TagRuler: Interactive Tool for Span-Level Data Programming by Demonstration

Dongjin Choi*
Georgia Institute of Techonology, USA
jin.choi@gatech.edu

Sara Evensen
Megagon Labs, USA
sara@megagon.ai

Çağatay Demiralp†
Sigma Computing, USA
cagatay@sigmacomputing.com

Estevam Hruschka
Megagon Labs, USA
estevam@megagon.ai

ABSTRACT

Despite rapid developments in the field of machine learning research, collecting high quality labels for supervised learning remains a bottleneck for many applications. This difficulty is exacerbated by the fact that state-of-the art models for NLP tasks are becoming deeper and more complex, often increasing the amount of training data required even for fine-tuning. Weak supervision methods, including data programming, address this problem and reduce the cost of label collection by using noisy label sources for supervision. However until recently, data programming was only accessible to users who knew how to program. In order to bridge this gap, the Data Programming by Demonstration framework was proposed to facilitate the automatic creation of labeling functions based on a few examples labeled by a domain expert. This framework has proven successful for generating high accuracy labeling models for document classification. In this work, we extend the DPBD framework to span-level annotation tasks, arguably one of the most time consuming NLP labeling tasks. We built a novel tool, TagRuler, that makes it easy for annotators to build span-level labeling functions without programming and encourages them to explore trade-offs between different labeling models and active learning strategies. We empirically demonstrated that an annotator could achieve a higher F1 score using the proposed tool compared to manual labeling for different span-level annotation tasks.

CCS CONCEPTS

• **Information systems → Web applications**.

KEYWORDS

Data programming, natural language processing, span annotation

*Work done during internship at Megagon Labs.
†Work done when at Megagon Labs.

1 INTRODUCTION

In recent machine learning research, supervised learning has remained a dominant approach in situations where accuracy is critical. This is despite the increasing complexity of models, which in turn requires more training data, even when taking advantage of pre-trained models (as they frequently still require large refined datasets for supervised fine-tuning). In many cases, labels are manually annotated by humans and can require from tens to hundreds of thousands of examples [17]. For example, one of the state-of-the-art open domain named entity recognition models [14], used more than a million data samples. Thus, the need for big amounts of labeled data and subsequent human efforts are still a big bottleneck in the recent NLP field.

Weak Supervision and Data Programmming. One common approach to reduce the need for labeled data is to rely on weak supervision methods [10, 15]. Such methods suggest using noisy (thus, cheaper) label sources for supervision to reduce the labeling cost. Crowd-sourcing [12] and user-defined heuristics [8] are among frequently used label sources to assign noisy training labels to available unlabeled data. Another alternative, which is gaining a lot of attention in recent years is data programming (DP) [17]. In particular, DP focuses on the process of generating domain knowledge from domain experts in the form of labeling functions using programming language.

```
def lf(x):
    if "http" in x:       return SPAM
    else:                 return ABSTAIN
```

The above code snippet shows a simple example of a labeling function (written in Python) for spam classification task. A main limitation of data programming is that domain experts for many difficult natural language tasks (e.g. medical document annotation, legal document annotation, etc.) may not have experience in programming, making it difficult for them to create such labeling functions.

Data Programmming by Demonstration (DPBD). Aiming at bridging the gap between domain experts and labeling functions creation, Ruler [6] proposed the concept of Data Programming by Demonstration. DPBD is a human-in-the-loop model for synthesizing labeling functions through interactive user feedback, eliminating the need for domain experts to learn programming skills to leverage data programming. In this paper, we introduce TagRuler,
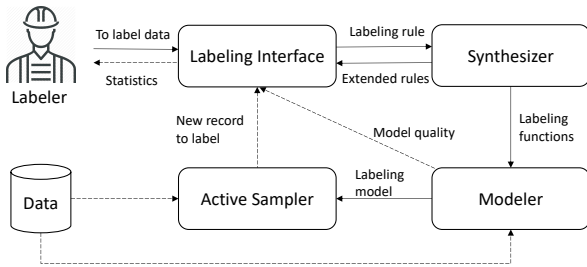
Figure 1: The Data Programming By Demonstration (DPBD) framework. Straight lines indicate the flow of knowledge, and dashed lines indicate the flow of data.

an open-source web application system that extends the DPBD framework to span-level annotation tasks. TagRuler includes a set of new features and characteristics that are not explored before in Ruler. More specifically, in TagRuler we adapted the DPBD main components to fit the span-level annotation problem and added new models. TagRuler implements four different models and allows users to explore the trade-offs afforded by each.

