PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts

Stephen H. Bach[*1,2] **Victor Sanh**[*3] **Zheng-Xin Yong**[1] **Albert Webson**[1] **Colin Raffel**[3]
Nihal V. Nayak[1] **Abheesht Sharma**[4] **Taewoon Kim**[5] **M Saiful Bari**[6] **Thibault Fevry**[7]
Zaid Alyafeai[8] **Manan Dey**[9] **Andrea Santilli**[10] **Zhiqing Sun**[11] **Srulik Ben-David**[12]
Canwen Xu[13] **Gunjan Chhablani**[7] **Han Wang**[14] **Jason Alan Fries**[15,2]
Maged S. Al-shaibani[8] **Shanya Sharma**[16] **Urmish Thakker**[17] **Khalid Almubarak**[18]
Xiangru Tang[19] **Dragomir Radev**[19] **Mike Tian-Jian Jiang**[20] **Alexander M. Rush**[3]
[1] Brown University [2] Snorkel AI [3] Hugging Face [4] BITS Pilani [5] VU Amsterdam
[6] NTU [7] BigScience [8] KFUPM [9] SAP [10] University of Rome [11] CMU [12] Technion
[13] UCSD [14] NYU [15] Stanford University [16] Walmart Labs [17] SambaNova Systems
[18] PSAU [19] Yale University [20] ZEALS [*] Equal Contribution

Abstract

PromptSource is a system for creating, sharing, and using natural language prompts. Prompts are functions that map an example from a dataset to a natural language input and target output. Using prompts to train and query language models is an emerging area in NLP that requires new tools that let users develop and refine these prompts collaboratively. *PromptSource* addresses the emergent challenges in this new setting with (1) a templating language for defining data-linked prompts, (2) an interface that lets users quickly iterate on prompt development by observing outputs of their prompts on many examples, and (3) a community-driven set of guidelines for contributing new prompts to a common pool. Over 2,000 prompts for roughly 170 datasets are already available in *PromptSource*. *PromptSource* is available at `https://github.com/bigscience-workshop/promptsource`.

1 Introduction

Prompt engineering is emerging as a new focus in NLP, particularly in zero- and few-shot learning settings. *Prompting* is the practice of representing a task as a natural language utterance in order to query a language model for a response (Liu et al., 2021). For example, if a language model is conditioned on the text *"She hit a home run. The previous sentence is about ..."*, then the model's subsequent generation would be interpreted as a prediction of the topic of the preceding sentence, e.g. by mapping a response such as *"sports"* to a label class. In specific contexts, prompting has been shown to have advantages over traditional classification, for example facilitating adaptation of language models to ad-hoc tasks and improving sample efficiency in low-data settings (Brown et al., 2020; Schick and Schütze, 2021b; Le Scao and Rush, 2021; Gao et al., 2021). These advantages motivate a practical challenge: *How can we enable users to create, refine, and share prompts?*

The process of prompt engineering is critical for successful deployment as choices in prompting can affect downstream predictions significantly, particularly in the zero-shot setting (Perez et al., 2021; Zhao et al., 2021; Webson and Pavlick, 2021). Furthermore, training directly on collections of prompts can enable large models to generalize to new prompts more robustly (Sanh et al., 2021; Wei et al., 2021; Min et al., 2021; Mishra et al., 2021). There is therefore a growing need for tools that support the creation of corpora of prompts.

PromptSource is an integrated development environment and repository for natural language prompts to use in the context of zero-shot (or gradient-based few-shot) learning. It provides a Web-based GUI that enables developers to write prompts in a templating language and immediately view their outputs on different examples. The system is integrated with the HuggingFace Datasets library (Lhoest et al., 2021), so that users can load any dataset automatically, browse existing prompts, and create new ones. Through the course of writing thousands of prompts, we converged on three key

aspects to the design of *PromptSource*:

- **Flexible Templating Language.** We adapt a templating language to represent prompts. Prompt authors can define prompts in terms of dataset fields, hard-coded text, and simple control logic. This choice provides the flexibility of a programming environment without the mental overhead of having to write and read arbitrary code. Prompt templates can easily be distributed and used in other systems.
- **Tools for Prompt Management.** *PromptSource* has multiple view to address the needs of prompt authors at different stages of the prompt engineering cycle. A global views let authors browse datasets and existing prompt templates. A local views facilitates iteration on prompt wording and metadata as well as testing on individual examples.
- **Community-Driven Quality Standards.** *PromptSource* includes a set of guidelines for prompting based on a large-scale prompt writing pilot. *PromptSource*'s collection is meant to be useful for a wide range of research, based on iterative refinement of a set of quality standards. Prompts in *PromptSource* are also annotated with various pieces of metadata to make finding and using prompts easier.

