned so much from both of you, and your mentorship had one of the most significant impacts on my PhD journey.

Thanks to all my NLP friends (Hyundong Cho, Deuksin Kwon, Pei Zhou, Rahul Khanna, Kian Ahrabian, Pegah Jandaghi, Yifan Jiang, Eric Boxer, Kexuan Sun, Avijit Thawani, Adyasha Maharana, Mohit Bansal, Zhihan Zhang, Wenhao Yu, Yupeng Hou, Brihi Joshi, Ziyi Liu, Akshen Kadakia, Ravi Kiran Selvam, Sheikh

Muhammad Sarwar, Dongwon Jung, and more), collaborators (Thomas Choi, Juhyung Lee, Eunsoo Sheen, Joohan Lee, and more), my supportive friends in Korea (Seungho Kim, Jason Ryu, Taewoo Kim, Seungwon Kim, Sukmin Kang, Soohyun Kim, and more).

Finally, I would like to thank my beloved parents, Jeonghwan Lee and Heejin Choi, my parents-in-law, Sunho Lee and Youme Kim, and my wife, Soyoung Lee. Your endless support and unconditional love have meant everything to me.

Table of Contents

Dedication . . . ii

Acknowledgements . . . iii

List of Tables . . . ix

List of Figures . . . xi

Abstract . . . xiv

Chapter 1: Introduction . . . 1
1.1 Background & Motivation . . . 2
1.1.1 In-context Learning of Language Models (LMs) . . . 2
1.1.2 Towards Better Understanding of In-context Learning . . . 3
1.2 Hypothesis . . . 3
1.3 Contributions . . . 4
1.3.1 Task-Specific Effectiveness of In-Context Learning in Language Models . . . 4
1.3.2 Training Language Models with Context Enhances Model Behavior . . . 5
1.3.3 Self-Refinement of Context by Language Models Improves Output Quality . . . 6
1.4 Outline . . . 6

Chapter 2: Background and Related Works . . . 8
2.1 Background: The Evolution of Context in NLP . . . 8
2.1.1 Static Representations: Early Views of Context . . . 8
2.1.2 Dynamic Contextualization: Contextual Language Models . . . 9
2.1.3 Task-Oriented Context: Instructions, Examples, and Explanations . . . 10
2.1.4 Interactive and Situated Context . . . 11
2.1.5 Summary . . . 12
2.2 Related Works: Contextual Learning Strategies for Language Models . . . 12
2.2.1 In-context Learning in Language Models . . . 13
2.2.2 Contextual Supervision for Language Model Training . . . 14
2.2.3 Language Models as Self-Refining Context Generators . . . 15
2.3 Summary . . . 16

Chapter 3: Context-Aware Inference in Language Models . . . 18
3.1 Background and Motivation . . . 18
3.1.1 In-context Learning with Task Examples . . . 18
3.1.2 Contextual Extrapolation from Interaction History . . . 20

3.2 In-context Learning for Structural Extrapolation . . . 20
3.2.1 Introduction . . . 20
3.2.2 Temporal Knowledge Graph Forecasting: Problem Formulation . . . 22
3.2.3 In-context Learning for Temporal Knowledge Graph Forecasting . . . 23
3.2.4 Experimental Setup . . . 25
3.2.5 Experimental Results . . . 27
3.2.6 Ablation Study: Understanding Prompt Effects . . . 30
3.3 In-context Learning for Semantic Extrapolation . . . 32
3.3.1 Introduction . . . 32
3.3.2 User Simulation: Dataset (REALTALK) and Evaluation Metrics . . . 34
3.3.2.1 Dataset: REALTALK . . . 34
3.3.2.2 Evaluation Metrics: Emotional Intelligence . . . 35
3.3.2.3 Data Insights from REALTALK . . . 39
3.3.3 In-context Learning for User Simulation . . . 41
3.3.4 Experimental Setup . . . 42
3.3.5 Experimental Results . . . 43
3.3.6 Ablation Study: Towards Better User Simulation through Fine-tuning . . . 43
3.4 Summary . . . 44

Chapter 4: Contextual Supervision for Language Model Training . . . 46
4.1 Background and Motivation . . . 46
4.1.1 Explicitly Training Language Models with Context Improves Performance . . . 46
4.1.2 Human Explanations as Context for Learning . . . 47
4.2 Training Language Models with Context . . . 48
4.2.1 Training for Named Entity Recognition Using In-Context Task Examples . . . 49
4.2.1.1 Introduction . . . 49
4.2.1.2 Named Entity Recognition: Problem Formulation . . . 51
4.2.1.3 Demonstration-based NER . . . 53
4.2.1.4 Experimental Setup . . . 56
4.2.1.5 Experimental Results . . . 57
4.2.2 Training for Social Norm Violation Detection Using Dialogue History as Context . 63
4.2.2.1 Introduction . . . 63
4.2.2.2 Norm Violation Detection: Dataset (NormVio-RT) . . . 65
4.2.2.3 Norm Classification in NormVio-RT . . . 71
4.2.2.4 Experimental Setup . . . 72
4.2.2.5 Experimental Results . . . 73
4.3 Human Explanations as Context for Learning . . . 76
4.3.1 Label-efficient Learning with Human-provided Labeling Explanation . . . 76
4.3.1.1 Introduction . . . 76
4.3.1.2 Entity Triggers as Explanation for NER . . . 80
4.3.1.3 TriggerNER: Human-Guided Learning for NER . . . 81
4.3.1.4 LEAN-LIFE: Explanation-Centric Annotation for NER . . . 86
4.3.1.5 Experimental Setup . . . 87
4.3.1.6 Experimental Results . . . 89
4.3.2 Model Refinement via Explanation-guided Regularization . . . 92
4.3.2.1 Introduction . . . 92
4.3.2.2 XMD: Explanation-Based NLP Model Debugger . . . 95
4.3.2.3 Experimental Setup . . . 100

4.3.2.4 Experimental Results . . . 101
4.4 Summary . . . 103

Chapter 5: Language Models as Self-Refining Context Generators . . . 104
5.1 Background and Motivation . . . 104
5.1.1 Self-Refinement Through Simulated Interaction as Contextual Supervision . . . 104
5.2 Self-Refinement for Educational Question Generation . . . 105
5.2.1 Introduction . . . 105
5.2.2 Question Generation: Problem Formulation and Dataset (TEXTBOOK-EXAM) . . . 108
5.2.2.1 Problem Formulation . . . 108
5.2.2.2 Dataset: TEXTBOOK-EXAM . . . 108
5.2.3 QUEST: Question Utility Estimation and Simulation Tests . . . 111
5.2.4 Experimental Setup . . . 113
5.2.5 Experimental Results . . . 115
5.3 Summary . . . 120

Chapter 6: Conclusions and Future Works . . . 121
6.1 Conclusions . . . 121
6.1.1 Context-Aware Inference in Language Models . . . 121
6.1.2 Contextual Supervision for Language Model Training . . . 122
6.1.3 Language Models as Self-Refining Context Generators . . . 122
6.1.4 Mechanisms, Limits, and Future Directions. . . . 123
6.2 Future work . . . 123
6.2.1 Toward Contextual Personalization in Language Models . . . 123
6.2.2 Toward Socially Intelligent Language Models . . . 124
6.2.3 Looking Ahead . . . 126

Bibliography . . . 127

List of Tables

3.1 Motivation example for structural pattern extrapolation using in-context learning 22

3.2 Prompt example of in-context learning for temporal knowledge graph forecasting 24

3.3 Performance (Hits@K) comparison between supervised models and ICL for temporal knowledge graph forecasting 27

3.4 Performance (Hits@K) comparison between rule-based predictions and ICL for temporal knowledge graph forecasting 28

3.5 Performance (Hits@1) comparison between `index` and `lexical` in ICL for temporal knowledge graph forecasting 29

3.6 Data statistics comparison between REALTALK and others in ICL for persona simulation . 35

3.7 Performance comparison between with and without fine-tuning in ICL for user simulation 44

4.1 Data statistics of named entity recognition task 56

4.2 Performance (F1-score) comparison in in-domain setting for named entity recognition . . 58

4.3 Performance (F1-score) comparison in domain-adaptation setting for named entity recognition 59

4.4 Performance (F1-score) comparison between different backbone models 60

4.5 Performance (F1-score) comparison w/ and w/o demonstration at training and inference time. 63

4.6 Performance (F1-score) in fully supervised setting by different percentages of train data . 63

4.7 Norms in live streaming domain (Twitch) 67

4.8 Data statistics of norm violation detection task 68

4.9 Inter-annotator agreement of data construction for norm violation detection task 70

4.10 Performance (F1-score) on norm violation detection 73

4.11 Performance (F1-score) on distribution shift between norm violations in Reddit and Twitch. 76

4.12 Data statistics of entity triggers 89

4.13 Performance (F1) comparison between BLSTM-CRF and trigger matching network 90

4.14 Performance (Accuracy) comparison with instance-level explanation on ID/OOD performance 101

4.15 Performance (Accuracy) comparison with task-level explanation on ID/OOD performance 102

5.1 Data statistics of TEXTBOOK-EXAM 110

5.2 End-of-chapter exam score (Accuracy) comparison between different question generation approaches across various subjects. 115

5.3 Spearman correlation between utility and other metrics 116

5.4 End-of-chapter exam score (Accuracy), average saliency, and average expected information gain (EIG) of generated questions for different QUEST-optimized models trained on datasets filtered by different selection criteria. 117

5.5 End-of-chapter exam score (Accuracy) for different model sizes across various subjects and modules. 119

List of Figures

1.1 **Chat with ChatGPT.** Context determines the interpretation of language model outputs, as illustrated by differing sentiment predictions depending on whether prior dialogue context reveals sarcasm. 2

1.2 Experiment setting for each hypothesis in the dissertation 4

3.1 Experiment setting for each study in in-context learning for extrapolation 21

3.2 Performance (Hit@1) comparison between w/ and w/o time, and w/ shuffling in ICL for temporal knowledge graph forecasting 28

3.3 Performance (Hit@1) adheres to the scaling law based on the history length in ICL for temporal knowledge graph forecasting 30

3.4 Performance (Hit@1) adheres to the scaling law based on the model size in ICL for temporal knowledge graph forecasting 30

3.5 Performance (Hit@1) analysis on prompt variation in ICL for temporal knowledge graph forecasting. 31

3.6 Comparison between LLM-simulated dialogues and real-world human dialogues. 33

3.7 Emotional intelligence comparison between LLM-simulated dialogues LoCoMo [158] and human dialogues REALTALK. 40

3.8 Persona consistency of individual speaker across two conversations, capturing how their emotional intelligence varies depending on their conversational partners. 41

3.9 Impact of context in user simulation 43

4.1 Experiment setting for training models with context 47

4.2 Experiment setting for each study in human explanations as context for learning 48

4.3 Overview of training for named entity recognition using in-context task examples 49

4.4 Task examples as context for training named entity recognition 51

4.5 Task example template for training named entity recognition 55

4.6 Performance (F1-score) comparison between different demonstration selection strategies . 60

4.7 Performance (F1-score) trend by different numbers of train data (15, 20, 30, 40, 50). 61

4.8 Performance (F1-score) variance by different permutation of entity type orders 61

4.9 Performance (F1-score) difference between original and perturbed demonstration 62

4.10 Motivating example of studying chat in synchronous domain for violation detection 64

4.11 Data construction overview for studying norm violation in synchronous chat domain. . . . 66

4.12 Explanation of different types of dialogue context for norm violation detection 73

4.13 Performance (F1-score) of norm violation detection by different ground truth label for each context. 74

4.14 Performance (F1-score) trend of norm violation detection with varying context length. . . 75

4.15 Motivation example of explanation (entity trigger) based learning 78

4.16 Example of entity trigger 79

4.17 Framework overview: Two stage training of the Trigger Matching Network 82

4.18 Framework overview: The inference process of Trigger Matching Network 84

4.19 Overview of LEAN-LIFE 86

4.20 The workflow to annotate label and trigger span in LEAN-LIFE 88

4.21 The study of cost effectiveness of trigger matching network 91

4.22 Case study of trigger attention during inference 92

4.23 Motivation example of spurious correlation in the machine learning model 93

4.24 System architecture of E**X**planation-Based NLP **M**odel **D**ebugger (XMD) 95

4.25 The workflow to provide human feedback on instance-level explanation 97

4.26 The workflow to provide human feedback on task-level explanation 98

4.27 The workflow of instance-level explanation-based model debugging 99

4.28 The workflow of task-level explanation-based model debugging 99

4.29 Performance (Accuracy) trend by annotation efforts 102

5.1 Experiment setting for self-refinement through simulated interaction as contextual supervision 105

5.2 Motivation example of high-utility question in education context 107

5.3 Data construction overview for studying educative question generation 109

5.4 Data statistics of TEXTBOOK-EXAM in a perspective of Bloom's taxonomy distribution . 111

5.5 Framework overview of QUEST 112

5.6 Impact of threshold in QUEST on end-of-chapter exam scores for Chemistry. 118

Abstract

Contextual cues play an important role in enhancing the reasoning capabilities and adaptability of language models (LMs) when faced with complex tasks. Effective integration of context can make LMs interpret human requests more accurately and generate more precise responses. This thesis investigates strategic integration and dynamic utilization of diverse forms of context (*e.g.,* explanations as context, illustrative task examples, dialogue history, data-driven context, and model-generated context) to systematically improve the performance of LMs. This thesis addresses three core questions:

Part I investigates whether LMs inherently possess the capability to leverage contextual cues to enhance their reasoning during the inference phase. We show tasks that can effectively be achieved through contextual integration and identify other tasks for which context alone is insufficient.

Part II evaluates the extent to which explicitly incorporating contextual cues during the training phase can result in measurable improvements in model behavior. We demonstrate that incorporating context during training facilitates label-efficient learning and effective model debugging.

Part III explores the dynamic capability of LMs to autonomously generate and iteratively refine their contextual cues, thereby continuously enhancing their performance. We show that LMs can simulate certain scenarios to generate context that significantly improves their outputs.

Overall, this thesis proposes foundational groundwork for the systematic study and practical incorporation of multiple contextual dimensions into language modeling. The findings demonstrate significant

improvements in task-specific reasoning and establish principles for the strategic application of context to advance the capabilities of future LMs.

Chapter 1

Introduction

Contextual information significantly influences the outputs of language models, often determining their interpretation and responses. For instance, a single phrase like "*Final exam is the best gift on my birthday*" can result in contrasting sentiment classifications depending on how contextual clues are considered. When a model identifies sarcasm in the previous context, recognizing the mismatch between the expectation of a joyful birthday gift and the reality of a stressful event like a final exam, it correctly interprets the sentiment as negative. Conversely, without appropriately interpreting such contextual nuances, another model might mistakenly classify this same phrase as positive, assuming that the phrase's literal meaning indicates joy or humor (See Figure 1.1).

Such examples observed in real-world scenarios naturally lead us to ask several questions: How does context influence a model's predictions?; What specific types of context can be leveraged to improve model predictions?; For which tasks does context effectively enhance performance, and for which does it have limited or no benefit?; Does explicitly training language models with contextual information improve their overall performance?; Can language models autonomously refine provided context to further improve their own predictions?

In the following sections, I briefly discuss each of these questions to motivate the broader contributions and structure of this thesis.

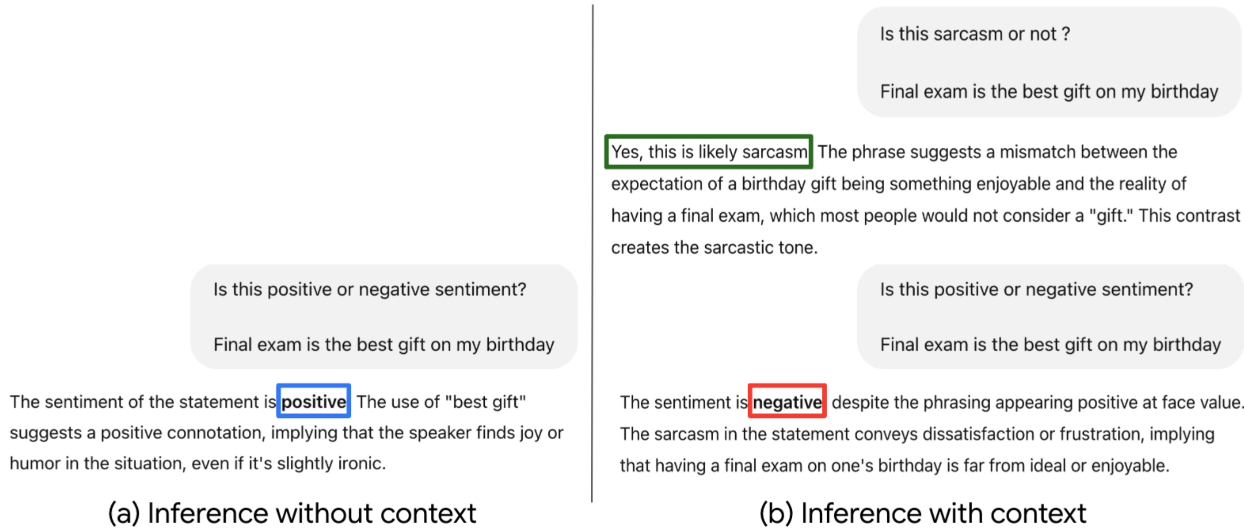


Figure 1.1: **Chat with ChatGPT.** Context determines the interpretation of language model outputs, as illustrated by differing sentiment predictions depending on whether prior dialogue context reveals sarcasm.

1.1 Background & Motivation

1.1.1 In-context Learning of Language Models (LMs)

Recent advancements in large language models (LLMs) have highlighted a powerful capability known as **in-context learning**. Unlike earlier generations of pre-trained language models, which typically required extensive labeled datasets to adapt to each new task, contemporary LLMs leverage meta-learning principles to rapidly generalize based on minimal contextual information. In this context, meta-learning describes the ability of a model to acquire broad, generalizable skills and sophisticated pattern-recognition capabilities during training, enabling the model to dynamically identify or adapt to specific tasks at inference time based solely on provided context, a process termed **in-context learning** [19].

In-context learning was initially studied primarily in the context of text classification, where models were provided with few-shot input–output examples per label to perform the classification task [165, 238]. However, the definition of in-context learning has since broadened to include any task where contextual information is incorporated directly into the input prompt. This context can take various forms, such as relevant documents for answering a question (i.e., retrieval-augmented generation) [188, 211, 61, 148, 131, 255, 160], dialogue history for maintaining coherent conversation [270, 243, 158], or prior event sequences for forecasting future events [167].

1.1.2 Towards Better Understanding of In-context Learning

While in-context learning has emerged as a powerful capability of modern language models, many fundamental questions about its behavior and limitations remain unanswered. For instance, we still lack a clear understanding of which types of tasks benefit most from contextual input, how different forms of context influence performance, and whether incorporating context during training can enhance a model's ability to reason from context at inference time. Furthermore, strategies for selecting, structuring, and refining contextual information remain underexplored.

This thesis is motivated by these gaps. By systematically studying in-context learning across diverse settings and contextual forms, I aim to build a deeper theoretical and empirical understanding of how context shapes language model behavior, and how this capability can be further enhanced.

1.2 Hypothesis

This dissertation hypothesizes the following:

1. **Language models can effectively leverage contextual information during inference to improve performance on various tasks.**

   To test this hypothesis, I set up diverse tasks (*i.e.,* temporal knowledge graph forecasting, persona simulation) under varying contextual conditions (*i.e.,* previous event history, dialogue history) to identify how context contributes to each task.

2. **Incorporating context during training enhances model behavior, enabling more robust generalization, label efficiency, and interpretability.**

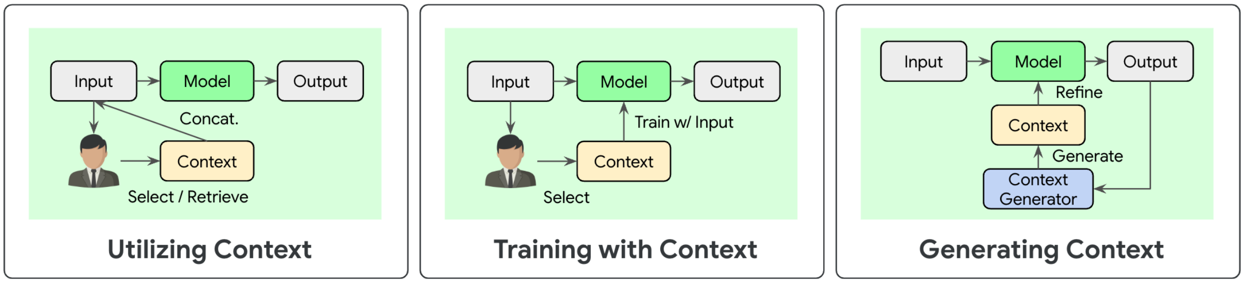


Figure 1.2: Experiment setting for each hypothesis in the dissertation

To test this hypothesis, I compare models trained with and without contextual signals, analyzing their performance, learning dynamics, and debugging capabilities.

3. **Language models can generate and refine context autonomously, improving their own downstream predictions.**

   To test this hypothesis, I design experiments where models iteratively construct and adapt contextual inputs, and assess the resulting impact on task performance.

Figure 1.2 presents the experiment setting for each hypothesis.

1.3 Contributions

This dissertation investigates how language models use, learn from, and generate context across training and inference. It offers three core contributions:

1.3.1 Task-Specific Effectiveness of In-Context Learning in Language Models

I conduct a systematic evaluation of in-context learning (ICL) across tasks that vary in the type of contextual signals they require. The results show that language models can effectively exploit contextual patterns to improve prediction performance, but the degree of success depends on the nature of the task and the kind of extrapolation required.

Specifically, I find that ICL is highly effective for tasks that involve **structural pattern extrapolation**, where models can identify and extend surface-level regularities in the input without requiring semantic interpretation. For example, in temporal knowledge graph forecasting, models accurately predict future facts by leveraging patterns in timestamped event sequences [120].

However, ICL performance declines in tasks that require **semantic extrapolation**, where models must interpret and internalize prior interaction history to simulate behavior or reasoning. For instance, when tasked with simulating a speaker's persona from multi-turn dialogue, models often struggle to ground their responses in meaningful context [125].

These findings show that while LLMs are capable of contextual generalization, their effectiveness varies with the depth of understanding required. This motivates the need for more robust modeling strategies, particularly when the context involves rich semantic or social information.

1.3.2 Training Language Models with Context Enhances Model Behavior

I demonstrate that explicitly incorporating contextual information during training leads to significant improvements in language model behavior, including enhanced robustness, generalization, and label efficiency. This chapter investigates two complementary forms of contextual supervision.

First, I show that simply appending auxiliary context (*e.g.*, task examples, dialogue history) to the input during both training and inference consistently improves model performance. Empirical results on tasks like named entity recognition (NER) and social norm violation detection confirm that contextualized inputs lead to higher accuracy and robustness [123, 170].

Second, I explore the use of human explanations as a rich form of contextual guidance. Explanations improve label efficiency by helping models internalize task-relevant reasoning, enabling effective learning from fewer examples [138, 124, 127]. In addition, I propose a post hoc refinement framework where explanations are used to regularize model parameters, correcting undesirable behavior when explanations reveal flaws [122].

Together, these findings highlight the versatility and effectiveness of contextual supervision as a principled strategy for improving language model learning and alignment.

1.3.3 Self-Refinement of Context by Language Models Improves Output Quality

I develop methods that enable language models to autonomously generate and iteratively refine contextual inputs, demonstrating that this self-refinement process enhances downstream output quality.

A central application explored is educational question generation. Here, the model first generates questions aimed at improving a learner's understanding of a given text. It then simulates learner responses and uses these simulated interactions as feedback (context) to revise and improve future questions. This iterative feedback loop enables the model to internalize what makes a question pedagogically effective, thereby improving performance over time [121].

These findings suggest that language models are not only consumers of context but also capable curators, able to construct and adapt context to support their own performance and learning.

1.4 Outline

The remainder of this dissertation is organized as follows: The main contributions of this dissertation are presented across the next three chapters:

- **Chapter 3** explores **inference with context** in language models. It analyzes the impact of in-context learning across two tasks: temporal knowledge graph forecasting [120] and persona simulation [125]. These studies have been published in *ACL* and *EMNLP*.

- **Chapter 4** investigates **training language models with context**. It examines three aspects: label efficiency [123, 138, 124, 127], performance improvement [170], and behavior correction [122]. These works have also appeared in *ACL* and *EMNLP*.
- **Chapter 5** introduces methods for **self-refining context** in language models. It focuses on how language models can generate and refine context to improve their own outputs, particularly in the setting of educational question generation [121].

Chapter 2

Background and Related Works

2.1 Background: The Evolution of Context in NLP

Context in natural language processing (NLP) refers to the surrounding information that shapes the meaning and interpretation of a given piece of text. Effective use of context is essential for language models to resolve ambiguity [23, 272, 76], identify referents [128, 101], interpret intent [233, 242], and generate coherent outputs over extended discourse [19, 187].

2.1.1 Static Representations: Early Views of Context

Early approaches to language understanding relied on fixed, context-independent word representations, such as one-hot encodings and count-based vector models like LSA and TF-IDF. These methods failed to capture semantic similarity or polysemy, which motivated the development of distributed word embeddings including Word2Vec [163], GloVe [182], and FastText [15].

These early embedding models incorporated a limited notion of context by leveraging co-occurrence statistics within a small, fixed-size window (typically 5 to 10 words) around the target token. In architectures such as CBOW and Skip-gram, this "**context**" consisted of immediate neighboring words used to predict or be predicted by the target word. While effective for learning general word associations, these models produced a single static vector for each word, regardless of its usage in different sentences or discourse contexts. As a result, they were unable to distinguish between context-dependent senses of words (*e.g.,* "bank" as a financial institution vs. a riverbank).

2.1.2 Dynamic Contextualization: Contextual Language Models

This limitation was addressed by the emergence of contextual language models, beginning with ELMo [184], which used deep bidirectional LSTMs to generate dynamic word embeddings conditioned on the full surrounding sentence. Unlike static embeddings, these representations vary depending on the specific context in which a word appears, marking a major shift in NLP toward contextualized language understanding.

The introduction of Transformer-based architectures grounded in attention mechanisms [225], particularly BERT [46] and GPT [186, 187], further advanced contextual language modeling by allowing models to flexibly attend over long-range input dependencies.

BERT employs bidirectional self-attention, enabling each token to incorporate information from both its left and right context. This design allows the model to capture sentence-level structure and disambiguate word meaning based on full contextual cues. Empirical results show that BERT achieves strong performance on tasks such as named entity recognition [215], co-reference resolution [101], and question answering [46], all of which require modeling relationships between distant tokens.

GPT, in contrast, adopts a unidirectional (left-to-right) auto-regressive formulation, enabling fluent text generation by conditioning on preceding tokens. Despite this directional constraint, GPT's ability to generate coherent paragraphs and dialogue reveals its strong contextual sensitivity over extended discourse [19].

2.1.3 Task-Oriented Context: Instructions, Examples, and Explanations

After the widespread success of large-scale generative models like GPT, which demonstrated strong generalization across diverse NLP tasks, the notion of "**context**" expanded beyond immediate textual surroundings such as adjacent words or sentences. Modern LMs increasingly leverage dynamic task-related context, including *task instructions*, *examples*, and *explanations*, to guide model behavior and improve performance:

Instruction as Context. Instruction describes the NLP task using natural language and often serve as the framework for guiding model behavior. Instructions may include other task-related forms of context, such as *examples* and *explanations* discussed in the following items, by specifying what the model is expected to do and under what constraints. After the emergence of auto-regressive language models, studies found that LMs can follow natural language instructions [54], demonstrating an implicit understanding of task intent. Building on this insight, substantial effort has been devoted to curating diverse collections of task instructions [169, 200, 175, 236], leading to the development of *instruction tuning*, a training paradigm in which models are fine-tuned on input–instruction–output triples. This approach has been shown to significantly improve generalization and zero-shot performance across a wide range of tasks.

Example as Context. Examples (*i.e.,* input–output pairs) are central to in-context learning [19], where models generalize to new inputs by conditioning on a few labeled demonstrations provided in the prompt. Examples support two types of generalization: (1) *Instance-level generalization*, where models learn to make predictions for unseen inputs within a given task by identifying patterns from a few labeled instances [60]; and (2) *Task-level generalization*, where models adapt to new tasks efficiently by leveraging prior experience with other tasks [252, 17]. Recent studies also show that including examples during *instruction tuning* further enhances generalization [252, 17], and that even incorporating *negative examples* (*e.g.,* incorrect outputs or counterexamples) can help models better distinguish correct behavior and improve task performance [232].

Explanation as Context. Explanation offers human rationale behind labels and have been shown to improve both label efficiency and interpretability. Several earlier studies explored training or fine-tuning language models with explanations as part of the supervision process [20, 234, 172, 275, 82]. In many of these works, explanations were provided in the form of span-level or token-level highlights rather than full natural language rationales [139, 130, 122]. More recently, LLMs have adopted *decomposition-style explanations*, which break down the reasoning process into intermediate steps, commonly referred to as chain-of-thought prompting [237]. These methods are particularly effective when augmented with *external scratchpads*, which serve as memory structures for storing intermediate computations [174]. Another line of research focuses on *input-level explanations*, where natural language justifications accompany task descriptions and examples. While such explanations can improve instruction tuning, their benefits appear to be most pronounced in larger models, suggesting that sufficient model capacity is necessary to effectively utilize explanatory signals [113, 232, 169].

These forms of context shift model behavior from implicit pattern recognition to more interpretable, instruction-following, and task-aware responses.

2.1.4 Interactive and Situated Context

This broadened understanding of context now also extends to external, conversational, and social dimensions.

Retrieved Passages as Context. Retrieval-augmented generation (RAG) incorporates external knowledge sources to inform model outputs. This can be achieved by combining retrieved passages with model inputs during pre-training [75, 16, 92], integrating retrieval-based token distributions into generation [107, 253, 271, 116, 166], or prompting/fine-tuning models using dynamically retrieved documents [188, 211]. These approaches allow LMs to go beyond the limitations of their static parametric memory and incorporate up-to-date or domain-specific knowledge on demand.

Dialogue History as Context. In multi-turn conversational systems, dialogue history serves as a critical form of context. By conditioning on previous utterances, models can maintain coherence, resolve coreference, and track user intent across extended interactions [265, 44, 248, 91, 207]. This type of context is essential for enabling goal-oriented, consistent, and natural conversations with users.

Persona as Context. In personalized and social NLP, persona information provides another layer of context that guides model behavior. Personas may take the form of demographic traits (*e.g.,* occupation, ethnicity, or personality) [86, 247, 73, 132], fictional character profiles [177, 208, 228], or individualized representations based on interaction histories [168, 199, 24, 178]. Integrating such context enables models to generate more socially coherent, consistent, and user-tailored responses.

2.1.5 Summary

In summary, the development of contextual models in NLP has evolved from static, word-level representations to dynamic, multi-scale architectures capable of capturing nuanced, real-world context. Today's LMs not only leverage linguistic context from surrounding text but also integrate task-specific instructions, retrieved external knowledge, dialog history, and social or personalized signals. This shift reflects a broader trend toward situated understanding, where language is interpreted in relation to users, settings, and social dynamics, enabling more robust, adaptive, and human-aligned language systems.

2.2 Related Works: Contextual Learning Strategies for Language Models

Building on the historical and conceptual foundations of contextual modeling, I now examine recent efforts that more directly explore how LMs can leverage, generate, or be trained with context. I organize this section into three threads: (i) in-context learning in LMs; (ii) contextual supervision during LM training; and (iii) LMs as self-refining context generators.

2.2.1 In-context Learning in Language Models

ICL allows LLMs to perform new tasks without any parameter updates, simply by conditioning on a few input–output examples provided in the prompt [19, 31]. These examples implicitly convey task intent, enabling the model to adapt its behavior based on context alone.

Early work has proposed two complementary hypotheses to explain how ICL works: (1) models rely on **semantic prior knowledge** acquired during pretraining to interpret and perform tasks; and (2) models can perform **function induction**, learning input–output mappings from few-shot demonstrations in the prompt.

1. **Semantic Prior Knowledge.** This view emphasizes the importance of semantic priors that knowledge of task structure and label meaning encoded during pretraining. For example, [191] show that ICL performance significantly deteriorates when the pretraining data lacks relevant task concepts, highlighting the need for exposure to semantically related tasks. Similarly, [165] demonstrate that ICL often fails under distribution shift, especially when the examples in the prompt differ from the test distribution in style or semantics.

2. **Function Induction.** The other view suggests that ICL can succeed even without semantic grounding. [63] and [238] show that sufficiently large models can learn to generalize from synthetic examples in tasks with no inherent meaning, indicating an ability to induce patterns from scratch. These results suggest that ICL works like a form of real-time learning, where the model figures out patterns directly from the examples in the prompt, similar to how meta-learning allows models to quickly adapt to new tasks.

Rather than being mutually exclusive, these two mechanisms may be jointly at play. Semantic priors help models recognize the type of task they are being asked to perform, while input–label induction enables them to generalize to new inputs within that task. The extent to which each contributes to performance may depend on the task type, prompt design, and model scale.

While ICL was initially studied in the context of text classification, recent work has expanded its application to more complex settings. Models now leverage context from retrieved documents for open-domain question answering [188, 211, 61], dialogue history for conversational coherence [270, 243], and temporally structured event sequences for forecasting [167]. This broader view positions ICL as a general mechanism for adapting model behavior at inference time through contextual prompts alone.

2.2.2 Contextual Supervision for Language Model Training

Beyond inference-time context, another line of research focuses on incorporating richer forms of context during training. This includes fine-tuning or instruction-tuning LMs on datasets augmented with task descriptions, exemplars, and explanations [200, 236, 175].

Before the rise of instruction-tuned generative LMs, earlier studies investigated contextual supervision in encoder-only models such as BERT. These approaches enriched input representations with auxiliary signals, including span-level annotations [25], and rationale-based highlights [259, 38, 9, 93, 48, 39] to improve model performance and interpretability. Additionally, several works incorporated natural language explanations into training data to support tasks such as arithmetic reasoning [36], multi-hop question answering [66], and natural language inference [20], demonstrating the value of explicit reasoning traces for improving generalization.

These lines of research laid the groundwork for modern instruction tuning paradigms, which unify task descriptions, input examples, and natural language explanations into a single training framework. Such contextual supervision not only enhances zero-shot generalization, but also improves model alignment with user intent, robustness to distribution shifts, and interpretability.

2.2.3 Language Models as Self-Refining Context Generators

Collecting human-provided or externally sourced context can be challenging and costly, motivating the use of synthetic context generated directly by language models. Recent studies have shown that such model-generated contexts can effectively enhance task performance and serve as robust contextual signals.

1. **Model-Generated Reasoning Traces as Context.** A core idea in recent work is that LMs can generate their own contextual information during inference to support more accurate predictions. One prominent example is *chain-of-thought* (CoT) prompting [237], where models are encouraged to produce intermediate reasoning steps before arriving at a final answer. These generated traces act as contextual scaffolding which extends the prompt with additional structure that the model itself uses to guide its decision-making. By explicitly unfolding its reasoning process, the model creates a richer internal context that improves performance on tasks involving multi-step logical or arithmetic inference. This approach is especially effective in larger models, where reasoning capabilities scale with size [109, 6, 51, 229, 201]. Further improvements come from *self-consistency decoding* [231], in which multiple reasoning traces are sampled, and the most consistent final answer is selected. This leverages diversity in model-generated context to enhance reliability, treating each reasoning path as a candidate context for robust aggregation.

2. **Model-Synthesized Knowledge as Context.** In knowledge-intensive tasks, models can generate factual or background knowledge relevant to the current input, effectively constructing their own retrieval context without relying on external databases. These synthesized *knowledge snippets* are prepended to the task input, providing models with supportive context that enhances understanding and performance [146, 56, 255, 153]. This approach treats the model not just as a consumer of context but as an active producer of it, capable of surfacing latent knowledge stored in its parameters.

Remarkably, generated knowledge often rivals or outperforms retrieved passages, especially in zero-shot or domain-limited scenarios where curated corpora may be unavailable.

3. **Model-Generated Feedback as Context for Iterative Refinement.** Another work explores how models can generate evaluative or corrective context (*e.g.,* feedback, rationales) to refine their own outputs. This context is used internally across iterative steps, enabling self-improvement during inference. For instance, STaR [261] prompts models to generate rationales and then revise their original answers using those rationales as context. Similarly, Self-Refine [157] introduces retrospective feedback loops where models reflect on and learn from their own mistakes. In all cases, generated feedback becomes a form of self-curated context that guides subsequent reasoning and output updates.

Together, these strategies illustrate how LMs can act not only as predictors but as active producers of their own context, using generated reasoning, knowledge, and feedback to improve accuracy, interpretability, and robustness—especially in settings where external supervision is limited or unavailable.

2.3 Summary

This chapter has reviewed prior work relevant to the three main contributions of this dissertation, each reflecting a distinct but interconnected perspective on how LMs utilize and benefit from context.

First, I summarized existing work on in-context learning (ICL), where language models adapt to tasks by conditioning on examples provided at inference time. This body of research proposes two complementary mechanisms (*i.e.,* **function induction**, **semantic prior knowledge**) to explain how models generalize from context. ICL has since expanded into a general framework for test-time adaptation across tasks such as open-domain QA, dialogue, and event forecasting. This dissertation builds on these foundations by distinguishing between **structural** and **semantic** forms of extrapolation, offering a finer-grained analysis of what kinds of patterns LMs can generalize from contextual input.

Second, I discussed contextual supervision during training, highlighting that integrating auxiliary signals like examples, explanations, and task descriptions into the training process significantly enhances model generalization, label efficiency, and robustness.

Third, I explored how LMs can serve as self-refining context generators, synthesizing their own contextual inputs to support downstream performance. By generating rationales, knowledge snippets, or iterative feedback, models can bootstrap their learning and output quality in settings where human-labeled context is scarce.

Together, these threads illustrate a growing understanding of context not merely as input, but as a dynamic mechanism for guiding LM behavior across training and inference. The following chapters build on this foundation, presenting empirical studies that advance our understanding of how context can be used, learned from, and generated by LMs.

Chapter 3

Context-Aware Inference in Language Models

This chapter explores how large language models (LLMs) perform in-context learning (ICL) beyond the standard few-shot task setting. I hypothesize that LLMs can generalize not only from explicit input-label pairs, but also from richer contextual structures, such as temporal sequences and dialogue histories, without parameter updates. To investigate this, I distinguish between two forms of extrapolation: structural (based on surface patterns) and semantic (based on contextual understanding). The following sections outline existing ICL mechanisms, then extend them to broader contextual learning tasks.

3.1 Background and Motivation

3.1.1 In-context Learning with Task Examples

ICL enables LLMs to perform new tasks without parameter updates by conditioning on a small number of labeled examples [19, 31]. In the ICL setup, task instructions are implicitly communicated through input-output exemplars provided in the prompt.

In order to effectively engage in ICL, models can leverage semantic prior knowledge to accurately predict labels following the structure of in-context exemplars [165, 191, 246, 22, 78], and learn the input-label mappings from the in-context examples presented [238]. To understand the mechanism of ICL, recent studies have explored the ICL capabilities of LLMs with regards to the impact of semantic prior knowledge by examining their correlation with training examples [165, 191, 246], data distribution [22], and language compositionality [78] in the pre-training corpus. Other recent works show that LLMs can actually learn input-label mappings from in-context examples by showing the transformer models trained on specific linear function class is actually predicting accurately on new unseen linear functions [63]. More recently, there is a finding that large-enough models can still do ICL using input-label mappings when semantic prior knowledge is not available [238].

Formally, a few-shot classification prompt is constructed by linearizing k input-output pairs $(\mathbf{x}_i, \mathbf{y}_i)$, followed by a new input $\mathbf{x}_{\text{test}}$. The model predicts the next token(s) to generate the output:

$$\mathbf{y}_{\text{test}} \sim \mathcal{P}_{\text{LLM}}(\mathbf{y}_{\text{test}} \mid \mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_k, \mathbf{y}_k, \mathbf{x}_{\text{test}}),$$

where the decoding strategy determines how the final prediction is sampled or selected.