We designed TagRuler to optimize usability by carefully selecting atomic conditions for span labeling rules. Based on results previously described in the literature (e.g. BabbleLabble [9]), lexical, semantic, syntactic, and entity-type match rules were incorporated into TagRuler, in addition to the possibility of using the negation operator. We implement semantic rules via similarity based on contextual token embeddings (BERT [5] and ELMo [16]). In addition to the active learning approach previously implemented in Ruler, TagRuler proposes a novel active learning component focusing on false positives, thus contributing to one of the main challenges in data programming: surfacing difficult (or borderline) labeled examples [20]. In this active learning approach, unlabeled examples that have higher potential to identify false positives will have higher probability to be sampled as next instance to be labeled. It aims to achieve maximum accuracy with a small number of interactions combined with uncertainty-based sampling.

Contributions. This work presents the following contributions:

- proposes a token labeling model which incorporates logical combinations of expressive atomic rules.
- implements state-of-the-art sequential label aggregation models.
- adds an active learning component based on false positives.
- demonstrates the effectiveness of TagRuler through evaluation on use-cases from two domains.
- makes the underlying code[1], example data, and video demo[2] publicly available in order to make span-level data programming for text more accessible to researchers and annotators.

2 SYSTEM OVERVIEW

Figure 1 shows the architecture of DPBD framework that was proposed by [6]. In this section, we describe how TagRuler extends each component of the DPBD framework for span-level annotation, or tagging task. We also describe how each component is connected to the TagRuler's user interface shown in Figure 2.

[1]Code available at https://github.com/megagonlabs/tagruler.
[2]A video demo can be found at https://youtu.be/MRc2elPaZKs.
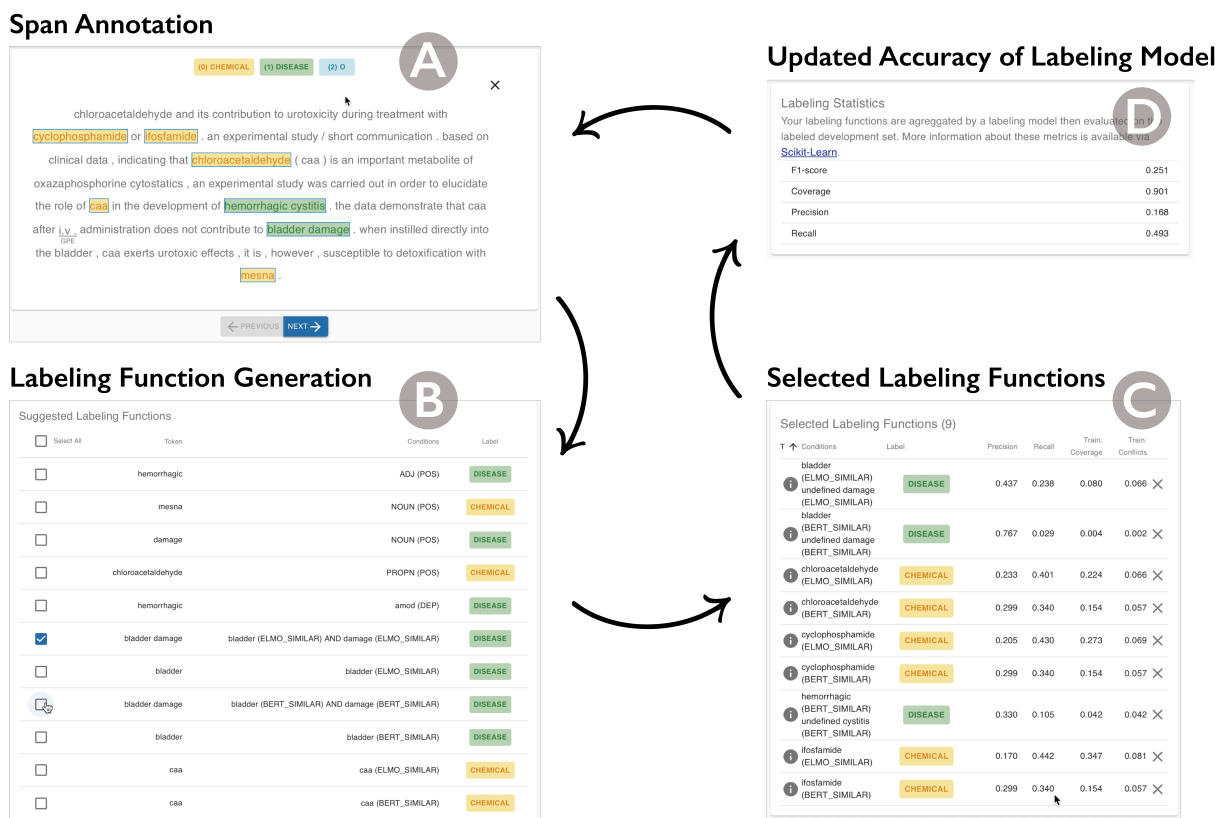
2.1 Labeling Interface