The *PromptSource* system includes over 2,000 open-source prompts for roughly 170 datasets, which have all been reviewed to meet the quality standards. This collection, which we call the Public Pool of Prompts (P3), allows users to materialize prompted forms of datasets for hundreds of different tasks. The T0 series of models (Sanh et al., 2021) for zero-shot inference were fine-tuned on a subset of P3. Since then, *PromptSource* and P3 have been extended for research on multi-lingual prompting (Lin et al., 2021) and priming, i.e., in-context few-shot learning (Min et al., 2021). The *PromptSource* system and associated content is a first step in the study of systems for prompt engineering, an area that is likely to continue to grow.

2 Background and Related Work

PromptSource builds on recent work in prompting and prompt engineering. It is also related to work on systems for other types of annotations.

Prompting Recently, prompting has emerged as a new focus within NLP as it can dramatically improve language models' few-shot and zero-shot performance in a wide range of downstream tasks (Brown et al., 2020; Schick and Schütze, 2021a; Sanh et al., 2021; Wei et al., 2021). Prompts and prompt engineering come in several varieties (Liu et al., 2021). *PromptSource* is focused on facilitating research with human-written prompts, in which natural language is the medium for describing tasks. This approach has the advantage that prompts can be understood, modified, and applied without being tied to a specific model. In contrast, past work has also aimed to automatically construct prompts by framing the search for a good prompt as a learning problem. These prompts can either be expressed in natural language (Gao et al., 2021; Shin et al., 2020) or as arbitrary vectors (a.k.a. "continuous" or "soft" prompts) not corresponding to words in the model's original vocabulary (Lester et al., 2021; Qin and Eisner, 2021)

When using human-written prompts, there are several possible approaches to learning. One is a zero-shot setting, where the goal is to generalize to prompts for which no training examples are given. Prompts can also be used in a few-shot setting, in which a model is either (1) trained on prompted examples of the target task via gradient updates, or (2) priming (i.e. in-context learning), in which labeled examples are included in an input sequence in order to prime models to make predictions without gradient updates (Brown et al., 2020).

PromptSource was originally designed for zero-shot learning, so it emphasizes explicit task instructions and no priming examples. If needed, users can extend *PromptSource* for few-shot learning (e.g., as done in Lin et al., 2021 and Min et al., 2021, described in §7).

Systems for Annotating Data Most work on collecting annotations has focused on labels and other annotations at the level of individual examples (Neves and Ševa, 2021). GATE (Cunningham et al., 2002) was an early system for annotating text, and includes support for many data types such as labels and entity tags. Since then, many Web-based systems for annotating text have been developed (Stenetorp et al., 2012; Salgado et al., 2012; Wei et al., 2013; Yimam et al., 2013; Chen and Styler, 2013; Eckart de Castilho et al., 2016; Putra et al., 2020). Other systems support collaboration among multiple annotators (Yang et al., 2018; Stewart et al., 2019). More recently, many annotation systems have begun to incorporate learned models to improve workflow, using techniques such as ac-

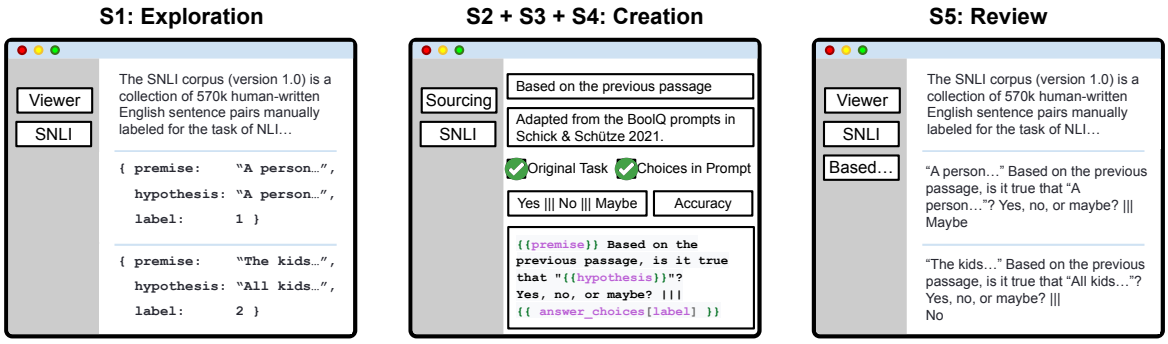


Figure 1: The five stages of creating prompts in *PromptSource*. The Browse view for Dataset Exploration (S1). The Sourcing view for Prompt Writing (S2), Prompt Documentation (S3), and Iteration and Variation (S4). The Browse view for performing a Global Review (S5).

tive learning (Lin et al., 2019; Li et al., 2021) and example recommendation (Lee et al., 2020; Kiela et al., 2021). These systems are possible because the annotations to be collected are labels, for which metrics like inter-annotator agreement and model confidence are available.

There has also been some work on collecting annotations other than labels. AlvisAE (Papazian et al., 2012) and TreeAnnotator (Helfrich et al., 2018) support creating ontologies and other structured annotations. Prompts differ from these annotations in that they are semi-structured functions, requiring new tools for developers.