While this setup captures task extrapolation, where models generalize across examples from the same task, it raises a broader question: **Can models extrapolate from patterns that go beyond task labels, such as temporal or social regularities embedded in interaction histories?**

This motivates a broader formulation of in-context learning, where the model is conditioned not on task examples $(\mathbf{x}_i, \mathbf{y}_i)$, but on a generic context $\mathbf{c}$, which may include temporally or socially structured sequences (e.g., events, dialogues). The objective becomes:

$$\mathbf{y}_{\text{test}} \sim \mathcal{P}_{\text{LLM}}(\mathbf{y}_{\text{test}} \mid \mathbf{c}, \mathbf{x}_{\text{test}})$$

Here, $\mathbf{c}$ is a context window composed of interaction history rather than curated exemplars, and the model must infer latent patterns or meanings from this history to make accurate predictions for $\mathbf{x}_{\text{test}}$.

3.1.2 Contextual Extrapolation from Interaction History

Recent work has shown that in-context learning can generalize from broader contextual structures beyond traditional task exemplars. In particular, sequences of events or dialogue history can serve as an implicit context for prediction, allowing LLMs to exhibit new forms of extrapolation.

We distinguish between two types of contextual extrapolation:

1. **Structural extrapolation:** The model identifies and extends latent patterns in sequential input (*e.g.*, event timelines, index sequences) without necessarily understanding their semantic meaning. For instance, even if the inputs are arbitrary tokens or indices, LLMs can infer regularities and generate the next item in the pattern.

2. **Semantic extrapolation:** The model must internalize and interpret prior interaction context to perform reasoning or simulation. This includes modeling user persona, goals, or emotions based on preceding dialogue, requiring deeper semantic understanding rather than pattern matching alone.

These two forms of contextual generalization expand the scope of in-context learning. While structural extrapolation highlights the model's ability to recognize statistical regularities, semantic extrapolation emphasizes its capacity for grounded reasoning and long-range coherence. The subsequent chapters explore these capabilities across a range of tasks involving both event sequences and interactive dialogues.

3.2 In-context Learning for Structural Extrapolation

3.2.1 Introduction

This section explores **in-context learning** for *structural extrapolation* using the temporal knowledge graph (TKG) forecasting task as a benchmark. An illustrative example of this task is the question "*Which team will win the Super Bowl in 2023?*" that can be expressed as $q =$ (*Super bowl*, *Champion*, ?, 2023) and

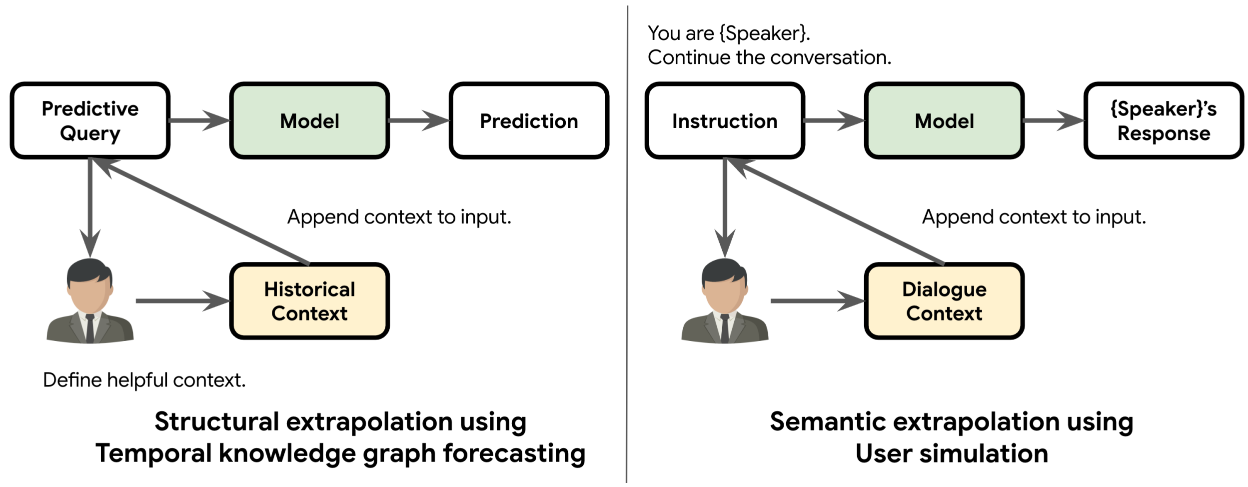


Figure 3.1: Experiment setting for each study in in-context learning for extrapolation

$\mathcal{E}_q$ = {(*Super bowl*, *Champion*, *Los Angeles*, 2022), (*Super bowl*, *Champion*, *Tampa Bay*, 2021), ...} (See Table 3.1). The ultimate objective is to identify the most suitable entity among all the football teams $\mathcal{E}$ (*e.g.*, *St Louis*, *Baltimore*) to fill the missing field. Here, we introduce a novel approach to TKG forecasting by framing it as an in-context learning (ICL) problem for LLMs. ICL is the ability of LLMs to quickly adapt to and perform new tasks with minimal examples provided in the form of input-label pairs [19]. Unlike previous ICL research that typically relies on equal numbers of examples for each label in classification tasks [165, 238], our study investigates what the model learns from irregular patterns of historical facts in the context.

We design a three-stage pipeline to control (1) the background knowledge selected for context, (2) the prompting strategy for forecasting, and (3) decoding the output into a prediction. The first stage uses the prediction query to retrieve a set of relevant past facts from the TKG that can be used as context. The second stage transforms these contextual facts into a lexical prompt representing the prediction task. The third stage decodes the output of the LLM into a probability distribution over the entities and generates a response to the prediction query. Our experimental evaluation performs competitively across a diverse collection of TKG benchmarks without requiring the time-consuming supervised training, or

Which team will win the Super Bowl in 2023?

```
2000: [Superbowl, Champion, St Louis]
2001: [Superbowl, Champion, Baltimore]
2002: [Superbowl, Champion, New England]
2003: [Superbowl, Champion, Tampa Bay]
...
2019: [Superbowl, Champion, New England]
2020: [Superbowl, Champion, Kansas City]
2021: [Superbowl, Champion, Tampa Bay]
2022: [Superbowl, Champion, Los Angeles]
2023: [Superbowl, Champion,
```

Model	# Params.	Prediction	G.T Rank
EleutherAI `gpt-j-6b`	6B	Los Angeles	3
EleutherAI `gpt-neox-20b`	20B	Kansas City	1
OpenAI `text-ada-001`	350M	Carolina	>5
OpenAI `text-babbage-001`	1.3B	Indianapolis	>5
OpenAI `text-curie-001`	6.7B	New England	>5
OpenAI `text-davinci-003`	175B	TBD	>5
OpenAI `gpt-3.5-turbo`	-	Sorry, I cannot predict future events.	

Table 3.1: Motivation example for structural pattern extrapolation using in-context learning

custom-designed architectures. Our findings are as follows: (1) LLMs demonstrate the ability to make predictions about future facts using ICL without requiring any additional training. Moreover, these models show comparable performance to supervised approaches, falling within the (-3.6%, +1.5%) Hits@1 margin, relative to the median approach for each dataset; (2) LLMs perform almost identically when we replace entities' and relations' lexical names with numerically mapped indices, suggesting that the prior semantic knowledge is not a critical factor for achieving such a high performance; and (3) LLMs outperform the best heuristic rule-based baseline on each dataset (*i.e.,* the most frequent or the most recent, given the historical context) by (+10%, +28%) Hits@1 relative margin, indicating that they do not simply select the output using frequency or recency biases in ICL [269].

3.2.2 Temporal Knowledge Graph Forecasting: Problem Formulation

Formally, a TKG, $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E}, \mathcal{T})$, is comprised of a set of entities $\mathcal{V}$, relations $\mathcal{R}$, facts $\mathcal{E}$, and timestamps $\mathcal{T}$. Moreover, since time is sequential, $\mathcal{G}$ can be split into a sequence of time-stamped snapshots, $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_t, \ldots\}$, where each snapshot, $\mathcal{G}_t = (\mathcal{V}, \mathcal{R}, \mathcal{E}_t)$, contains the facts at a specific point in time t. Each fact $f \in \mathcal{E}_t$ is a quadruple (s, p, o, t) where $s, o \in \mathcal{V}$, $p \in \mathcal{R}$, and $t \in \mathcal{T}$. The TKG forecasting task involves predicting a temporally conditioned missing entity in the future given a query quadruple, $(?, p, o, t)$ or $(s, p, ?, t)$, and previous graph snapshots $\mathcal{G}_{1:t-1} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{t-1}\}$. Here, the prediction typically involves ranking each entity's assigned score.

3.2.3 In-context Learning for Temporal Knowledge Graph Forecasting

In this work, we focus on 1) modeling appropriate history $\mathcal{E}_q$ for a given query quadruple q, 2) converting $\{\mathcal{E}_q, q\}$ into a prompt θ_q, and 3) employing ICL to get prediction $\mathbf{y}_q \sim \mathcal{P}_{\text{LLM}}(\mathbf{y}_q \mid \theta_q)$ in a zero-shot manner. Here, the history $\mathcal{E}_q$ is modeled on the facts from the previous graph snapshots $\mathcal{G}_{1:t-1} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_{t-1}\}$, and we employ token probabilities for $\mathbf{y}_q$ to get ranked scores of candidate entities in a zero-shot manner. In the rest of this section, we study history modeling strategies, response generation approaches, prompt construction templates, and common prediction settings.

History Modeling. To model the history $\mathcal{E}_q$, we filter facts that the known entity or relation in the query q has been involved in. Specifically, given the query quadruple $q = (s, p, ?, t)$ under the object entity prediction setting, we experiment with two different aspects of historical facts:

(1) Entity vs. Pair. `Entity` includes past facts that contain s, *e.g.,* all historical facts related to *Superbowl*. In contrast, `Pair` includes past facts that contain both s and p, *e.g.,* a list of (*Superbowl, Champion, Year*) as shown in Table 3.1;

(2) Unidirectional vs. Bidirectional. `Unidirectional` includes past facts $\mathcal{F}$ wherein s (`Entity`) or (s, p) (`Pair`) is in the same position as it is in q (*e.g.,* `Unidirectional` & `Pair` – s and p served as subject and predicate in $f \in \mathcal{F}$). `Bidirectional` includes past facts $\mathcal{F}$ wherein s (`Entity`) or (s, p) (`Pair`) appear in any valid position (*e.g.,* `Bidirectional` & `Entity` – s served as subject or object in $f \in \mathcal{F}$). As an example of the `Bidirectional` setting, given $q =$ (*Superbowl*, *Champion*, ?, 2023), we include $f =$ (*Kupp*, *Played*, *Superbowl*, 2022) because s (*i.e., Superbowl*) is present as the object in f. Moreover, in the `Bidirectional` setting, to preserve the semantics of the facts in the $\mathcal{E}_q$, we transform the facts where s appears as an object by 1) swapping the object and subject and 2) replacing the relation with its uniquely defined inverse relation (*e.g.,* $(f_s, f_p, f_o, f_t) \rightarrow (f_o, f_p^{-1}, f_s, f_t)$).

Response Generation. Given a prompt θ_q, we pass it to an LLM to obtain the next token probabilities. Then, we use the obtained probabilities to get a ranked list of entities. However, obtaining scores for entities based on these probabilities is challenging as they may be composed of several tokens. To address this challenge, we utilize a mapped numerical label as an indirect logit to estimate their probabilities [141].

Prompt Construction. Given the history $\mathcal{E}_q$ and query q, we construct a prompt using a pre-defined template θ. Specifically, given the query quadruple $q = (s, p, ?, t)$ under the object entity prediction setting, we present two versions of the template θ with varying levels of information. Our assumption is that each entity or relation has an indexed $\mathcal{I}(\cdot)$ (*e.g.,* 0) and a lexical $\mathcal{L}(\cdot)$ (*e.g., Superbowl*) form (See Table 3.2).

Category	Prompt
Lexical $\mathcal{L}(\cdot)$	2000: [Superbowl, Champion, 0. St Louis] 2001: [Superbowl, Champion, 1. Baltimore] … 2023: [Superbowl, Champion,
Index $\mathcal{I}(\cdot)$	2000: [0, 0, 0. 0] 2001: [0, 0, 1. 1] … 2023: [0, 0,

Table 3.2: Prompt example of in-context learning for temporal knowledge graph forecasting

Index. `Index` displays every fact, $(f_s, f_p, f_o, f_t) \in \mathcal{E}$ using the "f_t:[$\mathcal{I}(f_s), \mathcal{I}(f_p)$, $n_{f_o}. \mathcal{I}(f_o)$]" template where $f_s, f_o \in \mathcal{V}$, $f_p \in \mathcal{R}$, $f_t \in \mathcal{T}$, n_{f_o} denotes an incrementally assigned numerical label (*i.e.,* indirect logit), and $\mathcal{I}$ is a mapping from entities to unique indices. For example, in Table 3.1, we can use the following mappings are for the entities and relations, respectively: {*Superbowl* $\rightarrow$ 0, *St Louis* $\rightarrow$ 1, *Baltimore* $\rightarrow$ 2} and {*Champion* $\rightarrow$ 0}. The query q is then represented as "t:[$\mathcal{I}(s)$, $\mathcal{I}(p)$,", concatenated to the end of the prompt. For subject entity prediction, we follow the same procedure from the other side.

Lexical. `Lexical` follows the same process as `Index` but uses lexical form $\mathcal{L}(\cdot)$ of entity and relation. Each fact in $(f_s, f_p, f_o, f_t) \in \mathcal{E}$ is represented as "f_t:[$\mathcal{L}(f_s), \mathcal{L}(f_p), n_{f_o}.\ \mathcal{L}(f_o)$]" and the query q is represented as "t:[$\mathcal{L}(s), \mathcal{L}(p)$,", concatenated to the end of the prompt.

Prediction Setting. All the historical facts in the dataset are split into three subsets, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{valid}}$, and $\mathcal{D}_{\text{test}}$, based on the chronological order with train < valid < test. Given this split, during the evaluation phase, the TKG forecasting task requires models to predict over $\mathcal{D}_{\text{test}}$ under the following two settings:

(1) Single Step. In this setting, for each test query, the model is provided with ground truth facts from past timestamps in the *test* period. Hence, after making predictions for a test query in a specific timestamp, the ground truth fact for that query is added to the history before moving to the test queries in the next timestamp.

(2) Multi Step. In this setting, the model is *not* provided with ground truth facts from past timestamps in the *test* period and has to rely on its noisy predictions. Hence, after making predictions for a test query in a specific timestamp, instead of the ground truth fact for that query, we add the predicted response to the history before moving to the test queries in the next timestamp. This setting is considered more difficult as the model is forced to rely on its own noisy predictions, which can lead to greater uncertainty with each successive timestamp.

3.2.4 Experimental Setup

For our experiments, we use the WIKI [118], YAGO [159], ICEWS14 [62], and ICEWS18 [98] benchmark datasets with the unified splits introduced in previous studies [64]. Additionally, we extract a new temporal forecasting dataset from the Armed Conflict Location & Event Data Project (ACLED) project* which provides factual data of crises in a particular region. We specifically focus on incidents of combat and violence against civilians in Cabo Delgado from January 1900 to March 2022, using data from October

*https://data.humdata.org/organization/acled

2021 to March 2022 as our test set. This dataset aims to investigate whether LLMs leverage prior semantic knowledge to make predictions and how effective they are when deployed in real-world applications.

Evaluation We evaluate the models on well-known metrics for link prediction: Hits@k, with $k = 1, 3, 10$. Following [64], we report our results in two evaluation settings: 1) **Raw** retrieves the sorted scores of candidate entities for a given query quadruple and calculates the rank of the correct entity; and 2) **Time-aware filter** also retrieves the sorted scores but removes the entities that are valid predictions before calculating the rank, preventing them from being considered errors. To illustrate, if the test query is (NBA, Clinch Playoff, ?, 2023) and the true answer is Los Angeles Lakers, there may exist other valid predictions such as (NBA, Clinch Playoff, Milwaukee Bucks, 2023) or (NBA, Clinch Playoff, Boston Celtics, 2023). In such cases, the time-aware filter removes these valid predictions, allowing for accurate determination of the rank of the "Los Angeles Lakers." In this paper, we present performance with the time-aware filter.

ICL Implementation Details. We implement frameworks using PyTorch [179] and Hugging face [241]. We first collate the facts $f \in \mathcal{D}_{test}$ based on the identical test query to eliminate any repeated inference. To illustrate, suppose there exist two facts in the test set denoted as (s, p, a, t) and (s, p, b, t) in the object prediction scenario. We consolidate these facts into $(s, p, [a, b], t)$ and forecast only one for $(s, p, ?, t)$. Subsequently, we proceed to generate an output for each test query with history by utilizing the model, obtaining the probability for the first generated token in a greedy approach, and sorting the probability. The outputs are deterministic for every iteration. We retain the numerical tokens corresponding to the numerical label n that was targeted, selected from the top 100 probability tokens for each test query. To facilitate multi-step prediction, we incorporate the top-k predictions of each test query as supplementary reference history. In this paper, we present results with $k = 1$. It is important to acknowledge that the prediction may contain minimal or no numerical tokens as a result of inadequate in-context learning. This

Single-Step	Train	YAGO			WIKI			ICEWS14			ICEWS18			ACLED-CD22		
		H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10
`RE-GCN` [136]	✓	0.787	0.842	0.884	0.747	0.817	0.846	0.313	0.473	**0.626**	**0.223**	**0.367**	**0.525**	**0.446**	**0.545**	**0.608**
`xERTE` [79]	✓	0.842	0.902	**0.912**	0.703	0.785	0.801	0.330	0.454	0.570	0.209	0.335	0.462	0.320	0.445	0.497
`TLogic` [151]	✓	0.740	0.789	0.791	**0.786**	**0.860**	**0.870**	**0.332**	**0.476**	0.602	0.204	0.336	0.480	0.009	0.045	0.094
`TANGO` [80]	✓	0.590	0.646	0.677	0.483	0.514	0.527	0.272	0.408	0.550	0.191	0.318	0.462	0.327	0.482	0.599
`Timetraveler` [217]	✓	**0.845**	**0.908**	**0.912**	0.751	0.820	0.830	0.319	0.454	0.575	0.212	0.325	0.439	0.240	0.315	0.457
`GPT-NeoX` [13] (Entity)	✗	0.784	0.891	**0.927**	0.694	0.804	0.844	**0.324**	**0.460**	**0.565**	0.192	**0.313**	**0.414**	**0.324**	**0.492**	**0.604**
`GPT-NeoX` [13] (Pair)	✗	**0.787**	**0.892**	0.926	**0.721**	**0.812**	**0.847**	0.297	0.408	0.482	**0.196**	0.307	0.402	0.317	0.440	0.566

Multi-Step	Train	YAGO			WIKI			ICEWS14			ICEWS18			ACLED-CD22		
		H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10
`RE-GCN` [136]	✓	**0.717**	**0.776**	0.817	0.594	0.648	0.678	**0.278**	**0.421**	**0.575**	**0.195**	**0.326**	**0.475**	**0.421**	0.464	0.502
`RE-Net` [98]	✓	0.534	0.613	0.662	0.472	0.507	0.530	**0.278**	0.408	0.549	0.184	0.314	0.461	0.238	0.445	0.563
`CyGNet` [277]	✓	0.613	0.742	**0.834**	0.525	0.624	0.675	0.266	0.402	0.545	0.166	0.295	0.444	0.408	**0.500**	**0.588**
`TLogic` [151]	✓	0.631	0.706	0.715	**0.613**	**0.663**	**0.682**	0.265	0.395	0.531	0.155	0.272	0.412	0.009	0.045	0.094
`GPT-NeoX` [13] (Entity)	✗	0.686	**0.793**	**0.840**	0.543	0.622	**0.655**	**0.247**	**0.363**	**0.471**	0.136	0.224	0.321	**0.319**	**0.417**	**0.500**
`GPT-NeoX` [13] (Pair)	✗	**0.688**	**0.793**	0.839	**0.570**	**0.625**	0.652	0.236	0.324	0.395	**0.155**	**0.245**	**0.331**	0.289	0.410	0.464

Table 3.3: Performance (Hits@K) comparison between supervised models and ICL for temporal knowledge graph forecasting

can lead to problems when evaluating rank-based metrics. To mitigate this, we have established a protocol where the rank of the actual value within the predictions is assigned a value of 100, which is considered incorrect according to our evaluation metric.

3.2.5 Experimental Results

In this section, we present a multifaceted performance analysis of ICL under `Index` & `Unidirection` prompt strategy for both `Entity` and `Pair` history.

Q1: How well does ICL fare against the supervised learning approaches? We present a comparative analysis of the top-performing ICL model against established supervised learning methodologies for TKG reasoning, which are mostly based on graph representation learning. As evident from the results in Table 3.3, `GPT-NeoX` with a history length of 100 shows comparable performance to supervised learning approaches, without any fine-tuning on any TKG training dataset.

Q2: How do frequency and recency biases affect ICL's predictions? To determine the extent to which LLMs engage in pattern analysis, beyond simply relying on frequency and recency biases, we run a comparative analysis between `GPT-NeoX` and heuristic-rules (*i.e.,* `frequency` & `recency`) on the ICEWS14

Single-Step	ICEWS14			ICEWS18		
	H@1	H@3	H@10	H@1	H@3	H@10
frequency	0.243	0.387	0.532	0.141	0.265	0.409
recency	0.228	0.387	0.536	0.120	0.242	0.403
GPT-NeoX (Entity)	**0.324**	**0.460**	**0.565**	0.192	**0.313**	**0.414**
GPT-NeoX (Pair)	0.297	0.408	0.482	**0.196**	0.307	0.402

(a) Single-step

Multi-Step	ICEWS14			ICEWS18		
	H@1	H@3	H@10	H@1	H@3	H@10
frequency	0.222	0.349	0.460	0.121	0.207	0.307
recency	0.151	0.268	0.423	0.074	0.149	0.266
GPT-NeoX (Entity)	**0.247**	**0.363**	**0.471**	0.136	0.224	0.321
GPT-NeoX (Pair)	0.236	0.324	0.395	**0.155**	**0.245**	**0.331**

(b) Multi-step

Table 3.4: Performance (Hits@K) comparison between rule-based predictions and ICL for temporal knowledge graph forecasting

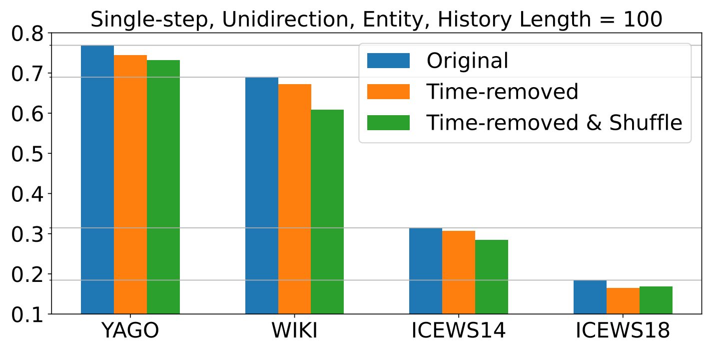


Figure 3.2: Performance (Hit@1) comparison between w/ and w/o time, and w/ shuffling in ICL for temporal knowledge graph forecasting

dataset, with history length set to 100. `frequency` identifies the target that appears most frequently in the provided history while `recency` selects the target associated with the most recent fact in the provided history. The reason for our focus on ICEWS is that each quadruple represents a single distinct event in time. In contrast, the process of constructing YAGO and WIKI involves converting durations to two timestamps to display events across the timeline. This step has resulted in `recency` heuristics outperforming all of the existing models, showcasing the shortcoming of existing TKG benchmarks. The experimental results presented in Table 3.4 demonstrate that ICL exhibits superior performance to rule-based baselines. This finding suggests that ICL does not solely rely on specific biases to make predictions, but rather it actually learns more sophisticated patterns from historical data.

Q3: How does ICL use the sequential and temporal information of events? To assess the ability of LLMs to comprehend the temporal information of historical events, we compare the performance of prompts with and without timestamps. Specifically, we utilize the original prompt format, "f_t:[$\mathcal{I}(f_s)$,

$\mathcal{I}(f_r), n_{f_o}. \mathcal{I}(f_o)]$", and the time-removed prompt format, "$[\mathcal{I}(f_s), \mathcal{I}(f_r), n_{f_o}. \mathcal{I}(f_o)]$", make the comparison. Additionally, we shuffle the historical facts in the time-removed prompt format to see how the model is affected by the corruption of sequential information. Figure 3.2 shows that the absence of time reference can lead to a deterioration in performance, while the random arrangement of historical events may further exacerbate this decline in performance. This observation implies that the model has the capability to forecast the subsequent event by comprehending the sequential order of events.

Q4: How does instruction tuning affect ICL's performance? To investigate the impact of instruction tuning on ICL, we employ the `gpt-3.5-turbo` model with manually curated system instruction. Since the size of this model is not publicly disclosed, it is challenging to make direct comparisons with other models featured in this paper. Moreover, since this model does not provide output probabilities, we are only able to report the Hit@1 metric. Table 3.5 showcases that the performance of the `lexical` prompts exceeds that of the `index` prompts by 0.024, suggesting that instruction-tuned models can make better use of semantic priors. This behavior is different from the other foundation LLMs, where the performance gap between the two prompt types was insignificant (See Figure 3.5 (a)).

Single-Step	**Prompt**	**ICEWS14**
		H@1
`gpt-3.5-turbo`	`index`	0.1615
`gpt-3.5-turbo`	`lexical`	**0.1858**

Table 3.5: Performance (Hits@1) comparison between `index` and `lexical` in ICL for temporal knowledge graph forecasting

Q5: How does history length affect ICL's performance? To evaluate the impact of the history length provided in the prompt, we conduct a set of experiments using varying history lengths. For this purpose, we use the best performing prompt format for each benchmark, *i.e.,* `Entity` for WIKI, YAGO, ICEWS18, and `Pair` for ICEWS14. Our results, as shown in Figure 3.3, indicate a consistent improvement in performance as the history length increases. This suggests that the models learn better as additional historical facts

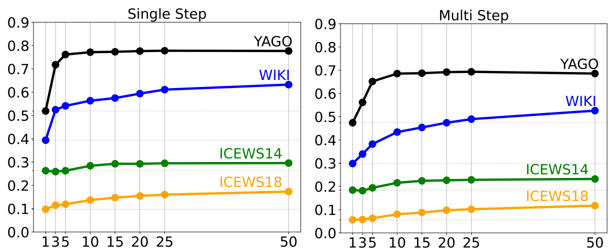


Figure 3.3: Performance (Hit@1) adheres to the scaling law based on the history length in ICL for temporal knowledge graph forecasting

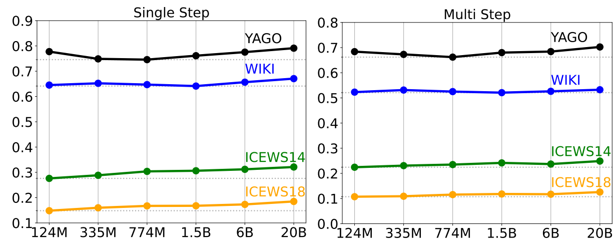


Figure 3.4: Performance (Hit@1) adheres to the scaling law based on the model size in ICL for temporal knowledge graph forecasting

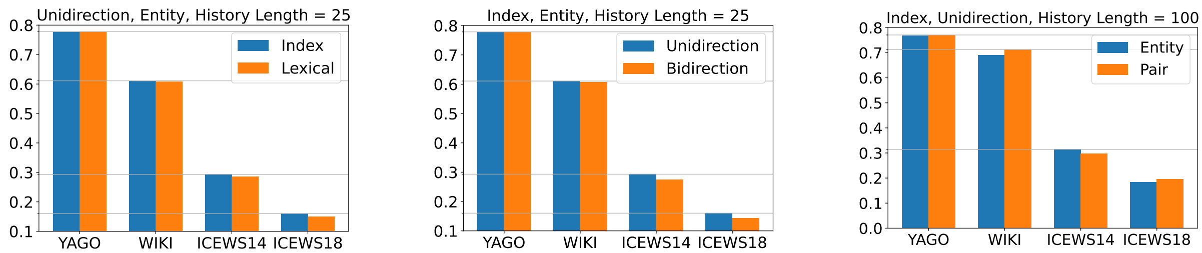
are presented. This observation is connected to few-shot learning in other domains, where performance improves as the number of examples per label increases. However, in our case, the historical patterns presented in the prompt do not explicitly depict the input-label mapping but rather aid in inferring the next step.

Q6: What is the relation between ICL's performance and model size? Here, we analyze the connection between model size and performance. Our results, as presented in Figure 3.4, conform to the expected trend of better performance with larger models. This finding aligns with prior works showing the scaling law of in-context learning performance. Our findings are still noteworthy since they show how scaling model size can facilitate more powerful pattern inference for forecasting tasks.

3.2.6 Ablation Study: Understanding Prompt Effects

To determine the most effective prompt variation, we run a set of experiments on all prompt variations, using GPT-J [227] and under the single-step setting.

Index vs. Lexical Our first analysis compares the performance of `index` and `lexical` prompts. This investigation aims to determine whether the model relies solely on input-label mappings or if it also incorporates semantic priors from pre-training to make predictions. Our results (Figure 3.5 (a)) show that the



(a) `Index` vs. `Lexical` (b) `Unidirection` vs. `Bidirection` (c) `Entity` vs. `Pair`

Figure 3.5: Performance (Hit@1) analysis on prompt variation in ICL for temporal knowledge graph forecasting.

performance is almost similar ($\pm 4e-3$ on average) across the datasets. This finding is aligned with previous studies indicating that foundation models depend more on input-label mappings and are minimally impacted by semantic priors [238].

Unidirectional vs. Bidirectional We next analyze how the relation direction in the history modeling impacts the performance. This analysis aims to ascertain whether including historical facts, where the query entity or pair appears in any position, can improve performance by offering a diverse array of historical facts. Our results (Figure 3.5 (b)) show that there is a slight decrease in performance when `Bidirectional` history is employed, with a significant drop in performance observed particularly in the ICEWS benchmarks. These observations may be attributed to the considerably more significant number of entities placed in both subject and object positions in ICEWS benchmarks than YAGO and WIKI benchmarks. This finding highlights the necessity of having robust constraints on the historical data for ICL to comprehend the existing pattern better.

Entity vs. Pair Finally, we examine the impact of the history retrieval query on performance. Our hypothesis posits that when the query is limited to a single entity, we can incorporate more diverse historical facts. Conversely, when the query is a pair, we can acquire a more focused set of historical facts related to the query. Our results (Figure 3.5 (c)) indicate that the performance of the model is dependent on the type of data being processed. Specifically, the WIKI and ICEWS18 benchmarks perform better when the query is focused on the entity, as a broader range of historical facts is available. In contrast, the ICEWS14 benchmark performs better when the query is focused on pairs, as the historical facts present a more focused pattern.

3.3 In-context Learning for Semantic Extrapolation

3.3.1 Introduction

This section explores **in-context learning** for *semantic extrapolation* by user simulation task, which the ability of large language models (LLMs) to simulate a user's communication style and behavior by conditioning on their past dialogue history. Unlike task-oriented settings, this focuses on modeling open-domain, long-term conversations where the goal is to maintain coherent, emotionally intelligent personas over time.

Achieving this ability is crucial for developing engaging chatbots that can remember previous interactions and respond with empathy, both of which are key aspects of emotional intelligence (EI) [70, 161]. However, evaluating large language models (LLMs) in this context is challenging. A key obstacle lies in collecting **real-world, human-to-human** conversations featuring the *same* individuals over extended periods, ensuring consistent personas—an important factor for passing the Turing test [221, 226]. To address this, recent efforts have turned to LLMs to simulate long-term dialogues between two human participants, resulting in synthetic conversation datasets [108, 94, 270, 53, 158]. Yet, questions remain about whether these simulated dialogues genuinely capture the nuances of real-world human interaction.

Here, we introduce REALTALK, a real-world, long-term dialogue dataset featuring pairs of individuals who initially connected through messaging apps. Over 21 days, 10 participants each engaged in two separate conversations with different partners, capturing evolving communication patterns. This resulted in a total of 10 unique conversations, each spanning approximately 21 sessions. These daily or near-daily interactions amounted to over 16,000 words per conversation (Section 3.3.2). The structure of these dialogues parallels the approach used in LoCoMo [158], where LLMs engage in extended conversations, exchanging small talk, and the occasional image. By collecting human dialogues of comparable length, REALTALK enables direct comparisons between real-world and LLM-simulated conversations.

Using our data, we analyze two key aspects of real-world long-term dialogues that distinguish them from LLM-simulated conversations.

First, we examine **emotional intelligence (EI)** by comparing authentic human conversations with LLM-generated dialogues using a dialogue-level EI assessment (Section 3.3.2.2 - 3.3.2.3). Our findings show that real conversations gradually build intimacy over time [5, 45], featuring a wide emotional range, whereas LLM-simulated chats often begin at a high intimacy level, predominantly maintain a positive tone, and display excessive empathy—even amid negative topics (See Figure 3.6). Next, we explore **persona consistency**, investigating whether individuals maintain a stable persona across multiple conversations and assessing how well models capture these shifts or continuities over time.

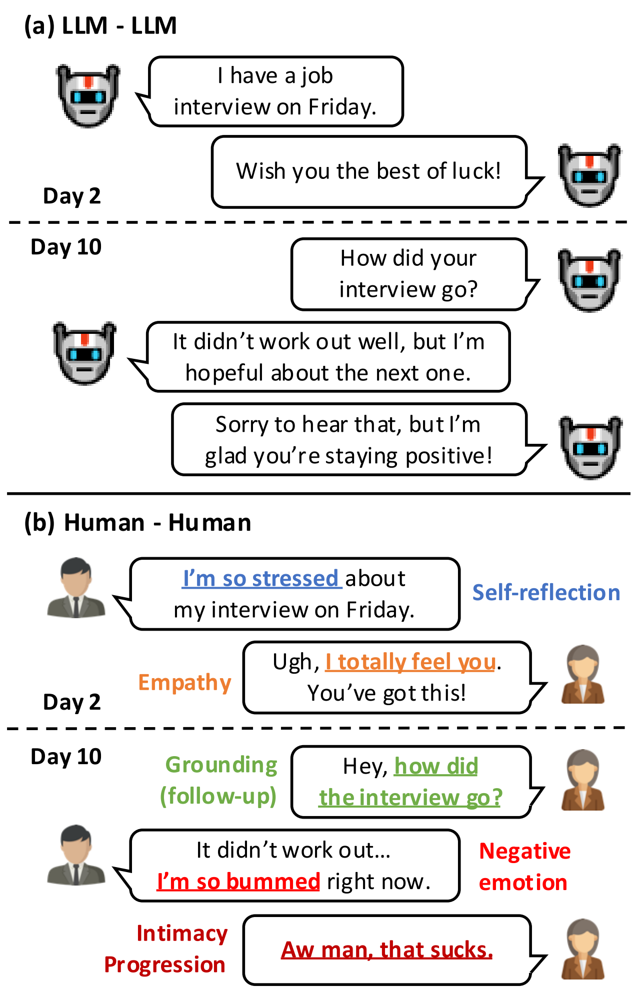


Figure 3.6: Comparison between LLM-simulated dialogues and real-world human dialogues.

Our analysis reveals that human conversational styles naturally fluctuate based on context, reflecting variations in emotional intelligence. In contrast, LLMs—despite being prompted with distinct personas—exhibit only minimal differences, struggling to adapt and replicate nuanced persona shifts over extended interactions.

Building on these insights, we introduce **persona simulation** benchmark to evaluate model performance in long-term dialogues. Maintaining a consistent persona is key to fostering trust, engagement, and personalization in AI interactions. This task assesses how well an LLM can simulate an individual's unique persona by continuing a conversation a dialogue given prior context. We evaluate the model's ability to reproduce a user's next message and match their EI attributes. Our experimental results show that models struggle to simulate a user solely from dialogue history. However, fine-tuning on a specific user's chat history improves persona emulation.

3.3.2 User Simulation: Dataset (REALTALK) and Evaluation Metrics

3.3.2.1 Dataset: REALTALK

To study real-world human dialogues, we first collect REALTALK, a dataset of 10 long-term conversations. We recruited 10 participants, pairing them to engage in conversations over 21 days. Once the conversations were collected, a separate group of annotators, distinct from the participants, labeled memory probing question-answer pairs for each conversation $\mathcal{C}$ to evaluate LLM memory retention capabilities. Additionally, they annotated speaker events for each session $\mathcal{S}_i \in \mathcal{C}$ to track contextual information over time. We provided specific guidelines to each participant, requiring them to send at least 50 messages to their conversation partner each day.

Participants. We recruited 10 participants who are native speakers from the US, aged between 18 and 25, with an approximately equal gender distribution. All participants provided signed release forms prior to the study.

Guidelines. We asked participants to engage in friendly, natural conversations that include small talk, personal stories, and occasional image sharing to create a realistic chat experience. Conversations should feel casual and relatable, with references to time and place (e.g., "last friday" or "when I was a kid") and should cover topics such as daily events, personal interests, and pop culture. Each session should feel like a conversation between friends, with participants sharing opinions and experiences. Moreover, we encouraged participants to thoughtfully integrate images into the conversation, sourcing them from the internet under a Creative Commons license and avoiding explicit descriptions of the image content. We also advised against using images with identifiable faces representing the participant directly and requested consistency if such images are reused across sessions.

Dataset	Dialogue Participants	# Turns / $\mathcal{C}$	# Session / $\mathcal{C}$	# Tokens / $\mathcal{C}$	Multimodal	Collection
MemoryBank [270]	Human-AI	3.7	10	257.8	✗	LLM-simulated
LongMemEval [243]	Human-AI	9.8	50.2	1,572.3	✗	LLM-simulated
SODA [108]	Human-Human	7.6	1	122.4	✗	LLM-simulated
Conversation Chronicles [95]	Human-Human	58.5	5	1,054.7	✗	LLM-simulated
LoCoMo [158]	Human-Human	588.2	27.2	13,377.2	✓	LLM-simulated
MPChat [3]	Human-Human	2.8	1	53.3	✓	Reddit
MMDialog [57]	Human-Human	4.6	1	72.5	✓	Social media
Daily Dialog [135]	Human-Human	7.9	1	114.7	✗	Crowdsourcing
MSC [249]	Human-Human	53.3	4	1,225.9	✗	Crowdsourcing
REALTALK	Human-Human	894.4	21.9	17,109.8	✓	Crowdsourcing

Table 3.6: Data statistics comparison between REALTALK and others in ICL for persona simulation

3.3.2.2 Evaluation Metrics: Emotional Intelligence

To systematically compare real-world and LLM-generated dialogues, and to evaluate persona consistency and simulation quality, we develop a comprehensive EI evaluation framework.

Problem formulation. Each conversation $\mathcal{C}$ consists of multiple sessions $\mathcal{S}_i$, such that $\mathcal{C} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$, where m is the total number of sessions in the conversation. Each session $\mathcal{S}_i$ consists of a sequence of messages exchanged between two speakers such that $\mathcal{S}_i = \{\mathcal{M}_{s_1,1}, \mathcal{M}_{s_2,2}, \ldots, \mathcal{M}_{s_n,n}\}$ where $\mathcal{M}_{s_j,j}$ represents the j-th message in the session, sent by speaker $s_j \in \{1, 2\}$. Unlike structured turn-taking, the order of messages can be variable, and a speaker may send consecutive messages (*e.g.,* multiple chat bubbles in a row), reflecting the natural flow of real-world conversations. To maintain coherence, we concatenate consecutive messages from the same speaker into a single message before analysis.

Message-level EI Attributes. We measure the following EI attributes for each individual message $\mathcal{M}$:

1. **Reflectiveness** ($R(\mathcal{M})$): A boolean indicator that captures whether a speaker explicitly recognizes and describes their own emotions, thoughts, or reactions. For example, statements like "I think I'm feeling this way because..." signal self-reflection. We use an LLM-based classification to label each turn as reflective or not.

2. **Grounding Act** ($G(\mathcal{M})$): A boolean indicator that captures whether a speaker uses acts—such as clarifying questions or follow-ups—that are essential for building common ground and preventing misunderstandings [34, 35, 205]. We use an LLM-based classification to label each turn as grounding or not.

3. **Emotion, Sentiment, and Intimacy** ($E(\mathcal{M})$, $S(\mathcal{M})$, $I(\mathcal{M})$): $E(\mathcal{M})$ and $S(\mathcal{M})$ represent the speaker's emotion and sentiment labels, respectively. Emotion is classified into 11 predefined categories (e.g., anger, fear, joy), while sentiment is classified into 3 categories (positive, negative, neutral). $I(\mathcal{M})$ is a floating-point value that indicates the speaker's level of intimacy. These attributes are measured using models trained for emotion, sentiment, and intimacy classification, leveraging annotated data from Twitter [7]. While LLMs could be used, fine-tuned models offer comparable or slightly better performance with significantly lower cost, making them more practical for large-scale evaluation [190].

