Daniel Fried: Statement of Career Goals

1 Introduction

My research aims to make it easier for people to use language to interact with computers to carry out real-world tasks. Advances in natural language processing (NLP) and machine learning have laid a foundation for **language interfaces**, such as ChatGPT and Github's Copilot, that are used by millions of people to aid in a variety of computer-assisted tasks. However, the full potential and range of these systems are still untapped. The main goals of my work are to make language interfaces more widely applicable and better collaborators with their human partners, and to develop new computational models of interaction inspired by human language and communication. Toward these goals, my work covers three core areas.

Language grounding. To cover the full range of tasks that people would like to do with computers, language interfaces must extend beyond text. My work on language grounding ties language to extra-linguistic modalities such as images, web pages, semi-structured data, and sequential actions or decisions in real-world environments. Grounding language in these ways enable people to carry out goals in contextually rich settings, using language and computers.

Interactive communication. Current language interfaces lack a variety of communicative abilities needed to make interactions with people efficient and consistently successful. To build interfaces that are easier to communicate with, we need to model linguistic pragmatics: tying language to context, conventions, and people's goals and likely interpretations. My work models the human partners that NLP systems interact with, and shows that reasoning about these partners improves the ability to communicate pragmatically.

Code generation. Language interfaces for code generation have made programming more accessible to novices and more efficient for professionals. However, current interfaces still fall far short of the collaborative and communicative process of human pair programming. My work has built models, benchmarks, and algorithms for language-based generation and refinement of program code. Going forward, I plan to use code generation — and, more broadly, software development — as a focal domain for grounding and modeling partners in interactive communication.

Research Growth at CMU

How has my research expanded since arriving at CMU? While I also worked on language grounding and interactive communication during my PhD, I've broaded my work in these areas. In language grounding, I've undertaken new directions on grounding LLMs to images [8, 6] and using LLMs as agents in web environments [28, 6]. My work on interactive communication builds on the work I did on RSA models of pragmatics during my PhD, but extends it to settings involving multi-turn interaction [5, 12, 4, 3] and program synthesis [20, 13].

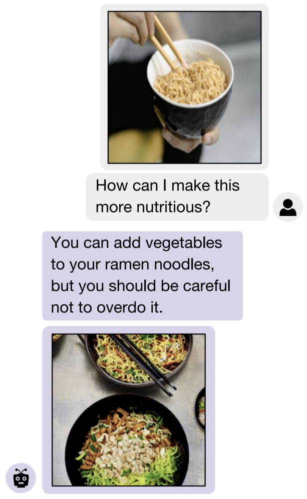
Code generation has been a larger area of growth. I began work on code generation while I was a post-doc at Meta [14, 15]. At CMU since then, my work in this area spans models, methods, and datasets for code generation tasks [25, 13, 20, 24, 23, 27, 1, 10, 21, 30], as well as using code as a representation of natural language meaning for other downstream tasks [22, 3, 2]. This work has led to a consulting role with GitHub and an advisory role with the BigCode research collaboration [1, 10, 30]. I have also sought to introduce students to this burgeoning research area by developing and teaching a new course, 11-891 Neural Code Generation, together with Sean Welleck.

My contributions to each of these areas, together with my advisees and collaborators, are outlined below.

2 Language Grounding

Large language models (LLMs) have had a major impact on the vast majority of NLP tasks, whether through direct application to the task or as a component in larger systems. However, the strongest publicly available LLMs have been text-only. To carry out a broader range of tasks in the world, we need to go beyond text and *ground language*: tie language to other modalities such as images and actions in an environment. Since starting at CMU, I have worked in two main directions to make LLMs usable in grounded environments.

Grounding LLMs to images. Multimodal generation models, such as DALL-E and Stable Diffusion, are capable of producing photorealistic images from text descriptions. Our recent work [8, 7] has extended these approaches to also use dialogue and other images as inputs to image retrieval and generation (Figure 1). To do this, we learn translation layers between the embedding spaces of frozen, pre-trained models (LLMs, image encoders, and image generation models). Using a relatively small amount of data and compute, we are able to lift many of the abilities of these frozen pre-trained LLMs learned from text-only data (in-context learning, dialogue and discourse, and world knowledge) to a multimodal setting. More broadly, by embedding images in a communicative context, we aim to lay a foundation for a range of tasks such as interactive image editing, multimodal article summarization, or illustrating how-to instructions.