TagRuler provides a web-based interactive system that facilitates users to encode their domain knowledge for span annotation into labeling rules. To this end, TagRuler provides the main text view (Figure 2 (A)) where user interacts with a text data sample. The main text view shows a single text data and supports labeling operations. The user annotates spans by highlighting them with the cursor. Next, one assigns a label to the selected spans by choosing one from the options displayed. The user also can indicate if the selected span is a positive or negative sample by clicking the (+) or (-) signs that pop up as a span is selected. Once labels are assigned to selected spans, the *synthesizer* compiles the selections and generates a set of relevant labeling functions.

2.2 Synthesizer

The synthesizer translates the annotations into atomic rules and generates a candidate set of labeling functions (in programming language) using the rules. TagRuler implements atomic rules which are specifically for span annotation tasks. Following previous works on rule-based labeling models [9, 19], we focus on capturing semantic, syntactic, and entity-type information of the spans. The created labeling functions are shown in the labeling function view (Figure 2 (B)), and the user selects relevant labeling functions and delivers them to the model. The selected labeling functions are shown in Figure 2 (C) with corresponding statistics (accuracy and coverage) of each labeling function. Lastly they are aggregated by the *modeler* and used to produce accuracy feedback in an iterative process.

2.2.1 Contextual similarity rules with neural embeddings. For semantic similarity, we implement two dense vector representation (embeddings) rules, `SIMILAR_BERT()` and `SIMILAR_ELMO()` rules, which are based on BERT [5] and ELMo [16], respectively. Both return True if the embeddings of the two input tokens have higher cosine similarity then pre-defined threshold values. We use the pre-trained models since they are able to capture contextual semantics considering the structure of sentences implicitly. During demonstration, we observed that `SIMILAR_BERT()` and `SIMILAR_ELMO()` work differently and are complementary. As BERT has been the state-of-the-art contextual language model, it has more strength in capturing semantic and syntactic information implicitly at the same time. ELMo outputs more stable results for out-of-vocabulary words since it is a character-based model. This functionality is especially important when a user works on highly domain-specific data sets since they tend to include rare vocabularies that were under-trained in the BERT model. We observed this effect when testing TagRuler on a medical dataset.