4. **Empathy** ($EP(\mathcal{M})$): A floating-point score capturing the speaker's empathy. It includes three components: emotional reaction, interpretation, and exploration. Each is scored by the LLM on a

0–2 Likert scale using the EPITOME framework [209]. The final empathy score is the sum of these three components.

Speaker-level EI Evaluation. To assess a speaker's overall emotional intelligence (EI), we aggregate message-level EI attributes into five categories based on Goleman's EI framework [70]:

1. **Self-awareness.** This reflects a speaker's ability to recognize and articulate emotions and perspectives. It is measured using two metrics:

   (a) *Reflective frequency* measures how often a speaker engages in self-reflection:

$$\text{reflective\_frequency}(s) = \frac{\sum_{\mathcal{M} \in \mathcal{M}_s} R(\mathcal{M})}{|\mathcal{M}_s|}$$

   (b) *Emotion & Sentiment Diversity* captures the range of emotions or sentiments expressed by a speaker s, computed using entropy. Let L be the set of sentiment labels (positive, negative, neutral), and $p_s(x)$ the proportion of label $x \in L$ in the speaker's messages $\mathcal{M}_s$:

$$p_s(x) = \frac{\sum_{\mathcal{M} \in \mathcal{M}_s} 1(S(\mathcal{M}) = x)}{|\mathcal{M}_s|},$$

$$\text{sentiment\_diversity}(s) = -\sum_{x \in L} p_s(x) \cdot \log_2 p_s(x).$$

   Emotion diversity is calculated analogously by replacing $S(\mathcal{M})$ with $E(\mathcal{M})$.

2. **Motivation.** A speaker's engagement in maintaining the conversation, measured by *grounding frequency*—the proportion of messages with grounding acts:

$$\text{grounding\_frequency}(s) = \frac{\sum_{\mathcal{M} \in \mathcal{M}_s} G(\mathcal{M})}{|\mathcal{M}_s|}$$

3. **Social Skills.** A speaker's ability to foster trust and engagement, captured by *intimacy progression*, assuming intimacy naturally increases over time.

   The average intimacy per session $\mathcal{S}_i$ is:

$$\text{intimacy\_average}(s, \mathcal{S}_i) = \frac{\sum_{\mathcal{M} \in \mathcal{M}_s^{\mathcal{S}_i}} I(\mathcal{M})}{|\mathcal{M}_s^{\mathcal{S}_i}|}$$

   where $\mathcal{M}_s^{\mathcal{S}_i}$ is the set of messages from speaker s in session $\mathcal{S}_i$. Intimacy progression is modeled using:

$$\text{linear\_progression}(s) = a, \text{ where } y = a \cdot x + b$$

$$\text{exp\_progression}(s) = b, \text{ where } y = a \cdot e^{b \cdot x}$$

4. **Self-regulation.** This reflects a speaker's ability to maintain emotional and sentiment stability during interaction. It is evaluated using two metrics:

   (a) *Emotion & Sentiment Stability:*

$$\text{stability}(s) = \frac{\sum_{k=2}^{|\mathcal{M}_s|} 1(E(\mathcal{M}_s^{(k)}) = E(\mathcal{M}_s^{(k-1)}))}{|\mathcal{M}_s| - 1}$$

   The same formula applies to sentiment by replacing $E(\mathcal{M})$ with $S(\mathcal{M})$.

   (b) *Emotion & Sentiment Alignment:*

$$\text{alignment}(s) = \frac{\sum_k 1(E(\mathcal{M}_s^{(k)}) = E(\mathcal{M}_p^{(k)}))}{|\mathcal{M}_s|}$$

   where $\mathcal{M}_p^{(k)}$ is the corresponding partner message. Sentiment alignment is computed similarly.

5. **Empathy.** The average empathy score across a speaker's messages:

$$\text{empathy}(s) = \frac{\sum_{\mathcal{M}\in\mathcal{M}_s} EP(\mathcal{M})}{|\mathcal{M}_s|}$$

where $EP(\mathcal{M})$ is the empathy score for message $\mathcal{M}$.

3.3.2.3 Data Insights from REALTALK

This section analyzes the collected data, starting with general observations and statistics. Next, we compare our dataset with LLM-generated dialogues, focusing on speaker-level emotional intelligence (EI) metrics. Finally, we examine persona consistency in real-world conversations, leveraging the fact that each speaker participates in at least two conversations.

Data Statistics & General Observations The dataset comprises 10 participant pairs engaging in 16–21 days of conversation, with daily word counts ranging from 691 to 861. Participants share 23–46 images per pair and revisit topics 6–8 times, showcasing sustained and rich interactions. Three key characteristics distinguish real-world human dialogues from LLM-generated dialogues: First, Human dialogues are inherently noisy, featuring typos, abbreviations, acronyms, and slang (*e.g.*, "imo" for "in my opinion" and "dunno" for "don't know"), reflecting the informal and natural flow of communication. Second, human conversations exhibit noticeable gaps between messages unlike LLM-generated dialogues, reflecting asynchronous and flexible communication. On average, participants take 20.98 minutes to respond, with a median gap of 2.22 minutes. Some gaps extend up to 27.90 hours, highlighting the non-linear nature of real-world interactions. Third, human dialogues often include varying lengths of consecutive messages (*i.e.,* chat bubbles) by the same speaker, reflecting diverse communication patterns. On average, participants send 2.31 consecutive messages per turn, with a median of 2.00 and a maximum of 68, showcasing occasional extended monologues.

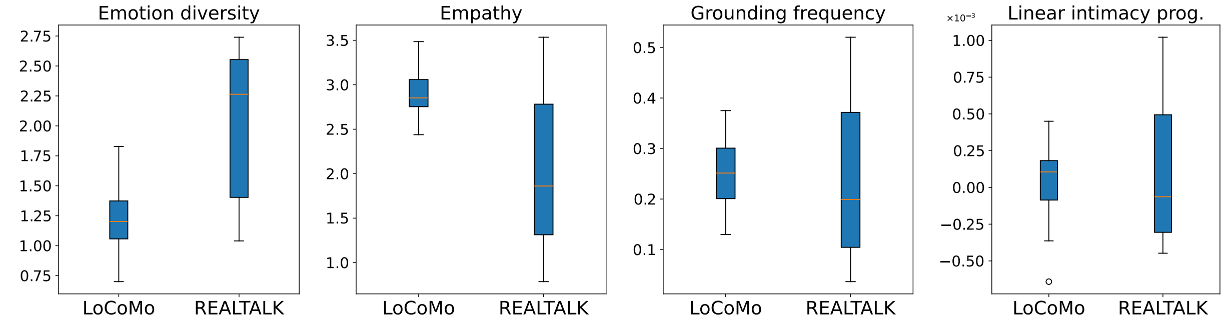


Figure 3.7: Emotional intelligence comparison between LLM-simulated dialogues LoCoMo [158] and human dialogues REALTALK.

vs. LLM-generated Dialogues We evaluate speaker-level EI in real-world and LLM-generated dialogues using the LoCoMo dataset [158]. Speaker-level EI is derived from message-level EI: reflectiveness, grounding acts, and empathy are computed using `gpt-4o-mini`, while sentiment, emotion, and intimacy are classified using task-specific fine-tuned RoBERTa models trained on labeled Twitter data [7]. Each speaker in each conversation receives an independent EI score. Figure 3.7 illustrates key differences between human (REALTALK) and LLM-generated (LoCoMo) dialogues by showing the distribution of speaker EI scores: (1) Humans exhibit greater emotion and sentiment diversity, while LLMs remain constrained; (2) LLMs display excessive empathy; and (3) Humans show high variance in EI attributes, whereas LLMs are more uniform, aligning with prior research [119].

High emotional intelligence—characterized by strong grounding, emotional alignment, and intimacy progression—enhances engagement and relationship-building [5, 83, 45, 88, 173]. However, since human EI naturally varies across individuals and interactions, enforcing uniformly high EI in LLMs does not make them more human-like. Instead, tailoring LLMs to specific individuals better reflects the diverse dynamics of real human interactions.

Persona Consistency Analysis We examine how speakers adapt their emotional intelligence (EI) based on their conversational partners. Figure 3.8 shows absolute differences in EI attributes per participant,

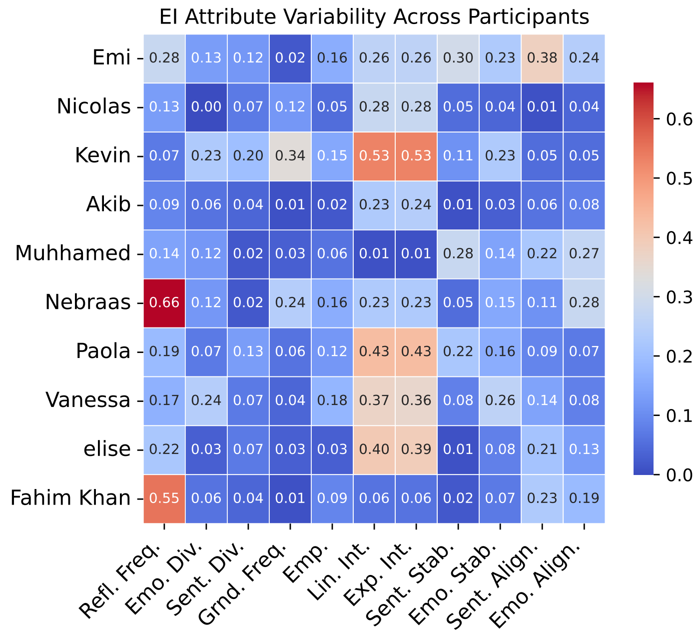


Figure 3.8: Persona consistency of individual speaker across two conversations, capturing how their emotional intelligence varies depending on their conversational partners.

capturing persona variability across interactions: (1) Some participants maintain a stable persona, while others dynamically adapt their EI; and (2) Intimacy progression shows the most variation, suggesting that rapport-building is highly influenced by the specific conversational partner. These findings indicate that a speaker's persona is flexible and adjusts according to social dynamics, reflecting adaptive emotional behavior.

3.3.3 In-context Learning for User Simulation

We simulate a speaker's next message $\hat{\mathcal{M}}_t$ based on prior conversation history $\mathcal{H}_t = \{\mathcal{M}_1, ..., \mathcal{M}_{t-1}\}$ and compare it to the ground truth $\mathcal{M}_t$. The input consists of a system and user prompt, where the system defines the user's persona and task, while the user prompt provides the dialogue history. The output serves as ground truth for evaluation.

System:

You are `{opponent_speaker}`. Continue the conversation.

Output only the message, not the speaker name.

User:

`{previous conversation}`

`{speaker}`

Assistant:

`{speaker's original message}`

3.3.4 Experimental Setup

Each speaker participates in two conversations ($\mathcal{C}_a$ and $\mathcal{C}_b$), allowing us to evaluate two baselines: (1) **w/o fine-tuning**: A generic LLM tested on $\mathcal{C}_b$; and (2) **w/ fine-tuning**: An LLM fine-tuned on $\mathcal{C}_a$ and tested on $\mathcal{C}_b$. We choose the conversation with lower overall EI as $\mathcal{C}_b$. During both training and testing, we use the prompt "*You are {speaker name}. Continue the conversation.*" along with the previous conversation history. The speaker's original message serves as the ground truth for this prompt. To measure how effectively the model simulates the speaker, we compare message-level EI attributes of predicted and ground truth messages using (1) accuracy for categorical and boolean attributes (*i.e.,* reflectiveness, grounding act, sentiment, and emotion); and (2) absolute difference for continuous attributes (*i.e.,* intimacy, empathy). Lower absolute difference and higher accuracy indicate stronger simulation accuracy. Additionally, we compare lexical and semantic similarity between $\hat{\mathcal{M}}_t$ and $\mathcal{M}_t$, using ROUGE [140] and BERTScore [267].

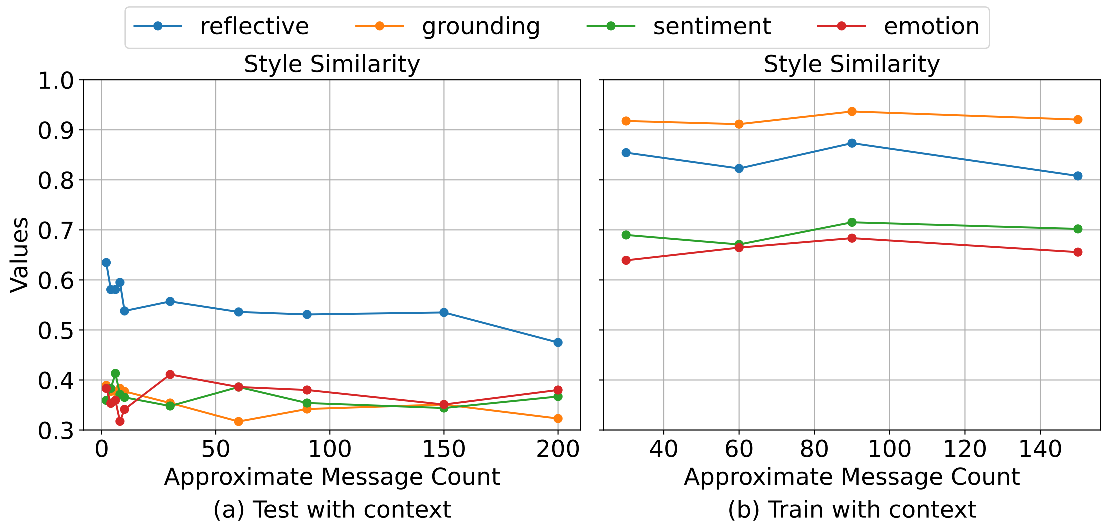


Figure 3.9: Impact of context in user simulation

3.3.5 Experimental Results

We analyze the role of conversational context in in-context persona simulation by varying the amount of dialogue history provided to the model. If performance remains unchanged despite additional context, it suggests that the model relies primarily on its parametric knowledge, rather than adapting to the speaker's style from context. As shown in left figure in Figure 3.9, increasing the length of prior conversation does not consistently improve message-level EI or reveal meaningful trends. These results indicate that LLMs struggle to simulate user personas based solely on in-context dialogue history.

3.3.6 Ablation Study: Towards Better User Simulation through Fine-tuning

We explore whether fine-tuning improves an LLM's ability to simulate a speaker's responses given conversational context. A potential approach is to train on all C_a conversations and evaluate on C_b. However, this risks data leakage, as a speaker in C_a may appear in C_b. To ensure fairness, we train and test each speaker separately. First, we train models using different amounts of conversation history as context and test them with the same number of sessions used in training. However, we observe no clear pattern indicating that increasing the amount of conversational context improves performance (See right figure in

	Content Similarity		Message-level EI (Emotional Intelligence)					
	Lexical ↑	Semantic ↑	Reflective ↑	Grounding ↑	Sentiment ↑	Emotion ↑	Intimacy ↓	Empathy ↓
w/o fine-tune	**0.14 ± 0.04**	0.76 ± 0.08	0.62 ± 0.13	0.40 ± 0.13	0.53 ± 0.22	0.43 ± 0.22	**0.06 ± 0.01**	1.80 ± 0.55
w/ fine-tune	**0.14 ± 0.05**	**0.78 ± 0.04**	**0.77 ± 0.09**	**0.62 ± 0.08**	**0.59 ± 0.18**	**0.46 ± 0.21**	0.07 ± 0.01	**1.24 ± 0.12**

Table 3.7: Performance comparison between with and without fine-tuning in ICL for user simulation

Figure 3.9. The results suggest that performance saturates after three sessions, implying that additional context beyond this point does not provide further benefits.

However, when comparing a fine-tuned model trained with three sessions of context to a non-fine-tuned version, we find that the model does learn the speaker's unique style. Table 3.7 presents the average performance across all speakers. Here, content similarity differences are not statistically significant, while message-level EI differences are significant ($p < 0.02$ for all attributes).

The findings highlight three key insights: (1) Fine-tuning effectively captures a speaker's style, including emotion, sentiment, reflectiveness, grounding, and empathy; (2) While stylistic adaptation improves, content similarity (lexical and semantic) remains largely unaffected; (3) Fine-tuning does not enhance intimacy, as intimacy emerges from mutual interaction rather than a single speaker's style.

Overall, the results suggest that while the model does not learn how to use conversational context to improve its simulation, it can learn the stylistic patterns of a speaker when trained exclusively on their messages.

3.4 Summary

This section explored the capability of in-context learning (ICL) in LLMs by moving beyond traditional few-shot task learning. We focused on extrapolation tasks where the context involves temporal or social regularities, rather than explicit input-label exemplars.

Our findings show that ICL is highly effective for **structural pattern extrapolation**, where models can generalize from statistical regularities in the input without semantic grounding, such as predicting future events from timestamped sequences. This demonstrates that LLMs possess strong capabilities for surface-level pattern recognition when sufficient context is provided.

However, ICL struggles with **semantic extrapolation** tasks, where meaningful understanding of the context is required—such as simulating a speaker's persona based on prior dialogue. In these cases, increasing contextual information does not consistently improve performance, showing the limits of in-context generalization when the task requires deeper social or emotional reasoning.

These results suggest that while LLMs are strong pattern matchers, they often fail to internalize and adapt to rich contextual signals in the absence of explicit supervision. Bridging this gap may require new modeling strategies that integrate contextual memory, user-specific adaptation, or grounded semantic representations.

Chapter 4

Contextual Supervision for Language Model Training

This chapter investigates whether explicitly incorporating context during training improves language model performance and behavior. While most prior work emphasizes using context at inference time (*e.g.*, in-context learning), we hypothesize that training models with contextual signals, such as task examples, dialogue history, or human explanations, can enhance generalization, robustness, and label efficiency. We explore this hypothesis through two forms of contextual supervision: (1) appending auxiliary context to model inputs during training; and (2) using human-provided explanations as training signals. Each section presents a specific mechanism and supporting experiments.

4.1 Background and Motivation

4.1.1 Explicitly Training Language Models with Context Improves Performance

A straightforward yet effective approach to enhancing language model performance is to explicitly incorporate contextual information during training. This involves appending auxiliary context, such as task examples or dialogue history, to the model input both during training and inference.

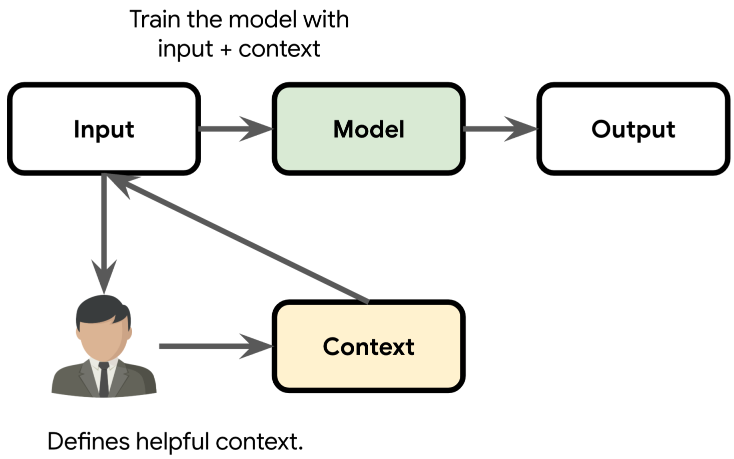


Figure 4.1: Experiment setting for training models with context

Formally, let $\mathbf{x}$ be the input instance and $\mathbf{c}$ be the contextual information (e.g., surrounding dialogue, prior task examples). The model is trained on sequences of the form $(\mathbf{c}, \mathbf{x}) \to \mathbf{y}$ and evaluated the correctness:

$$\mathbf{y} \sim \mathcal{P}_{\mathrm{LM}}(\mathbf{y} \mid \mathbf{c}, \mathbf{x})$$

Empirically, we observe that this simple form of contextual supervision consistently improves model performance across tasks. For instance, in named entity recognition (NER) and hate speech detection, models trained with additional contextual input achieve higher accuracy and robustness, validating the utility of context as a form of supervision.

4.1.2 Human Explanations as Context for Learning

Beyond task-specific context, a more nuanced and powerful form of contextual supervision comes from human-provided explanations. These explanations can serve as weak or indirect supervision signals to improve model generalization and enable behavior refinement.

We explore two key benefits of explanation-based contextual training:

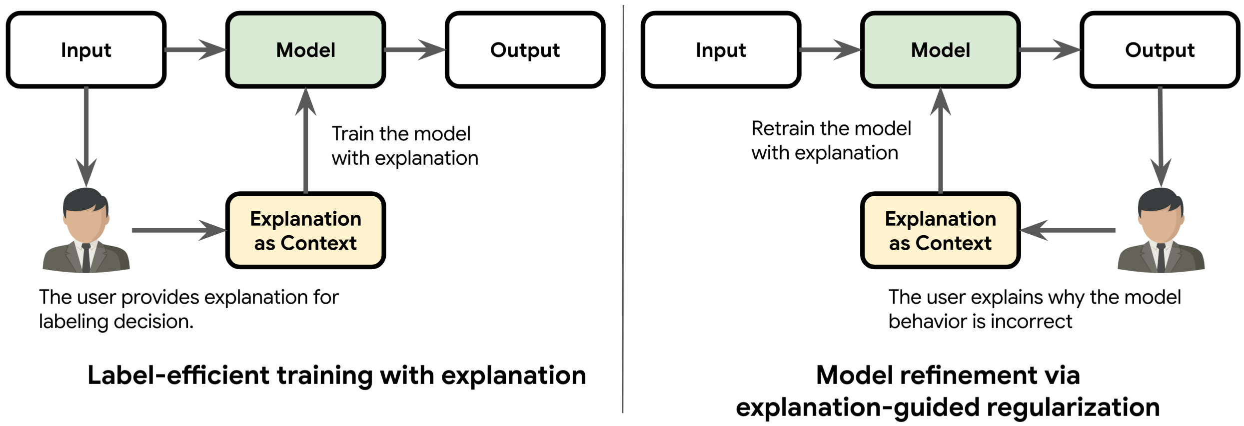


Figure 4.2: Experiment setting for each study in human explanations as context for learning

- **Label-efficient learning with human-provided labeling explanation:** Explanations help models understand the rationale behind a label, enabling generalization from fewer examples. This allows the model to learn more efficiently and perform better on unseen instances [138, 124].

- **Model refinement via explanation-guided regularization:** Explanations can also be used to adjust model behavior post hoc. When explanations highlight flawed reasoning in the model's outputs, they can be used to update the model through targeted regularization, improving alignment with human expectations [122].

This line of work demonstrates that explanations are not just useful at inference time for interpretability, but also at training time as a rich contextual signal for improving performance and controllability.

4.2 Training Language Models with Context

This section explores how incorporating contextual information into the input during training can improve model performance. We examine two settings: named entity recognition with task examples as contextual prompts (Section 4.2.1), and social norm violation detection using surrounding dialogue history as context (Section 4.2.2).

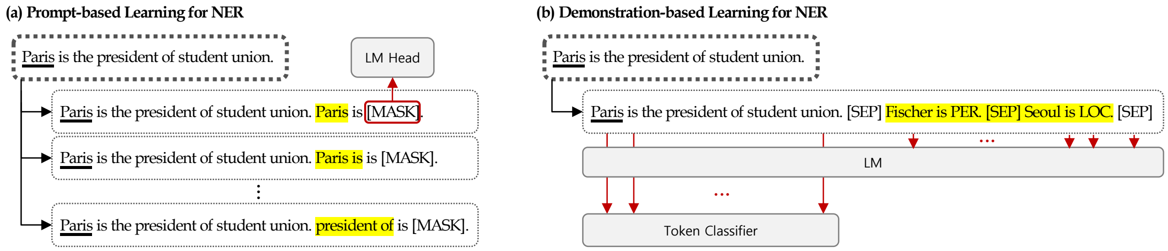


Figure 4.3: Overview of training for named entity recognition using in-context task examples

4.2.1 Training for Named Entity Recognition Using In-Context Task Examples

4.2.1.1 Introduction

Neural sequence models have become the *de facto* approach for named entity recognition (NER) and have achieved state-of-the-art results on various NER benchmarks [114, 155, 147]. However, these data-hungry models often rely on large amounts of labeled data manually annotated by human experts, which are expensive and slow to collect [87, 50], especially for specialized domains (*e.g.*, research papers). To improve NER performance on low-resource (label scarcity) settings, prior works seek auxiliary supervisions, such as entity dictionary [181, 206, 250, 150] and labeling rules [197, 97], to either augment human-labeled data with pseudo-labeled data, or incorporate meta information such as explanation [138, 124, 126], context [230], and prompts [49, 37] to facilitate training. However, such methods have the following challenges: (1) human efforts to create auxiliary supervisions (e.g., dictionaries, rules, and explanations); (2) high computational cost to make predictions. For example, [49] shows effectiveness on entity type prediction given the entity span by constructing a prompt with the structure *"[entity span] is [MASK]"*. However, when the entity span is not given, cloze-style prompts need to be constructed over all the entity candidates in the sentence with the structure *"[entity candidate] is [MASK]"* to make a prediction [37]. Such brute-force enumerations are often expensive.

In this paper, we propose *demonstration-based learning* [60, 149], a simple-yet-effective way to incorporate automatically constructed auxiliary supervision. The idea was originally proposed in prompt-based learning to show some task examples before the cloze-style template so that the model can better understand and predict the masked slot [60]. This paper proposes modified version of demonstration-based learning for NER task. Instead of reformatting the NER task into the cloze-style template, we augment the original input instances by appending automatically created task demonstrations and feed them into pre-trained language models (PTLMs) so that the model can output improved token representations by better understandings of the tasks. Unlike existing efforts which require additional human labor to create such auxiliary supervisions, our model can be automatically constructed by picking up proper task examples from the train data. Moreover, unlike approaches that need to change the format of token classification into cloze-style mask-filling prediction which can neglect latent relationships among token labels, our approach can be applied to existing token classification module in a plug-and-play manner (See Figure 4.3 (a) vs (b)).

We investigate the effectiveness of task demonstration in two different low-resource settings: (1) in-domain setting which is a standard NER benchmark settings where the train and test dataset come from the same domain; and (2) domain-adaptation setting which uses sufficient labeled data in source domain to solve new tasks in a target domain. Here, we study which variants of task demonstration are useful to train an accurate and label-efficient NER model and further explore ways to adapt the source model to target domain with a small amount of target data. We propose two ways of automatic task demonstration construction: (1) *entity-oriented demonstration* selects an entity example per entity type from train data to construct the demonstration. It allows the model to get a better sense of entity type by showing its entity example; and (2) *instance-oriented demonstration* retrieves instance example similar to input sentence in train data. It allows the model to get a better sense of the task by showing similar instances and their entities.

We show extensive experimental results on CoNLL03, Ontonotes 5.0 (generic domain), and BC5CDR (biomedical domain) over 3 different templates and 5 selection/retrieval strategies for task demonstrations.

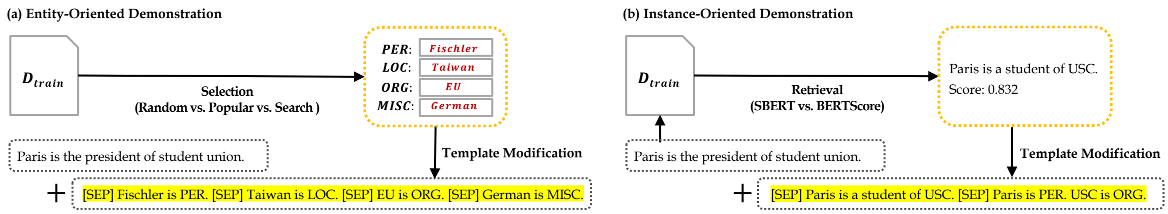


Figure 4.4: Task examples as context for training named entity recognition

For *entity-oriented demonstration*, we present 3 selection strategies to choose appropriate entity example per entity type: (1) `random` randomly selects entity example per entity type; (2) `popular` selects the entity example which occurs the most per entity type in the train data; and (3) `search` selects the entity example per entity type that shows the best performance in the development set. And for *instance-oriented demonstration*, we present 2 retrieval strategies to choose appropriate instance example (`SBERT` [192] vs. `BERTScore` [263]).

Our findings include: (1) good demonstration can save many labeled examples to reach a similar level of performance in low-resource settings. Our approach consistently outperforms standard fine-tuning by up to 3 points in terms of F1 score (p-value < 0.02); (2) demonstration becomes more effective when we also provide context. For example, not only showing 'Fischler is PER', but also the sentence that contains 'Fischler' as person, such as 'France backed Fischler's proposal'; and (3) consistency in demonstration contributes to better performance. Our experiments show that using consistent demonstration for all instances rather than varying per instance lead to better performance

4.2.1.2 Named Entity Recognition: Problem Formulation

In this section, we introduce basic concepts of named entity recognition, standard fine-tuning for sequence labeling, and domain adaptation for sequence labeling. We then formally introduce our goal – generating task demonstration and then developing a learning framework that uses them to improve NER models.

Named Entity Recognition Here, we let $\mathbf{x} = [x^{(1)}, x^{(2)}, \ldots x^{(n)}]$ denote the sentence composed of a sequence of n words and $\mathbf{y} = [y^{(1)}, y^{(2)}, \ldots y^{(n)}]$ denote the sequence of NER tags. The task is to predict the entity tag $y^{(i)} \in \mathcal{Y}$ for each word $x^{(i)}$, where $\mathcal{Y}$ is a pre-defined set of tags such as {B-PER, I-PER, …, O}. In *standard fine-tuning*, NER model $\mathcal{M}$ parameterized by θ is trained to minimize the cross entropy loss over token representations $\mathbf{h} = [h^{(1)}, h^{(2)}, \ldots h^{(n)}]$ which are generated from the pre-trained contextualized embedder as follows:

$$\mathcal{L} = -\sum_{i=1}^{n} \log f_{i,y_i}(\mathbf{h}; \boldsymbol{\theta}) \tag{4.1}$$

where f is the model's predicted conditional probability that can be either from linear or CRF layers.

In-domain Low-resource Learning We let $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ denote the labeled train and test dataset, respectively, consisting of $\{(\mathbf{x_i}, \mathbf{y_i})\}$. Here, we expect the number of labeled instances in $\mathcal{D}_{\text{train}}$ is extremely limited (e.g., $N < 50$). Given such small labeled instances, our goal is to train an accurate NER model with task demonstrations compared to standard fine-tuning and show the effectiveness of demonstration-based learning. We evaluate the trained models on $\mathcal{D}_{\text{test}}$.

Low-resource Domain Adaption Domain adaptation aims to exploit the abundant data of well-studied source domains to improve the performance in target domains of interest. We consider two different settings: (1) *label-sharing* setting in which the label space $\mathbf{L} = \{l_1, \ldots, l_{|L|}\}$ (e.g., $l_i = PERSON$) of source-domain data $\mathcal{S}$ and target-domain data $\mathcal{T}$ are equal; (2) *label-different* setting which $\mathbf{L}$ is different.

In domain adaptation, we first train a model $\mathcal{M}_s$ on source-domain data $\mathcal{S}$. Next, we initialize the weights of the new model $\mathcal{M}_t$ by weights of $\mathcal{M}_s$. Here, we can either transfer the whole model weights or only the weights of contextualized embedder from $\mathcal{M}_s$ to $\mathcal{M}_t$. Then, we further tune $\mathcal{M}_t$ on target-domain data $\mathcal{T}$. In our preliminary experiments, we find that transferring only the embedder from $\mathcal{M}_s$ to $\mathcal{M}_t$ is much more effective than transferring the whole model weights (See first rows in Table 4.2 and Table 4.3). For this paper, we focus on the effectiveness of our models to adapt to the target domain with a $\mathcal{T}$, for which the number of instances is extremely limited. We then compare the results of tasks with demonstration to those without demonstration.

4.2.1.3 Demonstration-based NER

In this work, we focus on how to create effective task demonstration $\tilde{\mathbf{x}}$ to elicit better token representations for $\mathbf{x}$, and then we propose an efficient learning framework that can be improved by the effect of $[\mathbf{x}; \tilde{\mathbf{x}}]$. This section introduces the concepts of *demonstration-based learning*, and provides details of the approach. Here, we study example sampling strategies and templates to construct the demonstration and how we can train the NER model with the demonstration.

Task Demonstration Task demonstration $\tilde{\mathbf{x}} = [\textit{[SEP]}; \hat{\mathbf{x}}_1; \cdots; \hat{\mathbf{x}}_l]$ is constructed by selecting entity example e or retrieving instance example s from $\mathcal{D}_{\text{train}}$ ($\mathcal{T}_{\text{train}}$ for domain adaptation) and modifying by template T to form $\hat{\mathbf{x}}_i$. The demonstration sequence $\tilde{\mathbf{x}}$ is then appended to the original input $\mathbf{x}$ to create a demonstration-augmented input $[\mathbf{x}; \tilde{\mathbf{x}}]$. Here, *[SEP]* in front of $\tilde{\mathbf{x}}$ is to separate $\mathbf{x}$ and $\tilde{\mathbf{x}}$. The key challenge of constructing task demonstration is to choose appropriate e or s and template T that can be helpful to demonstrate how the model should solve the task. As shown in Figure 4.4, we categorize the demonstration into (1) *entity-oriented demonstration*; and (2) *instance-oriented demonstration* by whether we choose e or s respectively, for demonstration.

1. **Entity-oriented demonstration.** Given an entity type label set $\mathbf{L} = \left\{l_1, \ldots, l_{|L|}\right\}$, we select an entity example e per label l from $\mathcal{D}_{\text{train}}$. Then, we modify it using template T. To select e per each l, we first enumerate all the $e \in \mathcal{D}_{\text{train}}$ and create a mapping $\{l_i : [e_1, \ldots, e_n] \mid l_i \in \mathbf{L}\}$ between l and corresponding list of entities. Then for each label l, we select e by three selection strategies: (1) `random` randomly chooses e from the list; (2) `popular` chooses e that occurs the most frequently in the list; and (3) `search` conducts grid search over possible entity candidates per label. Here, we

sample top-k frequent entities per label, and search over combinations of entity candidates ($= k^{|L|}$). We find the best combination that maximizes the F1 score on the dev set $\mathcal{D}_{\text{dev}}$. Here, $\tilde{\mathbf{x}}_\mathbf{i}$ for every $\mathbf{x}_\mathbf{i}$ is different in `random` while $\tilde{\mathbf{x}}_\mathbf{i}$ for every $\mathbf{x}_\mathbf{i}$ is same in `popular` and `search`.

2. **Instance-oriented demonstration.** Given an input $\mathbf{x}$, we retrieve an instance example s that is the most relevant to the input from $\mathcal{D}_{\text{train}}$. Then, we modify the s along with its $\{e, l\} \in s$ by template T. For retrieval, we present two strategies: (1) `SBERT` [192] retrieves semantically similar sentence using pre-trained bi-encoder. It produces CLS embeddings independently for an input $\mathbf{x}$ and $s \in \mathcal{D}_{\text{train}}$, and compute the cosine similarity between them to rank $s \in \mathcal{D}_{\text{train}}$; (2) `BERTScore` [263], which is originally used as a text generation metric, retrieves token-level semantically similar sentence by computing a sum of cosine similarity between token representations of two sentences. Since the NER task aims to token classification, sentence-level similarity may retrieve a sentence that is semantically relevant but has no relevant entities.

3. **Fixed vs Variable demonstration.** As described in previous sections, the demonstration in some strategies varies per instance while in others it stays fixed globally. We can divide the demonstration strategies into two categories: (1) Variable demonstration: `random`, `SBERT`, `BERTScore` (2) Fixed demonstration: `popular`, `search`

4. **Demonstration template.** As shown in Figure 4.5, we select three variants of template T:

   (1) `no-context` shows selected e per l with a simple template "e *is* l.", without including the specific sentence where the entities show up. Between each pair of (e, l) (of different entity labels l), we concatenate with separator *[SEP]*. This template is only applied to the entity-oriented demonstration.

   (2) `context` in entity-oriented demonstration shows selected e per l along with an instance sentence s that contains e as a type of l. For each triple of (e, l, s), it is modified into "s. e *is* l." and concatenated

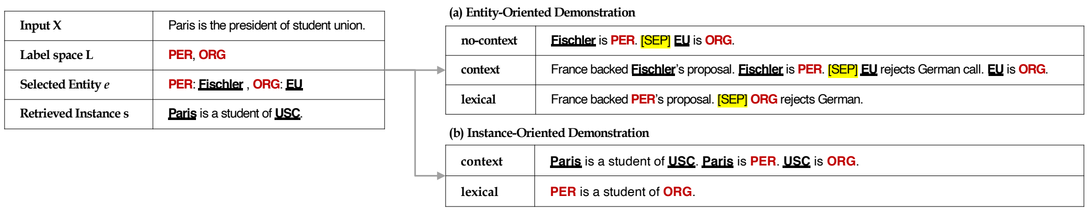


Figure 4.5: Task example template for training named entity recognition

with *[SEP]*. For instance-oriented demonstration, it shows the retrieved instance s along with all the entities mentioned in the sentence $e \in s$. It is modified into "s. e_1 *is* l_1. $\ldots e_n$ *is* l_n.".

(3) `lexical` in entity-oriented demonstration also shows selected e per l along with an instance sentence s. But here we only show s, which the entity span e is replaced by its label string l. For instance-oriented demonstration, we show retrieved s by replacing $e \in s$ with the corresponding l. We expect such templates can form labeling rules and let the model know how to label the sentence.

Model Training with Demonstration Transformer-based standard fine-tuning for NER first feeds the input sentence $\mathbf{x}$ into a transfomer-based PTLMs to get the token representations $\mathbf{h}$. The token representations $\mathbf{h}$ are fed into a CRF layer to get the conditional probability $p_\theta(\mathbf{y} \mid \mathbf{h})$, and the model is trained by minimizing the conditional probability by cross entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{n} \log p_\theta(\mathbf{y} \mid \mathbf{h}) \tag{4.2}$$

In our approach, we define a neural network parameterized by θ that learns from a concatenated input $[\mathbf{x}; \tilde{\mathbf{x}}]$. For both model training and inference, we feed the input and retrieve the representations:

$$[\mathbf{h}; \tilde{\mathbf{h}}] = [h^{(1)}, \ldots h^{(n)}, \tilde{h}^{(1)}, \ldots \tilde{h}^{(n)}] = \text{embed}([\mathbf{x}; \tilde{\mathbf{x}}]) \tag{4.3}$$

Dataset	Label	Train Data	
		25	50
CoNLL03	PER (Person)	$16.0_{\pm 3.52}$	$29.2_{\pm 4.52}$
	LOC (Location)	$15.6_{\pm 3.92}$	$30.4_{\pm 4.07}$
	ORG (Organization)	$21.8_{\pm 2.31}$	$32.6_{\pm 3.77}$
	MISC (Miscellaneous)	$11.0_{\pm 2.52}$	$15.6_{\pm 2.33}$
Ontonotes 5.0	PER (Person)	$10.8_{\pm 2.22}$	$21.4_{\pm 4.02}$
	LOC (Location)	$16.0_{\pm 3.52}$	$25.0_{\pm 7.32}$
	ORG (Organization)	$13.8_{\pm 3.48}$	$24.2_{\pm 6.17}$
	MISC (Miscellaneous)	$23.8_{\pm 5.56}$	$62.6_{\pm 7.93}$
BC5CDR	Disease	$25.8_{\pm 6.01}$	$29.2_{\pm 4.52}$
	Chemical	$51.0_{\pm 7.49}$	$65.8_{\pm 7.12}$

Table 4.1: Data statistics of named entity recognition task

As shown in Figure 4.3, we then feed **h** into the CRF layer to get predictions and train by minimizing the conditional probability $p_\theta(\mathbf{y} \mid \mathbf{h})$ as Equation 4.2.

For domain adaptation, we first train $\mathcal{M}_s$ with standard fine-tuning. Then, transfer the weights of embedder of $\mathcal{M}_s$ to $\mathcal{M}_t$ and further fine-tune $\mathcal{M}_t$ with our approach.

