🏴‍☠️ PyRater: A Python Toolkit for Annotation Analysis

Angelo Basile[1,2], **Marc Franco-Salvador**[2], **Paolo Rosso**[1,3]
[1]Universitat Politècnica de València
[2]Symanto Research
[3]ValgrAI - Valencian Graduate School and Research Network of Artificial Intelligence
{angelo.basile, marc.franco}@symanto.com, pross@dsic.upv.es

Abstract

We introduce PyRater, an open-source Python toolkit designed for analysing corpora annotations. When creating new annotated language resources, probabilistic models of annotation are the state-of-the-art solution for identifying the best annotators, retrieving the gold standard, and more generally separating annotation signal from noise. PyRater offers a unified interface for several such models and includes an API for the addition of new ones. Additionally, the toolkit has built-in functions to read datasets with multiple annotations and plot the analysis outcomes. In this work, we also demonstrate a novel application of PyRater to zero-shot classifiers, where it effectively selects the best-performing prompt. We make PyRater available to the research community.

1. Introduction

In recent years, the landscape of open-source NLP software has undergone substantial transformation. Custom codebases traditionally released alongside NLP research papers have partially been absorbed into more extensive libraries. To name a few examples, research on Transformer models has been streamlined by the popular Huggingface Transformers library (Wolf et al., 2020); tens of thousands of datasets can be sourced and loaded through the Datasets project (Lhoest et al., 2021); the AllenNLP (Gardner et al., 2018) and fairseq (Ott et al., 2019) projects have provided several foundational components for building custom neural models and sequence-to-sequence models respectively. It is reasonable to speculate that the ease of use of these frameworks has influenced and partially shaped the direction of NLP research. Despite these advancements in model building and data loading, there is still a lack of specialized libraries for annotation analysis. The most common practice for dealing with datasets with multiple annotations is to manage discrepancies using majority voting and to evaluate the quality of the annotations using inter-annotator agreement (IAA) metrics such as Cohen's Kappa (Artstein and Poesio, 2008). Recent studies (Hovy et al., 2013; Passonneau and Carpenter, 2014; Paun et al., 2018; Simpson and Gurevych, 2019) have shown that probabilistic models of annotations — popular in medical research (Dawid and Skene, 1979) — outperform both majority voting and traditional IAA metrics. An annotation merging strategy based on majority voting can fail if most of the annotators converge on the wrong label. Similarly, inter-annotator agreement metrics measure only consistency and not accuracy, implying that a pool of annotators can be consistently wrong; more importantly, agreement metrics cannot account for the different expertise of raters and assign equal value to both expert raters and noisy ones. Annotation models can avoid these pitfalls by jointly estimating directly the gold label, the difficulty of annotating each item and the skills of the raters. Despite these benefits, the use of probabilistic annotation models in NLP is still limited. We believe that one reason for this is the lack of user-friendly tools for the NLP community. This paper introduces a new Python library designed to handle repeated ratings using annotation models, aiming to make these models more accessible for NLP research and applications. More in detail, our contributions are as follows:

- We present PyRater, an easy-to-use and extensible Python library for annotation analysis with probabilistic models;
- We provide an implementation of popular models;
- We present a case study on a novel application of annotation models for unsupervised prompt optimization for zero-shot classifiers.

2. Annotation Analysis Models

Data annotation is crucial for supervised learning and model evaluation. Ensuring the reliability of annotations can be challenging due to the natural variance in human perspectives and to the objective difficulty of categorizing many linguistic structures (Plank et al., 2014b; Abercrombie et al., 2022). As multiple annotations for a single dataset are often collected to mitigate individual rater biases and provide a more robust label assignment, this introduces the need for a secondary layer of review, either through expert adjudication or automated methods. Few expert annotators can review smaller datasets, but for larger, crowdsourced projects, manual review becomes impractical. As larger annotated corpora are likely to contain errors (Abad and Moschitti, 2016; Gururangan et al., 2018), automated analysis methods become increasingly important.