3 System Design and Workflow

Creating prompts differs from other types of data collection and annotation. We note three different challenge axes along which prompting differs from traditional NLP annotation:

- **Functions, not Labels.** A single prompt is a function that maps dataset examples (dictionaries of arbitrary fields) to natural language input/target pairs. Creating a prompt is therefore more like programming than typical data annotation. How should a prompt format trade off between expressivity and simplicity?
- **Dataset-Level Choices.** Prompts are associated with datasets, unlike label annotations that are local to single examples. Prompt engineering requires developers to evaluate their choices across all examples. What interfaces do authors need to inspect and debug their prompts?
- **Variation in Prompt Construction.** Unlike with labels, it is often desirable to have variation within prompt construction, as different prompt choices may lead to different results. However, variation complicates quality judgment, and makes it impossible to apply simple metrics like inter-annotator agreement. How can multiple authors collaborate to build a high-quality corpus of prompts and associated metadata?

To illustrate these distinct axes, we start with a concrete overview of the prompt creation process of *PromptSource*. For this example, we imagine that a user of *PromptSource* is creating prompts for a natural language inference dataset, specifically SNLI (Bowman et al., 2015). The goal is to design a prompt query such that the answer can be mapped onto the SNLI classes. A prompt author can accomplish this goal with *PromptSource* via the following five steps (Figure 1):

S1: Dataset Exploration The prompt author first needs to read the dataset description, including linked READMEs and papers, and to browse through example rows. In this case, they would see that SNLI is a dataset for natural language inference: assume a given premise sentence is true, the goal is to determine whether a hypothesis sentence is true (entailment), false (contradiction), or undetermined (neutral).

S2: Prompt Writing The prompt author uses the *sourcing* mode to try out a prompt wording, and then adjusts it by observing prompted examples (Figure 1 middle, full example in Figure 5 and 6 in Appendix B).

S3: Prompt Documentation To facilitate using the prompt, the author fills in various metadata including possible metrics to evaluate the prompt , valid outputs if applicable , whether the prompt expresses the original intended task of the dataset and whether the template explicitly states the valid outputs.

S4: Iteration and Variation The prompt author then iterates through S2 and S3 to create multiple prompts for the dataset. Authors are encouraged to vary multiple factors such as the formulation of the prompt and the targeted task (see Section 6).

S5: Global Review The author saves the draft prompts in a structured file which are then verified by other contributors through code reviews. New prompts need to meet the quality standard with a series of automatic tests and by validation through prompted instances. Upon passing review, the new prompts can be merged into a global prompts collection.

Upon submission, prompts can be viewed through *PromptSource* by other users. The full collection is stored globally and can be used outside of the tool, for instance to be applied on an example from a dataset of the *Datasets* library (Lhoest et al., 2021).

```
from promptsource.templates import DatasetTemplates

prompts = DatasetTemplates("snli")
prompt_key = "Based on the previous passage"
p = prompts[prompt_key]

result = p.apply(example)
print("INPUT: ", result[0])
print("TARGET: ", result[1])
```

With this workflow in mind, we next describe the key aspects of the *PromptSource* system in greater detail.

4 Prompting Language

A key design decision is the format for prompts. We note that previous works on prompting tended to utilize code for specifying each prompt. We experimented with this format and found a trade-off between expressivity and explicit structure. On one side, a maximally expressive format such as pure Python code would let users write complex programs to manipulate the semi-structured examples into prompted examples. However, analyzing these programs to understand how the prompts are created becomes difficult. This difficulty limits its downstream manipulation and analysis of the prompts, for example for possible future work on automatic prompt augmentation. On the other side, a maximally structured format, such as rule-based generation, limits the kinds of prompts that users can create. We found it infeasible to enumerate types of rules sufficient for the wide range of tasks and data formats for which we wanted prompts.

We therefore settled on a middle ground between the two: a templating language. Specifically, we use the Jinja2 templating engine[1], originally designed for producing web markup. Users write templates as prompts with placeholders, such as `If {{premise}} is true, is it also true that {{hypothesis}}? ||| {{entailed}}.` The separator `|||` denotes the break between the conditioning text and the desired completion. Placeholders refer to fields in the underlying example (represented as a Python `dict` by *Datasets* (Lhoest et al., 2021)). Users also have access to Jinja's built-in functions, such as manipulating strings and structured data. For each prompt, prompted examples are created by applying the prompt to all examples in the corresponding dataset. While Jinja is a complete programming language, our review guidelines encourage simple functions with minimal additional logic (see Figure 5 and 6 for example).

During the development of *PromptSource*, we found that a few idioms were particularly useful. First, not all templates are applicable to all examples in a dataset. Users can wrap templates in Jinja's built-in conditional statements, and any example that results in an empty prompted example is simply skipped. Second, many examples can be used to make multiple training instances, such as a question that has multiple valid answers. We therefore added a `choice` function that selects an element from a list in a way that can be controlled during dataset generation, such as picking a random element using a seeded random number generator or generating different prompts for each combination of elements in the template. Third, many tasks such as classification and binary question answering have a small set of possible valid completions, and it is common to make predictions for these tasks by scoring only the valid completions and returning the highest one (Brown et al., 2020; Sanh et al., 2021; Wei et al., 2021). Users therefore can list the valid completions in a separate field and access them as a list in their prompts (displayed as `Answer choices` in Figure 5 in the appendix). These completions are then explicitly available when evaluating predictions for these prompted examples.