4.2.1.4 Experimental Setup

Datasets We consider three NER datasets as target tasks. We consider two datasets for a general domain (**CoNLL03** [219], **Ontonotes 5.0** [239]) and one dataset for a bio-medical domain (**BC5CDR** [134]). **CoNLL03** is a general domain NER dataset that has 22K sentences containing four types of general named entities: LOCATION, PERSON, ORGANIZATION, and MISCELLANEOUS entities that do not belong in any of the three categories. **Ontonotes 5.0** is a corpus that has roughly 1.7M words along with integrated annotations of multiple layers of syntactic, semantic, and discourse in the text. Named entities in this corpus were tagged with a set of general 18 well-defined proper named entity types. We split the data following [185]. **BC5CDR** has 1,500 articles containing 15,935 CHEMICAL and 12,852 DISEASE mentions.

Baselines To show its effectiveness in few-shot NER, we also show baselines of few-shot NER methods NNShot and StructShot [251]. NNshot is simple token-level nearest neighbor classification system while StructShot extends NNshot with a decoding process using abstract tag transition distribution. Here, both the classification model and the transition distribution should be pre-trained on the source dataset. Thus, we consider this as domain adaptation setting.

Experiments and Implementation Details We implement all the baselines and our frameworks using PyTorch [179] and HuggingFace [241]. We set the batch size and learning rate to 4 and 2e-5, respectively, and use `bert-base-cased` model for all the experiments. For each variant, we run 50 epochs over 5 different sub-samples and 3 random seeds with early-stopping 20 and show its average and standard deviation of F1 scores. Unlike existing sampling methods for few-shot NER [251], in which the training sample refers to one entity span in a sentence, we consider a real-world setting that humans annotate a sentence. We sub-sample data-points by random sampling with a constraint that sampled instances should cover all the BIOES labels [28] in the whole dataset. For Ontonotes, we aggregate all other entity types rather than person, location, and organization into miscellaneous to set the *label sharing* setting for domain adaptation experiments. Table 4.1 presents statistics of average number of entities per entity type over 5 different sub-samples.

4.2.1.5 Experimental Results

We first compare the overall performance of all baseline models and our proposed framework with the amount of training data 25 and 50 to show the impact of our approach in a low-resource scenario, assuming a task that needs to be annotated from scratch. Then, we show performance analysis to show the effectiveness of our approach and whether the model really learns from the demonstration.

Demonstration / Method	Strategy	Template	CoNLL03		Ontonotes 5.0		BC5CDR	
			25	50	25	50	25	50
BERT+CRF w/o demonstration	-	-	52.72 $_{\pm 2.44}$	62.75 $_{\pm 0.98}$	38.97 $_{\pm 4.62}$	54.51 $_{\pm 3.27}$	52.56 $_{\pm 0.46}$	60.20 $_{\pm 2.01}$
BERT+CRF w/ Instance-oriented demonstration	`SBERT` (`variable`)	`lexical`	48.92 $_{\pm 2.81}$	57.68 $_{\pm 0.37}$	36.58 $_{\pm 4.61}$	44.47 $_{\pm 2.58}$	49.41 $_{\pm 0.94}$	51.98 $_{\pm 2.14}$
		`context`	53.62 $_{\pm 1.64}$	64.21 $_{\pm 1.87}$	42.18 $_{\pm 5.21}$	53.07 $_{\pm 3.46}$	54.71 $_{\pm 2.09}$	59.78 $_{\pm 1.47}$
	`BERTScore` (`variable`)	`lexical`	49.55 $_{\pm 3.18}$	58.85 $_{\pm 1.06}$	35.42 $_{\pm 3.88}$	44.70 $_{\pm 2.41}$	49.37 $_{\pm 0.19}$	51.61 $_{\pm 2.45}$
		`context`	53.97 $_{\pm 1.52}$	64.66 $_{\pm 2.04}$	37.56 $_{\pm 5.29}$	53.13 $_{\pm 3.22}$	54.81 $_{\pm 2.11}$	59.63 $_{\pm 1.94}$
BERT+CRF w/ Entity-oriented demonstration	`random` (`variable`)	`no-context`	53.95 $_{\pm 1.89}$	63.31 $_{\pm 2.14}$	42.25 $_{\pm 3.61}$	55.71 $_{\pm 3.82}$	53.58 $_{\pm 0.48}$	59.97 $_{\pm 1.89}$
		`lexical`	55.20 $_{\pm 2.24}$	63.60 $_{\pm 2.32}$	44.02 $_{\pm 4.73}$	56.31 $_{\pm 3.83}$	53.79 $_{\pm 0.61}$	59.65 $_{\pm 1.71}$
		`context`	54.84 $_{\pm 2.12}$	63.51 $_{\pm 2.83}$	43.57 $_{\pm 3.73}$	56.76 $_{\pm 3.69}$	54.08 $_{\pm 0.97}$	59.94 $_{\pm 1.70}$
	`popular` (`fixed`)	`no-context`	54.34 $_{\pm 3.33}$	64.30 $_{\pm 2.76}$	43.02 $_{\pm 4.33}$	56.65 $_{\pm 3.35}$	53.86 $_{\pm 0.86}$	60.51 $_{\pm 1.77}$
		`lexical`	56.22 $_{\pm 3.88}$	64.95 $_{\pm 2.04}$	45.31 $_{\pm 5.02}$	58.24 $_{\pm 3.17}$	54.14 $_{\pm 0.67}$	60.67 $_{\pm 1.58}$
		`context`	56.52 $_{\pm 3.34}$	64.47 $_{\pm 2.35}$	45.52 $_{\pm 4.69}$	58.40 $_{\pm 3.24}$	54.31 $_{\pm 0.80}$	61.31 $_{\pm 1.51}$
	`search` (`fixed`)	`no-context`	54.63 $_{\pm 2.12}$	64.50 $_{\pm 2.76}$	42.88 $_{\pm 5.41}$	56.96 $_{\pm 4.09}$	53.97 $_{\pm 1.32}$	60.84 $_{\pm 2.14}$
		`lexical`	56.57 $_{\pm 3.61}$	**65.11** $_{\pm 2.71}$	44.87 $_{\pm 5.09}$	58.51 $_{\pm 3.42}$	54.39 $_{\pm 1.57}$	60.76 $_{\pm 2.12}$
		`context`	**57.00** $_{\pm 4.03}$	64.82 $_{\pm 3.16}$	**45.74** $_{\pm 5.57}$	**59.00** $_{\pm 3.27}$	**55.83** $_{\pm 1.25}$	**62.87** $_{\pm 2.41}$

Table 4.2: Performance (F1-score) comparison in in-domain setting for named entity recognition

In-domain setting In Table 4.2, we can observe that most variants of demonstration-based learning consistently and significantly (with p-value < 0.02) outperform the baseline by a margin ranging from 1.5 to 7 F1 score in three low-resource NER datasets (25, 50 train instances respectively). It demonstrates the potential of our approach for serving as a plug-and-play method for NER models.

Domain adaptation setting First, we observe that simple domain adaptation technique can improve the performance (First rows of Table 4.2 vs. Table 4.3). Here, we only transfer the embedder weights of $\mathcal{M}_s$ to $\mathcal{M}_t$, and we expect the performance gain can be attributed to the embedder of $\mathcal{M}_s$, which is trained in task adaptive pre-training manner on NER task formats [74]. In Table 4.3, we can see that the most variants of demonstration-based learning allow the source model $\mathcal{M}_s$ to be adapted to the target domain in fast with a small amount of target data $\mathcal{T}$, compared to baselines without demonstration including few-shot NER methods.

Entity vs. Instance-oriented demonstration. *instance-oriented demonstration* performs worse than *entity-oriented demonstration* due to the difficulty of finding an appropriate similar instance in a low resource train data. In our analysis, we find that the average cosine similarity between retrieved example

Baselines	Label Sharing: CoNLL03 -> Ontonotes 25	50	Label Different: CoNLL03 -> BC5CDR 25	50
BERT+CRF w/o demonstration	61.22 ±1.93	66.44 ±1.75	52.31 ±1.02	62.10 ±1.01
NNShot	46.67 ±5.48	46.34 ±2.66	44.93 ±1.78	48.12 ±2.72
StructShot	43.61 ±4.58	43.02 ±3.19	25.86 ±4.14	27.81 ±2.10

Strategy	Template	Label Sharing: CoNLL03 -> Ontonotes 25	50	Label Different: CoNLL03 -> BC5CDR 25	50
SBERT (variable)	lexical	**63.34 ±1.53**	68.52 ±0.98	53.50 ±2.26	60.52 ±0.71
	context	62.33 ±1.63	67.86 ±0.89	51.93 ±1.96	60.09 ±1.27
BERTScore (variable)	lexical	62.26 ±1.43	68.68 ±0.25	52.07 ±2.11	59.90 ±0.05
	context	62.46 ±1.69	67.46 ±0.79	53.58 ±1.98	58.95 ±0.38
random (variable)	no-context	62.28 ±1.70	69.32 ±1.34	53.61 ±1.04	62.57 ±0.97
	lexical	62.41 ±1.85	68.84 ±1.78	53.85 ±1.12	62.30 ±0.75
	context	62.58 ±2.20	69.26 ±1.51	54.05 ±0.63	63.04 ±0.31
popular (fixed)	no-context	62.31 ±1.60	69.39 ±1.59	54.33 ±0.80	62.87 ±0.23
	lexical	62.50 ±2.41	69.34 ±1.38	54.30 ±1.12	63.05 ±0.45
	context	62.59 ±2.38	69.91 ±1.24	54.45 ±0.96	63.40 ±0.33
search (fixed)	no-context	62.38 ±2.47	69.57 ±1.50	54.51 ±2.25	62.93 ±1.96
	lexical	62.51 ±2.43	68.93 ±1.69	54.70 ±2.26	62.88 ±2.90
	context	62.63 ±2.94	**69.98 ±1.63**	**54.97 ±1.99**	**63.55 ±1.58**

Table 4.3: Performance (F1-score) comparison in domain-adaptation setting for named entity recognition

s and input x is less than 0.4 which shows many of the retrieved examples are not appropriate similar examples to the input.

Fixed vs. Variable demonstration. `random` doesn't pick a fixed set of demonstrations the same way as `popular` and `search`. Instead, it picks random demonstrations for each input instance. In a low-resource setting, there are often no significantly popular entities. Therefore, the fact that `popular` outperforms `random` in our experiments might suggest that the consistency of demonstration selection, rather than popularity of selected entities, is a crucial factor in better few-shot learning. To test this, we randomly select one entity per entity type and attach it as the demonstration to all instances, we call it (`fixed random`). As shown in Figure 4.6, it outperforms `random` and is on par with `popular` and `search`. We believe this serves as evidence for two hypotheses: (1) consistency of demonstration is essential to performance, and (2) in low-resource settings, the effectiveness of combinations of entities as demonstrations might be a rather random function and not too affected by the combination's collective popularity in the training dataset, which further implies that the idea of `search` is on the right track.

Performance in other model variants To show the effectiveness of demonstration-based learning as plug-and-play method, we present performance in other model variants: `bert-large-cased`, `roberta-base` and `roberta-large`. As shown in Table 4.4, our method shows consistent improvement over baselines

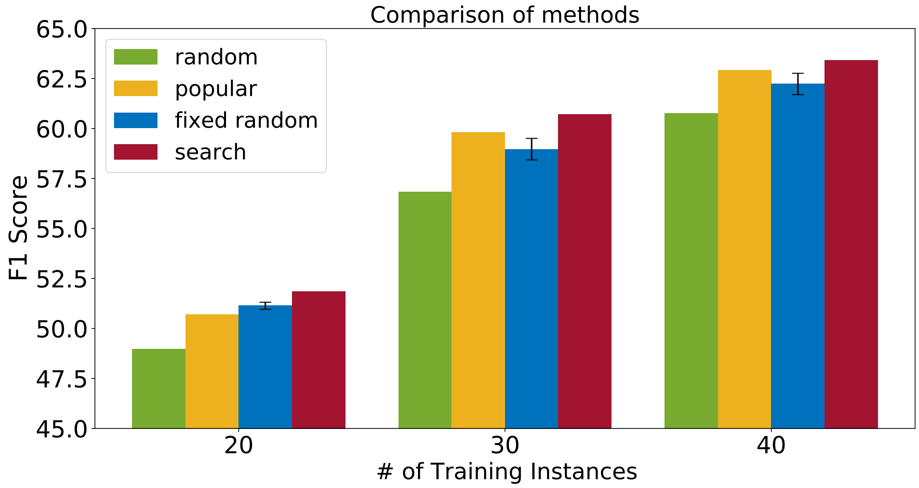


Figure 4.6: Performance (F1-score) comparison between different demonstration selection strategies

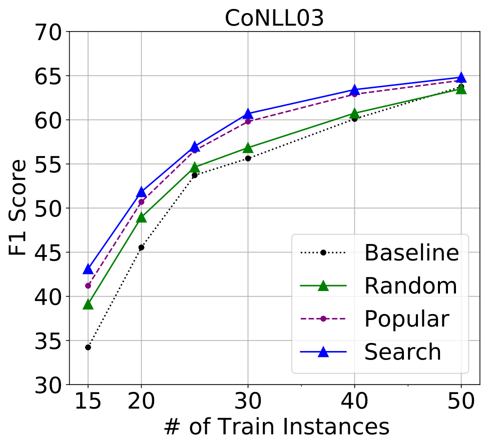
LM	Strategy	Template	*In-domain* CoNLL03 25	*In-domain* CoNLL03 50	*Label Sharing* CoNLL03 -> Ontonotes 25	*Label Sharing* CoNLL03 -> Ontonotes 50
BL	-	-	52.08 $\pm$2.02	66.42 $\pm$2.14	63.50 $\pm$0.96	70.59 $\pm$1.16
RB	-	-	59.67 $\pm$4.65	70.17 $\pm$3.93	68.43 $\pm$2.09	74.11 $\pm$1.19
RL	-	-	59.15 $\pm$2.93	71.51 $\pm$3.44	68.16 $\pm$2.65	74.45 $\pm$1.02
BL	`popular`	`context`	57.60 $\pm$3.37	67.11 $\pm$2.31	64.09 $\pm$2.95	70.88 $\pm$1.09
RB	`popular`	`context`	59.76 $\pm$4.27	70.21 $\pm$3.41	69.09 $\pm$2.63	74.53 $\pm$1.32
RL	`popular`	`context`	59.99 $\pm$2.16	72.15 $\pm$3.81	68.78 $\pm$2.89	74.93 $\pm$1.07

Table 4.4: Performance (F1-score) comparison between different backbone models

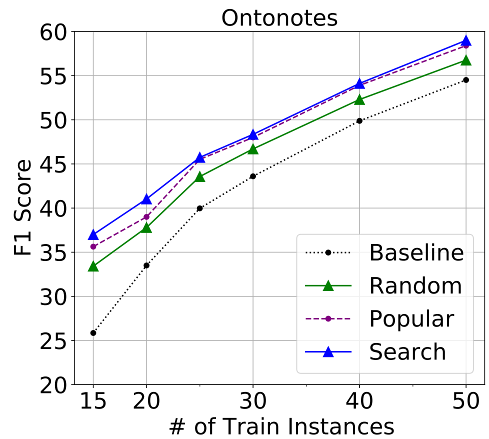
(p-value < 0.05). It shows that demonstration-based learning can be applied to any other model variants and output better contextualized representations for NER tasks and show its potential for scalability.

Effectiveness of `search`. `search` consistently outperforms all other strategies. It shows that not only the entity selection, but also the combination of entity examples per each entity type affects the performance. To see whether it consistently outperforms the baseline over various low-resource data points, we show the performance trend of *entity-oriented demonstration* in Figure 4.7.

Templates of entity-oriented demonstration. *entity-oriented demonstration* becomes more effective when not only showing the entity example per each entity type, but also the corresponding instance

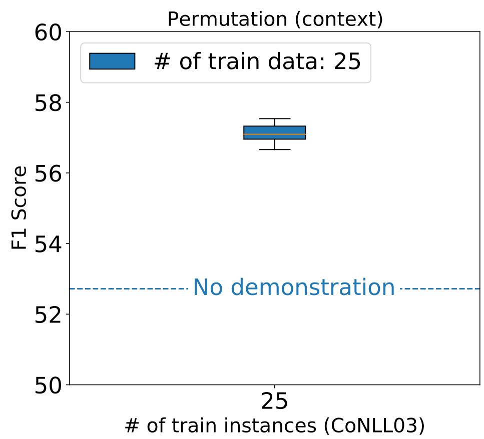


(a) CoNLL03

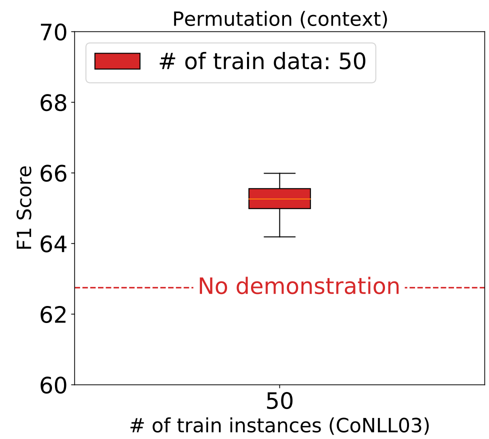


(b) Ontonotes

Figure 4.7: Performance (F1-score) trend by different numbers of train data (15, 20, 30, 40, 50).



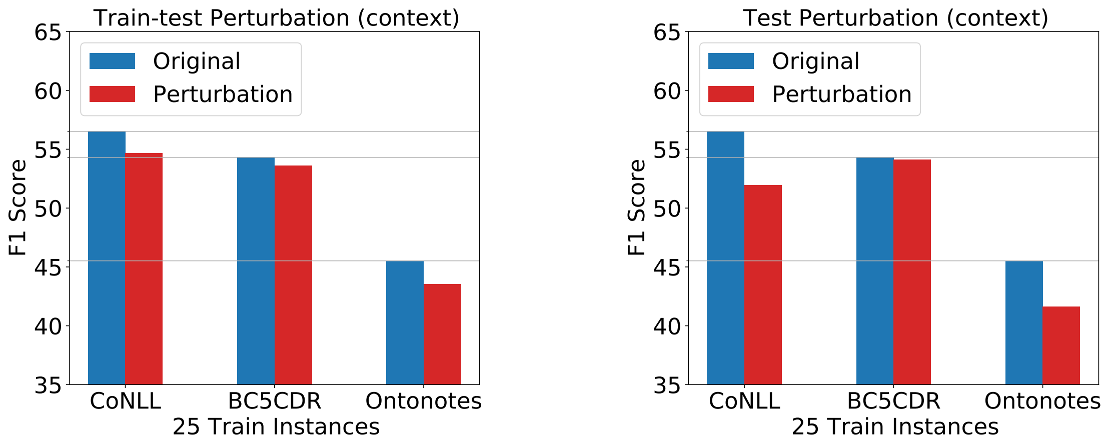
(a) 25 instances



(b) 50 instances

Figure 4.8: Performance (F1-score) variance by different permutation of entity type orders

example as a context. `context` and `lexical` consistently outperform `no-context`. We explore other templates as well, and these three are the best among them. To see whether the order of entity type in *entity-oriented demonstration* affects the performance, we present analysis of entity type permutation, e.g., `person` - `organization` - `location` - `miscellaneous`. There is no clear pattern of which entity type order is better (spearman correlation between F1-scores over different entity type orders with 25 and 50 training instances < 0), but all the permutations outperform the baseline as shown in Figure 4.8, which show that *demonstration-based learning* can be effective regardless of the order.



(a) Test perturbation

(b) Train-Test perturbation

Figure 4.9: Performance (F1-score) difference between original and perturbed demonstration

Demonstration perturbation. To investigate whether the model really learns from demonstration, we explore the performance of our approach with perturbed demonstration which selects random entities, labels, and context sentences as demonstration. Here, we present two studies: (1) *Test perturbation* which train with correct demonstration and test with perturbed demonstration; and (2) *Train-test perturbation* which both train and test with perturbed demonstration. Figure 4.9 shows perturbed demonstration disturbs the model in a large margin for both case. This shows that the model affects by demonstration, and proper demonstration can improve the model's performance.

Effects of demonstration in train & inference. Table 4.5 shows the effects of demonstration in training and inference stage. A comparison of row 0 with row 3 shows that applying demonstration in the training stage but not in the inference stage would make the model perform worse than the fine-tuning baseline. This is another evidence that consistency of demonstration is essential to the method's performance.

Train	Infer	CoNLL03		Ontonotes 5.0		BC5CDR	
		25	50	25	50	25	50
X	X	52.72 $\pm$2.44	62.75 $\pm$0.98	38.97 $\pm$4.62	54.51 $\pm$3.27	52.56 $\pm$0.46	60.20 $\pm$2.01
X	O	51.24 $\pm$2.10	61.02 $\pm$2.05	40.48 $\pm$3.90	52.12 $\pm$3.85	52.16 $\pm$0.55	58.12 $\pm$1.67
O	X	37.71 $\pm$4.65	53.17 $\pm$3.47	31.98 $\pm$4.25	45.27 $\pm$5.19	51.94 $\pm$1.04	57.73 $\pm$1.52
O	O	56.52 $\pm$3.34	64.47 $\pm$2.35	45.52 $\pm$4.69	58.40 $\pm$3.24	54.31 $\pm$0.80	61.31 $\pm$1.51

Table 4.5: Performance (F1-score) comparison w/ and w/o demonstration at training and inference time.

Strategy	Template	CoNLL03		BC5CDR	
		50%	100%	50%	100%
-	-	91.24 $\pm$0.13	91.82 $\pm$0.12	84.58 $\pm$0.17	85.89 $\pm$0.32
`random`	`context`	90.60 $\pm$0.13	91.22 $\pm$0.38	84.32 $\pm$0.07	85.58 $\pm$0.14
`popular`	`context`	90.81 $\pm$0.11	91.85 $\pm$0.07	84.12 $\pm$0.48	85.61 $\pm$0.12

Table 4.6: Performance (F1-score) in fully supervised setting by different percentages of train data

Fully supervised setting. Table 4.6 shows the performance in fully supervised setting, where the train data is sufficient. We can see that demonstration-based learning yields similar performance as baselines (p-value < 0.1), which shows that demonstrations are rather redundant when data is abundant.

4.2.2 Training for Social Norm Violation Detection Using Dialogue History as Context

4.2.2.1 Introduction

Interactive live streaming services such as Twitch [*] and YouTube Live [†] have emerged as one of the most popular and widely-used social platforms. Unfortunately, streamers on these platforms struggle with an increasing volume of toxic comments and norm-violating behavior.[‡] While there has been extensive research on mitigating similar problems for online conversations across various platforms such as Twitter [235, 42, 59, 11, 55], Reddit [40, 112, 176], Stackoverflow [26] and Github [164], efforts that extend them to live

[*] https://www.twitch.tv/
[†] https://www.youtube.com/
[‡] https://safety.twitch.tv/s/article/Community-Guidelines

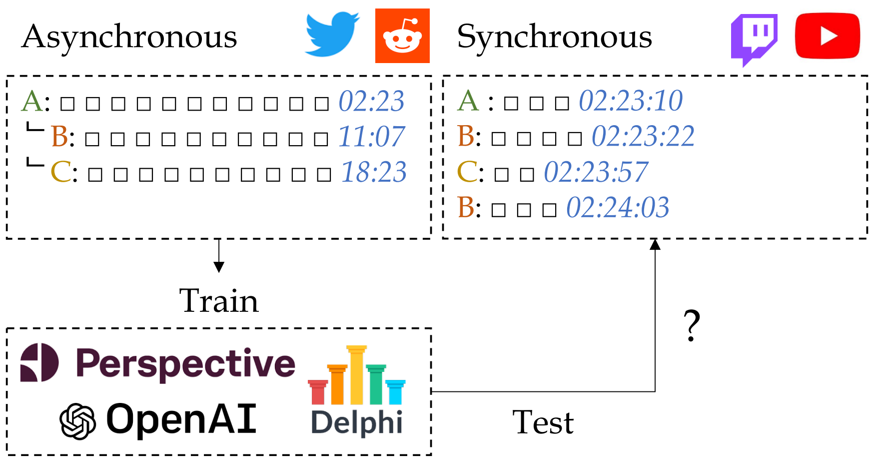


Figure 4.10: Motivating example of studying chat in synchronous domain for violation detection

streaming platforms have been absent. In this paper, we study unique characteristics of comments in live-streaming services and develop new datasets and models for appropriately using contextual information to automatically moderate toxic content and norm violations.

Conversations in online communities studied in previous work are *asynchronous*: utterances are grouped into threads that structurally establish conversational context, allowing users to respond to prior utterances without time constraints. The lack of time constraints allows users to formulate longer and better thought-out responses and more easily reference prior context.

On the other hand, conversations on live streaming platforms are *synchronous*, i.e. in real-time, as utterances are presented in temporal order without a thread-like structure. Context is mostly established by consecutive utterances [133]. The transient nature of live-stream utterances encourages fast responses, and encourages producing multiple short comments that may be more prone to typos (70% of comments are made up of < 4 words). Figure 4.10 shows an illustration of the contrasting temporal and length patterns between the asynchronous and synchronous platforms.

Owing to these different characteristics, we find that previous approaches for detecting norm violations are ineffective for live-streaming platforms. To address this limitation, we present the first NLP study of detecting norm violations in live-stream chats. We first establish norms of interest by collecting 329 rules from Twitch streamers' channels and define 15 different fine-grained norm categories through an iterative coding process. Next, we collect 4,583 moderated chats and their corresponding context from Twitch live streams and annotate them with these norm categories.

With our data, we explore the following research questions: (1) How are norm violations in live-stream chats, *i.e.* synchronous conversations, different from those in previous social media datasets, *i.e.* asynchronous conversations?; (2) Are existing norm violation or toxicity detection models robust to the distributional shift between the asynchronous and synchronous platforms?; and (3) Which features (*e.g.*, context and domain knowledge) are important for detecting norm violation in synchronous conversations?

From our explorations, we discover that (1) live-stream chats have unique characteristics and norm violating behavior that diverges from those in previous toxicity and norm-violation literature; (2) existing models for moderation perform poorly on detecting norm violations in live-stream chats; and (3) additional information, such as chat and video context, are crucial features for identifying norm violations in live-stream chats. We show that incorporating such information increases inter-annotator agreement for categorizing moderated content and that selecting temporally proximal chat context is crucial for enhancing the performance of norm violation detection models in live-stream chats.

4.2.2.2 Norm Violation Detection: Dataset (NormVio-RT)

To investigate norm-violations in live-stream chat, we first collect Norm Violations in Real-Time Conversations (**NormVio-RT**), which contains 4,583 norm-violating comments on Twitch that were moderated by channel moderators.[§] An overview of our data collection procedure is illustrated in Figure 4.11. We first select 200 top Twitch streamers and collect moderated comments from their streamed sessions. To understand why these chats are moderated, we collect chat rules from these streamers and aggregate them to define coarse and fine-grained norm categories. We design a three-step annotation process to determine

[§]Please contact the authors for the anonymized study data.

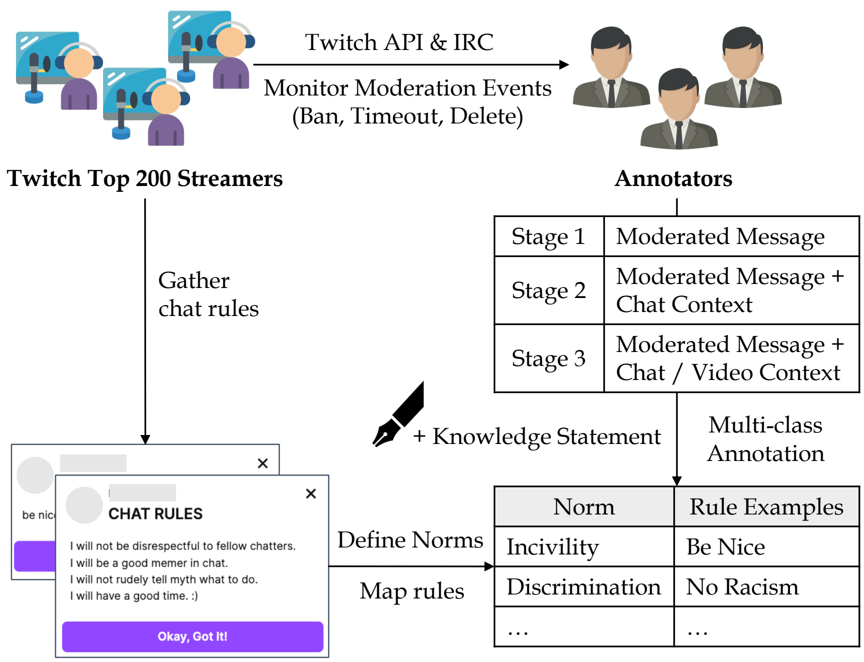


Figure 4.11: Data construction overview for studying norm violation in synchronous chat domain.

the impact of the chat history, video context, and external knowledge on labeling decisions. Lastly, we present analysis of the collected data.

Data Collection We collected data using the Twitch API and IRC[¶] from the streamers with videos that are available for download among the top 200 Twitch streamers as of June 2022[‖]. We specifically looked for comments that triggered a moderation event during a live stream (e.g. *user ban*, *user timeout*), and collected the moderated comment and the corresponding video and chat logs up to two minutes prior to the moderation event. Logs of moderated events from August 22, 2022 to September 3, 2022 were collected. We excluded comments that were moderated within less than 1 second of being posted, as they are likely to have been moderated by bots rather than humans.

Norm Categorization Twitch streamers can set their own rules for their channels, and these channel-specific rules are essential for understanding why comments were moderated. We first collect 329 rules

[¶] https://github.com/TwitchIO/TwitchIO
[‖] https://twitchtracker.com/channels/viewership/english

Coarse	Fine-grained	Target	Rule Examples
Discrimination	Discrimination	-	No racism, sexism or homophobia.
HIB (Harassment, Intimidation, Bullying)	HIB	Broadcaster OIB OOB	No HIB towards broadcaster No HIB towards viewers, moderators, etc. No HIB towards other broadcasters, politicians, etc.
Privacy	Doxing	-	Please no personal questions about me. Do not share any personal information about yourself or others.
Inappropriate Contents	NSFW Self-destructive Illegal Spoiler	- - - -	No NSFW content (e.g., Inappropriate ASCII arts). No talk of suicide. No drug discussion of any kind. Do not give game spoilers.
Off Topic	Controversial Topic Begging	- -	No drama, politics or religion. No begging for subscriptions or money.
Spam	Excessive & Repetitive Advertisements	- -	No walls of text. No self promotion unless authorized.
Meta-Rules (Live streaming specific)	Backseating & Tall order Mentioning other broadcasters Specific language only	- - -	Don't tell me what to do. Don't ask for mod. Don't talk down on other streamers. English only.
Incivility (Miscellaneous)	Incivility	-	Be nice, Be civil

Table 4.7: Norms in live streaming domain (Twitch)

from the top 200 Twitch streamers' channels. Next, following [58], we take an iterative coding process such that the authors of this paper individually code for rule types with certain categories, come together to determine differences and then repeat the coding process individually. With this process, we aggregated similar rules into 15 different fine-grained level norm categories (*e.g.*, controversial topics, begging) and cluster multiple fine-grained categories into 8 different coarse-grained norm categories (e.g., off-topic). To better understand the *targets* of offensive comments in the HIB (Harassment, Intimidation, Bullying) class, we added an additional dimension to consider whether the target is the broadcaster (streamer), participants in the channel (e.g., moderators and viewers), or someone not directly involved in the broadcast. We asked annotators to assign "Incivility" to cases where annotators do not believe that a specific predefined rule type has been violated although moderated. Table 4.7 shows the resulting norm categories and corresponding fine-grained norms with examples.

Violated Norm Type Annotation We recruited three annotators who are fluent in English and spend at least 10 hours a week on live streaming platforms to ensure that annotators understood live streaming content and conventions. Their fluency was verified through several rounds of pilot annotation work.

Internal auditors continuously conducted intermittent audits to ensure that annotators fully understood the guidelines.

Annotators were asked to annotate each moderation event (i.e. moderated comment) with the rule types it violates. To measure the importance of context in determining types of norm violations, annotators were asked to provide labels for three different scenarios with varying amounts of context: (1) **Stage 1**: annotate based on only the user's last message before the moderation event (*single utterance*); (2) **Stage 2**: annotate based on chat logs up to two minutes prior to the moderation (*+chat context*); (3) **Stage 3**: annotate based on chat logs and their corresponding video clip of the same duration (*+video context*). Since rules are not mutually exclusive (*e.g.,* a message can violate both discrimination & harassment), they are allowed to choose multiple rule types if there are more than one violated rule at each stage. All the annotations are done with our internal annotation user interface. To determine the final label for each moderated event, we aggregate the labels of annotators using a *majority vote* with heuristic rules.

Coarse	**Fine-grained**	**# Rules**	**# Violates**		
			stage 1	**stage 2**	**stage 3**
Discrimination	Discrimination	13.98% (46)	2.34% (104)	2.25% (101)	2.34% (105)
HIB	HIB	22.49% (74)	21.33% (947)	26.55% (1,190)	27.80% (1,246)
Privacy	Doxing	0.60% (2)	0.34% (15)	0.36% (16)	0.36% (16)
Inappropriate Contents	Spoiler	0.60% (2)	0.02% (1)	0.02% (1)	0.02% (1)
	NSFW	1.82% (6)	0.86% (38)	0.85% (38)	0.85% (38)
	Self-destructive	1.21% (4)	0.32% (14)	0.29% (13)	0.29% (13)
	Illegal	0.30% (1)	0.16% (7)	0.07% (3)	0.07% (3)
Off Topic	Controversial Topic	5.47% (18)	0.59% (26)	0.85% (38)	0.83% (37)
	Begging	1.51% (5)	1.44% (64)	1.36% (61)	1.36% (61)
Spam	Excessive & Repetitive	11.24% (37)	17.59% (781)	21.64% (970)	21.42% (960)
	Advertisements	11.24% (37)	4.64% (206)	4.40% (197)	4.42% (198)
Meta-Rules (Live streaming specific)	Mentioning other streamers	14.28% (47)	0.72% (32)	10.62% (476)	10.58% (474)
	Backseating & Tall order	5.16% (17)	3.45% (153)	3.70% (166)	3.77% (169)
	Specific language only	10.03% (33)	0.97% (43)	6.94% (311)	6.94% (311)
Incivility (Miscellaneous)	Incivility	-	12.30% (546)	11.57% (519)	11.51% (516)
	Non-Identifiable	-	32.93% (1,462)	8.52% (382)	7.45% (334)
Total		329	4,439	4,482	4,482

Table 4.8: Data statistics of norm violation detection task

Lastly, to examine how much external knowledge matters in understanding comments on live streaming platforms, we asked annotators to (1) indicate whether external knowledge is necessary to understand why a comment triggered a moderation event and if so (2) describe what that knowledge is. We focus on two types of external knowledge: platform- and streamer-specific. Platform-specific knowledge includes the implicit meaning of particular emojis, emotes, and slang that are commonly used on Twitch. Streamer-specific knowledge involves the streamer's personal background and previous streaming sessions. Examples of template for each type is as follow so that annotators can easily fill out:

1. **Platform.** <*span*> is {emoji, text} that means <*explanation*>

2. **Streamer.** <*streamer*> <*explanation*>

Data Insights from NormVio-RT

1. **General Observation** We identified three characteristics that distinguish real-time live-streaming chat from other domains. First, the majority of comments are very short; 70% of comments are made up of < 4 words. Additionally, they are often very noisy due to the real-time nature of communication, which leads to a high number of typos, abbreviations, acronyms, and slang in the comments. Lastly, some comments use unusual visual devices such as ASCII art and "all caps", to make them more noticeable. This is because each comment is visible only for a short time in popular streams (on average, there are around 316 chats per minute for the streamers in our data). The chat window in live streaming platforms can only display a limited number of comments, so viewers are incentivized to use visual devices to draw the streamer's attention in these fast-paced conditions.

2. **False Positives in Data** We find that the "Incivility" case contains many false positives, as they include cases that seem to have been moderated for no particular reason. We asked annotators to put all miscellaneous things into the "Incivility" category, and also to mark as "Incivility" if they could not identify any reason for the moderation. We found that many cases are not identifiable, as

% Agreement	Stage 1	Stage 2	Stage 3
Exact Match	39.71% (1,820)	41.10% (1,884)	40.67% (1,864)
Partial Match	54.11% (2,480)	75.03% (3,439)	75.21% (3,447)
Majority Vote	96.85% (4,439)	97.79% (4,482)	97.79% (4,482)

Table 4.9: Inter-annotator agreement of data construction for norm violation detection task

shown in Table 4.8. It is natural that many cases are non-identifiable in stage 1, as annotators are only given the moderated comment and no context. However, the 7.45% non-identifiable cases that remain even after stage 3 could be false positives, or they could be cases where the moderation event occurred more than two minutes after a problematic comment was made.

3. **Context Improves Inter-Annotator Agreement** Interestingly, providing context helps mitigate annotator bias, as shown by the increase in inter-annotator agreement from stage 1 to stages 2 and 3 in Table 4.9. Here, the *exact match* determines whether all three annotators have exactly the same rules; *partial match* determines whether there is at least one intersection rule between three annotators; and *majority vote* chooses the rule types that were selected by at least two people. Also, non-identifiable and disagreement cases drop significantly when the contexts are given as shown in Table 4.8. Similarly for determining rule types, context also helps annotators identify targets for HIB and reduces inconsistencies between annotators. Our observation emphasizes the importance of context in synchronous communication and differs from previous findings that context-sensitive toxic content is rare in asynchronous communication [180, 245].

4. **External Knowledge Helps Annotations** To investigate the impact of external knowledge on annotators' labeling decisions, we compare annotations made with and without external knowledge provided. For examples with knowledge statements, we expect to see differences in annotation if external knowledge is necessary to comprehend why they were moderated. Statistics show that

296 examples (6.6%) require knowledge, with 183 examples requiring streamer knowledge and 187 examples requiring platform knowledge. Note that there are some examples require both.

5. **Norm Category Distribution** Table 4.8 shows the norm category distribution of streamers' rules and the moderated comments. While the categories are not directly comparable to the ones defined in NormVio for Reddit [176], we identified a few similar patterns. First, in both domains, Harassment and Incivility (i.e., Discrimination, HIB, Incivility) take up a significant portion of the entire set of norm violations. Also, the two domains show a similar pattern where rules for Off-Topic, Inappropriate Contents, and Privacy exist but are relatively less enforced in practice. However, we also found that the two domains differ in various ways. For example, Spam and Meta-Rules cover significantly higher portions of both rules and moderated comments on Twitch than on Reddit. On the other hand, there are fewer rules about content on Twitch, which implies that streamers are less concerned about the content of the comments than Reddit community moderators. As our data shows that norm-violating comments on live chats exhibit distinctive rules and patterns, it suggests that the existing norm violation detection systems may not perform well without domain adaptation to account for these distributional differences. We examine this hypothesis empirically in the following section and suggest appropriate modeling adjustments to better detect toxicity for real-time comments.

4.2.2.3 Norm Classification in NormVio-RT

To understand the model's ability to detect norm violations and how additional information can affect detection, we train binary classification models for each category with different types of context including conversation history, broadcast category, and rule description following [176].

4.2.2.4 Experimental Setup

For each coarse-level category, we train a RoBERTa-base model with a binary cross entropy loss to determine whether the message is violating the certain norm or not. Following [176], we perform an 80-10-10 train/dev/test random split of moderated messages and add the same number of unmoderated messages in the same split. Next, for each binary classification, we consider the target category label as 1 and others as 0 and construct a balanced training data set. Here, the labels are based on stage 3.

To examine how context affects model performance, we experiment with four model variants with different input context: (1) **Single user context** is only the chat logs of the moderated user that took place up to two minutes before the moderation event; (2) **Multi-user context (event)** is N messages that directly precede the moderation event, regardless of whether it belongs to the moderated user; (3) **Multi-user context (utterance)** is N messages that directly precedes the *single utterance*, which is the moderated user's last message before the moderation event; (4) **Multi-user context (first)** is the first N messages of the collected two-minute chat logs. The intuition for this selection is that the moderation event may have taken place much earlier than the moderation event. In all the Multi-user contexts, we use $N = 5$; (5) **Broadcast category** is the category that streamers have chosen for their broadcast. It usually is the title of a game or set to "just chatting"; and (6) **Rule text** is a representative rule example shown in Table 4.7. The rule text is only used for training examples because it is not possible to know which rule was violated for unseen examples and we use randomly selected rule text for unmoderated negative examples in training examples. All contexts are appended to the input text (*single utterance*) with a special token ([SEP]) added between the input text and the context. Chat logs for multi-user context and single-user context are placed sequentially with spaces between chats.