In a typical data labelling scenario, we have a dataset of i items, a label space consisting of k classes, and a set of c coders. Each item i in the dataset is annotated with a label k by a coder c; a coder can annotate an instance 0 or more times. Annotation analysis aims to answer the following questions:

Q1 What is the most likely true class k for a given instance i?

Q2 Which instances are more difficult to annotate?

Q3 Among all coders c, who are the most reliable?

The state-of-the-art approach to answering these questions involves fitting a probabilistic generative model to the dataset. These probabilistic models are the core components of PyRater. The model parameters describe various aspects of the process, such as annotator reliability and item difficulty, based on a given annotation matrix. In this matrix, each element $a_{i,c}$ is a label k assigned to an item i by a coder c. Parameter estimation, conditioned on observed annotations, is performed through Bayesian inference. In PyRater, the full posterior distribution of the model parameters is estimated using the probabilistic programming language Stan (Carpenter et al., 2017).

3. Library Overview

```python
import pyrater as pyr

rte = pyr.load_dataset('rte')

# load Dawid&Skene model
model = pyr.PyraterModel.get('d&s')

# fit using Stan
model.fit(rte)

# RTE has 164 annotators ...
assert model.credibilities.shape==(164,)

# ... and 800 instances
# for which we estimate
# the gold label...
assert model.labels.shape == (800,)

# ... and the difficulty
assert model.difficulty.shape == (800,)
```

Listing 1: An overview of a PyRater workflow.

Design A high-level overview of an annotation analysis workflow with PyRater is provided in Listing 1. The focus of this work is twofold: *i)* to enable interoperability between diverse annotation models and *ii)* to improve the accessibility of these models. To achieve the first goal, PyRater defines a common API, as shown in Listing 2, and employs a registry system via Tango (Groeneveld et al., 2023). This allows users to easily list and register available models, as outlined in Listing 3. For the second goal, the library aligns loosely with the scikit-learn `Model` API (Pedregosa et al., 2011), providing familiar `model.fit()` and `model.predict()` methods. Additionally, PyRater includes built-in corpora with multiple annotations, such as RTE (Dagan et al., 2005; Snow et al., 2008), facilitating self-contained examples. For easier installation, PyRater can be installed via `pip`, and a Docker image is available to avoid portability issues.

Backend Choice In developing PyRater, we considered various backend options such as TensorFlow Probability (Abadi et al., 2015), NumPyro (Phan et al., 2019), and PyMC (Oriol et al., 2023) before finally choosing Stan (Carpenter et al., 2017). Several reasons informed this decision. First, Stan has a fast and reliable sampler, which is crucial for the computationally intense tasks of Bayesian inference. Second, compiled Stan models can be used with multiple programming languages. While Pyrater offers a Python interface to Stan through CmdStanPy, interfaces for R, MATLAB, Julia, and other languages are also available. Lastly, Stan's well-written documentation is helpful for new users who wish to create new models. It is worth noting that although GPU support in Stan is currently somewhat limited, it is rapidly maturing and is available through OpenCL, sidestepping the installation issues often associated with Nvidia CUDA[1].

Examples Several examples and tutorials for users are included in PyRater. These tutorials are available as easy-to-run Python notebooks. While the primary focus is on human annotation modeling, the examples also extend to other applications like dataset cartography (Swayamdipta et al., 2020) and prompt optimization (briefly presented in Section 4). To ensure the examples are self-contained, the library includes datasets with multiple annotations that are ready to be loaded.