5 The *PromptSource* UI

The *PromptSource* system is designed to enable prompt creators to view data (S1), write prompts

[1] https://jinja.palletsprojects.com

Figure 2: Prompt creators can browse through the dataset examples (left-column) and their prompted form (right column) using the *Browse* view.

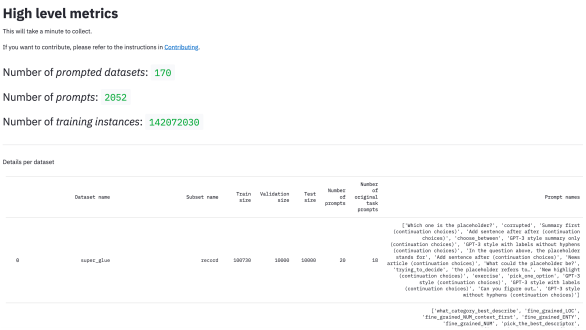


Figure 3: The *Helicopter* view indicates what datasets have prompts and how many prompts are available for each dataset.

in a standard format (S2, S3), and verify that their templates work correctly (S5). We implemented a lightweight interface for the tool in Streamlit[2] so that users could download, run locally in a web browser, and then upload their results to a central repository. Testing iterations of the interface on pilot template-writing tasks, we converged on three views for the interface.

V1: Sourcing This view (see Figure 5 in appendix) allows users to select a dataset to prompt, browse examples from that dataset in the form of tables, and enter a prompt for that dataset. As the user writes their template (S2), every time they save it, the output of the template applied to the current example is displayed next to the editor. We also collect metadata like a name for the template, and a reference for any bibliographic information or rationale for the template.

V2: Browse This view (see Figure 2) lets users select prompts and browse the prompted examples generated by them (S4). The original example is viewed side-by-side with the resulting prompted example, with the substituted text highlighted to distinguish from text hard-coded in the template. Users can quickly scroll through many examples, verify the behavior of their prompt, and return to the sourcing view if changes are needed.

V3 : Helicopter This view (see Figure 3) allows users to see what datasets are available for writing templates and how many are written for each, to prioritize user attention. This view is particularly useful for moving between datasets and for the prompt reviewers (S5).

[2] https://streamlit.io/

6 Community Guidelines and Process

Due to the variety of existing NLP datasets, we found out it challenging to exhaustively describe the characteristics of a good prompt: there are no simple metrics like inter-annotator agreement on example-level labels. Instead, over a few iterations, we converged on community guidelines[3] with 3 objectives in mind: (a) provide a standardized vocabulary for discussing prompts between prompt authors, reviewers and users, and minimum requirements for a valid prompt, (b) highlight common errors, and best practices, (c) collect the necessary information about the prompts to support current and future research on prompt engineering. The guidelines were enforced in the use of *PromptSource* by a code review process in which each prompt was reviewed before being committed to the central repository.

Guidelines apply to the combination of a template (a function that maps an example into an input/target pair in natural language) and a set of metadata about the template. The most important constraint we imposed for a template to be valid is to be formulated in natural language (both for the input and the target). We forbid the used of non-natural language prompts such as pure code. Each prompt should clearly state what task should be resolved, in a way a non-specialist adult can understand. We found this guideline strikes a good balance between freedom and expressivity in the wording of the prompts on one side and short generic prompts on the other side.

In early experiments, we found that user-written prompts that did not explicitly state the possible

[3] Complete guidelines can be found at https://github.com/bigscience-workshop/promptsource/blob/main/CONTRIBUTING.md.

valid completions tended to perform worse in experiments than their counterparts in which the possible valid completions were listed. We encouraged prompt authors to explicitly state the valid outputs in the prompt. In addition, when working with training prompts that include target text, we found it useful to remove variations on the target format that led to spurious ambiguity. For instance, the target template should only contain the answer to the task. It should not contain any extra text such as "The answer is ...", which can be equivalently moved to the input template.

One of the research question we hope to enable with *PromptSource* is whether the diversity of the prompt formulation during training leads to models that are more robust to the prompt formulation at test time. Therefore, we encouraged prompt authors to create between 5 and 10 (or more) prompts per dataset while varying the prompt formulation. For a given dataset, authors produce multiple prompts per example, sometimes for task formulations that differed from the original dataset. For instance, for question answering dataset, one prompt can ask to extract the answer to a given question from a given passage, while a second prompt can ask to generate a potential answer given an answer and a passage.