How can I make this more nutritious?

You can add vegetables to your ramen noodles, but you should be careful not to overdo it.

Figure 1: Our models FROMAGe [7] and GILL [8], combine large language models with image retrieval and generation to allow multimodal dialogue with images as inputs and outputs. Example interaction with GILL shown.

Grounded LLMs as web agents. Our work has developed benchmarks and models for using LLMs to take actions in web environments, conditioned on language guidance from a person. This enables language interfaces to web tasks, which for many people are a primary source of interaction with the digital world. Together with others at CMU, we built the WebArena [28] benchmark, which provided the first sandboxed set of realistic tasks in web environments; and VisualWebArena [6], which extended these tasks and environments to use images as grounding. In ongoing work, we are exploring methods to improve the performance of LLM-based systems in these and related domains. Our current focuses are on developing search-based planning methods and stronger pre-trained multimodal LLMs which are grounded in web modalities such as images and semi-structured data.

Future directions. I am interested in language grounding primarily because so many tasks that involve communication between people and machines rely on extra-linguistic context. This makes language grounding a necessary component of many interactive tasks and domains. I aim to do the core analysis, modeling, and benchmarking work necessary to make language grounding usable across a range of modalities. For high-resource modalities such as images and web trajectories, I plan to continue industry collaborations to steer progress in large multimodal model development. In other lower-resourced modalities such as visual CAD designs, plots and charts, and graphical interfaces, academia will be equally capable of developing performant models, and I expect many of these modalities to be a direct focus of my group going forward.

3 Interactive Communication

Language is often strategic. People choose what to say by predicting how listeners will interpret it and try to understand speakers by thinking about why they might have said what they did. However, current NLP systems lack the ability to communicate strategically. Systems do not adapt to the knowledge, skills, or communication styles of their human partners; are often either ambiguous or too wordy; and make assumptions rather than asking questions or accommodating corrections.

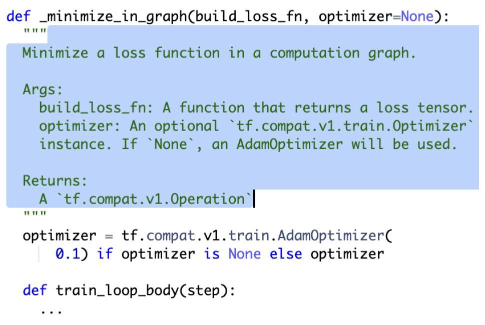
My work aims to *improve natural language interfaces by modeling communication as a cooperative game*, where speakers and listeners must understand each other in order to succeed. In our recent survey and position paper [19], we outline work in this area and make an argument for developing these capabilities in NLP systems. Our work has trained statistical *partner models* of people's communicative behavior and embedded these models within natural language interfaces for a range of tasks. These approaches allow language interfaces to address some of their current communicative shortcomings: conveying information concisely but adequately by modeling the costs of communication, adapting to their partners' linguistic conventions and preferences inferred over the course of interaction, and building common ground by inferring what the partner knows and wants, and what the system needs to ask. In work prior to CMU, we used game theoretic or Bayesian reasoning about partner models to improve state-of-the-art systems for a diverse set of tasks, including generating and interpreting navigational instructions [16, 18] and visually-grounded dialogue [17]. Below, we outline work we've done at CMU. In all of these works, I have found that real-world language systems are more successful when they learn models of their human partners and use these models to reason about cooperative goals.