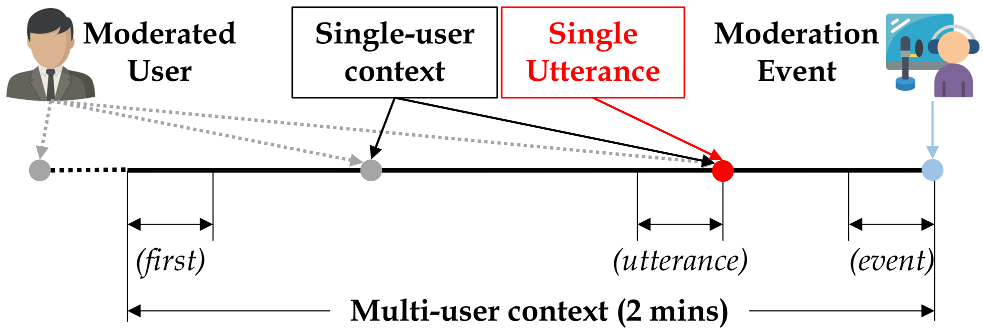


Figure 4.12: Explanation of different types of dialogue context for norm violation detection

Context	All	Discrimination	HIB	Privacy	Inapt. Contents	Off Topic	Spam	Meta-Rules	Incivility
-	0.70 $_{\pm 0.00}$	$\underline{0.11}$ $_{\pm 0.00}$	$\underline{0.52}$ $_{\pm 0.01}$	**0.05** $_{\pm 0.03}$	0.12 $_{\pm 0.01}$	0.07 $_{\pm 0.00}$	0.63 $_{\pm 0.01}$	**0.65** $_{\pm 0.01}$	0.28 $_{\pm 0.04}$
Single-user context	0.78 $_{\pm 0.00}$	0.07 $_{\pm 0.02}$	0.50 $_{\pm 0.01}$	$\underline{0.03}$ $_{\pm 0.02}$	**0.18** $_{\pm 0.05}$	0.05 $_{\pm 0.00}$	$\underline{0.67}$ $_{\pm 0.02}$	0.58 $_{\pm 0.04}$	0.28 $_{\pm 0.06}$
Multi-user context (event)	0.75 $_{\pm 0.04}$	0.03 $_{\pm 0.00}$	0.44 $_{\pm 0.02}$	0.01 $_{\pm 0.00}$	$\underline{0.14}$ $_{\pm 0.12}$	0.05 $_{\pm 0.01}$	0.66 $_{\pm 0.00}$	0.60 $_{\pm 0.03}$	0.17 $_{\pm 0.00}$
Multi-user context (utterance)	$\underline{0.91}$ $_{\pm 0.01}$	0.04 $_{\pm 0.00}$	**0.61** $_{\pm 0.05}$	0.00 $_{\pm 0.00}$	0.09 $_{\pm 0.03}$	0.10 $_{\pm 0.04}$	0.66 $_{\pm 0.01}$	**0.65** $_{\pm 0.04}$	0.24 $_{\pm 0.12}$
Multi-user context (first)	**0.95** $_{\pm 0.00}$	0.05 $_{\pm 0.01}$	**0.61** $_{\pm 0.01}$	0.01 $_{\pm 0.00}$	0.11 $_{\pm 0.02}$	0.08 $_{\pm 0.03}$	**0.70** $_{\pm 0.03}$	0.62 $_{\pm 0.02}$	**0.45** $_{\pm 0.03}$
Broadcast category	0.77 $_{\pm 0.03}$	**0.13** $_{\pm 0.03}$	0.48 $_{\pm 0.01}$	0.02 $_{\pm 0.01}$	0.13 $_{\pm 0.04}$	$\underline{0.13}$ $_{\pm 0.05}$	0.65 $_{\pm 0.01}$	$\underline{0.64}$ $_{\pm 0.02}$	$\underline{0.30}$ $_{\pm 0.02}$
Rule text	0.75 $_{\pm 0.01}$	0.05 $_{\pm 0.08}$	0.11 $_{\pm 0.17}$	0.00 $_{\pm 0.00}$	0.12 $_{\pm 0.06}$	**0.29** $_{\pm 0.18}$	0.58 $_{\pm 0.04}$	0.38 $_{\pm 0.02}$	0.13 $_{\pm 0.03}$

Table 4.10: Performance (F1-score) on norm violation detection

4.2.2.5 Experimental Results

Table 4.10 presents performance of norm classification for coarse-level norm categories. "All" refers to binary moderation detection, whether the message is moderated or not, and not the specific norm type. First, we can see that additional context improves the performance of "All," but context does not consistently improve the performance of category-specific norm classifiers. For example, context reduces performance for categories where the issues are usually limited to the utterance itself (*e.g.*, discrimination and privacy). In contrast, categories that rely on the relationships between utterances, such as HIB and incivility, show improved performance with context. Secondly, multi-user context performs quite well compared to the other contexts, indicating that a more global context that includes utterances from other users helps determine the toxicity of target utterances. Lastly, the strong performance of Multi-user context (first) suggests that earlier messages in the two-minute window are more important, meaning that the temporal distance between the moderation event and the actual offending utterance may be substantial in many cases. Thus, our results encourage future efforts on developing a more sophisticated approach for context selection.

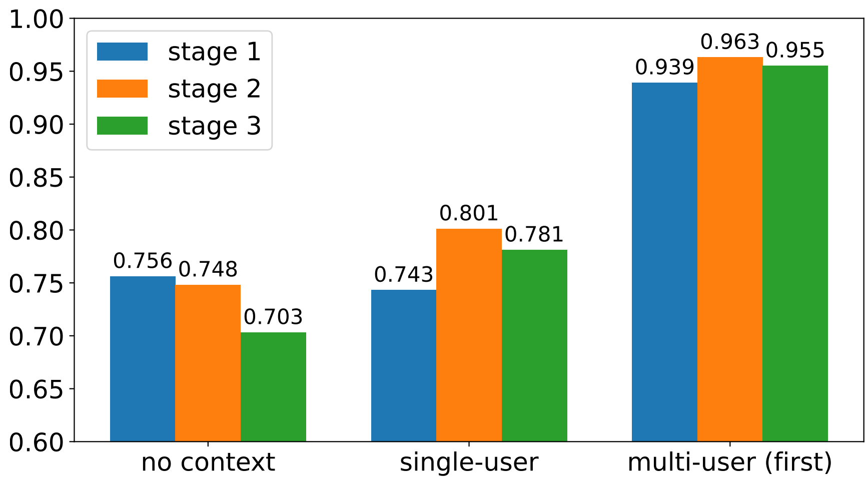


Figure 4.13: Performance (F1-score) of norm violation detection by different ground truth label for each context.

Availability of Context. To compare human decisions with those of our models, we conduct experiments varying the context available to annotators and models. For example, we expect models trained with only single utterances to perform best when using stage 1 (utterance only) labels as ground-truth labels since humans are also not given any context at stage 1. Indeed, in Figure 4.13, using the stage 1 labels as the ground truth labels yields the best performance for a model trained without any context, while using the stage 2 (context) labels as the ground truth labels shows the best performance for a model trained with previous chat history. Since our experiments only handle text inputs, it is not surprising that using stage 3 (video) labels as ground-truth labels yields worse performance than using stage 2 labels. However, interestingly, the gap is not large, which indicates that gains from a multi-modal model that incorporates information from the video may be small and that single modality (text-only) models can be sufficient for the majority of moderation instances.

Context Size. To understand how the amount of available context affects moderation performance, we compare the multi-user context configurations with various number of messages from one to 25. Figure 4.14 demonstrates that 15 to 20 messages prior to the moderated user's message helps with moderation performance the most (See *utterance* and *first*). However, increasing the number of messages that directly

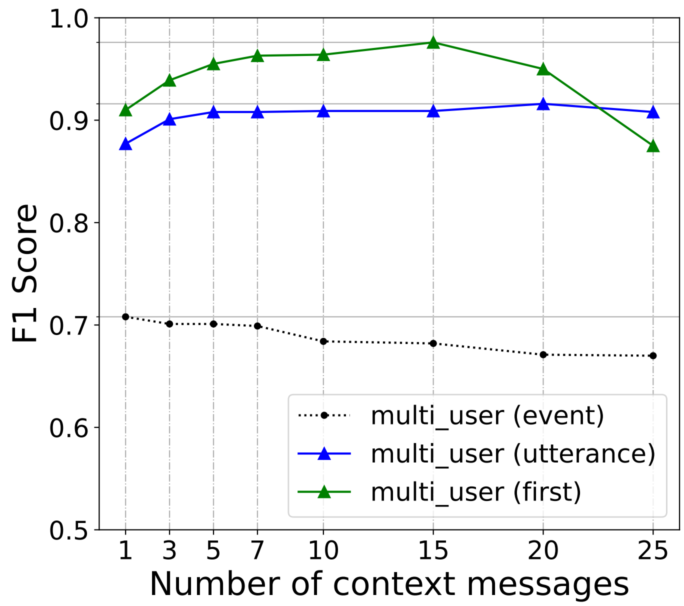


Figure 4.14: Performance (F1-score) trend of norm violation detection with varying context length.

precede the moderation event actually lowers moderation performance (See *event*). It may be that most of this context serves as noise.

Distribution Shift in Norm Classification. Existing tools often focus on identifying harmful speech, but NormVio [176] also considers a wider range of norm-violating comments on Reddit, similar to NormVio-RT but in a different domain. We compare NormVio and NormVio-RT by evaluating the performance of a model fine-tuned on NormVio with NormVio-RT, and vice versa, to examine the impact of distribution shift between these domains. We choose six coarse-level categories that overlap between the two, as shown in Table 4.11. To measure with-context performance, we use the previous comment history for Reddit and multi-user context (utterance) for Twitch to simulate the most similar setup in both domains. Overall, experimental results show a pronounced distribution shift between Reddit (*asynchronous*) and

Twitch (*synchronous*). Interestingly, models trained on Twitch are able to generalize better than models trained on Reddit despite having 6x less training data. Specifically, models trained using the out-of-domain Twitch+context data perform comparably on the Reddit test set to those trained using in-domain Reddit+context data.

Category		**Without Context**				**With Context**			
Reddit (Normvio)	**T**witch (Normvio-RT)	**R**	**T→R**	**T**	**R→T**	**R**	**T→R**	**T**	**R→T**
ALL	ALL	0.99 $\pm_{0.00}$	0.84 $\pm_{0.04}$	0.70 $\pm_{0.00}$	0.67 $\pm_{0.00}$	0.99 $\pm_{0.00}$	0.98 $\pm_{0.00}$	0.91 $\pm_{0.01}$	0.67 $\pm_{0.00}$
Incivility	Incivility	0.67 $\pm_{0.00}$	0.16 $\pm_{0.09}$	0.28 $\pm_{0.04}$	0.09 $\pm_{0.03}$	0.74 $\pm_{0.00}$	0.56 $\pm_{0.14}$	0.24 $\pm_{0.12}$	0.09 $\pm_{0.03}$
Harassment	HIB, Privacy	0.34 $\pm_{0.01}$	0.19 $\pm_{0.01}$	0.51 $\pm_{0.01}$	0.27 $\pm_{0.01}$	0.41 $\pm_{0.00}$	0.20 $\pm_{0.00}$	0.62 $\pm_{0.02}$	0.26 $\pm_{0.03}$
Spam	Spam	0.47 $\pm_{0.02}$	0.22 $\pm_{0.02}$	0.63 $\pm_{0.01}$	0.28 $\pm_{0.01}$	0.53 $\pm_{0.01}$	0.27 $\pm_{0.01}$	0.66 $\pm_{0.01}$	0.28 $\pm_{0.01}$
Off Topic	Off Topic	0.25 $\pm_{0.02}$	0.12 $\pm_{0.01}$	0.07 $\pm_{0.00}$	0.00 $\pm_{0.00}$	0.28 $\pm_{0.01}$	0.12 $\pm_{0.02}$	0.10 $\pm_{0.04}$	0.00 $\pm_{0.00}$
Hate Speech	Discrimination	0.17 $\pm_{0.02}$	0.05 $\pm_{0.04}$	0.11 $\pm_{0.00}$	0.02 $\pm_{0.00}$	0.19 $\pm_{0.00}$	0.06 $\pm_{0.04}$	0.04 $\pm_{0.00}$	0.02 $\pm_{0.00}$
Content	Inapt. Contents	0.30 $\pm_{0.06}$	0.08 $\pm_{0.03}$	0.12 $\pm_{0.01}$	0.00 $\pm_{0.00}$	0.37 $\pm_{0.01}$	0.05 $\pm_{0.02}$	0.09 $\pm_{0.03}$	0.00 $\pm_{0.00}$

Table 4.11: Performance (F1-score) on distribution shift between norm violations in Reddit and Twitch.

4.3 Human Explanations as Context for Learning

4.3.1 Label-efficient Learning with Human-provided Labeling Explanation

4.3.1.1 Introduction

Named entity recognition (NER) is a fundamental information extraction task that focuses on extracting entities from a given text and classifying them using pre-defined categories (e.g., persons, locations, organizations) [171]. Recent advances in NER have primarily focused on training neural network models with an abundance of human annotations, yielding state-of-the-art results [115]. However, collecting human annotations for NER is expensive and time-consuming, especially in social media messages [137] and technical domains such as biomedical publications, financial documents, legal reports, etc. As we seek to advance NER into more domains with less human effort, how to learn neural models for NER in a cost-effective way becomes a crucial research problem.

The standard protocol for obtaining an annotated NER dataset involves an annotator selecting token spans in a sentence as mentions of entities, and labeling them with an entity type. However, such annotation process provides limited supervision *per example*. Consequently, one would need large amount of annotations in order to train high-performing models for a broad range of entity types, which can clearly be cost-prohibitive. The key question is then *how can we learn an effective NER model in presence of limited quantities of labeled data*?

We, as humans, recognize an entity within a sentence based on certain words or phrases that act as cues. For instance, we could infer that *'Kasdfrcxzv'* is likely to be a location entity in the sentence "*Tom traveled a lot last year in Kasdfrcxzv.*" We recognize this entity because of the cue phrase "*travel ... in*," which suggests there should be a location entity following the word 'in.'

We call such phrases "entity triggers." Similar to the way these triggers guide our recognition process, we hypothesize that they can also help the model to learn to generalize efficiently, as shown in Figure 4.15.

Specifically, we define an ***"entity trigger"*** (or trigger for simplicity) as a group of words that can help explain the recognition process of a particular entity in the same sentence. For example, in Figure 4.16, *"had ... lunch at"*[**] and *"where the food"* are two distinct *triggers* associated with the RESTAURANT entity *"Rumble Fish."* An entity trigger should be a necessary and sufficient cue for humans to recognize its associated entity even if we mask the entity with a random word. Thus, unnecessary words such as "*fantastic*" should not be considered part of the entity trigger.

In this paper, we argue that a combination of entity triggers and standard entity annotations can enhance the generalization power of NER models. This approach is more powerful because unlabeled sentences, such as *"Bill enjoyed a great dinner with Alice at Zcxlbz."*, can be matched with the existing trigger *"had ... lunch at"* via their semantic relatedness. This makes it easier for a model to recognize *"Zcxlbz"* as

[**] Note that a trigger can be a discontinuous phrase.

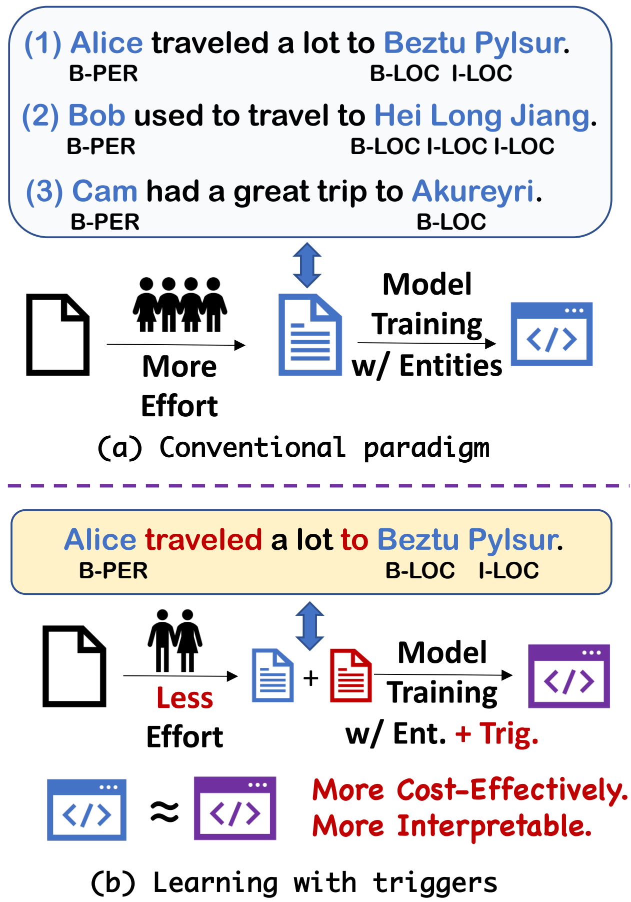


Figure 4.15: Motivation example of explanation (entity trigger) based learning

a Restaurant entity. In contrast, if we only have the entity annotation itself (i.e., *"Rumble Fish"*) as supervision, the model will require many similar examples in order to learn this simple pattern. Annotation of triggers, in addition to entities, does not incur significantly additional effort because the triggers are typically short, and more importantly, the annotator has already comprehended the sentence, identifying their entities as required in the traditional annotation. On the benchmark datasets we consider, the average length of a trigger in our crowd-sourced dataset is only 1.5-2 words. Thus, we hypothesize that using triggers as additional supervision is a more cost-effective way to train models.

We crowd-sourced annotations of 14,708 triggers on two well-studied NER datasets to study their usefulness for the NER task. We propose a novel framework named Trigger Matching Network that learns trigger representations indicative of entity types during the training phase, and identifies triggers in an

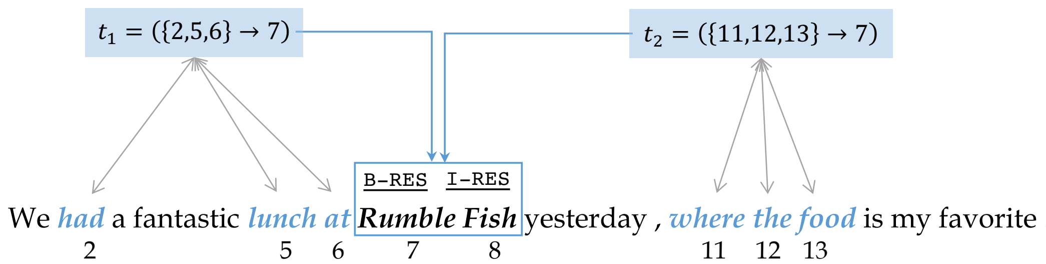


Figure 4.16: Example of entity trigger

unlabeled sentence at inference time to guide a traditional entity tagger for delivering better overall NER performance. Our TMN framework consists of three components: 1) a trigger encoder to learn meaningful trigger representations for an entity type, 2) a semantic trigger matching module for identifying triggers in a new sentence, and 3) an entity tagger that leverages trigger representations for entity recognition (as present in existing NER frameworks). Different from conventional training, our learning process consists of two stages, in which the first stage comprises jointly training a trigger classifier and the semantic trigger matcher, followed by a second stage that leverages the trigger representation and the encoding of the given sentence using an attention mechanism to learn a sequence tagger.

Our contributions in this paper are as follows:

- We introduce the concept of "entity triggers," a novel form of explanatory annotation for named entity recognition problems. We crowd-source and publicly release 14k annotated entity triggers on two popular datasets: *CoNLL03* (generic domain), *BC5CDR* (biomedical domain).

- We propose a novel learning framework, named *Trigger Matching Network*, which encodes entity triggers and softly grounds them on unlabeled sentences to increase the effectiveness of the base entity tagger.

- Experimental results show that the proposed trigger-based framework is significantly more cost-effective. The TMN uses 20% of the trigger-annotated sentences from the original CoNLL03 dataset, while achieving a comparable performance to the conventional model using 70% of the annotated sentences.

- We present an annotation framework `LEAN-LIFE`, which is the first framework to capture and leverage explanations for improved model training and performance.

4.3.1.2 Entity Triggers as Explanation for NER

We consider the problem of *how to cost-effectively learn a model for NER using entity triggers*. In this section, we introduce basic concepts and their notations, present the conventional data annotation process for NER, and provide a formal task definition for learning using entity triggers.

In the conventional setup for supervised learning for NER, we let $\mathbf{x} = [x^{(1)}, x^{(2)}, \cdots, x^{(n)}]$ denote a sentence in the labeled training corpus $\mathcal{D}_L$. Each labeled sentence has a NER-tag sequence $\mathbf{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(n)}]$, where $y^{(i)} \in \mathcal{Y}$ and $\mathcal{Y}$ can be {`O`, `B-PER`, `I-PER`, `B-LOC`, `I-LOC`, $\cdots$}. The possible tags come from a BIO or BIOES tagging schema for segmenting and typing entity tokens. Thus, we have $\mathcal{D}_L = \{(\mathbf{x_i}, \mathbf{y_i})\}$, and an unlabeled corpus $\mathcal{D}_U = \{\mathbf{x_i}\}$.

We propose to annotate entity triggers in sentences. We use $T(\mathbf{x}, \mathbf{y})$ to represent the set of annotated entity triggers, where each trigger $t_i \in T(\mathbf{x}, \mathbf{y})$ is associated with an entity index e and a set of word indices $\{w_i\}$. Note that we use the index of the first word of an entity as its entity index. That is, $t = (\{w_1, w_2, \cdots\} \rightarrow e)$, where e and w_i are integers in the range of $[1, |\mathbf{x}|]$. For instance, in the example shown in Figure 4.16, the trigger *"had ... lunch at"* can be represented as a trigger $t_1 = (\{2, 5, 6\} \rightarrow 7)$, because this trigger specifies the entity starting at index 7, *"Rumble"*, and it contains a set of words with indices: *"had"* (2), *"lunch"* (5), and *"at"* (6). Similarly, we can represent the second trigger *"where the food"* as $t_2 = (\{11, 12, 13\} \rightarrow 7)$. Thus, we have $T(\mathbf{x}, \mathbf{y}) = \{t_1, t_2\}$ for this sentence.

Adding triggers creates a new form of data $\mathcal{D}_T = \{(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i})\}$. Our goal is to learn a model for NER from a trigger-labeled dataset $\mathcal{D}_T$, such that we can achieve comparable learning performance to a model with a much larger $\mathcal{D}_L$.

4.3.1.3 TriggerNER: Human-Guided Learning for NER

We now present our framework for a more cost-effective learning method for NER using triggers. We assume that we have collected entity triggers.

At a high-level, we aim to learn trigger representations for entity types that allow the entity tagger to generalize for sentences beyond the training phase. Our intuition is that triggers acting as cues for the same named-entity type should have similar trigger representations, and thus triggers can be identified in an unlabeled sentence at inference time by soft-matching between the sentence representation and trigger representations seen during training. We perform such soft-matching using a self-attention mechanism.

We propose a straightforward yet effective framework, named *Trigger Matching Networks* (TMN), consisting of a trigger encoder (`TrigEncoder`), a semantic-based trigger matching module (`TrigMatcher`), and a base sequence tagger (`SeqTagger`). We have two learning stages for the framework: the first stage jointly learns the `TrigEncoder` and `TrigMatcher`, and the second stage uses the trigger vectors to learn NER tag labels. Figure 4.17 shows this pipeline.

Trigger Encoding & Semantic Matching Learning trigger representations and semantically matching them with sentences are inseparable tasks. Desired trigger vectors capture the semantics in a shared embedding space with token hidden states, such that sentences and triggers can be semantically matched. Recall the example we discussed in Figure 4.16, *"enjoyed a great dinner at"* versus *"had ... lunch at."* Learning an attention-based matching module between entity triggers and sentences is necessary so that triggers and sentences can be semantically matched. Therefore, in the first stage, we propose to jointly train the

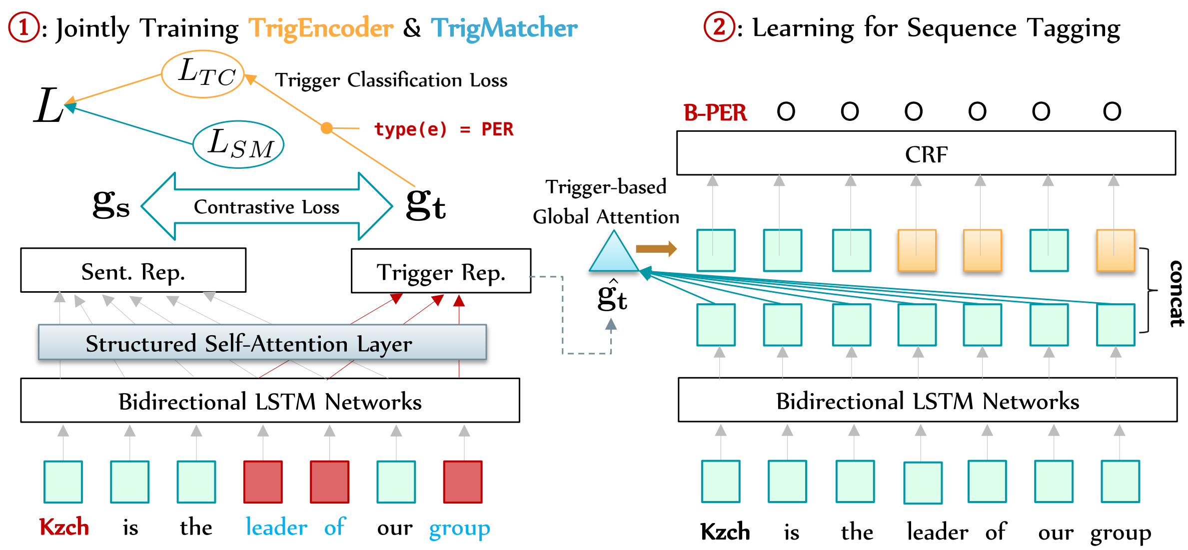


Figure 4.17: Framework overview: Two stage training of the Trigger Matching Network

trigger encoder (`TrigEncoder`) and the attention-based trigger matching module (`TrigMatcher`) using a shared embedding space.

Specifically, for a sentence $\mathbf{x}$ with multiple entities $\{e_1, e_2, \cdots\}$, for each entity e_i we assume that there is a set of triggers $T_i = \{t_1^{(i)}, t_2^{(i)}, \cdots\}$ without loss of generality. To enable more efficient batch-based training, we reformat the trigger-based annotated dataset $\mathcal{D}_T$ such that each new sequence contains only one entity and one trigger. We then create a training instance by pairing each entity with one of its triggers, denoted $(\mathbf{x}, e_i, t_j^{(i)})$.

For each reformed training instance $(\mathbf{x}, e, t)$, we first apply a bidirectional LSTM (BLSTM) on the sequence of word vectors[††] of $\mathbf{x}$, obtaining a sequence of hidden states that are the contextualized word representations $\mathbf{h}_i$ for each token x_i in the sentence. We use $\mathbf{H}$ to denote the matrix containing the hidden vectors of all of the tokens, and we use $\mathbf{Z}$ to denote the matrix containing the hidden vectors of all trigger tokens inside the trigger t.

[††]Here, by "word vectors" we mean the concatenation of external GloVe [183] word embeddings and char-level word representations from a trainable CNN network [156].

In order to learn an attention-based representation of both triggers and sentences, we follow the self-attention method introduced by [142] as follows:

$$\vec{a}_{sent} = \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{H}^T\right)\right)$$

$$\mathbf{g_s} = \vec{a}_{sent}\mathbf{H}$$

$$\vec{a}_{trig} = \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{Z}^T\right)\right)$$

$$\mathbf{g_t} = \vec{a}_{trig}\mathbf{Z}$$

W_1 and W_2 are two trainable parameters for computing self-attention score vectors $\vec{a}_{sent}$ and $\vec{a}_{trig}$. We obtain a vector representing the weighted sum of the token vectors in the entire sentence as the final sentence vector $\mathbf{g_s}$. Similarly, $\mathbf{g_t}$ is the final trigger vector, representing the weighted sum of the token vectors in the trigger.

We want to use the type of the associated entity as supervision to guide the trigger representation. Thus, the trigger vector $\mathbf{g_t}$ is further fed into a multi-class classifier to predict the *type* of the associated entity e (such as `PER`, `LOC`, etc) which we use $\texttt{type}(e)$ to denote. The loss of the trigger classification is as follows:

$$L_{TC} = -\sum \log \text{P}\left(\texttt{type}(e) \mid \mathbf{g_t}; \theta_{TC}\right),$$

where θ_{TC} is a model parameter to learn.

Towards learning to match triggers and sentences based on attention-based representations, we use contrastive loss [77]. The intuition is that similar triggers and sentences should have close representations (i.e., have a small distance between them, d). We create negative examples (i.e., mismatches) for training by randomly mixing the triggers and sentences, because `TrigMatcher` needs to be trained with both positive and negative examples of the form (sentence, trigger, label). For the negative examples, we expect a margin

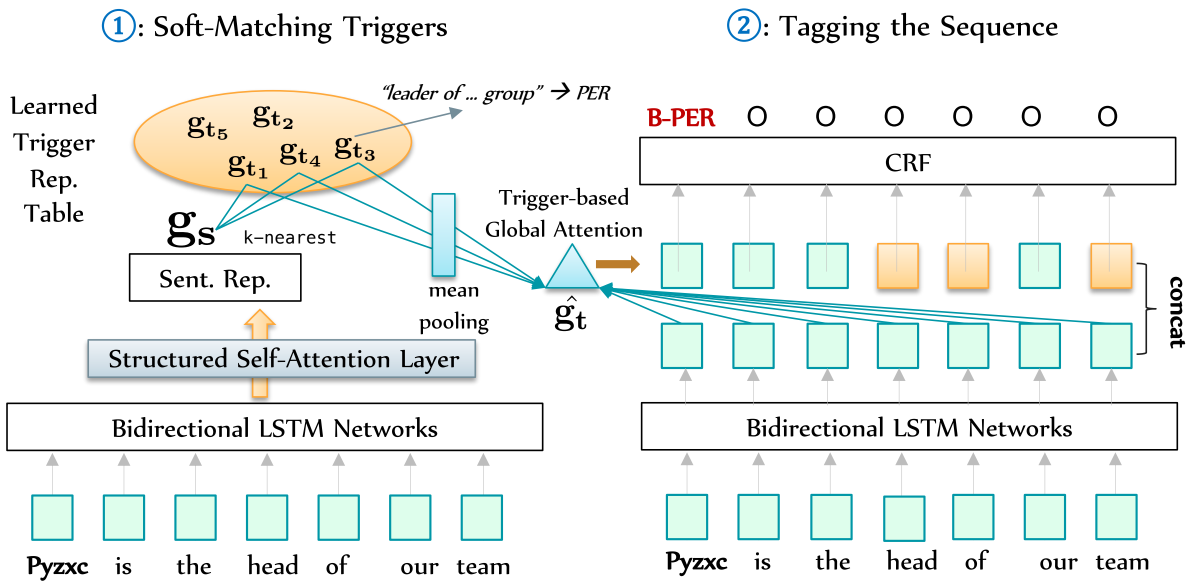


Figure 4.18: Framework overview: The inference process of Trigger Matching Network

m between their embeddings. The contrastive loss of the soft matching is defined as follows, where $\mathbf{1}_{\text{matched}}$ is 1 if the trigger was originally in this sentence and 0 if they are not:

$$d = \|\mathbf{g_s} - \mathbf{g_t}\|_2$$

$$L_{SM} = \mathbf{1}_{\text{matched}} \frac{1}{2} (d)^2 +$$

$$(1 - \mathbf{1}_{\text{matched}}) \frac{1}{2} \{\max(0, m - d)\}^2$$

The joint loss of the first stage is thus $L = L_{TC} + \lambda L_{SM}$, where λ is a hyper-parameter to tune.

Trigger-Enhanced Sequence Tagging The learning objective in this stage is to output the tag sequence $\mathbf{y}$. Following the most common design of neural NER architecture, BLSTM-CRF [156], we incorporate the entity triggers as attention queries to train a trigger-enhanced sequence tagger for NER. Note that the BLSTM used in the the `TrigEncoder` and `TrigMatcher` modules is the same BLSTM we use in the `SeqTagger` to obtain $\mathbf{H}$, the matrix containing the hidden vectors of all of the tokens. Given a sentence $\mathbf{x}$, we use the previously trained `TrigMatcher` to compute the mean of all the trigger vectors $\hat{\mathbf{g_t}}$ associated with this sentence. Following the conventional attention method [154], we incorporate the mean trigger vector as the query, creating a sequence of attention-based token representations, $\mathbf{H}'$.

$$\vec{\alpha} = \mathrm{SoftMax}\left(\boldsymbol{v}^\top \tanh\left(U_1\mathbf{H}^T + U_2\hat{\mathbf{g}_\mathbf{t}}^T\right)^\top\right)$$

$$\mathbf{H}' = \vec{\alpha}\,\mathbf{H}$$

U_1, U_2, and v are trainable parameters for computing the trigger-enhanced attention scores for each token. Finally, we concatenate the original token representation $\mathbf{H}$ with the trigger-enhanced one $\mathbf{H}'$ as the input ($[\mathbf{H}; \mathbf{H}']$) to the final CRF tagger. Note that in this stage, our learning objective is the same as conventional NER, which is to correctly predict the tag for each token.

Inference on Unlabeled Sentences When inferencing tags on unlabeled sentences, we do not know the sentence's triggers. Instead, we use the `TrigMatcher` to compute the similarities between the self-attended sentence representations and the trigger representations, using the most suitable triggers as additional inputs to the `SeqTagger`. Specifically, we have a trigger dictionary from our training data, $\mathcal{T} = \{t|(\cdot, \cdot, t) \in \mathcal{D}_T\}$. Recall that we have learned a trigger vector for each of them, and we can load these trigger vectors as a look-up table in memory. For each unlabeled sentence $\mathbf{x}$, we first compute its self-attended vector $\mathbf{g_s}$ as we do when training the `TrigMatcher`. Using L2-norm distances to compute the contrastive loss, we efficiently retrieve the most similar triggers in the shared embedding space of the sentence and trigger vectors.

Then, we calculate $\hat{\mathbf{g}_\mathbf{t}}$, the mean of the top k nearest semantically matched triggers, as this serves a proxy to triggers mentioned for the entity type in the labeled data. We then use it as the attention query for `SeqTagger`. Now, we can produce trigger-enhanced sequence predictions on unlabeled data, as shown in Figure 4.18.

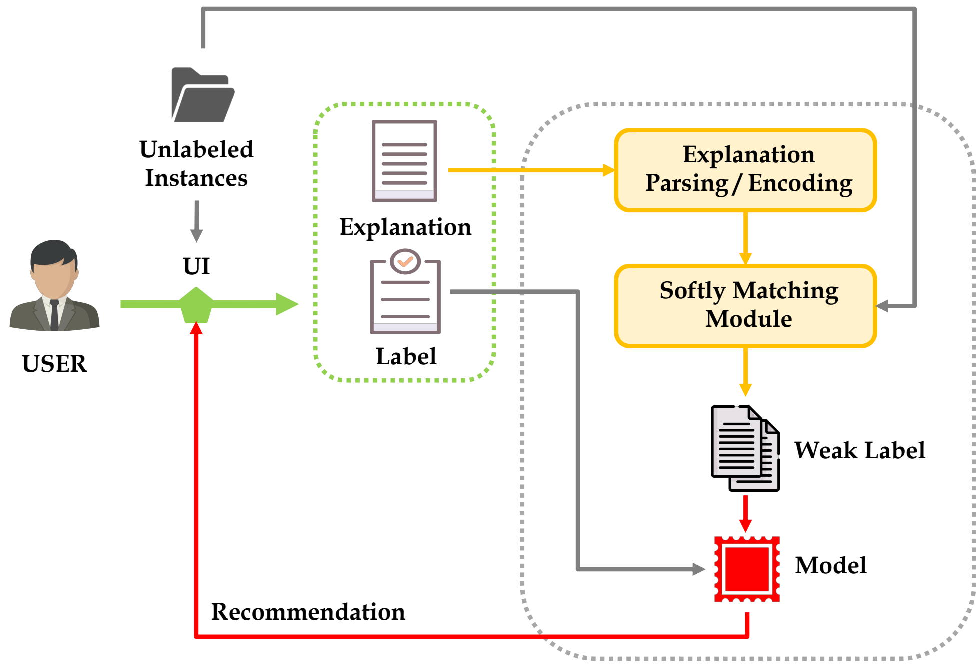


Figure 4.19: Overview of LEAN-LIFE

4.3.1.4 LEAN-LIFE: Explanation-Centric Annotation for NER

As shown in Figure 4.19, our annotation framework consists of two main components, a user-friendly web-UI that can capture labels and explanations for labeling decisions, and a weak supervision framework that parses explanations for the creation of weakly labeled data. The framework then uses this weakly labeled data in conjunction with user-provided labels to train models for improved annotation recommendations. Our UI shows annotators unlabeled instances (can be sampled using active learning), along with annotation recommendations in an effort to reduce annotation costs. We use PyTorch to build our models and implement an API for communication between the web-UI and our weak supervision framework. The learned parameters of our framework are updated in an online fashion, thus improving in near real time.

UI for Capturing Human Explanation The emphasis of our front-end design is to simplify the capture of both label and explanation for each labeling decision, while reducing annotation effort via accessible annotation recommendation. A Trigger is a group of words in the sentence being annotated that aided the annotator's labeling decision, while Natural Language is a written explanation of the labeling decision.

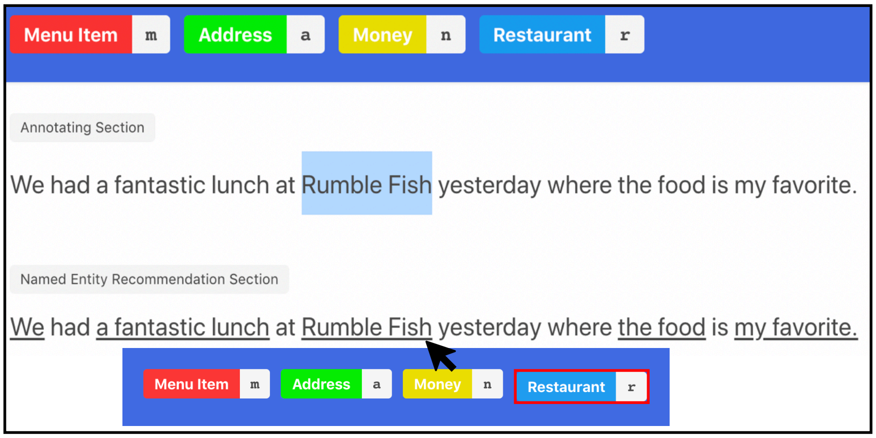
Figure 4.20 illustrates how our framework can capture both a named entity (NE) label and triggers for the sentence "We had a fantastic lunch at Rumble Fish yesterday where the food is my favorite". The user is first presented with a piece of text to annotate (Annotating Section), the available labels that may be applied to sub-sequences (spans) of text (in the blue header) and recommendations of what spans of text should be considered as NE mentions (Named Entity Recommendation Section). The user may choose to select a span of text to label, or they may click on one of the recommended spans below (Figure 4.20 (a)). If the user clicks on a recommended span, a small pop-up displaying the available labels appear with the recommended label circled in red (Figure 4.20 (a)). Once the user selects a label for a span of text by either clicking on the desired label button or via a predefined shortcut key (ex: for Restaurant the shortcut key is **r**), a pop-up appears (Figure 4.20 (b)), asking the user to select helpful spans (triggers) from the text that provide useful context in deciding the label for the NEM−−multiple triggers may be selected. The user may cancel their decision to label a span of text with a label by clicking the *x* button in the pop-up, but if the user wants to proceed and has selected at least one trigger, they finish the labeling by hitting done. Then, their label is visualized in the Annotating Section by highlighting the NEM.

4.3.1.5 Experimental Setup

In this section, we first discuss how to collect entity triggers, and empirically study the data-efficiency of our proposed framework. We use a general domain dataset CoNLL2003 [219] and a bio-medical domain dataset BC5CDR [134]. Both datasets are well-studied and popular in evaluating the performance of neural named entity recognition models such as BLSTM-CRF [156].