As part of the community process and to facilitate future research, *PromptSource* asks prompt authors to include additional few metadata for each prompt. Metadata fields include a name for the prompt, a reference to the paper it was extracted from (or any relevant explanation), whether the prompt expresses the task originally intended by the dataset, the valid outputs (if relevant), whether the input template states the valid outputs and possible metrics to evaluate the prompted examples. These can be used in future systems to evaluate how the style and structure of prompts leads to different downstream results.

7 Case Studies

A system for creating, maintaining and using prompts is a key tool for supporting the emerging research area of prompt engineering in a standardized and reproducible manner. We highlight three recent research for which *PromptSource* was a necessary resource.

Massively multitask fine-tuning Sanh et al. (2021) study the question of zero-shot behaviors in large language models and asks whether zero-shot generalization can be induced by training a language model on a massively multitask mixture of tasks. To test this question, they use *PromptSource* to create diverse prompts for a large collection of NLP datasets. P3 is the result of this collection and allows training a language model on a massively multitask mixture of prompted datasets and evaluate the ability of models trained with such a procedure to perform unseen tasks.

Multilingual prompting Lin et al. (2021) study the zero- and few-shot learning abilities of an multilingual autoregressive language model trained on 30 languages. In particular, they are interested in the cross-lingual generalization of such models and benchmark a variety of tasks in multiple languages. *PromptSource* allows using a massive set of high-quality English prompts. Moreover, the English prompts serve as support to create prompts in other languages (through either machine or human translation).

Priming (In-Context Learning) Min et al. (2021) study improving models' few-shot priming performance by first fully training a model (with gradient updates) on a multitask mixture formatted with priming examples. They find that incorporating templates from P3 significantly further improves performance compared to training on priming examples alone. Although *PromptSource* was not originally designed for this specific form priming, users were able to easily use P3's template collection and the templating language for their own priming methods.

8 Conclusion

PromptSource is an open-source system for creating, sharing and using natural language prompts and addresses the need for new collaborative and centralized tools to support the emerging research around prompt engineering. The tool is designed to answer three key needs: a flexible template language, a suite of tools for prompt management and community-driven quality standards. As of January 2022, *PromptSource* includes a growing collection of 2,000 public prompts for roughly 170 datasets, and has already been instrumental resource for multiple recent research.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference.

In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: an architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan. The COLING 2016 Organizing Committee.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. 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)*, pages 3816–3830, Online. Association for Computational Linguistics.

Philipp Helfrich, Elias Rieb, Giuseppe Abrami, Andy Lücking, and Alexander Mehler. 2018. TreeAnnotator: Versatile visual annotation of hierarchical text relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020. 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*, pages 372–379, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yanzeng Li, Bowen Yu, Li Quangang, and Tingwen Liu. 2021. FITAnnotator: A flexible and intelligent text annotation system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies: Demonstrations, pages 35–41, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. 2019. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin Stoyanov, and Xian Li. 2021. Few-shot learning with multilingual language models. *CoRR*, abs/2112.10668.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *CoRR*, abs/2110.15943.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Mariana Neves and Jurica Ševa. 2021. An extensive review of tools for manual annotation of documents. *Briefings in bioinformatics*, 22(1):146–163.

Frédéric Papazian, Robert Bossy, and Claire Nédellec. 2012. AlvisAE: a collaborative web text annotation editor for knowledge acquisition. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 149–152, Jeju, Republic of Korea. Association for Computational Linguistics.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *NeurIPS*.

Jan Wira Gotama Putra, Simone Teufel, Kana Matsumura, and Takenobu Tokunaga. 2020. TIARA: A tool for annotating discourse relations and sentence reordering. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6912–6920, Marseille, France. European Language Resources Association.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

David Salgado, Martin Krallinger, Marc Depaule, Elodie Drula, Ashish V. Tendulkar, Florian Leitner, Alfonso Valencia, and Christophe Marcelle. 2012. MyMiner: a web application for computer-assisted biocuration and text annotation. *Bioinformatics*, 28(17):2285–2287.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Michael Stewart, Wei Liu, and Rachel Cardell-Oliver. 2019. Redcoat: A collaborative annotation tool for hierarchical entity typing. 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): System Demonstrations, pages 193–198, Hong Kong, China. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *ArXiv*, abs/2109.01247.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41(W1):W518–W522.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *CoRR*, abs/2102.09690.

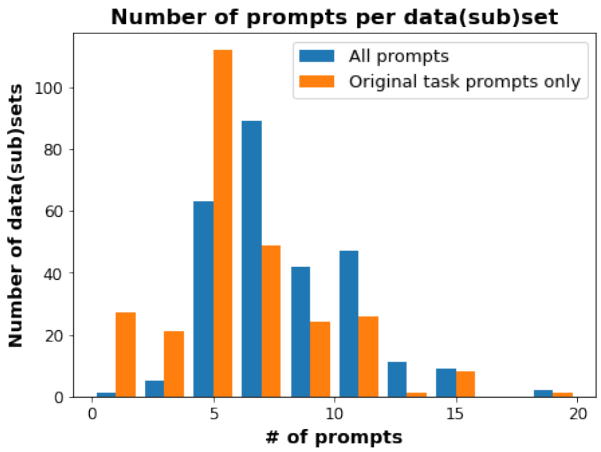


Figure 4: Most of the datasets have between 5 and 10 prompts.