Pragmatic code synthesis. Programs can be specified using a variety of communicative modalities, such as natural language descriptions, partial code snippets, and example inputs and outputs. But there is an inherent tension between the ambiguity of these specifications, which are consistent with many possible programs, and the precise semantics of any single program. To address this, we've developed approaches for resolving ambiguity in human specifications of programs. Our approaches use partner models and reasoning about how humans choose specifications to be maximally informative [27]. The precise, but non-pragmatic, semantics of programs allows us to extend our previous work on RSA-based pragmatics [16] to generate data and amortize pragmatic reasoning into learned models [20, 13]. This enables the models to accurately resolve ambiguity in human-written specifications, despite using minimal human-produced data in training.

Multi-turn settings. We've proposed approaches for building common ground in multi-turn interactions using probabilistic modeling of partner intentions and information-theoretic planning. Much of this work has been on dialogue-based visual reference games [5, 4, 3]. In all settings, we use partner models to infer what a human interlocutor intended, updating a distribution over the referents that they may have. We then have the system generate informative queries for the user, using an information gain objective and partner models of how the user might respond to queries. We've shown that these approaches improve system accuracy and communicative efficiency in interaction with human users. In future work, we plan to apply similar approaches to interactive code generation and grounded collaborative construction tasks.

Future directions. I aim to develop interactive systems in settings where communicative intents cannot be explicitly enumerated or represented symbolically. Examples include non-literal language and social settings, which we have began to explore in collaboration with others at CMU [11, 26, 29]. I am also interested in systems that adapt to their human partners — not just in a single interaction with a single person, but over the course of time with both individuals and communities. I expect explicit partner models to be a fruitful avenue here as well. As in our past work, we will evaluate our systems in interactions with people: deploying the systems with real end-users and seeing if they help people achieve tasks more successfully with lower communicative cost, learn more, and enjoy their interactions with the systems.

4 Code Generation



```
def _minimize_in_graph(build_loss_fn, optimizer=None):
  """
  Minimize a loss function in a computation graph.

  Args:
    build_loss_fn: A function that returns a loss tensor.
    optimizer: An optional `tf.compat.v1.train.Optimizer`
    instance. If `None`, an AdamOptimizer will be used.

  Returns:
    A `tf.compat.v1.Operation`
  """
  optimizer = tf.compat.v1.train.AdamOptimizer(
      0.1) if optimizer is None else optimizer

  def train_loop_body(step):
    ...
```

Figure 2: Our InCoder models [15] were the first open-source models which can *infill*: complete arbitrary regions of code (example output highlighted), laying a foundation for our work on interactive code generation in broad domains. The InCoder models have been downloaded around 200,000 times. As part of the BigCode research consortium, we helped scale a similar approach to produce the StarCoder models [1, 10].

Current systems for machine learning-augmented code generation, such as GitHub's Copilot and OpenAI's Codex, are used by millions of software developers. These systems provide sophisticated auto-complete functionality: proposing completions for a programmer's partial code or natural language instructions. However, these systems still fall far short of the experience of collaboratively writing software with another person: people hold dialogues about code to ask each other questions and give feedback, point at the screen and write test cases for each other, and adapt to each others' communication styles and preferences. In fact, the communication abilities of current systems are limited enough that they can actually sometimes make programmers write *less efficient* or *less accurate code*. The long-term aim of my work is to use grounded interaction, with varied communicative modalities, to fix these issues and recapture some of the benefits of human-human interaction for collaborative software development. Below, I describe progress we've made in these directions.

Language-to-code. Our recent work has developed models, algorithms, and benchmark tasks for producing code from natural language. While code generation is a natural fit for LLMs, prior work applying LLMs to code generation largely focused on limited settings of left-to-right generation of solutions to code interview problems. A major focus of our work has been improving the usefulness of code models for everyday tasks, with modeling contributions such as infilling [15, 10, 1] (Figure 2) and evaluations on open-domain code [9, 23, 21, 25, 30]. In future work, we plan to continue to develop language-to-code components that will be useful for downstream systems for interactive software development.

Code for grounded tasks. Most performant LLMs are now trained on mixtures of natural language and code, making them able to perform semantic parsing using code as an representation of language meaning. Given a library of grounded functions, these code representations can be executed in an environment. This has allowed us to develop code LLM-based methods that achieve state-of-the-art performance on question answering tasks involving tables [2], visual dialogue [3], and images [22]. In recent work, we have shown that we can have the LLMs write their own reusable grounded functions [22], increasing task performance and making it more efficient for humans to verify the code's correctness.