2.2.2 POS, dependency parse, and NER tag rules. Syntactic information of word is also important to build expressive rules. Two representative ways of extracting the syntactic information are Part-of-Speech (POS) tagging and dependency parsing, which capture shallow and deeper syntactic structure, respectively. Specifically, we implement `POS()` and `DEP()` rules. Both return True if two input tokens have same POS and DEP tags, respectively. Similarly, TagRuler supports usage of `NER()` rule which returns True if inputs have a same named entity recognition (NER) tag (in our case, 18 classes provided by spaCy [11]) To detect POS, dependency,

Span Annotation

Labeling Function Generation

Updated Accuracy of Labeling Model

Selected Labeling Functions

Figure 2: Overview of TagRuler. (A) Main text view lets users annotate text spans with relevant class names. (B) Labeling function view shows a list of labeling functions suggested by TagRuler. The user is expected to select relevant labeling functions that align with their domain knowledge. (C) Selected labeling function view shows statistics about the user-selected functions. (D) Label model view shows statistics of the label model trained on currently selected labeling functions.

and NER tags, we used the spaCy Python library [11] which have achieved near-human performance for each task.

2.3 Modeler

The modeler creates weak supervision labels and associated probabilities by using the output of labeling functions selected by the user. That is, given a set of labels $\mathcal{L}$, and a label matrix $M \in \{1, 2, \ldots, |\mathcal{L}|\}^{n \times l}$ generated by l labeling functions applied to n tokens, the modeler fits an aggregation model and produce a matrix $P \in \mathbb{R}^{n \times |\mathcal{L}|}$ where $P_{i,j}$ denotes the probability $P(label(token_i) == j|token_i)$. The resulting probabilistic labels are applied to a small labeled development set, and their accuracy is shown to the user in the label model view (Figure 2(D)) in real time. TagRuler implements four different models: Snorkel [17], HMM [19], FlyingSquid [7], and Majority Voter to aggregate and generate probabilistic labels. Snorkel, HMM, and FlyingSquid are probabilistic graphical models that capture dependency structure between tokens and labeling functions. In particular, HMM and FlyingSquid support sequential dependence structure, while others do not. It is important to use the appropriate method as modeler, since there is a tradeoff between the time it takes to fit the model (which hampers interactivity) and

the accuracy. At the beginning of a session, users can select the model most appropriate for their use-case.

2.4 Active Sampler

TagRuler is a system designed to learn from expert knowledge, and expert manual annotation is expensive, so it is important to obtain informative data with as few annotations as possible. The active learning approach focuses on contributing to this main challenge in data programming: generating difficult (or borderline) examples [20]. TagRuler samples the text to be displayed in Figure 2 (A) after each annotation using an active learning technique that leverages the trained label model and a small labeled development set.

TagRuler's active learning approach is motivated by the fact that there is a small set of ground truth in the DPBD setting. In this approach, unlabeled examples that have higher potential in identifying and reducing false positives will have higher probability to be sampled as next instance to be labeled. First, the active sampler selects the data record x^* from the development set, which shows the greatest error in predicting the true label, i.e. $x^* = \arg\min_x p(label_{true}(x)|x)$. Then, the sampler samples the most similar text from train set using cosine similarity between

sentence embedding vectors. For sentence embeddings, we use the Sentence-bert [18] model. This model is based on the BERT model so that the embeddings can be comprehensive representation of syntax and semantics of texts. TagRuler implements an active sampler that alternately samples based on the proposed method and the uncertainty-based sampler used by Ruler [6] so that it can advantage of both methods in turn, reducing variance as *ensemble models* do.
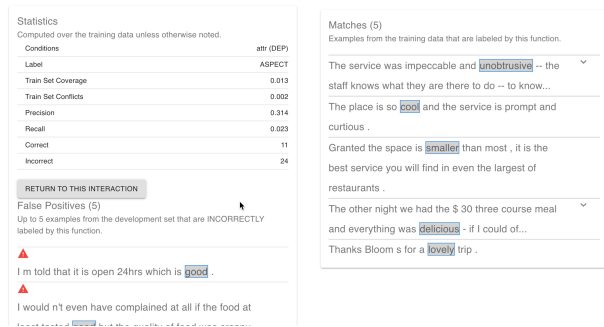
Figure 3: Individual selected labeling function view with false positives and statistics. A user can identify falsely labeled ground truth examples and potentially labeled train examples.