In order to collect the entity triggers from human annotators, we use *Amazon SageMaker Ground Truth*[‡‡] to crowd-source entity triggers. Specifically, we sample 20% of each training set as our inputs,

[‡‡] An advanced version of *Amazon Mechanical Turk*. https://aws.amazon.com/sagemaker/



(a) the labels appear in the header, followed by an annotating section; tagging suggestions are shown as underlined spans at the bottom of the page. A user may hover over a tagging suggestion or select a span in order to apply a label to a substring.

(b) after clicking a label to assign to a text span, a pop up appears asking the user to explain their decision by selecting nearby "trigger" text spans.

Figure 4.20: The workflow to annotate label and trigger span in LEAN-LIFE

and then reform them to be the same format. Annotators are asked to annotate a group of words that would be helpful in typing and/or detecting the occurrence of a particular entity in the sentence. We masked the entity tokens with their types so that human annotators are more focused on the non-entity words in the sentence when considering the triggers. We consolidate multiple triggers for each entity by

Dataset	Entity Type	# of Entities	# of Triggers	Avg. # of Triggers per Entity	Avg. Trigger Length
CONLL 2003	PER	1,608	3,445	2.14	1.41
	ORG	958	1,970	2.05	1.46
	MISC	787	2,057	2.61	1.4
	LOC	1,781	3,456	1.94	1.44
	Total	5,134	10,938	2.13	1.43
BC5CDR	Disease	906	2,130	2.35	2.00
	Chemical	1,085	1,640	1.51	1.99
	Total	1,991	3,770	1.89	2.00

Table 4.12: Data statistics of entity triggers

taking the intersection of the three annotators' results. Statistics of the final curated triggers are summarized in Table 4.12. We release the 14k triggers to the community for future research in trigger-enhanced NER.

Base model We require a base model to compare with our proposed TMN model in order to validate whether the TMN model effectively uses triggers to improve model performance in a limited label setting. We choose the CNN-BLSTM-CRF [156] as our base model for its wide usage in research of neural NER models and applications. Our TMNs are implemented within the same codebase and use the same external word vectors from GloVE [183]. The hyper-parameters of the CNNs, BLSTMs, and CRFs are also the same. This ensures a fair comparison between a typical non-trigger NER model and our trigger-enhanced framework.

4.3.1.6 Experimental Results

Labeled data efficiency. We first seek to study the cost-effectiveness of using triggers as an additional source of supervision. Accordingly, we explore the performance of our model and the baseline for different fractions of the training data. The results on the two datasets are shown in Table 2. We can see that by using only 20% of the trigger-annotated data, TMN model delivers comparable performance as the baseline

CONLL 2003										
	BLSTM-CRF				TMN			TMN + SELF-TRAINING		
sent.	Precision	Recall	F1	**trig.**	Precision	Recall	F1	Precision	Recall	F1
5%	70.85	67.32	69.04	3%	76.36	74.33	75.33	80.36	75.18	77.68
10%	76.57	77.09	76.83	5%	81.28	79.16	80.2	81.96	81.18	81.57
20%	82.17	80.35	81.3	7%	82.93	81.13	82.02	82.92	81.94	82.43
30%	83.71	82.76	83.23	10%	84.47	82.61	83.53	84.47	82.61	83.53
40%	85.31	83.1	84.18	13%	84.76	83.69	84.22	84.64	84.01	84.33
50%	85.07	83.49	84.27	15%	85.61	84.45	85.03	86.53	84.26	85.38
60%	85.58	84.54	85.24	17%	85.25	85.46	85.36	86.42	84.63	85.52
70%	86.87	85.3	**86.08**	**20%**	86.04	85.98	**86.01**	87.09	85.91	**86.5**
BC5CDR										
	BLSTM-CRF				TMN			TMN + SELF-TRAINING		
sent.	Precision	Recall	F1	**trig.**	Precision	Recall	F1	Precision	Recall	F1
5%	63.37	43.23	51.39	3%	66.47	57.11	61.44	65.23	59.18	62.06
10%	68.83	60.37	64.32	5%	69.17	73.31	66.11	68.02	66.76	67.38
20%	79.09	62.66	69.92	7%	64.81	69.82	67.22	69.87	66.03	67.9
30%	80.13	65.3	71.87	10%	71.89	69.57	70.71	69.75	72.75	71.22
40%	82.05	65.5	72.71	13%	73.36	70.44	71.87	75.11	69.31	72.1
50%	82.56	66.58	**73.71**	15%	70.91	72.89	71.89	71.23	73.31	72.26
60%	81.73	70.74	75.84	17%	75.67	70.6	73.05	77.47	70.47	73.97
70%	81.16	75.29	76.12	**20%**	77.47	70.47	**73.97**	75.23	73.83	**74.52**

Table 4.13: Performance (F1) comparison between BLSTM-CRF and trigger matching network

model using 50-70% traditional training data. The drastic improvement in the model performance obtained using triggers thus justifies the slightly additional cost incurred in annotating triggers.

Self-training with triggers. We also do a preliminary investigation of adopting self-training [195] with triggers. We make inferences on unlabeled data and take the predictions with high confidences as the weak training examples for continually training the model. The confidence is computed following the MNLP metric [210], and we take top 20% every epoch. With the self-training method, we further improve the TMN model's F-1 scores by about 0.5∼1.0%.

Annotation time vs. performance. Although it is hard to accurately study the time cost on the crowd-sourcing platform we use[§§], based on our offline simulation we argue that annotating both triggers and

[§§]Annotators may suspend jobs and resume them without interaction with the crowd-sourcing platform.

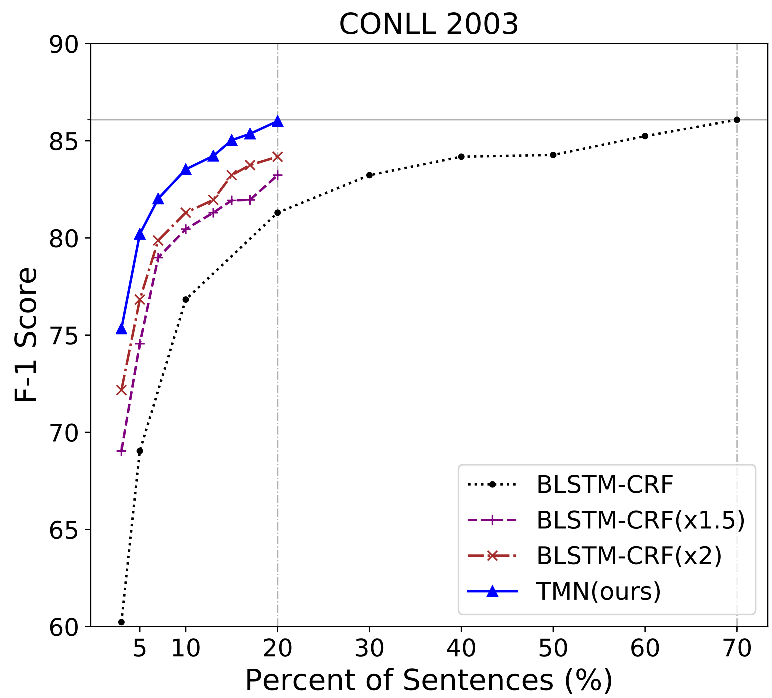


Figure 4.21: The study of cost effectiveness of trigger matching network

entities are about 1.5 times ("BLSTM-CRF (x1.5)") longer than only annotating entities. our offline simulation. In Figure 4.21, The x-axis for BLSTM-CRF means the number of sentences annotated with only entities, while for TMN means the number of sentences tagged with both entities and triggers. In order to reflect human annotators spending 1.5 to 2 times as long annotating triggers and entities as they spend annotating only entities, we stretch the x-axis for BLSTM-CRF. For example, the line labeled ("BLSTM-CRF (x2)") associates the actual F1 score for the model trained on 40% of the sentences with the x-axis value of 20%. We can clearly see that the proposed TMN outperforms the BLSTM-CRF model by a large margin. Even if we consider the extreme case that tagging triggers requires twice the human effort ("BLSTM-CRF (x2)"), the TMN is still significantly more labor-efficient in terms of F1 scores.

Interpretability. Figure 4.22 shows two examples illustrating that the trigger attention scores help the TMN model recognize entities. The training data has 'per day' as a trigger phrase for chemical-type entities,

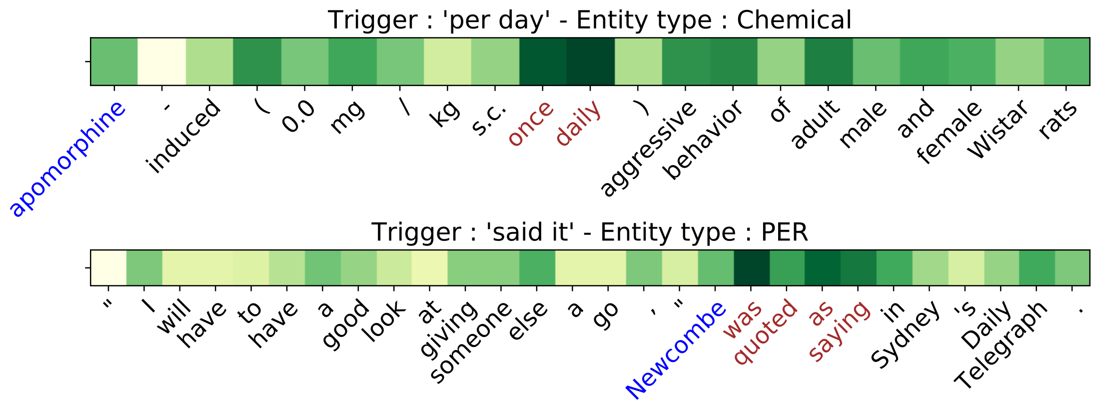


Figure 4.22: Case study of trigger attention during inference

and this trigger matches the phrase 'once daily' in an unseen sentence during the inference phase of `TrigMatcher`. Similarly, in CoNLL03 the training data trigger phrase 'said it' matches with the phrase 'was quoted as saying' in an unlabeled sentence. These results not only support our argument that trigger-enhanced models such as TMN can effectively learn, but they also demonstrate that trigger-enhanced models can provide reasonable interpretation, something that lacks in other neural NER models.

4.3.2 Model Refinement via Explanation-guided Regularization

4.3.2.1 Introduction

Neural language models have achieved remarkable performance on a wide range of natural language processing (NLP) tasks [216]. However, studies have shown that such NLP models are susceptible to learning spurious biases (*i.e.,* bugs) that work on specific datasets but do not properly reflect the underlying task [2, 65, 52, 198]. For example, in hate speech detection, existing NLP models often associate certain group identifiers (*e.g.*, *black*, *muslims*) with hate speech, regardless of how these words are actually used [106] (Fig. 4.23). This poses serious concerns about the usage of NLP models for high-stakes decision-making [12].

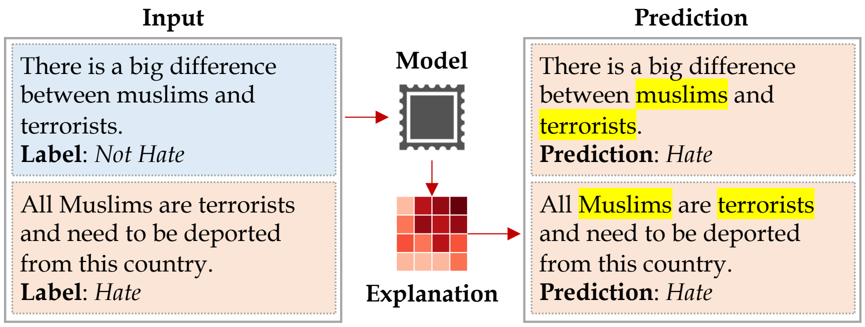


Figure 4.23: Motivation example of spurious correlation in the machine learning model

In response, many methods have been proposed for debiasing either the model or the dataset. Model debiasing can be done via techniques like instance reweighting [204], confidence regularization [222], and model ensembling [84, 33]. Dataset debiasing can be done via techniques like data augmentation [96, 102] and adversarial filtering [262, 117]. However, these methods lack knowledge of which spurious biases actually impacted the model's decisions, which greatly limits their debiasing ability.

On the other hand, *explanation-based model debugging* focuses on addressing spurious biases that actually influenced the given model's decision-making [213, 130, 81]. In this paradigm, a human-in-the-loop (HITL) user is given explanations of the model's behavior [218, 212] and asked to provide feedback about the behavior. Then, the feedback is used to update the model, in order to correct any spurious biases detected via the user feedback. While existing model debugging methods have shown promise [90, 129, 278, 193], their prototype-level implementations provide limited end-to-end utility (*i.e.,* explanation generation, explanation visualization, user feedback collection, model updating, model deployment) for practical use cases.

Given the interactive nature of explanation-based model debugging, it is important to have a user-friendly framework for executing the full debugging pipeline. To achieve this, we propose the E**X**planation-Based NLP **M**odel **D**ebugger (**XMD**). Compared to prior works, XMD makes it simple for users to debug NLP models and gives users significant control over the debugging process (Fig. 4.24). Given either task (model behavior over all instances) or instance (model behavior w.r.t. a given instance) explanations, users can flexibly provide various forms of feedback (*e.g., add* or *remove* focus on a given token) through an easy-to-use, web-based user interface (UI). To streamline user feedback collection, XMD's UI presents intuitive visualizations of model explanations as well as the different options for adjusting model behavior (Fig. 4.25-4.26). After receiving user feedback, XMD automatically updates the model in real time, by regularizing the model so that its explanations align with the user feedback [100]. XMD also provides various algorithms for conducting model regularization. The newly debugged model can then be downloaded and imported into real-world applications via Hugging Face [241]. We summarize our contributions as follows:

1. **End-to-End Model Debugging**: XMD packages the entire model debugging pipeline (*i.e.,* explanation generation, explanation visualization, user feedback collection, model updating, model deployment) as a unified system. XMD is agnostic to the explanation method, user feedback type, or model regularization method. XMD can improve models' out-of-distribution (OOD) performance on text classification tasks (*e.g.,* hate speech detection, sentiment analysis) by up to 18%.

2. **Intuitive UI**: XMD's point-and-click UI makes it easy for non-experts to understand model explanations and give feedback on model behavior.

3. **Easy Model Deployment**: Given user feedback, XMD automatically updates the model in real time. Users can easily deploy debugged models into real-world applications via Hugging Face.

Framework Overview As shown in Figure 4.24, our framework consists of three main components: *Explanation Generation*; *Explanation Visualization*; and *Explanation-based Model Debugging*. Here, the explanation generation and debugging process are done on the back-end while visualizing explanations and capturing human feedback on them are done on front-end UI.

1. **Explanation Generation**: Humans first input the training data and the model trained on that data into the framework. On the backend, the framework applies a heuristic post-hoc explanation method to generate rationales for the training instances.

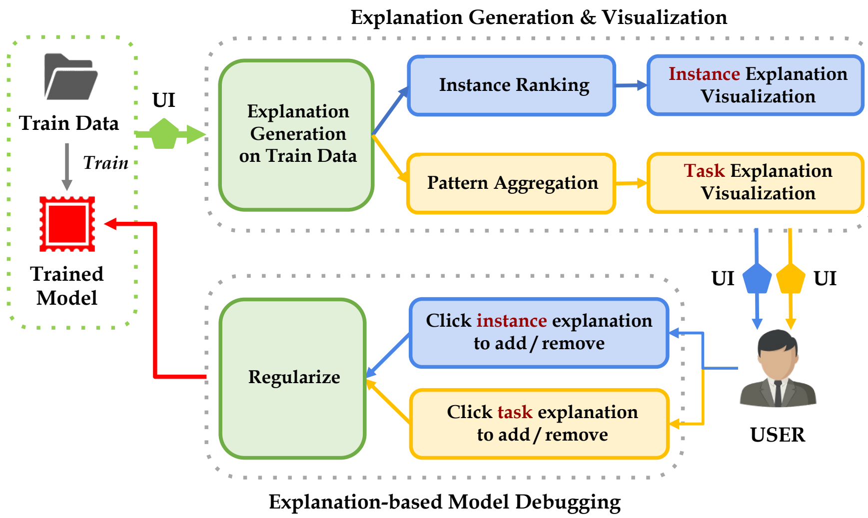


Figure 4.24: System architecture of E**X**planation-Based NLP **M**odel **D**ebugger (XMD)

2. **Explanation Visualization**: The framework displays the generated rationales in two forms: (a) *Instance Explanation*, which shows explanations for individual training instances, and (b) *Task Explanation*, which shows word-level importance aggregated across the dataset.

3. **Explanation-Based Model Debugging**: Users interactively select words—either for individual instances or across tasks—to increase or decrease their importance. Once debugging is complete, the framework retrains the model using a regularization approach and provides a downloadable version of the updated model.

4.3.2.2 XMD: Explanation-Based NLP Model Debugger

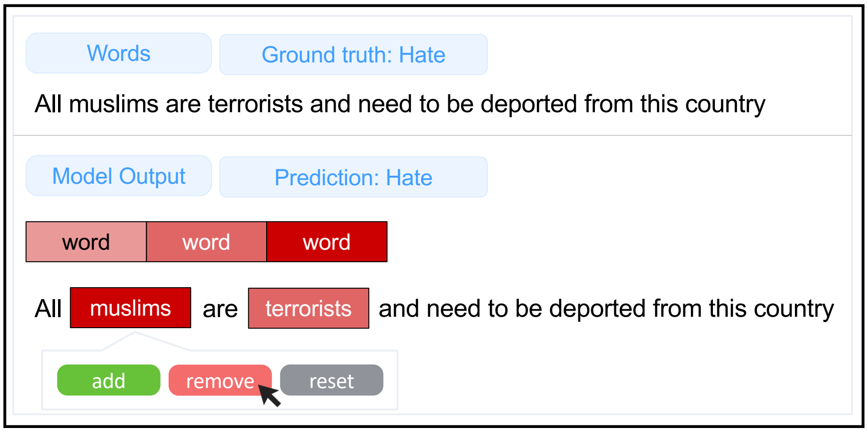
In this section, we present each module of XMD in processing order. To start the process, the user needs to place a training dataset $\mathcal{D}_T$ and a classification model $\mathcal{M}$ that is trained on $\mathcal{D}_T$.

Explanation Generation Our explanation generation module outputs rationales from $\mathcal{M}$. For each instance $\mathbf{x} \in \mathcal{D}_T$, $\mathcal{M}$ generates rationales $\phi(\mathbf{x}) = [\phi(\mathbf{w}_1), \phi(\mathbf{w}_2), \ldots, \phi(\mathbf{w}_n)]$ where $\mathbf{w}_i$ denotes the i-th token in the sentence. Each importance score $\phi(\mathbf{w}_i)$ has a score with regard to all the classes. Our module is exploiting $\phi^p(\mathbf{w}_i)$ which the importance score is attributed to model predicted label p. Here, we exploit heuristic methods that assign importance scores ϕ based on gradient changes in $\mathcal{M}$ [212, 218].

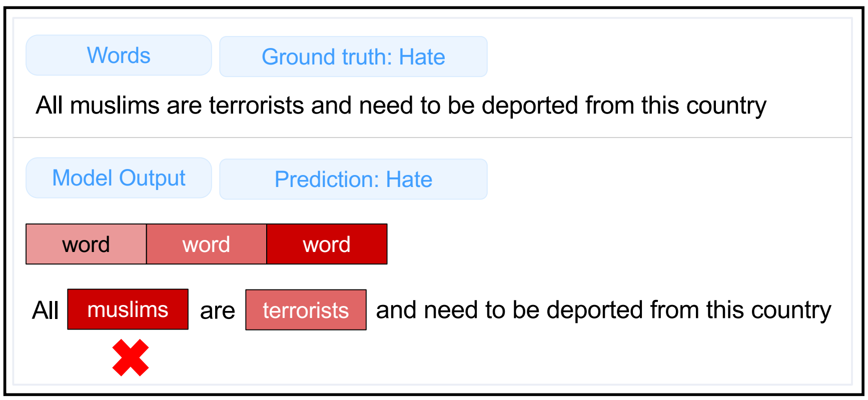
Explanation Visualization Our framework supports visualizing the generated rationale in two different forms, *instance* and *task* explanations. Instance explanations display word importance scores for model predictions for each train instance, while task explanations aggregate and rank words according to their importance to the predicted label. In this section, we first present a UI for visualizing and capturing human feedback for instance explanations and then a UI for task explanations.

1. **Instance Explanation**: Figure 4.25 illustrates how our framework visualizes instance explanations and captures human feedback on them. First, the trained model makes a prediction and generates explanations for one of the train instances that the model correctly predicts: "All muslims are terrorists and need to be deported from this country". The reason why we present only the instances that the model correctly predicts is that we are asking users to provide feedback for the ground truth label and comparing it with $\phi^p(\mathbf{w}_i)$ which the importance score is attributed to the model predicted label p. If p is not equal to the ground truth label, the human feedback would act as a source of incorrect prediction.

   Next, the user is presented with the sentence and its ground truth label on the upper deck (Words Section), and the sentence with highlighted rationales and its predicted label on the lower deck (Model Output Section). Then, the user can choose to select words to decrease or increase its importance toward the ground truth label (Figure 4.25 (a)). If the user clicks the word (*muslims*) that the model is focusing on to predict *hate*, a small pop-up displaying buttons for operation options (*i.e.*, *add*, *remove* and *reset*) appear. Once the user selects a desired operation (*remove*) for the selected word (*muslims*) that is not a right reason for *hate*, that word in the model output section is marked



(a) As a user clicks on a word in the sentence, pop-up displaying operation options and a user selects an appropriate operation for that word.



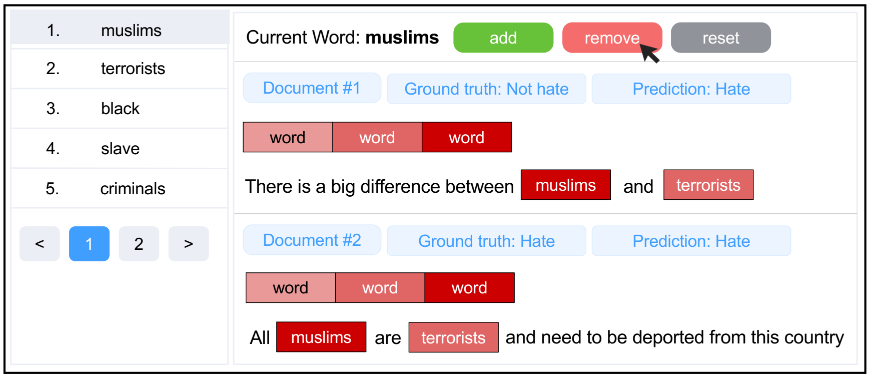
(b) Once the user selects an operation for the selected word, that word in the model output section is marked with an operation symbol (remove: X, add: +).

Figure 4.25: The workflow to provide human feedback on instance-level explanation

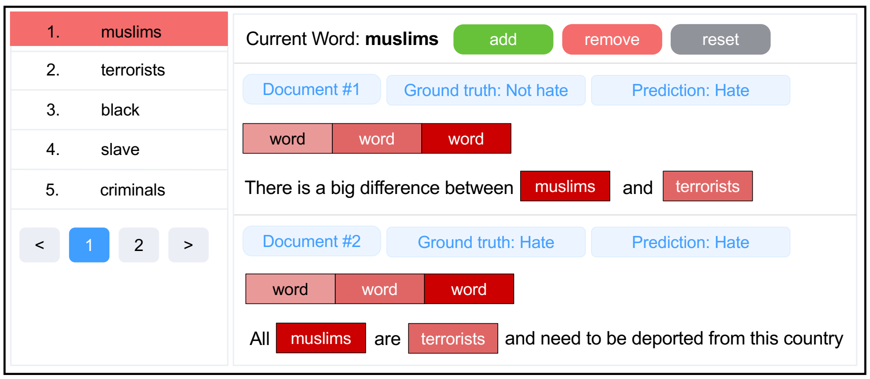
with operation symbol ('X' for *remove*, '+' for *add* – Figure 4.25 (b)). The user may cancel their decision to operation for the word by clicking *reset* in the pop-up.

2. **Task Explanation**

   Figure 4.26 illustrates how our framework visualizes task explanations and captures human feedback on them. First, task explanations are presented in list format on the left panel in descending order of its importance (Figure 4.26 (a)). Here, the importance is a score averaged by the word importance score of all examples containing that word. As user clicks on a word in the list, all the examples



(a) As a user clicks on a word in the list of global explanations in the left panel, examples containing that word are displayed. The user can select the appropriate operation for the word.



(b) After the operation for a word is selected, the word in the left panel is marked with a color of the operation.

Figure 4.26: The workflow to provide human feedback on task-level explanation

containing that word are displayed. The user can then choose to two different operations (*remove* and *add*). If user clicks *remove* for the word (*muslims*) that should not be conditioned on any label (both *hate* and *not hate*), the model will consider it as an unimportant word in all cases. Here, we don't need to consider whether the prediction is correct or not since the word is not important for all the cases (Figure 4.26 (a)). If user clicks *add* for the word that should be useful for the correct prediction, the model will consider it as an important word for the ground truth label. Here, we consider it as an important word only for the correct prediction. After the operation for a word is selected, the word in the left panel is marked with a color of that operation (red for *remove* and green for *add*).

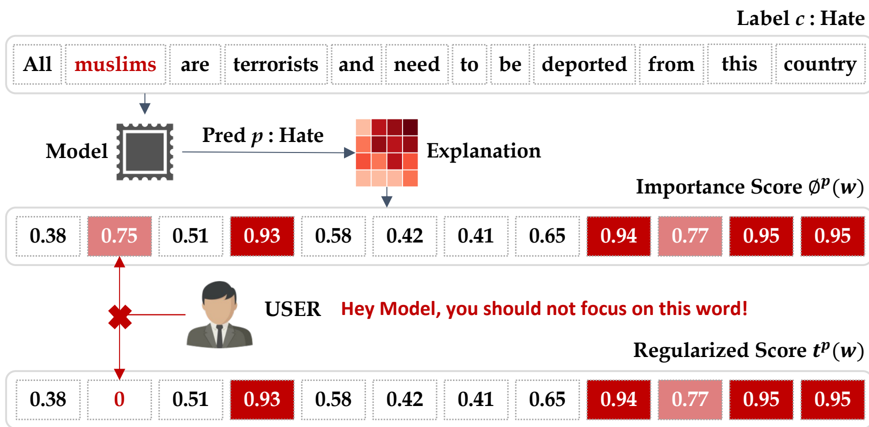


Figure 4.27: The workflow of instance-level explanation-based model debugging

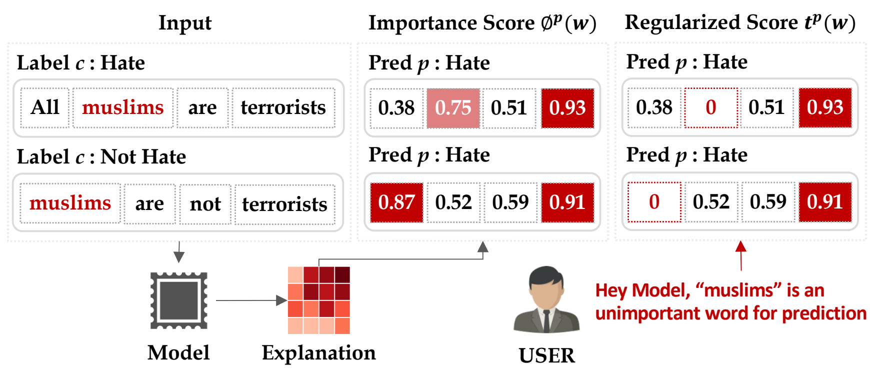


Figure 4.28: The workflow of task-level explanation-based model debugging

Explanation-based Model Debugging Our explanation-based model debugging module is based on explanation regularization (ER) which regularizes model to produce rationales that align to human rationales [258, 196, 145, 67, 105, 194, 139, 89, 100]. Existing works require the human to annotate rationales for each training instance or apply task-level human priors (*e.g.*, task-specific lexicons) across all training instances before training. Despite its effectiveness, the regularized model may not be fully free of hidden biased patterns. To catch all the hidden biased patterns, our framework asks the human to provide binary feedback (*i.e.*, click to add or remove) given the current model explanations and use them to regularize the model. Here we ask the human to provide feedback to the model in order to output the "**correct prediction**".

For the instance explanation, as shown in Figure 4.27, the trained model $\mathcal{M}$ generates rationales $\phi^p(\mathbf{x}) = [\phi^p(w_1), \phi^p(w_2), \ldots, \phi^p(w_n)]$, where $\phi^p(w_i)$ denotes the importance score of i-th token in the sentence $\mathbf{x}$ towards model predicted label p. As the user selects a word (*muslims*) that is spuriously correlated with the correct prediction p (*hate*), the regularized score $t^p(w_i)$ where i is a user-selected word index ($w_2 =$ *muslims*) becomes 0 (See Figure 4.27). For the task explanation, we aggregate words based on its score averaged by the word importance score of examples containing that word, and present them in a descending order. When the user clicks a word w (*muslims*) to decrease its importance, then regularized score $t^p(w)$ where w is user-selected word ($w =$ *muslims*) for all the examples become 0 (See Figure 4.28).

After the click process, the user can start the debugging process based on the examples labeled so far. Here, the learning objective for re-training the model $\mathcal{M}$ is $\mathcal{L} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{ER}}$, where $\mathcal{L}_{\text{task}}$ is a cross-entropy loss for traditional sequence classification tasks and $\mathcal{L}_{\text{ER}}$ is an explanation regularization loss which minimizes the distance between $\phi^p(w)$ and $t^p(w_i)$ [100]. In this framework, we support two different regularization loss: Mean Squared Error (MSE) [144, 106, 196], Mean Absolute Error (MAE) [194].

4.3.2.3 Experimental Setup

We conduct extensive experiments investigating how our debugging process affects the performance on in-distributed (ID) and out-of-distribution (OOD) data, and the model explanation. Here, we present experimental results on sentiment analysis for the instance explanation and hate speech detection for the task explanation. For base model, we use `BigBird-Base` [257].

Tasks and Datasets For sentiment analysis, we exploit SST [214] as the ID dataset, and Yelp (restaurant reviews) [264], Amazon (product reviews) [162] and Movies (movie reviews) [258, 47] as OOD datasets. To simulate human feedback for the instance explanation, we leverage ground truth rationales for SST [21] as human feedback. For hate speech detection, we use STF [68] as the ID dataset, and HatEval [10], Gab Hate Corpus (GHC) [104] and Latent Hatred [55] for OOD datasets. To simulate human feedback for the task

Regularize	ER Loss	Sentiment Analysis			
		In-distribution	Out-of-Distribution		
		SST	Amazon	Yelp	Movies
None	None	93.4	<u>89.1</u>	89.0	82.0
Correct	MSE	**94.7**	88.4	<u>91.8</u>	**94.5**
	MAE	<u>94.0</u>	**92.3**	**94.4**	<u>94.0</u>

Table 4.14: Performance (Accuracy) comparison with instance-level explanation on ID/OOD performance

explanations, we leverage group identifiers (*e.g.*, *black*, *muslims*) [106] as words that need to be discarded for determining whether the instance is hate or not.

Implementation Details To start XMD, users should input the trained model following Hugging Face model structure [241]. After users input the train data and the model, our framework uses Captum [110] to generate explanation. For visualizing the explanation and capturing the human feedback, we implement UI using Vue.js [¶¶]. Here, we re-use UI components from LEAN-LIFE, an explanation-based annotation framework [124], for capturing human feedback. To train the model with ER, we use PyTorch [179] and Huggingface [241].

4.3.2.4 Experimental Results

ID/OOD Performance Table 4.14 shows the performance on ID and OOD when regularize on correct predictions using its instance explanation. We see that our framework helps model to not only do much better on ID data, but also generalize well to OOD data. For task explanation, we present performance by regularizing on correct and incorrect prediction and all the instances regardless of prediction. Table 4.14 presents the performance with *remove* operations for task explanations (*i.e.*, group identifiers) for incorrect predictions, correct predictions, and for all instances, respectively. We observe that our framework helps model not to focus on the words that should not be conditioned on any label and lead to performance enhancement on both ID and OOD data.

[¶¶]https://vuejs.org/

Regularize	ER Loss	Hate Speech Analysis			
		In-distribution	Out-of-Distribution		
		STF	HatEval	GHC	Latent
None	None	89.5	88.2	64.5	67.2
Correct	MSE	89.2	**90.1**	62.3	67.9
	MAE	89.1	**90.1**	59.3	64.9
Incorrect	MSE	88.9	86.3	**67.9**	**70.3**
	MAE	89.3	88.8	64.2	67.6
ALL	MSE	**90.0**	88.4	63.8	67.0
	MAE	89.7	86.9	66.5	70.2

Table 4.15: Performance (Accuracy) comparison with task-level explanation on ID/OOD performance

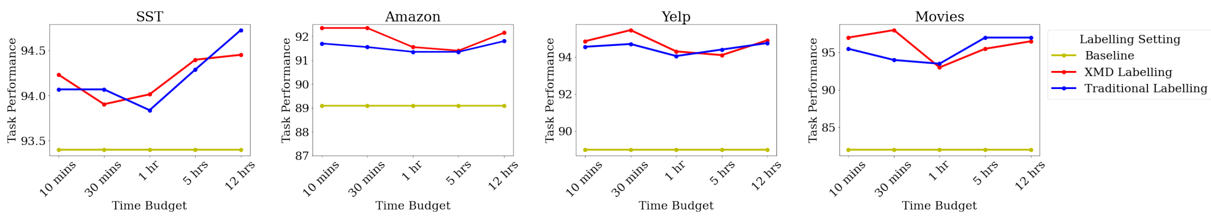


Figure 4.29: Performance (Accuracy) trend by annotation efforts

Efficiency To quantify the advantage that XMD provides, we compare the time taken to annotate instances using XMD versus traditional labeling for instance explanations. While XMD requires humans to interact with a trained model and decrease or increase importance scores of words, traditional labeling is not model-in-the-loop in nature, and requires users to directly annotate binary importance scores to words in the instance [47, 21]. We ask two graduate students to annotate 50 instances, using the traditional and the XMD labeling methods. For both of these labeling settings, we ensure that there is no overlap between the instances, so as to avoid familiarity and record the time taken to annotate each instances. Upon aggregating across all instances and both annotators, it is found that one instance takes ~ 60 seconds and ~ 110 seconds to be annotated using the framework and the traditional labeling method respectively. Using this time estimate, we simulate the time-efficiency of these two labeling methods with varying amounts of time budgets for annotations. Figure 4.29 presents our results for this experiment. We note that although both labeling methods outperforms the baseline of no explanation annotation, using XMD is particularly helpful when the time budget given is limited (< 1 hour), especially in the OOD setting (Amazon, Yelp, Movies datasets).

4.4 Summary

This chapter demonstrates that contextual supervision during training significantly enhances language model behavior across robustness, generalization, and label efficiency.

We first showed that appending contextual information, such as in-context task demonstrations and dialogue histories, leads to consistent improvements in predictive accuracy and resilience across tasks like named entity recognition and social norm violation detection. These results show that **language models benefit from being explicitly trained to utilize context**, rather than relying solely on context at inference time.

Next, we explored how human-provided explanations serve as a powerful form of contextual guidance. Through explanation-driven supervision, models can achieve stronger performance with fewer labeled examples, as demonstrated in tasks like NER using **TriggerNER** and **LEAN-LIFE**. Furthermore, we introduced a post hoc refinement method, **XMD**, where explanations regularize model behavior, enabling targeted debugging and correction when humans provide explanations for the errors.

Together, these findings highlight that contextual signals, whether provided as additional input or through explanations, play a central role in shaping and aligning model behavior. This highlights the importance of further research into incorporating structured, semantically rich context directly into model training pipelines.

Chapter 5

Language Models as Self-Refining Context Generators

This chapter explores how language models can improve their behavior through self-refinement, using simulated interaction and feedback as a source of contextual supervision. Rather than relying solely on static, human-curated context, this thesis hypothesizes that models can actively generate, evaluate, and revise their outputs by constructing internal feedback loops. This chapter investigates this in the domain of educational question generation, where the model simulates learner interactions to iteratively improve the quality of its questions.

5.1 Background and Motivation

5.1.1 Self-Refinement Through Simulated Interaction as Contextual Supervision

We hypothesize that models can improve their outputs through a self-refinement process, where context is constructed, evaluated, and iteratively updated based on feedback. To test this, we develop methods that enable models to simulate interactions and use the resulting signals as contextual input for further improvement.

A key case study is educational question generation. Here, the model first generates questions aimed at enhancing a learner's understanding of a given text. It then simulates a learner's response and uses that feedback as contextual supervision to update its parameters. Through this process, the model learns

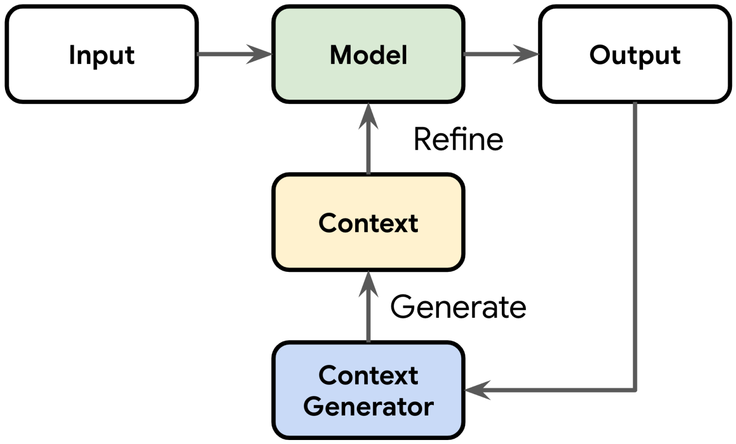


Figure 5.1: Experiment setting for self-refinement through simulated interaction as contextual supervision

to generate more effective questions over time. This approach highlights how simulated interaction and feedback-driven refinement can lead to more socially intelligent and adaptive language models.

5.2 Self-Refinement for Educational Question Generation

5.2.1 Introduction

Asking good questions offers broad benefits. Effective teachers ask questions to promote deeper understanding [14, 220] and encourage divergent thinking [32, 72, 223], while high-performing students ask clarifying questions or self-reflect to refine their understanding [1, 41, 71, 18, 224, 43]. In addition, asking questions helps establish common ground [189, 4, 254, 260, 240] and thus catalyzes favorable outcomes in various applications, such as therapy [205], response generation [273, 30], and content moderation [29].

But what characterizes *good* questions and how do we craft them? Prior work in psychology literature is limited to general guidelines for assessing question quality based on heuristics [72, 223] or experiments within a specific domain such as language learning [85] or chemistry [43]. On the other hand, recent work in natural language processing measures question quality with indirect metrics, such as *expected information gain*, the additional information provided by an answer to a question [143, 189, 254, 240, 103], and *saliency*, how effectively a question introduces new concepts or clarifies key ideas that appear in future content [244]. These works cannot directly evaluate a question's usefulness since its impact on downstream tasks is challenging to quantify.

In this work, we overcome this challenge by simulating a person's knowledge using language models (LMs) and introduce a direct measure of question *utility*. Building on growing work that advocates LMs as world models [178, 266, 256, 99, 152, 203, 268], we propose QUEST (**Q**uestion **U**tility **E**stimation and **S**imulation **T**ests), a learning environment simulator that quantifies a question's utility—how much it contributes to improved performance on a relevant exam (as illustrated in Figure 5.2)—and fine-tunes question generators with high-utility questions.

Specifically, QUEST assigns an LM to roleplay as a novice learner that asks questions and receives answers while acquiring new information. We estimate the *utility* of each question by simulating the learner's performance on a related exam using different question subsets. Using these utility measurements, we fine-tune the question generator through a rejection sampling approach [8], retaining only questions that surpass a predefined utility threshold.

We evaluate QUEST on TEXTBOOK-EXAM, a curated dataset where each sample consists of a chapter with multiple sections and corresponding exam questions. Using TEXTBOOK-EXAM, we experiment with various question generation strategies and quality metrics (*i.e., utility*, *saliency*, *expected information gain*).