Models PyRater includes implementations of several models that are ready to use and also serve as a reference for the development of new models. A registry system makes it possible to list all the available models:

```python
from pyrater import PyRaterModel as pm
```

[1]https://developer.nvidia.com/cuda-toolkit

```
class PyRaterModel:

  def fit(...):
    ...

  @property()
  def labels()
    """
    Return the predicted true labels.
    This can answer Q1: given multiple
    annotations for an instance, which
    is the true label?
    """
    ...

  @property()
  def difficulty()
    """
    Return the predicted difficulty
    of each item. This can answer Q2:
    which are the most difficult
    instances to label?
    """
    ...

  @property()
  def credibilities()
    """
    Return the credibility of each
    rater. This can answer Q3: how
    is the best rater?
    """
    ...
```

Listing 2: A partial view of the main PyRaterModel class. After fitting, the object exposes the `labels()`, `difficulty()` and `credibilities()` methods that can be used to address the most common problems in annotation analysis.

```
assert pm.list_available() == [
  'd&s',
  'hier_d&s',
  'mace',
  ...
  'kappa_majority_voting']

@pm.register('my-model')
class MyNewModel(pm):
    ...

assert 'my-model' in pm.list_available()
```

Listing 3: An example usage of the model registry.

Additional Features PyRater also provides a set of utilities such as plotting functions for visualizing the posterior distributions computed by the Stan model, data loading tools for easier data preparation and data loading, and a command-line interface:

```
$ pyrater [OPTIONS] -m MODEL <input file>
```

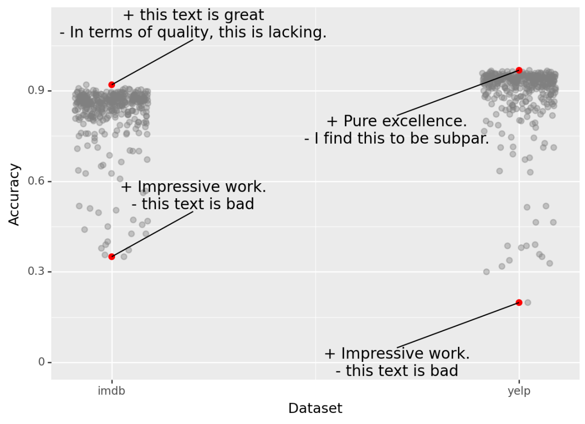

Figure 1: Accuracy score distributions for IMDB and Yelp using different prompts. Each point denotes zero-shot model accuracy. In red: the best and worst scores, each annotated with its corresponding prompt text. + and - denote the prompt for the positive and negative class, respectively. Jittering is added for enhanced readability.

4. Case Study: Unsupervised Prompt Optimization

In addition to analyzing human annotations, Pyrater can also be used for scenarios that involve repeated model predictions. This is particularly relevant for prompt selection in zero-shot models. In a genuine zero-shot environment, selecting the optimal prompt is challenging due to the lack of labeled development data. A solution to this issue involves collecting repeated predictions from a zero-shot model prompted in different ways. Pyrater can then analyze these multiple predictions and *i*) estimate the performance of different prompts and *ij*) predict the true label, effectively simulating an ensemble model.

In this section, we conduct a case study on prompt selection, using the Yelp (Zhang et al., 2015) and IMDb (Maas et al., 2011) binary sentiment analysis datasets and a dual-encoder architecture (`sentence-t5-large`, (Ni et al., 2022)) for zero-shot learning. First, we craft a diverse set of prompts per class, as shown in Table 1. Second, we assess how the model's performance varies depending on the selected prompts. Next, we evaluate how the inferred gold labels using PyRater contrast with those obtained through simple majority voting. Finally, we investigate the correlation between predicted prompt reliability and the model's true accuracy, evaluated using gold label data.

Positive	Negative
this is good	quite bad
text is great	this is terrible
very positive	very negative
…	…

Table 1: Sample Label Descriptions for binary sentiment analysis.

Dual-encoder zero-shot classifiers (Müller et al., 2022) work by calculating the similarity between the embedding of an input text and label descriptions, commonly referred to as prompts. The class corresponding to the label description that is closest to the input text is selected as the predicted class. For binary sentiment classification tasks, the user begins by crafting a label description for each class: for instance, *This text is positive* and *This text is negative* could serve as label descriptions for the positive and negative classes, respectively. Such zero-shot classifiers are sensitive to the phrasing of label descriptions: well-crafted but differently phrased descriptions can result in significant performance variations (Shin et al., 2020; Lu et al., 2022).