2.5 False Postive Feedback

TagRuler helps users correct their *mistakes* and develop improved labeling functions. When a user clicks the ⓘ button located to the left of each labeling function in the selected labeling function view (Figure 2(C)), false positive feedback and detailed statistics of the corresponding labeling function are displayed as shown in Figure 3. During our demonstration session, the false positive examples shown by the system were crucial and helped achieving better F1 score and also helped users to identify *wrong ground-truth* instances (meaning that the labeling guideline had changed since the ground truth was labeled, or that the labelers used for the ground truth were not reliable).

3 USE CASES AND OBSERVATIONS

In order to demonstrate the efficacy of TagRuler, we demonstrate our findings from <30 minutes trial labeling sessions on two domains. We recorded the accuracy obtained in these sessions by time and found that TagRuler was superior when compared to a simulated manual labeling session for the same time period.

3.1 Data and Tasks

We used two real world datasets: BC5CDR [23], and Yelp Restaurants [1]. Each dataset was labeled for 30 minutes using TagRuler. Table 1 shows more information about the data. BC5CDR is a medical dataset [23] where each token is annotated as either a chemical, disease, or neither. The medical domain is a prime example of a field where training data is expensive or difficult to obtain, as the labeler requires a high level of expertise and their time is very valuable and limited. This makes it a good use-case for weak-supervision approaches if we can make them accessible. Yelp Restaurants is a set of restaurant reviews [1] annotated with aspects (ex: food, service)

Table 1: Summary of datasets used for our trial sessions.

Dataset	# Doc	Avg. Tokens/Doc	Class Frequency
BC5CDR	866	381.6	Chemical: 0.063 Disease: 0.079 Other: 0.858
Yelp Restaurants	838	36.5	Aspect: 0.100 Opinion: 0.114 Other: 0.786

and opinions about those aspects. Once TagRuler is used to build relevant labeling functions, we can apply the labeling functions to any extension of the datasets and automatically generate training labels.

The labeling tasks of TagRuler were carried out by one of the authors who is not a subject matter expert for these tasks. We hope that with a little background information about data programming, domain experts would achieve similar results (or better, due to their increased knowledge of the domains). As a baseline to compare the results of TagRuler's weak labeling and manual labeling, we sampled text examples randomly from the labeled ground truth, at a rate equivalent to the annotator speed for the task. We believe that this is a generous estimate, since with TagRuler the user only has to label a few representative tokens, and not every instance in the document. TagRuler's majority voter model's output and the randomly sampled examples were then used to train CRF models which we evaluate based on their test accuracy scores (micro F1 scores).
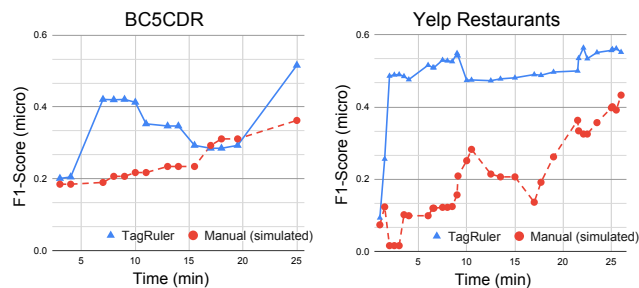
Figure 4: Comparison of TagRuler and simulated manual labeling. Line charts show change in accuracy (micro F1 score) per each text example over time.

3.2 Observations

From less than 30 minutes of labeling BC5CDR dataset, we were able to achieve an F1 score 0.516 from a CRF model trained on TagRuler's generated labels. With the same time period, the simulated manual labeling ended up an F1 score of 0.363 (Figure 4 left). On Yelp Restaurants dataset, we achieved an F1 score of 0.564 with a CRF model trained on TagRuler's output in 26 minutes while the simulated manual labeling reached 0.433 in the same time period (Figure 4 right). The active sampling helped discover some inconsistencies in the ground truth labels. With TagRuler, we don't need to have high confidence in the ground truth labels– the most reliable ground truth is the domain expert, and TagRuler can leverage this to surface edge cases or changing guidelines.

4 RELATED WORK

Both TagRuler and Ruler share concepts and ideas from rule-based information extraction [4] and exploit those ideas to make it easier for domain expert to leverage Machine Learning models and data. This is in contrast to systems like PropMiner [2], which focus on generating highly reliable rules for information extraction, as opposed to data programming which relies on a mix of high and low precision rules that make independent errors (like those generated by TagRuler and Ruler).