Future directions. We plan to develop grounded code systems: allowing a developer to communicate using a combination of natural language, code edits, example inputs and outputs, and deictic gestures; and allowing the system to issue queries that the user can answer with any of these modalities. We have three aims in this work, all of which will be enabled by explicit pragmatic modeling and partner models (here, programmer models): (1) Allowing systems to proactively ask for feedback from the user when the user's intent is unclear or seems mistaken, using a learned model of the user. (2) Modeling the communicative cost of the user's feedback: e.g., if the user gives a correction in natural language rather than editing the code directly, it probably means it would be difficult for the user to edit the current code. (3) Adapting to a varied range of user preferences and abilities, for example accommodating the working memory needs of neuro-divergent programmers. Our overall goal is to produce better collaborative code generation tools that developers can build working relationships with.

Expected Impacts

Our approaches aim to (1) ground language interfaces in a wider range of modalities and (2) make interfaces more practical and efficient to use by modeling user intent, interpretations, and communication style. A major obstacle to the widespread adoption of language-only interfaces has been their communicative inefficiency compared to graphical interfaces or to a sufficiently-skilled user simply performing the task themselves without the computer's assistance. While interfaces based on LLMs are now widely known and, particularly in the case of code generation, used (by millions of developers), these interfaces still are suboptimal communicators. Improving interface efficiency, helpfulness, and efficacy through grounding and pragmatics could benefit the the broad section of the population that is already using language interfaces. Our work should also allow more people — in particular those who are less able to use non-language-based or unimodal interfaces, e.g., visually-impaired users and novice programmers — to more easily use the computational tools that the language interfaces operate.

I am excited to continue work on these directions in collaboration with others.

References

[1] L. B. Allal, R. Li, D. Kocetkov, C. Mou, C. Akiki, C. M. Ferrandis, N. Muennighoff, M. Mishra, A. Gu, M. Dey, L. K. Umapathi, C. J. Anderson, Y. Zi, J. L. Poirier, H. Schoelkopf, S. Troshin, D. Abulkhanov, M. Romero, M. Lappert, F. D. Toni, B. G. d. Río, Q. Liu, S. Bose, U. Bhattacharyya, T. Y. Zhuo, I. Yu, P. Villegas, M. Zocca, S. Mangrulkar, D. Lansky, H. Nguyen, D. Contractor, L. Villa, J. Li, D. Bahdanau, Y. Jernite, S. Hughes, D. Fried, A. Guha, H. d. Vries, and L. v. Werra. SantaCoder: Don't reach for the stars. In *Deep Learning for Code Workshop*, 2023.

[2] Y. Cao, S. Chen, R. Liu, Z. Wang, and D. Fried. API-assisted code generation for question answering on varied table structures. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[3] J. Chiu, W. Zhao, D. Chen, S. Vaduguru, A. Rush, and D. Fried. Symbolic planning and code generation for grounded dialogue. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[4] J. Chiu, W. Zhao, A. M. Rush, and D. Fried. Modeling perspective-dependent ambiguity in grounded collaborative dialogue. In *Wordplay: When Language Meets Games Workshop*, 2022.

[5] S. Keh, J. T. Chiu, and D. Fried. Asking more informative questions for grounded retrieval. In *Findings of NAACL*, 2024.

[6] J. Y. Koh, R. Lo, L. Jang, V. Duvvur, M. C. Lim, P.-Y. Huang, G. Neubig, S. Zhou, R. Salakhutdinov, and D. Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[7] J. Y. Koh, R. Salakhutdinov, and D. Fried. Grounding language models to images for multimodal inputs and outputs. In *International Conference on Machine Learning (ICML)*, 2023.

[8] J. Y. Koh, D. Fried, and R. Salakhutdinov. Generating images with multimodal language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[9] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, S. W.-t. Yih, D. Fried, S. I. Wang, and T. Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning (ICML)*, 2023.