Figure 1 illustrates the distribution of scores for the same model when prompted with different label descriptions. As observed, the performance can vary widely, ranging from nearly random to approaching the state-of-the-art. When multiple plausible label descriptions are available and labeled data cannot be used to identify the best-performing description, the situation becomes analogous to human annotation analysis. However, in this case, the "annotations" are zero-shot model predictions rather than human ratings.

We evaluate PyRater on two tasks: label prediction, assessed by accuracy (**Q1**), and prompt ranking, evaluated by Pearson coefficients (**Q2**). As shown in Table 2, the model-based approach significantly outperforms both the Kappa agreement metric and majority voting. The correlation parameters for the ranking task — 0.83 for Yelp and 0.94 for IMDB — indicate that probabilistic models can accurately select the best prompt from a pool of available options, largely outperforming ranking using Kappa.

Dataset	Task	Method	Metric
Yelp	Ranking	Kappa	0.76
		PyRater	**0.83**
	Accuracy	Majority Voting	72.5
		PyRater	**74.9**
IMDb	Ranking	Kappa	0.87
		PyRater	**0.94**
	Accuracy	Majority Voting	66.5
		PyRater	**68.7**

Table 2: Performance comparison on Yelp and IMDb datasets. The ranking metric is Pearson correlation coefficient.

5. Related Work

Annotation Analysis While probabilistic, generative models of annotation have been extensively applied in fields such as psychometrics, epidemiology, and education, their adoption in NLP remains relatively limited. In contrast, NLP researchers frequently rely on agreement metrics like Kappa (Artstein and Poesio, 2008). Although suitable for small pools of expert raters, these metrics become less reliable with larger, noisy annotation sets (Passonneau and Carpenter, 2014; Paun et al., 2018). As crowdsourcing platforms like Mechanical Turk gained popularity for annotation tasks (Callison-Burch and Dredze, 2010), the use of annotation models in NLP has increased (Snow et al., 2008; Hovy et al., 2013) to maintain the quality of expert annotators using a broader set of noisy raters. Concurrently, separate research lines aim to avoid the annotation analysis and label adjudication steps by directly training models on multiple labels per instance, rather than a single, aggregated one (Plank et al., 2014a; Rodrigues and Pereira, 2018; Fornaciari et al., 2021). For a comprehensive overview of statistical annotation analysis, readers are advised to refer to Paun et al. (2022).

NLP Open Source Software Although various implementations of models for annotation analysis are available (Hovy et al., 2013; Pullin, Jeffrey and Vukcevic, Damjan and Saxhaug, Lars-Mølgaard, 2020; Simpson and Gurevych, 2019; Carpenter, 2013), a dedicated, comprehensive framework is still lacking[2]. This work synthesizes ideas from two lines of existing libraries. First, like Transformers and AllenNLP, it consolidates various implementations under a unified interface. Second, in terms of user experience, it draws inspiration from the simplicity and ease of use found in NLTK (Loper and Bird, 2002) and scikit-learn (Pedregosa et al., 2011).

Applications to NLP Models Basile et al. (2022) discuss the application of annotation models to zero-shot classifiers, and similar efforts have been made in the context of few-shot models (Zhao et al., 2022) and ensemble learning (Simpson et al., 2013).

[2]It is worth noting that the probabilistic programming library Numpyro (Phan et al., 2019) lists several annotation models in its example documentation.

6. Conclusion and Future Work

We introduced PyRater, an open-source library designed to simplify annotation analysis. We believe PyRater will facilitate broader adoption of state-of-the-art annotation models. Going forward, we welcome community feedback to inform the development of future features for PyRater.