The idea of systems and agents that learn by demonstration or by natural language explanation have also been explored in other domains, such as in PUMICE, an agent proposed by [13] that uses the combination of natural language programming and programming-by-demonstration to allow users to describe tasks in natural language at a high level, and then starts a collaborative process with the user to recursively resolve ambiguities or vagueness through conversations and demonstrations. Or in the classical artificial intelligence problem of automated synthesis of programs that satisfy a given specification [22].

Natural Language Programming has also been applied to Data Programming as in BabbleLabble [9]. BabbleLabble allows the labelers to use natural language explanations during the labeling process to generate labeling functions. A semantic parser is then used to convert the natural language explanations into logical forms, which represent labeling functions. The labeling functions are then aggregated using the traditional Data Programming approach proposed in Snorkel [17]. We hope TagRuler can complement such an approach by giving users a different, more expressive set of atomic rules.

The difficulty in having Domain Experts capable of generating labeling functions that can go beyond easy and redundant examples is also tackled in SNUBA [21] and in the framework described in [3]. In SNUBA, a semi-supervised data programming setting is proposed. But instead of counting on user interactions, SNUBA proposes an automatic generation of labeling functions by using a small set of labeled data, and without user interaction. In [3], queries to be annotated are not data points but labeling functions, and the approach is based on training supervised ML models (instead of Data Programming, as done in TagRuler) with weak supervision through an interactive process.

Finally, span-level annotation has also been explored in the context of Data Programming in the approach proposed in FlyingSquid [7], but as happens in Snorkel, FlyingSquid also requires that the user have programming literacy.

5 CONCLUSIONS

TagRuler is an open-sourced web-based system that enables users to create span-level annotation functions through intuitive interactions. It requires no programming literacy, yet reliably expands user annotations to expressive rules. TagRuler's labeling model builds on, and extends, state-of-the-art Data Programming methods, improving user experience and making the techniques more accessible. Through experiments on two datasets, we demonstrated that an annotator using TagRuler could achieve a higher F1 score compared to manual labeling, allowing faster development of tagging models. By open-sourcing this technology, we hope to improve the accessibility of deep learning models for NLP.

REFERENCES

[1] 2014. Yelp Dataset. http://www.yelp.com/dataset_challenge.
[2] Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 157–162.
[3] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. 2020. Interactive Weak Supervision: Learning Useful Heuristics for Data Labeling. *arXiv preprint arXiv:2012.06046* (2020).
[4] Laura Chiticariu, Yunyao Li, and Frederick Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems!. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 827–832.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[6] Sara Evensen, Chang Ge, and Cagatay Demiralp. 2020. Ruler: Data Programming by Demonstration for Document Labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 1996–2005.
[7] Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*. PMLR, 3280–3291.
[8] Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. 98–108.
[9] Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, Vol. 2018. NIH Public Access, 1884.
[10] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 541–550.
[11] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.
[12] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. NeurIPS.
[13] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 577–589.
[14] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019. Dice loss for data-imbalanced NLP tasks. *arXiv preprint arXiv:1911.02855* (2019).
[15] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 1003–1011.
[16] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).
[17] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.
[18] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[19] Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5570–5578.
[20] Sahaana Suri, Raghuveer Chanda, Neslihan Bulut, Pradyumna Narayana, Yemao Zeng, Peter Bailis, Sugato Basu, Girija Narlikar, Christopher Ré, and Abishek Sethi. 2020. Leveraging organizational resources to adapt models to new data modalities. *arXiv preprint arXiv:2008.09983* (2020).
[21] Paroma Varma and Christopher Ré. 2018. Snuba: automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 12. NIH Public Access, 223.
[22] RJ Waldinger and RCT PROW LEE. [n.d.]. A step toward automatic program writing. In *Proc. Int. Joint Conf. on Artificial Intelligence (Washington DC, May 1969)*. 241–252.
[23] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2015. Overview of the BioCreative V chemical disease relation (CDR) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, Vol. 14.