Our experimental results show that (1) Question generators trained with QUEST improve learners' understanding, leading to an average exam score increase of at least 20% across simulations in five subjects; (2) Indirect metrics (*i.e., saliency*, *expected information gain*) show weak correlations (<0.1) with utility, suggesting they do not accurately reflect a question's impact on learning outcomes; (3) Optimizing for indirect metrics does not improve learners' understanding, whereas utility-driven training achieves the best overall performance, even in indirect metrics; (4) Utility has a weak positive correlation with lexical similarity (<0.04) and semantic similarity (<0.25) between generated and exam questions. This suggests that high-utility questions are not simple rephrasings but introduce new concepts that enhance learning;

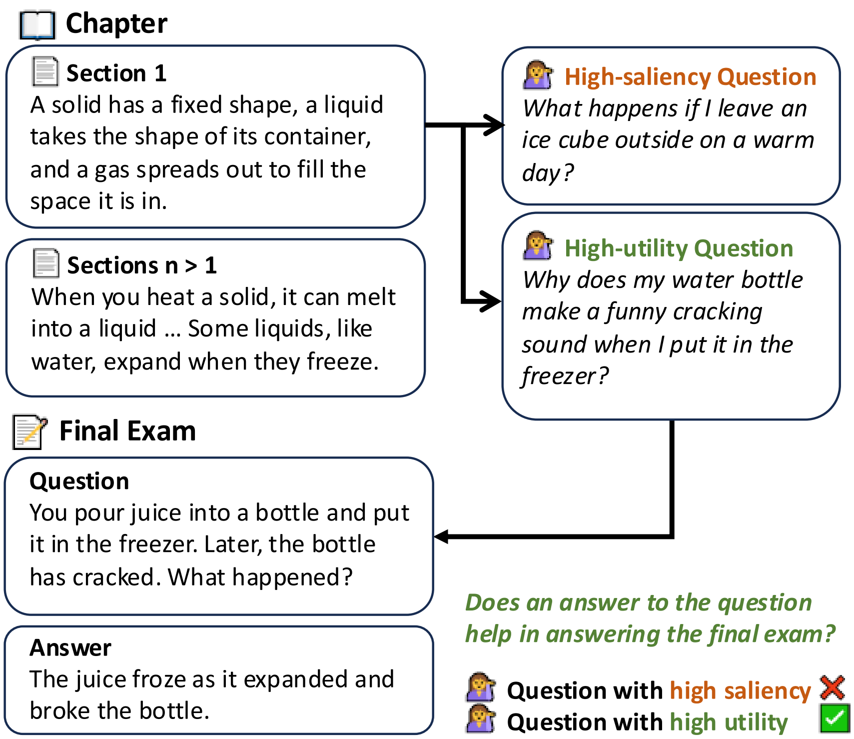


Figure 5.2: Motivation example of high-utility question in education context

(5) There are no clear stylistic patterns among high-utility questions, indicating that their effectiveness is not driven by specific question formats or phrasing.

To summarize, we introduce QUEST, a novel framework that quantifies question utility based on its direct impact on learning outcomes, rather than relying on indirect heuristics. Our approach leverages LLMs as simulators to evaluate a question's contribution to learning, using performance improvements in downstream tasks as a reward signal. We further propose a utility-driven training paradigm for question generation, fine-tuning models through rejection sampling by retaining only high-utility questions. Our results demonstrate that optimizing for direct utility significantly enhances the simulated learner's understanding, surpassing existing methods and offering new insights into what makes a question effective.

5.2.2 Question Generation: Problem Formulation and Dataset (TEXTBOOK-EXAM)

5.2.2.1 Problem Formulation

In this work, we evaluate the system in an educational setting where a student is trying to learn a textbook chapter's content.

Let D be a document and E be a set of exam questions that can be solved using D. The document D is structured as a sequence of sections, denoted by $S_k \subset D$, where each section S_k represents the content at the k-th position in D. For each section S_k, we define $S_{[1:k-1]} = \{S_1, S_2, \ldots, S_{k-1}\} \subset D$ as the context, which includes all preceding content in the document up to section S_k.

Let M_q be a question generator that processes the document sequentially, section by section. For each section S_k, it generates a set of questions $Q_k = \{Q_k^1, Q_k^2, \ldots, Q_k^n\}$, where $Q_k \sim M_q(S_k, S_{[1:k-1]})$, indicating that the questions are generated based on the anchor section S_k and the preceding context $S_{[1:k-1]}$.

Let M_s be a reader simulator. We assess the effectiveness of the question generator M_q by measuring the performance of M_s on the exam E using only the generated questions Q, where $Q = \{Q_1, Q_2, \ldots, Q_k\}$ represents the set of all questions produced by M_q across all sections. This is expressed as $M_s(E \mid Q)$, evaluating how well the generated questions contribute to solving E without direct access to D.

Our objective is to design a question generator M_q that maximizes $M_s(E \mid Q)$, under the assumption that questions contributing more effectively to solving E are high-utility questions.

5.2.2.2 Dataset: TEXTBOOK-EXAM

In order to evaluate, we curate TEXTBOOK-EXAM, a dataset where each entry contains a document D along with a corresponding set of exam questions E. An overview of TEXTBOOK-EXAM is illustrated in Figure 5.3.

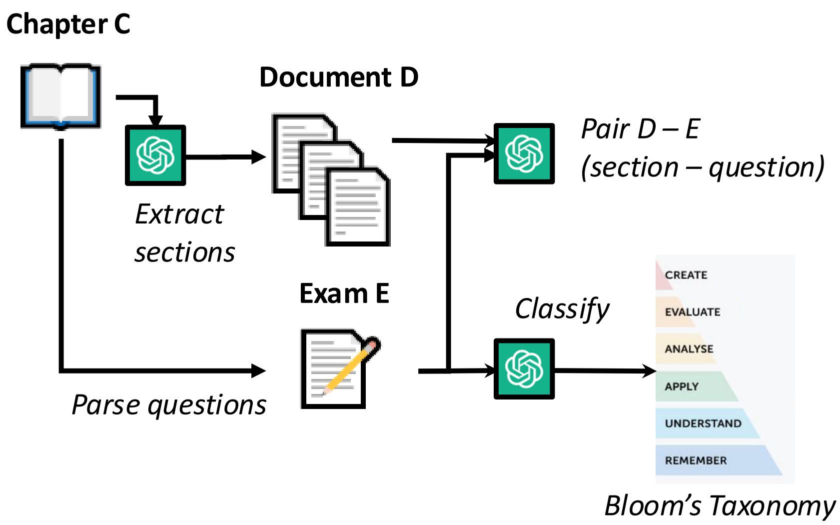


Figure 5.3: Data construction overview for studying educative question generation

Data Processing Our pipeline starts with textbooks from the OpenStax repository[*]. Each textbook is divided into chapters, where each chapter C contains learning objectives, main content, and review questions. For each C, we parse the main content to build D and the review questions to form E.

1. **Extracting Sections.** To simulate a learner incrementally progressing through a chapter, we divide each chapter into sections using an LM-based document structuring method. The language model segments D into n sections, denoted as $\{S_1, S_2, \ldots, S_n\} \subset D$, while also extracting the corresponding review questions E. However, not all review questions include ground-truth answers, as some textbooks omit them (see Table 5.1 for the proportion of E with answers). To ensure consistent segmentation across different subjects, we manually annotate the first 2–5 sections from one sample per subject. These annotated samples are used as few-shot examples in our LM prompt.

2. **Extracting Questions.** To balance evaluation depth with computational feasibility, we include only chapters that contain at least 10 review questions—ensuring sufficient coverage for assessment—while capping the maximum number of questions at 25 to keep learning simulations tractable.

[*] https://github.com/philschatz/textbooks

Subject	**#** C	**Split**	**#** E **/** C	**%** E **w/ answer**	**#** S **/** C
Microbiology	20	Train	12.4	64%	16.4
	5	Test	13.4	58%	17.0
Chemistry	20	Train	14.2	51%	11.0
	5	Test	16.2	49%	6.4
Economics	20	Train	12.2	23%	14.1
	5	Test	12.2	23%	14.4
Sociology	20	Train	10.4	62%	16.6
	5	Test	11.2	67%	19.0
US History	20	Train	7.2	51%	14.9
	5	Test	8.4	38%	13.2

Table 5.1: Data statistics of TEXTBOOK-EXAM

Data Statistics For each subject, we curate 25 sequential chapters C, each containing both D and E. The chapters are arranged in their natural order, with the first 20 used for training and the last five reserved for evaluation. There is the risk that content in later chapters may include information from prior chapters (e.g., revisiting prerequisite knowledge). Therefore, preserving this sequential structure between the training and test set is essential for preventing information leakage and fairly assessing a model's learning process. Table 5.1 shows an overview of the statistics of the resulting TEXTBOOK-EXAM.

Distribution of Question Types To better understand how final exams assess a learner's comprehension on multiple dimensions, we categorize questions in TEXTBOOK-EXAM based on the revised *Bloom's Taxonomy* [111]. Using an LM, we assign a cognitive depth d_j to each question $E_j \in E$, classifying them into six categories: *Remembering, Understanding, Applying, Analyzing, Evaluating*, and *Creating*. Additionally, we identify the relevant sections $S_j \subset D$ that correspond to each question.

The distribution, shown in Figure 5.4, indicates that different subjects emphasize different cognitive skills. For instance, questions in Microbiology and Sociology primarily focus on *Remembering* and *Understanding*, whereas Chemistry and Economics exhibit a more varied distribution. This analysis highlights

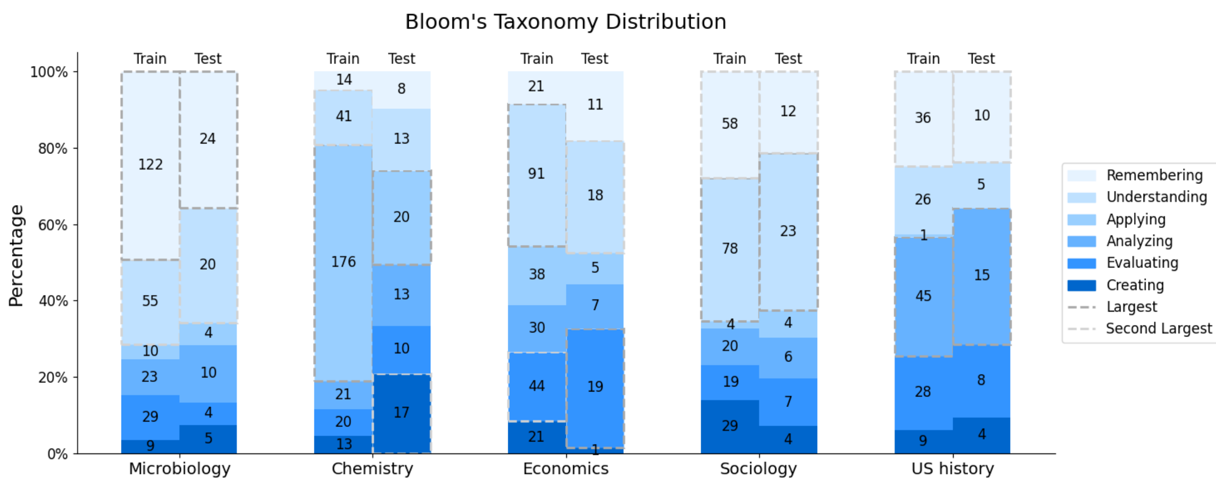


Figure 5.4: Data statistics of TEXTBOOK-EXAM in a perspective of Bloom's taxonomy distribution

the diverse cognitive demands across subjects and underscores how TEXTBOOK-EXAM provides a multifaceted evaluation of learning outcomes through final exams.

5.2.3 QUEST: Question Utility Estimation and Simulation Tests

In this section, we introduce QUEST as a method for measuring question utility and using high-utility questions to fine-tune question generators. We first provide an overview of QUEST and delve into the details of each its procedures.

Overview QUEST consists of the following: (1) a **question generator** (M_q), which takes a document D and generates a set of questions $Q = \{q_1, q_2, \ldots, q_n\}$; (2) an **answer generator** (M_a) that then produces an answer a_i for each question $q_i \in Q$, forming the set of question-answer pairs QA $= \{(q_1, a_1), (q_2, a_2), \ldots, (q_n, a_n)\}$ using parametric knowledge; (3) a **learner simulator**, which models a learner (M_l)'s understanding by having an evaluator (M_e) assess their performance on a final exam E using only QA; and (4) a **utility estimator**, which runs the learner simulator multiple times with different subsets of QA, estimating the contribution of individual questions to the learner's overall performance. See Figure 5.5.

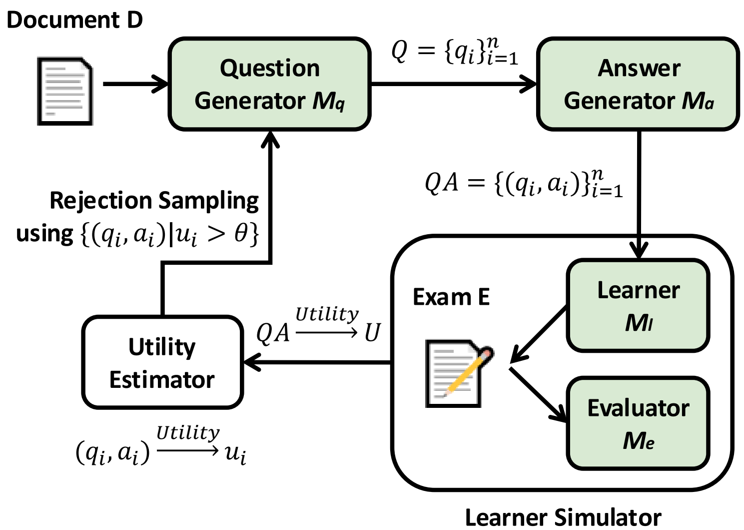


Figure 5.5: Framework overview of QUEST

Question Generation The **Question Generator** (M_q) generates a set of questions (Q) based on the input document. For each ordered section S_k in D, where S_k represents the part of the document the learner is currently reading (referred to as the *anchor*), the question generator considers both S_k and its preceding context $C_k = \{S_1, S_2, \ldots, S_{k-1}\}$ to generate a corresponding set of questions $Q_k = \{q_k^1, q_k^2, \ldots, q_k^n\}$, formulated as $Q_k = M_q(S_k, C_k)$.

This approach ensures that the generated questions are contextually relevant and informed by the surrounding content, aligning with prior research on inquisitive question generation [244]. Once the questions are generated, the **Answer Generator** (M_a) produces an answer a for each question q using the model's parametric knowledge, forming a set of question-answer pairs: QA $= \{(q_1, a_1), (q_2, a_2), \ldots, (q_n, a_n)\}$. These QA pairs serve as the foundation for subsequent evaluation and simulation stages.

Evaluating the Question Generator Once the QA pairs for the document D are generated using M_q and M_a, we assess the effectiveness of the question generator by employing the **Learner Simulator**, which consists of a **Learner** (M_l) and an **Evaluator** (M_e). The learner model M_l simulates a learner's understanding by attempting the final exam E using only the generated QA pairs, producing responses

$P = M_l(E, \mathrm{QA})$. The evaluator model M_e then assesses the learner's responses P by comparing them against ground-truth answers when available or using parametric knowledge to assign a score. The score can serve as a metric to assess the quality of M_q, providing insights into the usefulness (*i.e., utility*) of the generated questions in answering the final exam.

Improving the Question Generator Our objective is to design a question generator M_q that maximizes the learner simulator's performance on E when provided with QA. This assumes that questions contributing more effectively to solving E are high-utility questions.

To enhance the quality of the question generator M_q, we aim to learn the patterns of high-utility questions—those that lead to better performance in the learner simulator. To achieve this, we estimate the utility u of each question-answer pair $(q, a) \in \mathrm{QA}$. The utility estimator runs the learner simulator multiple times with different subsets of QA, computing the *single-one gain*, which is the score obtained using only (q, a), and the *all-but-one gain*, which is the score with all pairs except (q, a). The utility of (q, a) is then computed as the average of these two scores. We retain only the pairs where utility exceeds a threshold θ and train M_q using only these high-utility pairs, following a rejection sampling strategy [8].

5.2.4 Experimental Setup

To assess the effectiveness of QUEST, we compare various question generators based on utility and examine the correlation between utility-based evaluation and existing metrics.

Question Generator. We evaluate our approach against multiple baselines to assess the effectiveness of utility-based question generation (QG). These baselines include: (1) **zero-shot**, which directly prompts the model to generate a question that helps the student better understand the section; (2) **few-shot**, which extends zero-shot by providing five example exam questions from the training set, each paired with its relevant section, to guide the model in generating final exam-style questions [19]; (3) **chain-of-thought**

(CoT), which prompts the model to generate reasoning steps before formulating a question, improving logical coherence [237, 274]; (4) **Bloom-based** prompting, which selects a level in the revised Bloom's Taxonomy [111] with weighted random sampling based on the distribution of the levels among the exam questions in the training data and generates a question that requires this cognitive level for answering. (5) **supervised fine-tuning (SFT)** similar to few-shot but trains the model on exam questions from the training set instead of using in-context prompting, allowing it to learn question patterns over multiple optimization steps; and (6) **QUEST** employs rejection sampling to generate questions and select those with the highest simulated utility, maximizing learning impact.

Both SFT and QUEST are evaluated using two variations: (1) **subject-specific**: Training and testing are performed separately for each subject, meaning the model is trained only on data from a specific subject and evaluated on that same subject; and (2) **cross-subject**: Training is conducted across all subjects collectively by aggregating the entire training dataset, and the model is then tested on each subject individually.

Evaluation Metric. To evaluate whether our measure of utility is a good indicator of question quality, we compare with experiments using the following evaluation metrics: (1) **Saliency** measures a question's relevance and importance within the document D [244]. It is rated on a Likert scale from 1 to 5, where a score of 1 indicates that the question in section k is unrelated to $S_{[1:k]}$ and contributes minimally to understanding, while a score of 5 indicates strong relevance to $S_{[1:k]}$ and essential comprehension support for S_k by clarifying key concepts or introducing new information. Saliency is evaluated using an LLM. (2) **Expected Information Gain (EIG)** quantifies the reduction in uncertainty about a student's knowledge state after answering a question [143, 202, 189, 254, 240, 103]. We estimate EIG using an LLM by computing the entropy of the model's token probability distribution before and after conditioning on the first token of the correct answer, capturing the immediate shift in the model's belief distribution. (3) **Utility** measures

	Microbiology	**Chemistry**	**Economics**	**Sociology**	**US History**	**Average**
Base score	0.46	0.09	0.00	0.61	0.03	0.24
Zero-shot	0.62 (+0.16)	0.40 (+0.31)	0.40 (+0.40)	0.61 (+0.00)	0.19 (+0.16)	0.44 (+0.20)
Few-shot	0.62 (+0.16)	<u>0.45</u> (+0.36)	<u>0.47</u> (+0.47)	0.62 (+0.01)	0.16 (+0.13)	0.46 (+0.22)
Chain-of-thought	0.61 (+0.15)	<u>0.45</u> (+0.36)	0.46 (+0.46)	0.61 (+0.00)	0.19 (+0.16)	0.46 (+0.22)
Bloom-based	0.57 (+0.11)	0.37 (+0.28)	0.29 (+0.29)	0.62 (+0.01)	0.22 (+0.19)	0.41 (+0.17)
SFT (Subject-Specific)	0.65 (+0.19)	0.24 (+0.15)	0.46 (+0.46)	<u>0.64</u> (+0.03)	0.20 (+0.17)	0.44 (+0.20)
SFT (Cross-Subject)	0.59 (+0.13)	0.21 (+0.12)	<u>0.47 (+0.47)</u>	0.63 (+0.02)	<u>0.26</u> (+0.23)	0.43 (+0.19)
QUEST (Subject-Specific)	**0.76 (+0.30)**	**0.46 (+0.37)**	**0.58 (+0.58)**	**0.65 (+0.04)**	**0.31 (+0.28)**	**0.55 (+0.31)**
QUEST (Cross-Subject)	<u>0.73</u> (+0.27)	<u>0.41</u> (+0.32)	<u>0.47</u> (+0.47)	**0.65 (+0.04)**	0.25 (+0.22)	<u>0.50</u> (+0.26)

Table 5.2: End-of-chapter exam score (Accuracy) comparison between different question generation approaches across various subjects.

the impact of a generated question on overall learning by evaluating simulated exam performance after exposure to the question.

Experiments and Implementation Details All LLMs used in data preprocessing and throughout the framework, including the question generator M_q, answer generator M_a, learner M_l, and evaluator M_e in the reader simulator, are based on `gpt-4o-mini`. For SFT in the baseline and rejection sampling in QUEST, we use OpenAI fine-tuning API to further train `gpt-4o-mini`.

5.2.5 Experimental Results

In this section, we first compare the overall performance of all question generation baselines based on the learner's exam score. Next, we analyze evaluation metrics by examining their correlations with utility and assessing the impact of optimizing models on high-scoring questions for each metric. We then conduct a qualitative analysis of high-utility questions to understand their characteristics. Finally, we perform ablation studies on the framework by varying the criteria for selecting high-utility questions for training and replacing `gpt-4o-mini` to `gpt-4o`.

Metric 1	Metric 2	**Spearman correlation**	**p-value**
Utility	Saliency	0.097	0.003
Utility	EIG	-0.022	0.512
Saliency	EIG	0.030	0.363

Table 5.3: Spearman correlation between utility and other metrics

Overall Performance Table 5.2 presents the learner's exam performance of different question generators, measured by exam scores using all generated question-answer pairs. Here are findings: (1) **Prompting techniques (Few-shot, CoT, Bloom-based)** offer only marginal performance gains. While advanced prompting enhances reasoning and task accuracy [19, 237, 274], it does not directly optimize utility, which reflects real-world impact—how well generated questions enhance learning. Without explicit selection or optimization, prompting cannot systematically improve this measure; (2) **SFT** shows no performance gains, indicating that while it learns the style of exam questions, it fails to generate questions that enhance learner understanding. This highlights the key distinction between producing syntactically valid questions and generating those that effectively promote learning; (3) **QUEST** achieves the highest performance gain, improving by approximately 20% on average. Performance gap between subject-specific and cross-subject rejection sampling suggests that the definition of a "high-utility" question varies by domain. The results indicate that outcome-based learning is most effective when applied within a specific domain.

Evaluation Metrics Analysis

1. **Correlation Analysis.** To analyze the relationship between *utility* and existing metrics (*saliency*, *EIG*), we estimate all three metrics on generated questions from the training set. Table 5.3 shows that both saliency and EIG have weak correlations with utility. While saliency has a weak but statistically significant correlation with utility, EIG shows no meaningful relationship. This indicates that existing indirect metrics may not accurately reflect a question's impact on learning outcomes.

2. **Optimization on Indirect Metrics.** To further investigate the impact of indirect metrics on ques-

	Microbiology			Chemistry			Economics			Sociology			US History		
Train Metric	**Utility**	**Saliency**	**EIG**	**Utility**	**Saliency**	**EIG**	**Utility**	**Saliency**	**EIG**	**Utility**	**Saliency**	**EIG**	**Utility**	**Saliency**	**EIG**
$utility > 0.1$	**0.76**	4.27	-0.18	**0.46**	4.65	-0.20	**0.58**	**4.70**	-0.04	**0.65**	**4.49**	-0.02	**0.31**	4.65	**-0.01**
$saliency = 5$	0.73	**4.42**	-0.24	0.39	**4.46**	-0.22	0.46	4.66	-0.08	0.64	**4.49**	-0.03	0.23	**4.68**	-0.02
$EIG > 0$	0.61	4.21	**-0.17**	0.32	4.40	**-0.09**	0.47	4.65	**0.01**	0.62	4.46	**0.01**	0.21	4.65	**-0.01**

Table 5.4: End-of-chapter exam score (Accuracy), average saliency, and average expected information gain (EIG) of generated questions for different QUEST-optimized models trained on datasets filtered by different selection criteria.

tion generation performance, we compare the results of QUEST when trained using different selection criteria: (1) only questions with a *utility* score greater than 0.1 (Ours), (2) only questions with a *saliency* score of 5, and (3) only questions with an *expected information gain (EIG)* greater than 0. We evaluate the generated questions based on their overall utility (*i.e.,* end-of-chapter exam scores), as well as their average saliency and EIG, to assess the question generator's performance across different quality metrics. Table 5.4 shows that while saliency- and EIG-based training improves their respective scores, it does not enhance utility. In contrast, the utility-trained model consistently achieves the highest utility across all subjects and even improves some indirect metrics (*e.g.,* it matches or outperforms saliency-based training in Economics and Sociology). Furthermore, utility-based training outperforms EIG-based training on saliency and saliency-based training on EIG, demonstrating its broader effectiveness. These results emphasize that optimizing for indirect metrics does not improve real-world learning, whereas utility-driven training yields the best overall performance.

High Utility Questions Analysis

1. **Overlap with Exam Questions.** To evaluate the relationship between generated high-utility questions and exam questions, we measured their semantic and lexical similarity. For each generated question, we computed embedding similarity using `text-3-embedding-small`[†] for semantic overlap, and the ROUGE score for lexical overlap with all exam questions in the same chapter. We then

[†] https://platform.openai.com/docs/guides/embeddings/

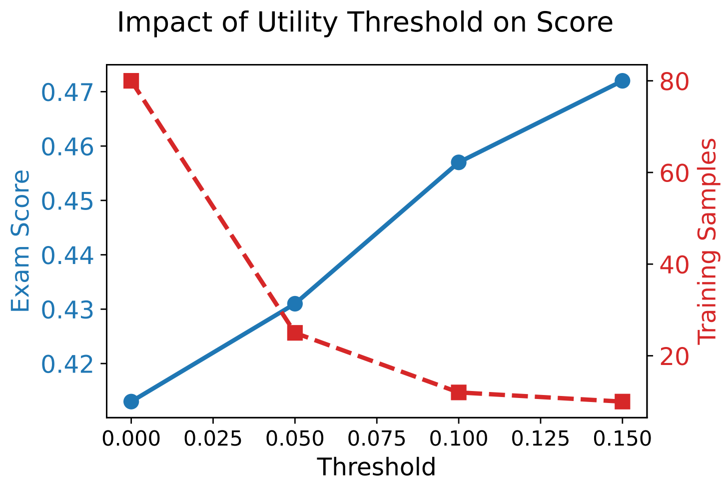


Figure 5.6: Impact of threshold in QUEST on end-of-chapter exam scores for Chemistry.

assessed the correlation between utility and the most similar exam question. The correlation between utility and semantic similarity was 0.25 ($p < 0.001$), indicating a weak positive relationship, while the correlation with ROUGE was nearly zero at 0.04 ($p < 0.01$). These findings suggest that high-utility questions are not merely paraphrased versions of exam questions but often introduce novel content that aids learning beyond surface-level similarity.

2. **Qualitative Analysis.** Qualitative examples of high-utility questions did not reveal consistent stylistic patterns. Interestingly, Bloom's taxonomy—which classifies questions by cognitive depth (e.g., "what" for recall, "why"/"how" for deeper reasoning)—did not show a strong correlation with utility. When using Bloom's taxonomy as a Likert-scale measure of depth (1–6), the correlation with utility was 0.12 ($p < 0.001$), indicating only a weak positive relationship.

Rejection Sampling Analysis Filtering for high-utility questions through rejection sampling is crucial for improving question generation. As shown in Figure 5.6, increasing the utility threshold enhances question quality, leading to higher exam scores. However, stricter filtering reduces the available training data, posing challenges for model training. These results suggest that increasing the dataset size while applying a higher threshold could further boost performance.

	QG (M_q)	AG (M_a)	RS (M_l)	Microbiology	Chemistry	Economics	Sociology	US History
Zero-Shot	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	0.620	0.414	0.398	0.609	0.233
	gpt-4o	gpt-4o-mini	gpt-4o-mini	0.681	0.457	0.466	0.634	0.180
	gpt-4o-mini	gpt-4o	gpt-4o-mini	0.682	0.422	0.480	0.634	0.232
	gpt-4o-mini	gpt-4o-mini	gpt-4o	0.710	0.173	0.476	0.564	0.263
QUEST	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini	**0.756**	**0.457**	**0.582**	**0.649**	**0.311**

Table 5.5: End-of-chapter exam score (Accuracy) for different model sizes across various subjects and modules.

Model Variants Analysis To evaluate the robustness of our framework and the impact of model size on different components, we conduct experiments to analyze how a larger model affects each module. Table 5.5 presents results for different configurations of the question generator (M_q), answer generator (M_a), and reader simulator (*i.e.,* learner M_l). The baseline corresponds to the `zero-shot` setting in Table 5.2. In the M_a experiment, we use the same questions from the `zero-shot` setting but generate new answers. For the reader simulator experiment, we keep the same questions and answers from `zero-shot` and re-run the simulation only.

Our main findings are the following: (1) **Question Generator**: A larger model (`gpt-4o`) improves the utility score by 5.7%. However, it still underperforms compared to the smaller, utility-optimized model (`gpt-4o-mini`) by 12.3%; (2) **Answer Generator**: A larger model improves performance by 7.1%, suggesting that higher answer quality provides additional information to the QA pair. However, it remains 11% behind the optimized `gpt-4o-mini` in utility. (3) **Reader Simulator**: Using a larger model (`gpt-4o`) as the reader simulator leads to mixed results, with performance gains in some subjects but a sharp decline in Chemistry. This suggests that larger models may introduce different reasoning strategies or evaluation biases, leading to inconsistencies in scoring. Additionally, since our framework is optimized for `gpt-4o-mini`, the larger model may not align well with the training dynamics. These results highlight the importance of consistency in simulation for reliable utility estimation.

5.3 Summary

This chapter explores how language models can autonomously improve their outputs by generating, evaluating, and adapting context.

We focus on the task of educational question generation, where the model creates questions to enhance comprehension and continuously refines them by simulating learner responses. These simulated interactions serve as contextual signals for estimating question utility and guiding improvement.

Experimental results show that this self-refinement process significantly enhances downstream performance, even without explicit human supervision. These findings demonstrate that language models can act as both producers and consumers of context, using their own outputs to iteratively refine future behavior. This opens promising directions for building self-improving systems driven by simulated interaction and internal feedback.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

This dissertation explored the role of context in shaping language model behavior across inference, training, and self-refinement. It examined how language models (LMs) leverage context to make more accurate predictions, how context during training can improve generalization and alignment, and how models can autonomously generate and refine context to support their own performance.

6.1.1 Context-Aware Inference in Language Models

Chapter 3 examined the capabilities and limitations of in-context learning (ICL) beyond traditional few-shot learning, focusing on tasks that require extrapolation from temporal and conversational patterns. The thesis found that LLMs good at **structural extrapolation**, effectively leveraging statistical regularities (*e.g.,* timestamped sequences) to make accurate predictions. However, performance decreases on **semantic extrapolation** tasks, where meaningful understanding of prior context is required, such as simulating a speaker's persona in dialogue. These results highlight that while LLMs are strong pattern matchers, they often fail to internalize rich contextual signals when explicit supervision is absent. Overcoming this limitation may require new modeling approaches that incorporate user-specific adaptation, contextual memory, or grounded semantic representations.

6.1.2 Contextual Supervision for Language Model Training

Chapter 4 demonstrated that contextual supervision during training significantly improves LM behavior in terms of robustness, generalization, and label efficiency. Appending contextual signals, such as task demonstrations or dialogue histories, to training inputs consistently enhanced performance across tasks like named entity recognition and social norm violation detection, underscoring that **models benefit from being explicitly trained to utilize context**, not just prompted with it at inference.

Additionally, the thesis showed that human explanations offer a powerful form of contextual guidance. Explanation-driven supervision enabled stronger performance with fewer labels, while our post hoc refinement method, XMD, allowed explanations to regularize and debug model behavior. These results emphasize the importance of integrating structured, semantically rich context into training pipelines to better align models with human reasoning and expectations.

6.1.3 Language Models as Self-Refining Context Generators

Chapter 5 explored how LMs can autonomously improve their outputs by generating, evaluating, and adapting contextual information. Focusing on educational question generation, we showed that models can simulate learner responses to their own questions and use these interactions as feedback signals for iterative refinement. This self-refinement process led to substantial improvements in output quality, even in the absence of explicit human supervision. These results illustrate that LMs can serve as both producers and consumers of context, leveraging their own outputs to guide future behavior. This opens new directions for building self-improving systems powered by simulated interaction and internal feedback loops.

6.1.4 Mechanisms, Limits, and Future Directions.

While this dissertation demonstrates the importance of context in improving LM performance, it also has limitations in how current models process and reason with context. Most models process context as a flat input sequence, without understanding which parts are most relevant or how they relate to the task. This can lead to issues like overemphasizing recent information, struggling with long-range dependencies, or misinterpreting noisy signals.

To move forward, future research should explore new ways for models to better organize, filter, and reason over context. This includes adding memory or retrieval mechanisms, training with objectives that reward useful context use, and combining language models with tools like user simulators or symbolic reasoning.

Ultimately, context should not be treated as mere prompt engineering. Better handling of context is key to building more adaptive, reliable, and user-aligned language systems.

6.2 Future work

Building on the insights from this dissertation, I highlight two promising directions for future research: (1) contextual personalization and (2) socially intelligent language models.

6.2.1 Toward Contextual Personalization in Language Models

A natural extension of contextual modeling is personalization, adapting model behavior to individual users by conditioning on long-term interaction history, preferences, or inferred goals.

Recent approaches often introduce a memory layer to store and retrieve user-specific context [27]:

$$\text{Conversation History} + \text{User Memory} \rightarrow \text{AI Response}$$

However, these systems face core limitations, including the lack of criteria for determining whether a response is truly personalized, and limited ability to generalize across implicit user signals.

To address these issues, future work can explore:

1. **Memory Probing Benchmarks:** Designing tasks that evaluate a model's ability to retrieve user-specific information, both explicit (*e.g.,* stated facts) and implicit (*e.g.,* inferred preferences), from conversational history or memory representations.

2. **User Simulation Tasks:** Predicting user behavior, such as the next user utterance in a conversation, based on prior AI turns. These tasks can be used to jointly train models with realistic user simulators and improve alignment with user expectations.

3. **Preference Embedding Techniques:** Learning and applying user embeddings to guide generation. Embeddings may be derived via contrastive learning, bi-encoders, or in-context prompting, and can be used to personalize outputs through attention mechanisms or prompt conditioning.

These directions extend this dissertation's contributions on contextual supervision and self-refinement to a more user-centered modeling framework.

6.2.2 Toward Socially Intelligent Language Models

Beyond personalization, a broader and more ambitious goal is to build socially intelligent AI systems. Humans excel at achieving complex, multifaceted objectives in social interactions, which is a key feature of our social intelligence [69, 276]. Similarly, socially intelligent AI should be capable of pursuing explicit social goals, such as improving user comprehension or satisfaction, rather than merely mimicking human behavior.

Achieving this vision requires advances in three areas:

1. **Outcome-based Reinforcement Learning:** Optimizing models for measurable social outcomes (e.g., comprehension gains, satisfaction scores, or click-through rates).

2. **Realistic User Simulation:** Creating authentic simulations of user interactions to train and evaluate AI behavior in socially meaningful contexts.

3. **Behavior Correction via Feedback:** Structuring and integrating feedback to correct undesired behavior and align model outputs with human-centered goals.

These will enable the development of AI systems that are not only accurate and fluent, but also socially effective and aligned.

Conversational Recommendation Systems One concrete application of socially intelligent AI is in conversational recommendation systems (CRS). Traditional CRS focus on item relevance; future systems should instead optimize for social outcomes such as increased engagement, satisfaction, or successful goal completion.

To enable this, outcome-based reinforcement learning can be applied with a multi-level reward structure:

- **Item-level rewards:** e.g., +1 for clicks, +5 for purchases, +2 for 5-star ratings.
- **Turn-level rewards:** e.g., +0.5 for a helpful clarification, +1 for accepted suggestions.
- **Session-level rewards:** e.g., +3 for completing a goal, +2 for positive feedback at the end.

This structured reward design enables more intentional and effective dialogue planning, shifting the focus from next-best response generation to outcome-oriented interaction.

Infographic Generation for Educational Outcomes Another exciting direction is outcome-optimized infographic and visual content generation. the aim is to guide diffusion-based image generators to produce educational visuals, such as diagrams and infographics, that improve user understanding.

Rather than optimizing for aesthetics or preference, the generation process would be guided by learning outcomes:

- After viewing a visualization, a real or simulated user completes a comprehension quiz.
- The user's performance is used as a reward signal for the generation model.
- This feedback loop enables training models that generate visuals aligned with specific educational goals.

This direction reframes structured image generation as an outcome-driven task, where success is measured not by visual appeal but by actual comprehension gains—aligning visual AI more closely with human learning and communication needs.

6.2.3 Looking Ahead

These future directions highlight the growing role of context as a foundation for building AI systems that are more adaptive, personalized, and socially intelligent. Rather than treating context as static input, this thesis views it as a dynamic signal, something that can be learned from, generated, and used to guide behavior. By developing methods for training on context, reasoning with context, and refining behavior through contextual feedback, this thesis takes key steps toward AI that truly understands and responds to human needs. In the long run, advancing how models represent and reason over context may be one of the most promising paths to achieving more human-aligned and socially effective language systems.

Bibliography

[1] Gerry Abbott. "Teaching the learner to ask for information". In: *TESOL Quarterly* (1980), pp. 5–16.

[2] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. "Debugging tests for model explanations". In: *arXiv preprint arXiv:2011.05429* (2020).

[3] Jaewoo Ahn, Yeda Song, Sangdoo Yun, and Gunhee Kim. "MPCHAT: Towards Multimodal Persona-Grounded Conversation". In: *arXiv preprint arXiv:2305.17388* (2023).

[4] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. "Asking clarifying questions in open-domain information-seeking conversations". In: *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval.* 2019, pp. 475–484.

[5] Irwin Altman. "Social penetration: The development of interpersonal relationships". In: *Rinehart, & Winston* (1973).

[6] Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. "Exploring Length Generalization in Large Language Models". In: *Advances in Neural Information Processing Systems.* Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=zSkYVeX7bC4.

[7] Dimosthenis Antypas, Asahi Ushio, Francesco Barbieri, Leonardo Neves, Kiamehr Rezaee, Luis Espinosa-Anke, Jiaxin Pei, and Jose Camacho-Collados. "SuperTweetEval: A Challenging, Unified and Heterogeneous Benchmark for Social Media NLP Research". In: *Findings of the Association for Computational Linguistics: EMNLP 2023.* 2023.

[8] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. "Constitutional ai: Harmlessness from ai feedback". In: *arXiv preprint arXiv:2212.08073* (2022).

[9] Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. "Deriving Machine Attention from Human Rationales". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1903–1913. DOI: 10.18653/v1/D18-1216.

[10] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1644–1650. DOI: 10.18653/v1/2020.findings-emnlp.148.

[11] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. "SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter". In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Ed. by Jonathan May, Ekaterina Shutova, Aurelie Herbelot, Xiaodan Zhu, Marianna Apidianaki, and Saif M. Mohammad. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 54–63. DOI: 10.18653/v1/S19-2007.

[12] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 2021, pp. 610–623.

[13] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. "Gpt-neox-20b: An open-source autoregressive language model". In: *arXiv preprint arXiv:2204.06745* (2022).

[14] Benjamin S Bloom, Max D Engelhart, EJ Furst, Walker H Hill, and David R Krathwohl. "Handbook I: cognitive domain". In: *New York: David McKay* (1956), pp. 483–498.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017). Ed. by Lillian Lee, Mark Johnson, and Kristina Toutanova, pp. 135–146. DOI: 10.1162/tacl_a_00051.

[16] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. "Improving Language Models by Retrieving from Trillions of Tokens". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 2206–2240. URL: https://proceedings.mlr.press/v162/borgeaud22a.html.

[17] Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. "Flex: Unifying evaluation for few-shot nlp". In: *Advances in neural information processing systems* 34 (2021), pp. 15787–15800.

[18] Danny Brassell and Timothy Rasinski. *Comprehension that works: Taking students beyond ordinary understanding to deep comprehension*. Teacher Created Materials, 2008.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[20] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. "e-snli: Natural language inference with natural language explanations". In: *Advances in Neural Information Processing Systems* 31 (2018).

[21] Samuel Carton, Anirudh Rathore, and Chenhao Tan. "Evaluating and characterizing human rationales". In: *arXiv preprint arXiv:2010.04736* (2020).

[22] Stephanie C.Y. Chan, Adam Santoro, Andrew Kyle Lampinen, Jane X Wang, Aaditya K Singh, Pierre Harvey Richemond, James McClelland, and Felix Hill. "Data Distributional Properties Drive Emergent In-Context Learning in Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=lHj-q9BSRjF.