7. Acknowledgements

We thank the anonymous reviewers for their helpful comments. The work of Paolo Rosso was in the framework of the FairTransNLP research project (PID2021-124361OB-C31), funded by MCIN/AEI/10.13039/501100011033 and by ERDF, EU A way of making Europe. The work from Symanto has been partially funded by the Pro^{2}Haters - Proactive Profiling of Hate Speech Spreaders (CDTi IDI-20210776), the XAI-DisInfodemics: eXplainable AI for disinformation and conspiracy detection during infodemics (MICIN PLEC2021-007681) and the ANDHI - ANomalous Diffusion of Harmful Information (CPP2021-008994) R&D grants.

Bibliographical References

Azad Abad and Alessandro Moschitti. 2016. Taking the best from the crowd:learning question passage classification from noisy data. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 136–141, Berlin, Germany. Association for Computational Linguistics.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Gavin Abercrombie, Valerio Basile, Sara Tonelli, Verena Rieser, and Alexandra Uma, editors. 2022. *Proceedings of the 1st Workshop on Perspectivist Approaches to NLP @LREC2022*. European Language Resources Association, Marseille, France.

Ron Artstein and Massimo Poesio. 2008. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Angelo Basile, Marc Franco-Salvador, and Paolo Rosso. 2022. Unsupervised ranking and aggregation of label descriptions for zero-shot classifiers. In *International Conference on Applications of Natural Language to Information Systems*, pages 119–126. Springer.

Chris Callison-Burch and Mark Dredze, editors. 2010. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Los Angeles.

Bob Carpenter. 2013. pyanno. python package for dawid and skene (1979) maximum likelihood estimator for categorical data coding models. `https://github.com/bob-carpenter/pyanno`.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28.

Tommaso Fornaciari, Alexandra Uma, Silviu Paun, Barbara Plank, Dirk Hovy, and Massimo Poesio. 2021. Beyond black & white: Leveraging annotator disagreement via soft-label multi-task learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2591–2597, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of*

Workshop for NLP Open Source Software (NLP-OSS), pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Dirk Groeneveld, Akshita Bhagia, and Pete Walsh. 2023. AI2 Tango. `https://github.com/allenai/tango`.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. 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.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. `https://www.nltk.org/`.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Thomas Müller, Guillermo Pérez-Torró, and Marc Franco-Salvador. 2022. Few-shot learning with Siamese networks and label tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8532–8545, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Abril-Pla Oriol, Andreani Virgile, Carroll Colin, Dong Larry, Fonnesbeck Christopher J., Kochurov Maxim, Kumar Ravin, Lao Jupeng, Luhmann Christian C., Martin Osvaldo A., Osthege Michael, Vieira Ricardo, Wiecki Thomas, and Zinkov Robert. 2023. Pymc: A modern and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9:e1516.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Silviu Paun, Ron Artstein, and Massimo Poesio. 2022. *Statistical methods for annotation analysis*. Springer Nature.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. 2019. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014a. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751, Gothenburg, Sweden. Association for Computational Linguistics.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014b. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 507–511, Baltimore, Maryland. Association for Computational Linguistics.

Pullin, Jeffrey and Vukcevic, Damjan and Saxhaug, Lars-Mølgaard. 2020. rater: Statistical models of repeated categorical rating data. `https://cran.r-project.org/package=rater`.

Filipe Rodrigues and Francisco Pereira. 2018. Deep learning from crowds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

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.

Edwin Simpson and Iryna Gurevych. 2019. A Bayesian approach for sequence tagging with crowds. 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)*, pages 1093–1104, Hong Kong, China. Association for Computational Linguistics.

Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. 2013. Dynamic bayesian combination of multiple imperfect classifiers. *Decision making and imperfection*, pages 1–35.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

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. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Mengjie Zhao, Fei Mi, Yasheng Wang, Minglei Li, Xin Jiang, Qun Liu, and Hinrich Schuetze. 2022. LMTurk: Few-shot learners as crowdsourcing workers in a language-model-as-a-service framework. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 675–692, Seattle, United States. Association for Computational Linguistics.