Weak Supervision in the Age of Transfer Learning for Natural Language Processing

Fei Fang[1] **& Zihan Xie**[2]
[1]ffang19@stanford.edu, [2]xiezihan@stanford.edu

1 Introduction

In deep learning-based natural language processing (NLP), obtaining high-quality training data is one of the most important and challenging tasks. Traditionally, due to the heterogeneity and complexity of natural language, in order to train a performant DL model for an NLP task, a significantly large labeled dataset must first be collected. However, human-labeled training data is usually expensive to obtain due to the large amount of manual effort, the human expertise required for labelling (e.g. medical records), and the importance of data privacy (e.g. financial records).

Such demand for scalable training data motivated recent research, leading to the development of Snorkel, a system for programmatically building training sets under weak supervision, and the study of the tradeoffs between quantity of data and quality of labels [11]. Meanwhile, the advent of large pretrained natural language processing (NLP) models, such as BERT, GPT-2, and XLNet, has made transfer learning a powerful technique for NLP and drastically reduced the amount of training data necessary for individual downstream tasks.

In light of these two breakthroughs, we performed an ablation study to analyze the effects of applying weak supervision to data generation and of transfer learning. In this project, we trained DL models, both with and without transfer learning (BERT and LSTM, respectively), for a binary sentiment classification task on data with programmatically generated weak labels and on hand-labeled data. The input to the DL models is a text review, and the models would output a predicted sentiment, which is either "positive" or "negative." Our goal was to investigate the following questions:

- When applying transfer learning to text classification, what are the effects of using data generated with weak supervision instead of hand-curated data to finetune a pre-trained language model (e.g. BERT)?
- Without transfer learning (e.g. training an LSTM from scratch), does weak supervision have similar effects?

Without transfer learning, weak supervision helped achieve a 10-point increase in AUC for LSTM. With transfer learning, although weak supervision did not demonstrate a strong impact, the performance of BERT trained on 800 hand-labeled examples surpassed the performance of LSTM on 25k hand-labeled examples. In both settings, we observed that the DL models were capable of denoising the weakly-labeled dataset and generalizing to the test set.

2 Related Work

2.1 Weak supervision

Weak supervision is the approach of using noisy or imprecise sources as signals to label large amounts of training data, which are then fed into a supervised learning algorithm. Various techniques have been proposed to extract such imperfect signals programmatically from the training data.

Rule-based heuristics The core of rule-based heuristics involves leveraging existing manually-maintained knowledge of the task at hand, e.g. by accessing databases. For instance, for the task of relation extraction from text, maintaining a knowledge base of known relations between keywords could provide a first set of input features for further supervised learning [8, 12].

Generative models Generative models can be used to learn and model the process of assigning labels to the training data. In the example of relation extraction, instead of directly using a knowledge base of known relations, one can alternatively use a generative model to directly predict which textual *patterns* are more likely to appear in a text given the relation it expresses [14].

Snorkel: Data programming with weak supervision Snorkel is a flexible weak supervision system that allows the user to leverage multiple techniques at once, including the two mentioned above. With Snorkel, the user first develops labeling functions, which are heuristics that leverage prior knowledge. The labeling functions are then fed into a generative model that automatically assigns weights to each labeling function [11]. The effects and efficiency of using Snorkel have been demonstrated, with applications to domains ranging from medicine to autonomous driving [3, 15].

2.2 Transfer learning

Transfer learning is the technique of reusing a model pre-trained for a different but related task as the starting point for the task at hand. In NLP, the most widely adopted pretrained model today is BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is a generic natural language understanding model pretrained on English Wikipedia and BookCorpus [16], which can then be fine-tuned for downstream natural language tasks. At the time of BERT's debut, BERT obtained state-of-the-art results on eleven NLP tasks, including a 7.7-point absolute improvement on the GLUE benchmark. Since then, many variants of BERT have been presented, such as RoBERTa [6] and DistilBERT [13].

3 Dataset

We used the IMDB movie review dataset [7]. The original dataset contains a training set of 25k examples and a test set of 25k examples, each of which contains 12.5k positive examples and 12.5k negative examples.

To simulate different levels of data scarcity, we randomly sampled scarce datasets (SD) with size decreasing logarithmically from the original train set (abbreviated OG), each with balanced classes. The sub-dataset sizes are 12800, 6400, 3200, 1600, 800, respectively, with the assumption that a dataset of size 800 is small enough that an individual machine learning practitioner can viably obtain. Any SD dataset is a strict subset of arbitrary larger SD dataset.