A Data and Statistics

P3 is the largest public collection of English prompts and is actively growing. As of January 2022, it contains 2'052 English prompts for 170 English datasets (or 269 subsets, one dataset can contain multiple subsets with different prompts). There is an average of 7.6 prompts per data subset and an average 5.6 original-task prompts per data subset (see Figure 4).

P3 was developed as part of the BigScience project for open research[4]. There was a open hackathon to collect prompts for as many English NLP dataset (or English subsets of datasets) as possible. Almost 50 unique contributors affiliated with more than 25 institutions in 10 countries participated.

B Examples of prompt creating in the "Sourcing" mode

The "Sourcing" mode lets prompt authors write prompts, and iterate on them while observing the resulting prompted example. Figure 5 and 6 shows two examples of different complexities.

C Complete views

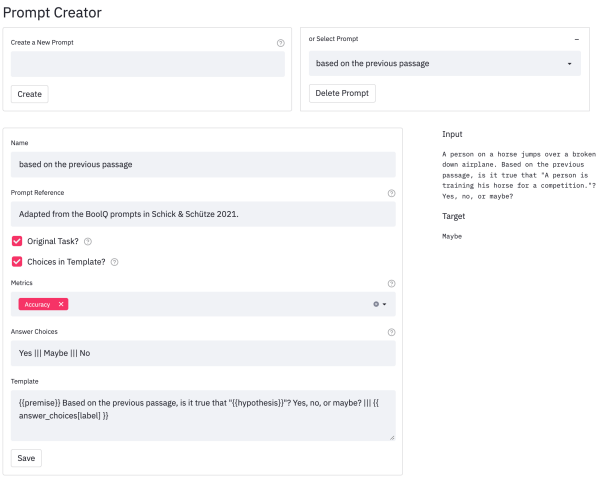


Figure 5: With the *Sourcing* mode, prompt authors can write new prompts, fill in the associate metadata, observe the result on examples and iterate.

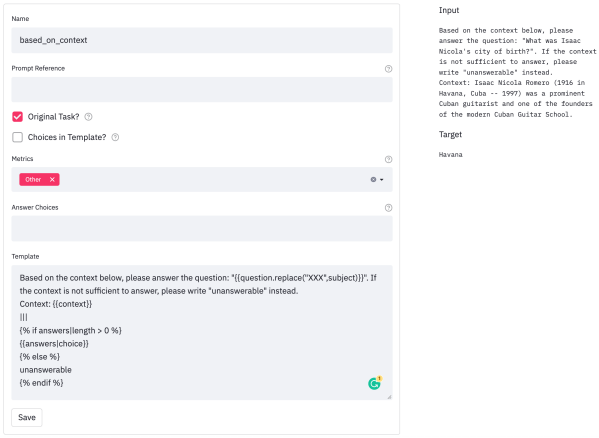


Figure 6: The templating language strikes a proper balance between expressivity and explicit structure. The following prompt for QA-ZRE (Levy et al., 2017), a dataset for zero-shot relation extraction, shows how to manipulate strings and do conditional statements with Jinja.