[23] Yee Seng Chan, Hwee Tou Ng, and David Chiang. "Word Sense Disambiguation Improves Statistical Machine Translation". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Ed. by Annie Zaenen and Antal van den Bosch. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 33–40. URL: https://aclanthology.org/P07-1005/.

[24] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. "When large language models meet personalization: Perspectives of challenges and opportunities". In: *World Wide Web* 27.4 (2024), p. 42.

[25] Qian Chen, Zhu Zhuo, and Wen Wang. "Bert for joint intent classification and slot filling". In: *arXiv preprint arXiv:1902.10909* (2019).

[26] Jithin Cheriyan, Bastin Tony Roy Savarimuthu, and Stephen Cranefield. "Norm violation in online communities–A study of Stack Overflow comments". In: *Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIII*. Springer, 2017, pp. 20–34.

[27] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory". In: *arXiv preprint arXiv:2504.19413* (2025).

[28] Jason P.C. Chiu and Eric Nichols. "Named Entity Recognition with Bidirectional LSTM-CNNs". In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 357–370. DOI: 10.1162/tacl_a_00104.

[29] Hyundong Cho, Shuai Liu, Taiwei Shi, Darpan Jain, Basem Rizk, Yuyang Huang, Zixun Lu, Nuan Wen, Jonathan Gratch, Emilio Ferrara, and Jonathan May. "Can Language Model Moderators Improve the Health of Online Discourse?" In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 7478–7496. DOI: 10.18653/v1/2024.naacl-long.415.

[30] Hyundong Justin Cho, Nicolaas Paul Jedema, Leonardo F. R. Ribeiro, Karishma Sharma, Pedro Szekely, Alessandro Moschitti, Ruben Janssen, and Jonathan May. "Speechworthy Instruction-tuned Language Models". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 10652–10670. DOI: 10.18653/v1/2024.emnlp-main.595.

[31] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. "Palm: Scaling language modeling with pathways". In: *arXiv preprint arXiv:2204.02311* (2022).

[32] Angelo V Ciardiello. "Did you ask a good question today? Alternative cognitive and metacognitive strategies". In: *Journal of adolescent & adult literacy* 42.3 (1998), pp. 210–219.

[33] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. "Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases". 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4069–4082. DOI: 10.18653/v1/D19-1418.

[34] Herbert H Clark. *Using language*. Cambridge university press, 1996.

[35] Herbert H Clark and Edward F Schaefer. "Contributing to discourse". In: *Cognitive science* 13.2 (1989), pp. 259–294.

[36] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. "Training verifiers to solve math word problems". In: *arXiv preprint arXiv:2110.14168* (2021).

[37] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. "Template-Based Named Entity Recognition Using BART". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, 2021, pp. 1835–1845. DOI: 10.18653/v1/2021.findings-acl.161.

[38] Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. "Human Attention in Visual Question Answering: Do Humans and Deep Networks look at the same regions?" In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by Jian Su, Kevin Duh, and Xavier Carreras. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 932–937. DOI: 10.18653/v1/D16-1092.

[39] Devleena Das and Sonia Chernova. "Leveraging rationales to improve human task performance". In: *Proceedings of the 25th international conference on intelligent user interfaces*. 2020, pp. 510–518.

[40] Srayan Datta and Eytan Adar. "Extracting inter-community conflicts in reddit". In: *Proceedings of the international AAAI conference on Web and Social Media*. Vol. 13. 2019, pp. 146–157.

[41] Beth Davey and Susan McBride. "Effects of question-generation training on reading comprehension." In: *Journal of Educational Psychology* 78.4 (1986), p. 256.

[42] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. "Automated hate speech detection and the problem of offensive language". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 11. 1. 2017, pp. 512–515.

[43] Mohammad Davoudi and Narges Amel Sadeghi. "A Systematic Review of Research on Questioning as a High-Level Cognitive Strategy." In: *English Language Teaching* 8.10 (2015), pp. 76–90.

[44] Jan Deriu, Don Tuggener, Pius von Däniken, Jon Ander Campos, Alvaro Rodrigo, Thiziri Belkacem, Aitor Soroa, Eneko Agirre, and Mark Cieliebak. "Spot The Bot: A Robust and Efficient Framework for the Evaluation of Conversational Dialogue Systems". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 3971–3984. DOI: 10.18653/v1/2020.emnlp-main.326.

[45] Daantje Derks, Agneta H Fischer, and Arjan ER Bos. "The role of emotion in computer-mediated communication: A review". In: *Computers in human behavior* 24.3 (2008), pp. 766–785.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[47] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. "Eraser: A benchmark to evaluate rationalized nlp models". In: *arXiv preprint arXiv:1911.03429* (2019).

[48] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. "ERASER: A Benchmark to Evaluate Rationalized NLP Models". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 4443–4458. DOI: 10.18653/v1/2020.acl-main.408.

[49] Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. "Prompt-learning for fine-grained entity typing". In: *ArXiv preprint* abs/2108.10604 (2021). URL: https://arxiv.org/abs/2108.10604.

[50] Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. "Few-NERD: A Few-shot Named Entity Recognition Dataset". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3198–3213. DOI: 10.18653/v1/2021.acl-long.248.

[51] David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, et al. "Language model cascades". In: *arXiv preprint arXiv:2207.10342* (2022).

[52] Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. "Towards Interpreting and Mitigating Shortcut Learning Behavior of NLU models". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 915–929. DOI: 10.18653/v1/2021.naacl-main.71.

[53] Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. "PerLTQA: A Personal Long-Term Memory Dataset for Memory Classification, Retrieval, and Synthesis in Question Answering". In: *arXiv preprint arXiv:2402.16288* (2024).

[54] Avia Efrat and Omer Levy. "The turking test: Can language models understand instructions?" In: *arXiv preprint arXiv:2010.11982* (2020).

[55] Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. "Latent Hatred: A Benchmark for Understanding Implicit Hate Speech". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 345–363. DOI: 10.18653/v1/2021.emnlp-main.29.

[56] Yuwei Fang, Shuohang Wang, Yichong Xu, Ruochen Xu, Siqi Sun, Chenguang Zhu, and Michael Zeng. "Leveraging Knowledge in Multilingual Commonsense Reasoning". In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3237–3246. DOI: 10.18653/v1/2022.findings-acl.255.

[57] Jiazhan Feng, Qingfeng Sun, Can Xu, Pu Zhao, Yaming Yang, Chongyang Tao, Dongyan Zhao, and Qingwei Lin. "Mmdialog: A large-scale multi-turn dialogue dataset towards multi-modal open-domain conversation". In: *arXiv preprint arXiv:2211.05719* (2022).

[58] Casey Fiesler, Joshua McCann, Kyle Frye, Jed R Brubaker, et al. "Reddit rules! characterizing an ecosystem of governance". In: *Twelfth International AAAI Conference on Web and Social Media*. 2018.

[59] Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. "Large scale crowdsourcing and characterization of twitter abusive behavior". In: *Twelfth International AAAI Conference on Web and Social Media*. 2018.

[60] Tianyu Gao, Adam Fisch, and Danqi Chen. "Making Pre-trained Language Models Better Few-shot Learners". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, pp. 3816–3830. DOI: 10.18653/v1/2021.acl-long.295.

[61] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. "Enabling Large Language Models to Generate Text with Citations". In: *arXiv preprint arXiv:2305.14627* (2023).

[62] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. "Learning Sequence Encoders for Temporal Knowledge Graph Completion". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 4816–4821. DOI: 10.18653/v1/D18-1516.

[63] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. "What can transformers learn in-context? a case study of simple function classes". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 30583–30598.

[64] Julia Gastinger, Timo Sztyler, Lokesh Sharma, and Anett Schuelke. "On the Evaluation of Methods for Temporal Knowledge Graph Forecasting". In: *NeurIPS 2022 Temporal Graph Learning Workshop*. 2022. URL: https://openreview.net/forum?id=J_SNklR-KR.

[65] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. "Shortcut learning in deep neural networks". In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[66] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. "Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 346–361.

[67] Reza Ghaeini, Xiaoli Z Fern, Hamed Shahbazi, and Prasad Tadepalli. "Saliency learning: Teaching the model where to pay attention". In: *arXiv preprint arXiv:1902.08649* (2019).

[68] Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. "Hate Speech Dataset from a White Supremacy Forum". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 11–20. DOI: 10.18653/v1/W18-5102.

[69] Daniel Goleman. *Social intelligence*. Random house, 2007.

[70] Daniel Goleman. "Working with emotional intelligence". In: *NY: Bantam Books* (1998).

[71] Thomas L Good, Ricky L Slavings, and DeWayne A Mason. "Learning to ask questions: Grade and school effects". In: *Teaching and Teacher Education* 4.4 (1988), pp. 363–378.

[72] Art Graesser, Yasuhiro Ozuru, and Jeremiah Sullins. "What is a good question?" In: (2010).

[73] Shashank Gupta, Vaishnavi Shrivastava, Ameet Deshpande, Ashwin Kalyan, Peter Clark, Ashish Sabharwal, and Tushar Khot. "Bias runs deep: Implicit reasoning biases in persona-assigned llms". In: *arXiv preprint arXiv:2311.04892* (2023).

[74] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of ACL*. 2020.

[75] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. “Retrieval augmented language model pre-training”. In: *International conference on machine learning*. PMLR. 2020, pp. 3929–3938.

[76] Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. “Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations”. 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)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5297–5306. DOI: 10.18653/v1/D19-1533.

[77] R. Hadsell, S. Chopra, and Y. LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. 2006, pp. 1735–1742. URL: https://ieeexplore.ieee.org/document/1640964.

[78] Michael Hahn and Navin Goyal. “A Theory of Emergent In-Context Learning as Implicit Structure Induction”. In: *arXiv preprint arXiv:2303.07971* (2023).

[79] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. “Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=pGIHq1m7PU.

[80] Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. “Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 8352–8364. DOI: 10.18653/v1/2021.emnlp-main.658.

[81] Mareike Hartmann and Daniel Sonntag. “A survey on improving NLP models with human explanations”. In: *arXiv preprint arXiv:2204.08892* (2022).

[82] Peter Hase and Mohit Bansal. “When can models learn from explanations? a formal framework for understanding the roles of explanation data”. In: *arXiv preprint arXiv:2102.02201* (2021).

[83] Elaine Hatfield, John T Cacioppo, and Richard L Rapson. “Emotional contagion”. In: *Current directions in psychological science* 2.3 (1993), pp. 96–100.

[84] He He, Sheng Zha, and Haohan Wang. “Unlearn Dataset Bias in Natural Language Inference by Fitting the Residual”. In: *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 132–142. DOI: 10.18653/v1/D19-6115.

[85] Avi Hofstein, Oshrit Navon, Mira Kipnis, and Rachel Mamlok-Naaman. “Developing students’ ability to ask more and better questions resulting from inquiry-type chemistry laboratories”. In: *Journal of research in science teaching* 42.7 (2005), pp. 791–806.

[86] Jen-tse Huang, Wenxuan Wang, Eric John Li, Man Ho Lam, Shujie Ren, Youliang Yuan, Wenxiang Jiao, Zhaopeng Tu, and Michael Lyu. "On the humanity of conversational ai: Evaluating the psychological portrayal of llms". In: *The Twelfth International Conference on Learning Representations*. 2023.

[87] Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. "Few-Shot Named Entity Recognition: A Comprehensive Study". In: *arXiv preprint arXiv:2012.14978* (2020).

[88] Karen Huang, Michael Yeomans, Alison Wood Brooks, Julia Minson, and Francesca Gino. "It doesn't hurt to ask: Question-asking increases liking." In: *Journal of personality and social psychology* 113.3 (2017), p. 430.

[89] Quzhe Huang, Shengqi Zhu, Yansong Feng, and Dongyan Zhao. "Exploring Distantly-Labeled Rationales in Neural Network Models". In: *arXiv preprint arXiv:2106.01809* (2021).

[90] Maximilian Idahl, Lijun Lyu, Ujwal Gadiraju, and Avishek Anand. "Towards benchmarking the utility of explanations for model debugging". In: *arXiv preprint arXiv:2105.04505* (2021).

[91] Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, et al. "Rewarding chatbots for real-world engagement with millions of users". In: *arXiv preprint arXiv:2303.06135* (2023).

[92] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. "Few-shot learning with retrieval augmented language models". In: *arXiv preprint arXiv:2208.03299* (2022).

[93] Alon Jacovi and Yoav Goldberg. "Aligning Faithful Interpretations with their Social Attribution". In: *Transactions of the Association for Computational Linguistics* 9 (2021). Ed. by Brian Roark and Ani Nenkova, pp. 294–310. DOI: 10.1162/tacl_a_00367.

[94] Jihyoung Jang, Minseong Boo, and Hyounghun Kim. "Conversation Chronicles: Towards Diverse Temporal and Relational Dynamics in Multi-Session Conversations". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 13584–13606. DOI: 10.18653/v1/2023.emnlp-main.838.

[95] Jihyoung Jang, Minseong Boo, and Hyounghun Kim. "Conversation Chronicles: Towards Diverse Temporal and Relational Dynamics in Multi-Session Conversations". In: *arXiv preprint arXiv:2310.13420* (2023).

[96] Robin Jia and Percy Liang. "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2021–2031. DOI: 10.18653/v1/D17-1215.

[97] Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. “Cold-Start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 3193–3207. DOI: 10.18653/v1/2020.emnlp-main.258.

[98] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. “Recurrent Event Network: Autoregressive Structure Inferenceover Temporal Knowledge Graphs”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 6669–6683. DOI: 10.18653/v1/2020.emnlp-main.541.

[99] Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. “AgentReview: Exploring Peer Review Dynamics with LLM Agents”. In: *arXiv preprint arXiv:2406.12708* (2024).

[100] Brihi Joshi, Aaron Chan, Ziyi Liu, Shaoliang Nie, Maziar Sanjabi, Hamed Firooz, and Xiang Ren. “ER-TEST: Evaluating Explanation Regularization Methods for NLP Models”. In: *arXiv preprint arXiv:2205.12542* (2022).

[101] Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. “BERT for Coreference Resolution: Baselines and Analysis”. 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)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5803–5808. DOI: 10.18653/v1/D19-1588.

[102] Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. “Learning The Difference That Makes A Difference With Counterfactually-Augmented Data”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=Sklgs0NFvr.

[103] Sedrick Keh, Justin Chiu, and Daniel Fried. “Asking More Informative Questions for Grounded Retrieval”. In: *Findings of the Association for Computational Linguistics: NAACL 2024*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 4429–4442. DOI: 10.18653/v1/2024.findings-naacl.276.

[104] Brendan Kennedy, Mohammad Atari, Aida M Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs, Shreya Havaldar, Gwenyth Portillo-Wightman, Elaine Gonzalez, and et al. *Introducing the Gab Hate Corpus: Defining and applying hate-based rhetoric to social media posts at scale*. 2018. DOI: 10.1007/s10579-021-09569-x.

[105] Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. “Contextualizing hate speech classifiers with post-hoc explanation”. In: *arXiv preprint arXiv:2005.02439* (2020).

[106] Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. "Contextualizing Hate Speech Classifiers with Post-hoc Explanation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5435–5442. DOI: 10.18653/v1/2020.acl-main.483.

[107] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. "Generalization through memorization: Nearest neighbor language models". In: *arXiv preprint arXiv:1911.00172* (2019).

[108] Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. "SODA: Million-scale Dialogue Distillation with Social Commonsense Contextualization". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12930–12949. DOI: 10.18653/v1/2023.emnlp-main.799.

[109] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. "Large Language Models are Zero-Shot Reasoners". In: *arXiv preprint arXiv:2205.11916* (2022).

[110] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. arXiv: 2009.07896 [cs.LG].

[111] DR Krathwohl. "A Revision Bloom's Taxonomy: An Overview". In: *Theory into Practice* (2002).

[112] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. "Community interaction and conflict on the web". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 933–943.

[113] Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. "Can language models learn from explanations in context?" In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 537–563. DOI: 10.18653/v1/2022.findings-emnlp.38.

[114] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. "Neural Architectures for Named Entity Recognition". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030.

[115] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. "Neural Architectures for Named Entity Recognition". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030.

[116] Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. "Copy is All You Need". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=CROlOA9Nd8C.

[117] Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. "Adversarial filters of dataset biases". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1078–1088.

[118] Julien Leblay and Melisachew Wudage Chekol. "Deriving validity time in knowledge graph". In: *Companion proceedings of the the web conference 2018*. 2018, pp. 1771–1776.

[119] Bruce W Lee, Yeongheon Lee, and Hyunsoo Cho. "Language models show stable value orientations across diverse role-plays". In: *arXiv preprint arXiv:2408.09049* (2024).

[120] Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. "Temporal Knowledge Graph Forecasting Without Knowledge Using In-Context Learning". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 544–557. DOI: 10.18653/v1/2023.emnlp-main.36.

[121] Dong-Ho Lee, Hyundong Cho, Jonathan May, and Jay Pujara. "What is a Good Question? Utility Estimation with LLM-based Simulations". In: *arXiv preprint arXiv:2502.17383* (2025).

[122] Dong-Ho Lee, Akshen Kadakia, Brihi Joshi, Aaron Chan, Ziyi Liu, Kiran Narahari, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara, and Xiang Ren. "XMD: An End-to-End Framework for Interactive Explanation-Based Debugging of NLP Models". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Ed. by Danushka Bollegala, Ruihong Huang, and Alan Ritter. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 264–273. DOI: 10.18653/v1/2023.acl-demo.25.

[123] Dong-Ho Lee, Akshen Kadakia, Kangmin Tan, Mahak Agarwal, Xinyu Feng, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara, and Xiang Ren. "Good Examples Make A Faster Learner: Simple Demonstration-based Learning for Low-resource NER". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2687–2700. DOI: 10.18653/v1/2022.acl-long.192.

[124] Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. "LEAN-LIFE: A Label-Efficient Annotation Framework Towards Learning from Explanation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Ed. by Asli Celikyilmaz and Tsung-Hsien Wen. Online: Association for Computational Linguistics, July 2020, pp. 372–379. DOI: 10.18653/v1/2020.acl-demos.42.

[125] Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. "Realtalk: A 21-day real-world dataset for long-term conversation". In: *arXiv preprint arXiv:2502.13270* (2025).

[126] Dong-Ho Lee, Ravi Kiran Selvam, Sheikh Muhammad Sarwar, Bill Yuchen Lin, Mahak Agarwal, Fred Morstatter, Jay Pujara, Elizabeth Boschee, James Allan, and Xiang Ren. "AutoTriggER: Named Entity Recognition with Auxiliary Trigger Extraction". In: *ArXiv preprint* abs/2109.04726 (2021). URL: https://arxiv.org/abs/2109.04726.

[127] Dong-Ho Lee, Ravi Kiran Selvam, Sheikh Muhammad Sarwar, Bill Yuchen Lin, Fred Morstatter, Jay Pujara, Elizabeth Boschee, James Allan, and Xiang Ren. "AutoTriggER: Label-Efficient and Robust Named Entity Recognition with Auxiliary Trigger Extraction". In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by Andreas Vlachos and Isabelle Augenstein. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 3011–3025. DOI: 10.18653/v1/2023.eacl-main.219.

[128] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. "End-to-end Neural Coreference Resolution". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 188–197. DOI: 10.18653/v1/D17-1018.

[129] Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. "FIND: Human-in-the-Loop Debugging Deep Text Classifiers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 332–348. DOI: 10.18653/v1/2020.emnlp-main.24.

[130] Piyawat Lertvittayakumjorn and Francesca Toni. "Explanation-based human debugging of nlp models: A survey". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1508–1528.

[131] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. "Standing on the shoulders of giant frozen language models". In: *arXiv preprint arXiv:2204.10019* (2022).

[132] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. "Camel: Communicative agents for" mind" exploration of large language model society". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 51991–52008.

[133] Han Li, Robert E Kraut, and Haiyi Zhu. "Technical Features of Asynchronous and Synchronous Community Platforms and their Effects on Community Cohesion: A Comparative Study of Forum-based and Chat-based Online Mental Health Communities". In: *Journal of Computer-Mediated Communication* 26.6 (2021), pp. 403–421.

[134] Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. "BioCreative V CDR task corpus: a resource for chemical disease relation extraction". In: *Database : the journal of biological databases and curation* (2016).

[135] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2017, pp. 986–995.

[136] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. "Temporal knowledge graph reasoning based on evolutional representation learning". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 408–417.

[137] Bill Y. Lin, Frank Xu, Zhiyi Luo, and Kenny Zhu. "Multi-channel BiLSTM-CRF Model for Emerging Named Entity Recognition in Social Media". In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 160–165. DOI: 10.18653/v1/W17-4421.

[138] Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. "TriggerNER: Learning with Entity Triggers as Explanations for Named Entity Recognition". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 8503–8511. DOI: 10.18653/v1/2020.acl-main.752.

[139] Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. "Triggerner: Learning with entity triggers as explanations for named entity recognition". In: *arXiv preprint arXiv:2004.07493* (2020).

[140] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[141] Stephanie Lin, Jacob Hilton, and Owain Evans. "Teaching models to express their uncertainty in words". In: *arXiv preprint arXiv:2205.14334* (2022).

[142] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. "A Structured Self-Attentive Sentence Embedding". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=BJC_jUqxe.

[143] Dennis V Lindley. "On a measure of the information provided by an experiment". In: *The Annals of Mathematical Statistics* 27.4 (1956), pp. 986–1005.

[144] Frederick Liu and Besim Avci. "Incorporating Priors with Feature Attribution on Text Classification". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6274–6283. DOI: 10.18653/v1/P19-1631.

[145] Frederick Liu and Besim Avci. "Incorporating priors with feature attribution on text classification". In: *arXiv preprint arXiv:1906.08286* (2019).

[146] Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. "Generated Knowledge Prompting for Commonsense Reasoning". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3154–3169. DOI: 10.18653/v1/2022.acl-long.225.

[147] Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. "Empower Sequence Labeling with Task-Aware Neural Language Model". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 5253–5260. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17123.

[148] Nelson F Liu, Tianyi Zhang, and Percy Liang. "Evaluating verifiability in generative search engines". In: *arXiv preprint arXiv:2304.09848* (2023).

[149] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". In: *ArXiv preprint* abs/2107.13586 (2021). URL: https://arxiv.org/abs/2107.13586.

[150] Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. "Towards Improving Neural Named Entity Recognition with Gazetteers". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5301–5307. DOI: 10.18653/v1/P19-1524.

[151] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. "Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 4120–4127.

[152] Michael Lu, Hyundong Justin Cho, Weiyan Shi, Jonathan May, and Alexander Spangher. *NewsInterview: a Dataset and a Playground to Evaluate LLMs' Ground Gap via Informational Interviews*. 2024. arXiv: 2411.13779 [cs.CL]. URL: https://arxiv.org/abs/2411.13779.

[153] Ziyang Luo, Can Xu, Pu Zhao, Xiubo Geng, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. "Augmented Large Language Models with Parametric Knowledge Guiding". In: *arXiv preprint arXiv:2305.04757* (2023).

[154] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166.

[155] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, pp. 1064–1074. DOI: 10.18653/v1/P16-1101.

[156] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074. DOI: 10.18653/v1/P16-1101.

[157] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. "Self-refine: Iterative refinement with self-feedback". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 46534–46594.

[158] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. "Evaluating Very Long-Term Conversational Memory of LLM Agents". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 13851–13870. DOI: 10.18653/v1/2024.acl-long.747.

[159] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. "Yago3: A knowledge base from multilingual wikipedias". In: *7th biennial conference on innovative data systems research*. CIDR Conference. 2014.

[160] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. "When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 9802–9822. DOI: 10.18653/v1/2023.acl-long.546.

[161] John D Mayer, Peter Salovey, and David R Caruso. "Emotional intelligence: New ability or eclectic traits?" In: *American psychologist* 63.6 (2008), p. 503.

[162] Julian McAuley and Jure Leskovec. "Hidden factors and hidden topics: understanding rating dimensions with review text". In: *Proceedings of the 7th ACM conference on Recommender systems*. 2013, pp. 165–172.

[163] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[164] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. ""Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions". In: *In 44th International Conference on Software Engineering (ICSE'22)*. 2022.

[165] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?" In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 11048–11064. URL: https://aclanthology.org/2022.emnlp-main.759.

[166] Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. "Nonparametric Masked Language Modeling". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 2097–2118. URL: https://aclanthology.org/2023.findings-acl.132.

[167] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. "Large language models as general pattern machines". In: *arXiv preprint arXiv:2307.04721* (2023).

[168] Bhavana Dalvi Mishra, Oyvind Tafjord, and Peter Clark. "Towards teachable reasoning systems: Using a dynamic memory of user feedback for continual system improvement". In: *arXiv preprint arXiv:2204.13074* (2022).

[169] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. "Cross-Task Generalization via Natural Language Crowdsourcing Instructions". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3470–3487. DOI: 10.18653/v1/2022.acl-long.244.

[170] Jihyung Moon, Dong-Ho Lee, Hyundong Cho, Woojeong Jin, Chan Park, Minwoo Kim, Jonathan May, Jay Pujara, and Sungjoon Park. "Analyzing Norm Violations in Live-Stream Chat". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 852–868. DOI: 10.18653/v1/2023.emnlp-main.55.

[171] David Nadeau and Satoshi Sekine. "A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26. DOI: https://doi.org/10.1075/li.30.1.03nad.

[172] Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. "Wt5?! training text-to-text models to explain their predictions". In: *arXiv preprint arXiv:2004.14546* (2020).

[173] Kate G Niederhoffer and James W Pennebaker. "Linguistic style matching in social interaction". In: *Journal of Language and Social Psychology* 21.4 (2002), pp. 337–360.

[174] Maxwell Nye, Anders Andreassen, Guy Gur-Ari, Henryk Witold Michalewski, Jacob Austin, David Bieber, David Martin Dohan, Aitor Lewkowycz, Maarten Paul Bosma, David Luan, Charles Sutton, and Augustus Odena. *Show Your Work: Scratchpads for Intermediate Computation with Language Models.* https://arxiv.org/abs/2112.00114. 2021.

[175] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[176] Chan Young Park, Julia Mendelsohn, Karthik Radhakrishnan, Kinjal Jain, Tushar Kanakagiri, David Jurgens, and Yulia Tsvetkov. "Detecting Community Sensitive Norm Violations in Online Conversations". In: *Findings of the Association for Computational Linguistics: EMNLP 2021.* Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3386–3397. DOI: 10.18653/v1/2021.findings-emnlp.288.

[177] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. "Generative agents: Interactive simulacra of human behavior". In: *arXiv preprint arXiv:2304.03442* (2023).

[178] Joon Sung Park, Carolyn Q Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S Bernstein. "Generative Agent Simulations of 1,000 People". In: *arXiv preprint arXiv:2411.10109* (2024).

[179] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[180] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. "Toxicity Detection: Does Context Really Matter?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 4296–4305. DOI: 10.18653/v1/2020.acl-main.396.

[181] Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuanjing Huang. "Distantly Supervised Named Entity Recognition using Positive-Unlabeled Learning". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 2409–2419. DOI: 10.18653/v1/P19-1231.

[182] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[183] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[184] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202.

[185] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. “Towards Robust Linguistic Analysis using OntoNotes”. In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 143–152. URL: https://aclanthology.org/W13-3516.

[186] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. “Improving language understanding by generative pre-training”. In: (2018).

[187] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[188] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. “In-context retrieval-augmented language models”. In: *arXiv preprint arXiv:2302.00083* (2023).

[189] Sudha Rao and Hal Daumé III. “Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2737–2746. DOI: 10.18653/v1/P18-1255.

[190] Steve Rathje, Dan-Mircea Mirea, Ilia Sucholutsky, Raja Marjieh, Claire E Robertson, and Jay J Van Bavel. “GPT is an effective tool for multilingual psychological text analysis”. In: *Proceedings of the National Academy of Sciences* 121.34 (2024), e2308950121.

[191] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. “Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning”. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 840–854. URL: https://aclanthology.org/2022.findings-emnlp.59.

[192] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. 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)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

[193] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[194] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In: *International conference on machine learning*. PMLR. 2020, pp. 8116–8126.

[195] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. "Semi-Supervised Self-Training of Object Detection Models". In: *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1* 1 (2005), pp. 29–36. URL: https://ieeexplore.ieee.org/document/4129456.

[196] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. "Right for the right reasons: Training differentiable models by constraining their explanations". In: *arXiv preprint arXiv:1703.03717* (2017).

[197] Esteban Safranchik, Shiying Luo, and Stephen H. Bach. "Weakly Supervised Sequence Tagging from Noisy Rules". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2020. URL: http://cs.brown.edu/people/sbach/files/safranchik-aaai20.pdf.

[198] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. "An Investigation of Why Overparameterization Exacerbates Spurious Correlations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8346–8356.

[199] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. "Lamp: When large language models meet personalization". In: *arXiv preprint arXiv:2304.11406* (2023).

[200] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. "Multitask prompted training enables zero-shot task generalization". In: *arXiv preprint arXiv:2110.08207* (2021).

[201] Abulhair Saparov and He He. "Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=qFVVBzXxR2V.

[202] Nora Cate Schaeffer and Stanley Presser. "The science of asking questions". In: *Annual review of sociology* 29.1 (2003), pp. 65–88.

[203] Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. "AgentClinic: a multimodal agent benchmark to evaluate AI in simulated clinical environments". In: *arXiv preprint arXiv:2405.07960* (2024).

[204] Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. "Towards Debiasing Fact Verification Models". 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3419–3425. DOI: 10.18653/v1/D19-1341.

[205] Omar Shaikh, Kristina Gligoric, Ashna Khetan, Matthias Gerstgrasser, Diyi Yang, and Dan Jurafsky. "Grounding Gaps in Language Model Generations". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 6279–6296. DOI: 10.18653/v1/2024.naacl-long.348.

[206] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. “Learning Named Entity Tagger using Domain-Specific Dictionary”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2054–2064. DOI: 10.18653/v1/D18-1230.

[207] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. “Multi-turn reinforcement learning from preference human feedback”. In: *arXiv preprint arXiv:2405.14655* (2024).

[208] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. “Character-llm: A trainable agent for role-playing”. In: *arXiv preprint arXiv:2310.10158* (2023).

[209] Ashish Sharma, Adam Miner, David Atkins, and Tim Althoff. “A Computational Approach to Understanding Empathy Expressed in Text-Based Mental Health Support”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 5263–5276. DOI: 10.18653/v1/2020.emnlp-main.425.

[210] Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. “Deep Active Learning for Named Entity Recognition”. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 252–256. DOI: 10.18653/v1/W17-2630.

[211] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. “Replug: Retrieval-augmented black-box language models”. In: *arXiv preprint arXiv:2301.12652* (2023).

[212] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning important features through propagating activation differences”. In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153.

[213] Alison Smith-Renner, Ron Fan, Melissa Birchfield, Tongshuang Wu, Jordan Boyd-Graber, Daniel S Weld, and Leah Findlater. “No explainability without accountability: An empirical study of explanations and feedback in interactive ml”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–13.

[214] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[215] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. “Portuguese named entity recognition using BERT-CRF”. In: *arXiv preprint arXiv:1909.10649* (2019).

[216] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. “Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models”. In: *arXiv preprint arXiv:2206.04615* (2022).

[217] Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. "Timetraveler: Reinforcement learning for temporal knowledge graph forecasting". In: *arXiv preprint arXiv:2109.04101* (2021).

[218] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[219] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

[220] Toyin Tofade, Jamie Elsner, and Stuart T Haines. "Best practice strategies for effective use of questions as a teaching tool". In: *American journal of pharmaceutical education* 77.7 (2013), p. 155.

[221] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[222] Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. "Mind the trade-off: Debiasing NLU models without degrading the in-distribution performance". In: *arXiv preprint arXiv:2005.00315* (2020).

[223] Ronald D Vale. "The value of asking questions". In: *Molecular biology of the cell* 24.6 (2013), pp. 680–682.

[224] Robert Van Rooy. "Questioning to resolve decision problems". In: *Linguistics and Philosophy* 26 (2003), pp. 727–763.

[225] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[226] Oriol Vinyals and Quoc Le. "A neural conversational model". In: *arXiv preprint arXiv:1506.05869* (2015).

[227] Ben Wang. *Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[228] Noah Wang, Z.y. Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. "RoleLLM: Benchmarking, Eliciting, and Enhancing Role-Playing Abilities of Large Language Models". In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 14743–14777. DOI: 10.18653/v1/2024.findings-acl.878.

[229] PeiFeng Wang, Aaron Chan, Filip Ilievski, Muhao Chen, and Xiang Ren. "PINTO: Faithful Language Reasoning Using Prompt-Generated Rationales". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=WBXbRs63oVu.

[230] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. "Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, pp. 1800–1812. DOI: 10.18653/v1/2021.acl-long.142.

[231] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. "Self-consistency improves chain of thought reasoning in language models". In: *arXiv preprint arXiv:2203.11171* (2022).

[232] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5085–5109. DOI: 10.18653/v1/2022.emnlp-main.340.

[233] Yu Wang, Yilin Shen, and Hongxia Jin. "A Bi-Model Based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 309–314. DOI: 10.18653/v1/N18-2050.

[234] Ziqi Wang, Yujia Qin, Wenxuan Zhou, Jun Yan, Qinyuan Ye, Leonardo Neves, Zhiyuan Liu, and Xiang Ren. "Learning from explanations with neural execution tree". In: *arXiv preprint arXiv:1911.01352* (2019).

[235] Zeerak Waseem and Dirk Hovy. "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter". In: *Proceedings of the NAACL Student Research Workshop*. Ed. by Jacob Andreas, Eunsol Choi, and Angeliki Lazaridou. San Diego, California: Association for Computational Linguistics, June 2016, pp. 88–93. DOI: 10.18653/v1/N16-2013.

[236] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. "Finetuned language models are zero-shot learners". In: *arXiv preprint arXiv:2109.01652* (2021).

[237] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[238] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. "Larger language models do in-context learning differently". In: *arXiv preprint arXiv:2303.03846* (2023).

[239] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. "Ontonotes release 5.0 ldc2013t19". In: *Linguistic Data Consortium, Philadelphia, PA* 23 (2013).

[240] Julia White, Gabriel Poesia, Robert Hawkins, Dorsa Sadigh, and Noah Goodman. "Open-domain clarification question generation without question examples". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 563–570. DOI: 10.18653/v1/2021.emnlp-main.44.

[241] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[242] Di Wu, Liang Ding, Fan Lu, and Jian Xie. "SlotRefine: A Fast Non-Autoregressive Model for Joint Intent Detection and Slot Filling". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 1932–1937. DOI: 10.18653/v1/2020.emnlp-main.152.

[243] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. "LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory". In: *arXiv preprint arXiv:2410.10813* (2024).

[244] Yating Wu, Ritika Mangla, Alexandros G Dimakis, Greg Durrett, and Junyi Jessy Li. "Which questions should I answer? Salience Prediction of Inquisitive Questions". In: *arXiv preprint arXiv:2404.10917* (2024).

[245] Alexandros Xenos, John Pavlopoulos, and Ion Androutsopoulos. "Context Sensitivity Estimation in Toxicity Detection". In: *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*. Ed. by Aida Mostafazadeh Davani, Douwe Kiela, Mathias Lambert, Bertie Vidgen, Vinodkumar Prabhakaran, and Zeerak Waseem. Online: Association for Computational Linguistics, Aug. 2021, pp. 140–145. DOI: 10.18653/v1/2021.woah-1.15.

[246] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. "An Explanation of In-context Learning as Implicit Bayesian Inference". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=RdJVFCHjUMI.

[247] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. "Expertprompting: Instructing large language models to be distinguished experts". In: *arXiv preprint arXiv:2305.14688* (2023).

[248] Jing Xu, Arthur Szlam, and Jason Weston. "Beyond Goldfish Memory: Long-Term Open-Domain Conversation". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5180–5197. DOI: 10.18653/v1/2022.acl-long.356.

[249] Jing Xu, Arthur Szlam, and Jason Weston. "Beyond Goldfish Memory: Long-Term Open-Domain Conversation". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 5180–5197.

[250] Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. "Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 2159–2169. URL: https://aclanthology.org/C18-1183.

[251] Yi Yang and Arzoo Katiyar. "Simple and Effective Few-Shot Named Entity Recognition with Structured Nearest Neighbor Learning". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 6365–6375. DOI: 10.18653/v1/2020.emnlp-main.516.

[252] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. "CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7163–7189. DOI: 10.18653/v1/2021.emnlp-main.572.

[253] Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. "Adaptive Semiparametric Language Models". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 362–373. DOI: 10.1162/tacl_a_00371.

[254] Lili Yu, Howard Chen, Sida I. Wang, Tao Lei, and Yoav Artzi. "Interactive Classification by Asking Informative Questions". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 2664–2680. DOI: 10.18653/v1/2020.acl-main.237.

[255] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. "Generate rather than Retrieve: Large Language Models are Strong Context Generators". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=fB0hRu9GZUS.

[256] Joy He-Yueya, Noah D Goodman, and Emma Brunskill. "Evaluating and Optimizing Educational Content with Large Language Model Judgments". In: *arXiv preprint arXiv:2403.02795* (2024).

[257] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. "Big Bird: Transformers for Longer Sequences." In: *NeurIPS*. 2020.

[258] Omar Zaidan and Jason Eisner. "Modeling annotators: A generative approach to learning from annotator rationales". In: *Proceedings of the 2008 conference on Empirical methods in natural language processing*. 2008, pp. 31–40.

[259] Omar Zaidan, Jason Eisner, and Christine Piatko. "Using "Annotator Rationales" to Improve Machine Learning for Text Categorization". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Ed. by Candace Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai. Rochester, New York: Association for Computational Linguistics, Apr. 2007, pp. 260–267. URL: https://aclanthology.org/N07-1033/.

[260] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. "Generating clarifying questions for information retrieval". In: *Proceedings of the web conference 2020*. 2020, pp. 418–428.

[261] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. "Star: Bootstrapping reasoning with reasoning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15476–15488.

[262] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. "SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 93–104. DOI: 10.18653/v1/D18-1009.

[263] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. "BERTScore: Evaluating Text Generation with BERT". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

[264] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification". In: *arXiv:1509.01626 [cs]* (Sept. 2015). arXiv: 1509.01626 [cs].

[265] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. "DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Ed. by Asli Celikyilmaz and Tsung-Hsien Wen. Online: Association for Computational Linguistics, July 2020, pp. 270–278. DOI: 10.18653/v1/2020.acl-demos.30.

[266] Zheyuan Zhang, Daniel Zhang-Li, Jifan Yu, Linlu Gong, Jinchang Zhou, Zhiyuan Liu, Lei Hou, and Juanzi Li. "Simulating classroom education with llm-empowered agents". In: *arXiv preprint arXiv:2406.19226* (2024).

[267] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

[268] Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. "CompeteAI: Understanding the Competition Dynamics of Large Language Model-based Agents". In: *Forty-first International Conference on Machine Learning*.

[269] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. "Calibrate before use: Improving few-shot performance of language models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12697–12706.

[270] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. "Memorybank: Enhancing large language models with long-term memory". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 17. 2024, pp. 19724–19731.

[271] Zexuan Zhong, Tao Lei, and Danqi Chen. "Training Language Models with Memory Augmentation". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5657–5673. URL: https://aclanthology.org/2022.emnlp-main.382.

[272] Zhi Zhong and Hwee Tou Ng. "Word Sense Disambiguation Improves Information Retrieval". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Haizhou Li, Chin-Yew Lin, Miles Osborne, Gary Geunbae Lee, and Jong C. Park. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 273–282. URL: https://aclanthology.org/P12-1029/.

[273] Pei Zhou, Hyundong Cho, Pegah Jandaghi, Dong-Ho Lee, Bill Yuchen Lin, Jay Pujara, and Xiang Ren. "Reflect, Not Reflex: Inference-Based Common Ground Improves Dialogue Response Quality". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 10450–10468. DOI: 10.18653/v1/2022.emnlp-main.714.

[274] Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed H Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. "Self-discover: Large language models self-compose reasoning structures". In: *arXiv preprint arXiv:2402.03620* (2024).

[275] Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, Xiaodan Liang, Maosong Sun, Chenyan Xiong, and Jian Tang. "Towards interpretable natural language understanding with explanations as latent variables". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6803–6814.

[276] Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, et al. "Sotopia: Interactive evaluation for social intelligence in language agents". In: *arXiv preprint arXiv:2310.11667* (2023).

[277] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. "Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 4732–4740.

[278] Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. "HILDIF: Interactive Debugging of NLI Models Using Influence Functions". In: *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1–6. DOI: 10.18653/v1/2021.internlp-1.1.