[10] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov,

Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. v. Werra, and H. d. Vries. StarCoder: May the source be with you! *Transactions on Machine Learning Research (TMLR)*, 2023.

[11] A. Liu, M. T. Diab, and D. Fried. Evaluating large language model biases in person-steered generation. In *Findings of ACL*, 2024.

[12] J. Ou, B. Krojer, and D. Fried. Pragmatic inference with a CLIP listener for contrastive captioning. In *Findings of ACL*, 2023.

[13] Y. Pu, S. Vaduguru, P. Vaithilingam, E. Glassman, and D. Fried. Amortizing pragmatic program synthesis with rankings. In *International Conference on Machine Learning (ICML)*, 2024.

[14] F. Shi, D. Fried, M. Ghazvininejad, L. Zettlemoyer, and S. I. Wang. Natural language to code translation with execution. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[15] D. Fried, A. Aghajanyan, J. Lin, S. I. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, and M. Lewis. InCoder: A generative model for code infilling and synthesis. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] D. Fried, J. Andreas, and D. Klein. Unified pragmatic models for generating and following instructions. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[17] D. Fried, J. Chiu, and D. Klein. Reference-centric models for grounded collaborative dialogue. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[18] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[19] D. Fried, N. Tomlin, J. Hu, R. Patel, and A. Nematzadeh. Pragmatics in language grounding: Phenomena, tasks, and modeling approaches. In *Findings of EMNLP*, 2023.

[20] S. Vaduguru, D. Fried, and Y. Pu. Generating pragmatic examples to train neural program synthesizers. In *International Conference on Learning Representations (ICLR)*, 2024.

[21] Z. Wang, A. Asai, X. V. Yu, F. F. Xu, Y. Xie, D. Fried, and G. Neubig. CodeRAG-Bench: Can retrieval augment code generation? *in preparation*, 2024.

[22] Z. Wang, G. Neubig, and D. Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *International Conference on Machine Learning (ICML)*, 2024.

[23] Z. Wang, S. Zhou, D. Fried, and G. Neubig. Execution-based evaluation for open-domain code generation. In *Findings of EMNLP*, 2023.

[24] Y. Xie, A. Naik, D. Fried, and C. Rose. Data augmentation for code translation with comparable corpora and multiple references. In *Findings of EMNLP*, 2023.

[25] Y. Xie, A. Xie, D. Sheth, P. Liu, D. Fried, and C. Rose. CodeBenchGen: Creating scalable execution-based code generation benchmarks. *arXiv*, 2024.

[26] A. Yerukola, S. Vaduguru, D. Fried, and M. Sap. Is the pope Catholic? yes, the pope is Catholic. generative evaluation of intent resolution in llms. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[27] T. Zhang, T. Yu, T. B. Hashimoto, M. Lewis, W.-t. Yih, D. Fried, and S. I. Wang. Coder reviewer reranking for code generation. In *International Conference on Machine Learning (ICML)*, 2023.

[28] S. Zhou, F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, Y. Bisk, D. Fried, U. Alon, and G. Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

[29] X. Zhou, H. Zhu, L. Mathur, R. Zhang, H. Yu, Z. Qi, L.-P. Morency, Y. Bisk, D. Fried, G. Neubig, and M. Sap. Sotopia: Interactive evaluation for social intelligence in language agents. In *International Conference on Learning Representations (ICLR)*, 2024.

[30] T. Y. Zhuo, V. M. Chien, H. Hu, J. Chim, W. Yu, R. Widyasari, I. N. B. Yusuf, H. Zhan, J. He, I. Paul, S. Brunner, C. GONG, J. Hoang, A. R. Zebaze, X. Hong, W.-D. Li, J. Kaddour, M. Xu, Z. Zhang, P. Yadav, N. Jain, A. Gu, Z. Cheng, L. B. allal, Q. Liu, Z. Wang, D. Lo, B. Hui, N. Muennighoff, D. Fried, X. Du, H. d. Vries, and L. V. Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. *in preparation*, 2024.