[4] https://bigscience.huggingface.co

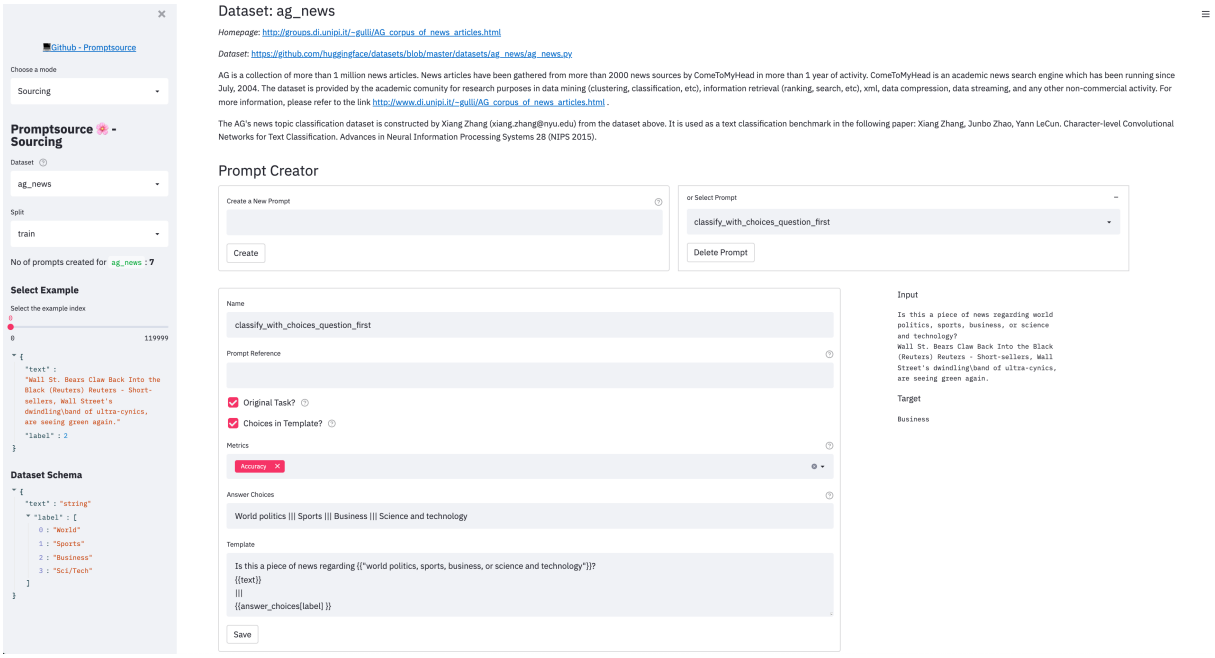


Figure 7: Complete view of the *Sourcing* mode.

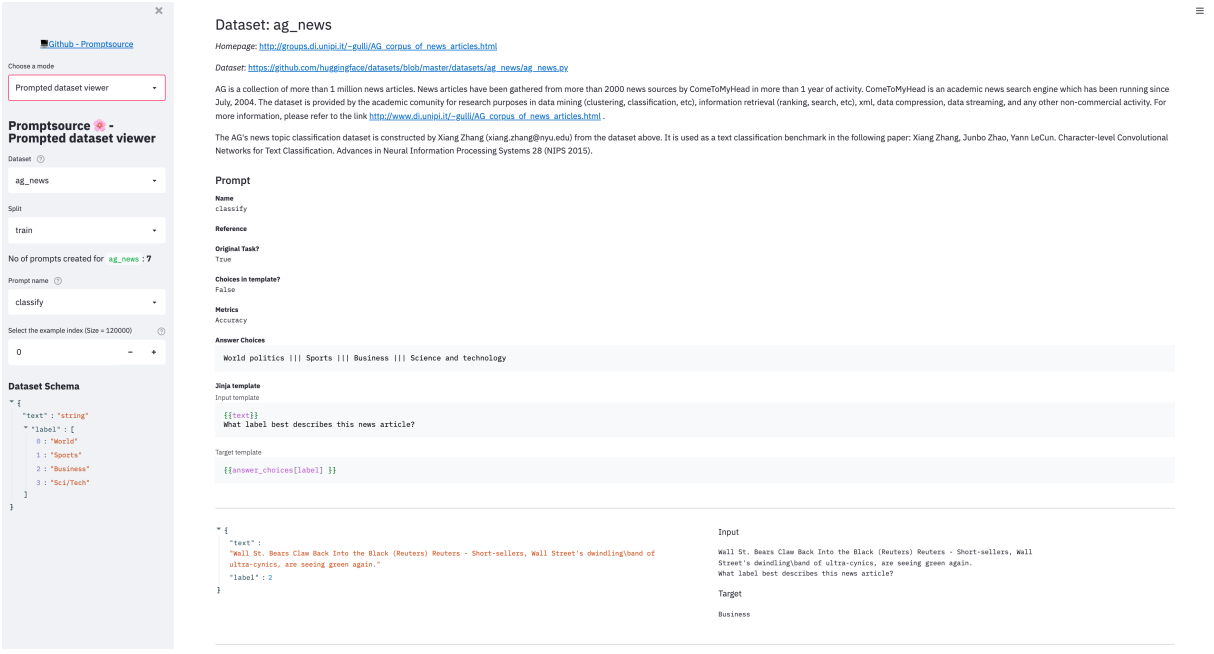
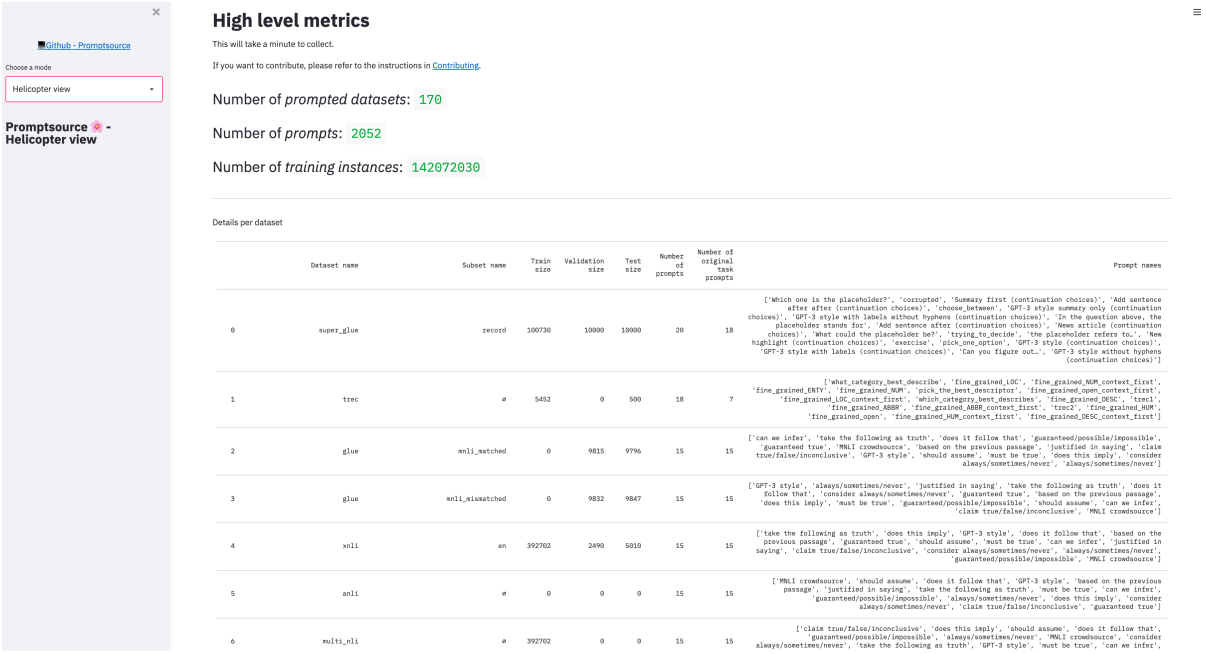


Figure 8: Complete view of the *Browse* mode.



	Dataset name	Subset name	Train size	Validation size	Test size	Number of prompts	Number of original task prompts	Prompt names
0	super_glue	record	100730	10000	10000	20	18	['Which one is the placeholder?', 'corrupted', 'Summary first (continuation choices)', 'Add sentence after after (continuation choices)', 'choose_between', 'GPT-3 style summary only (continuation choices)', 'GPT-3 style with labels without hyphens (continuation choices)', 'In the question above, the placeholder stands for', 'Add sentence after (continuation choices)', 'News article (continuation choices)', 'What could the placeholder be?', 'trying_to_decide', 'the placeholder refers to…', 'New highlight (continuation choices)', 'exercise', 'pick_one_option', 'GPT-3 style (continuation choices)', 'GPT-3 style with labels (continuation choices)', 'Can you figure out…', 'GPT-3 style without hyphens (continuation choices)']
1	trec	ø	5452	0	500	18	7	['what_category_best_describe', 'fine_grained_LOC', 'fine_grained_NUM_context_first', 'fine_grained_ENTY', 'fine_grained_NUM', 'pick_the_best_descriptor', 'fine_grained_open_context_first', 'fine_grained_LOC_context_first', 'which_category_best_describes', 'fine_grained_DESC', 'trec1', 'fine_grained_ABBR', 'fine_grained_ABBR_context_first', 'trec2', 'fine_grained_HUM', 'fine_grained_open', 'fine_grained_HUM_context_first', 'fine_grained_DESC_context_first']
2	glue	mnli_matched	0	9815	9796	15	15	['can we infer', 'take the following as truth', 'does it follow that', 'guaranteed/possible/impossible', 'guaranteed true', 'MNLI crowdsource', 'based on the previous passage', 'justified in saying', 'claim true/false/inconclusive', 'GPT-3 style', 'should assume', 'must be true', 'does this imply', 'consider always/sometimes/never', 'always/sometimes/never']
3	glue	mnli_mismatched	0	9832	9847	15	15	['GPT-3 style', 'always/sometimes/never', 'justified in saying', 'take the following as truth', 'does it follow that', 'consider always/sometimes/never', 'guaranteed true', 'based on the previous passage', 'does this imply', 'must be true', 'guaranteed/possible/impossible', 'should assume', 'can we infer', 'claim true/false/inconclusive', 'MNLI crowdsource']
4	xnli	en	392702	2490	5010	15	15	['take the following as truth', 'does this imply', 'GPT-3 style', 'does it follow that', 'based on the previous passage', 'guaranteed true', 'should assume', 'must be true', 'can we infer', 'justified in saying', 'claim true/false/inconclusive', 'consider always/sometimes/never', 'always/sometimes/never', 'guaranteed/possible/impossible', 'MNLI crowdsource']
5	anli	ø	0	0	0	15	15	['MNLI crowdsource', 'should assume', 'does it follow that', 'GPT-3 style', 'based on the previous passage', 'justified in saying', 'take the following as truth', 'must be true', 'can we infer', 'guaranteed/possible/impossible', 'always/sometimes/never', 'does this imply', 'consider always/sometimes/never', 'claim true/false/inconclusive', 'guaranteed true']
6	multi_nli	ø	392702	0	0	15	15	['claim true/false/inconclusive', 'does this imply', 'should assume', 'does it follow that', 'guaranteed/possible/impossible', 'always/sometimes/never', 'MNLI crowdsource', 'consider always/sometimes/never', 'take the following as truth', 'GPT-3 style', 'must be true', 'can we infer',

Figure 9: Complete view of the *Helicopter* mode.