We broke the original test set into a DEV set of size 800, and a TEST set of all remaining 24.2k samples. The DEV set is used for hyperparameter tuning, model selection, overfitting prevention, etc, while the TEST set is used once for each model as the final evaluation. The DEV set and the TEST set are shared across all models and experiment runs, to control for the random factors during hyperparameter tuning and model selection, and to ensure fair performance evaluation.

4 Methodology: Weak Label Generation

We generated weakly-supervised labels under the assumption that we have access to a significant number of training examples, but limited resources to manually produce training labels. Specifically, for weak label generation, we simulated the scenario where we had full access to the 25k review texts in the original training set but only 800 manual labels, all taken from the gold labels of SD-800.

We used the following two approaches to generate weak labels using heuristic functions. The gold labels in SD-800 are used to tune the heuristic functions.

Baseline: Naive keyword labeler Our baseline approach is to use a single labeling function that detects lists of keywords. The naive labeler generated WD-N (Weak Dataset - Naive) based on the presence of positive (e.g. "wonderful" and "fascinating") and negative (e.g. "horrible" and "waste of

time") keywords in a piece of text, where the sets of keywords POS, NEG and the weights w_p, w_n are manually maintained and tuned using DEV set. For each text review r, we compute:

$$score(r) = |\{x : x \in POS \cap r\}| \times w_p + |\{x : x \in NEG \cap r\}| \times w_n$$

If $score(r) > 0$, 'pos' is assigned. If $score(r) < 0$, 'neg' is assigned. If $score(r) = 0$, the naive labeler abstains from assigning a label to r.

Snorkel labeler With Snorkel, we started with writing a set of labeling functions. Snorkel used a generative model to estimate the weights to assign to each labeling function output, with the intuition that the more a labeling functions agrees with the majority on the training set overall, the higher its weight will be. Eventually for each piece of review text, Snorkel emitted the weak label as a linear combination of the outputs of all labeling functions.

Note that in addition to labeling functions, Snorkel includes two other programming abstractions to help with dataset creation via other techniques, such as data augmentation. For the scope of this project, we are only concerned with the use of labeling functions.

The labeling functions we developed include:

1. Textblob, a rule-based sentiment classifier. Note that since Textblob is rule-based, it has no prior knowledge of our dataset distribution.
2. Keyword detection with multinomial event model. We used Textblob's predictions on the training set as weak labels, and fit a multinomial Naive Bayes on the train set with the rule-based labels. However, instead of using Naive Bayes to make predictions, we made use of the model parameters. Recall that when fitting a Naive Bayes, for each high-frequency word w in the corpus, we estimate the probability that w appears in a review given the review score:

   $$T(w) = p(w \text{ occurs in review}|\text{review is positive}).$$

   We took the 50 words with the highest $T(\cdot)$ values to form the list of positive keywords that are most indicative of positive sentiments, and the 50 words with the lowest $T(\cdot)$ values to form the list of negative keywords. Then we assign a label for each example using the naive keyword labeler with these two lists of keywords.
3. Key phrase detection using regular expressions. One example is "10 out of 10" or "10/10".

After the generative model assigns a weight to each labeling function, the Snorkel labeler is now able to output a probabilistic label for each example that it does not abstain from labeling. The probabilistic label represents the probability that the review is positive. We used AUC score as our dev metric to filter out the training samples whose probabilistic labels are close to 0.5. By tuning on the dev set, the training samples whose probabilistic labels fall in the interval [0.4, 0.6] are filtered out in order to maintain an appropriate balance between coverage and AUC. The resulting dataset thus forms WD-S (weak dataset - Snorkel). Note that the probabilistic labels would then become the input labels to the DL models.

Here are the statistics for the weak labels we generated compared to the gold labels in the original training set.

Dataset	Coverage	ROC-AUC	Precision	Recall
WD-N	82.6%	0.75	0.74	0.74
WD-S	94.3%	0.88	0.82*	0.82*

Table 1: Statistics For Weak Supervision Datasets. Metrics are calculated using the gold labels in OG.
* *Calculated by casting probabilistic labels to binary labels using a threshold = 0.5.*

5 Experiments & Results

To compare the effect of weak supervision on transfer learning and non-transfer learning, we trained two parallel sets of models on each of the OG, SD, and WD datasets. The transfer learning model was a fine-tuning classification layer on top of a pre-trained Uncased Bert-Base model, while the

non-transfer learning model was a LSTM model [4] with random weight initialization. Each training sample was tokenized using the same vocabulary and tokenizer, and padded to a fixed-length sequence. The models trained in mini-batches, and early stopped when evaluation metric on the DEV set did not improve (based on pearson correlation) for several consecutive epochs to prevent overfitting.

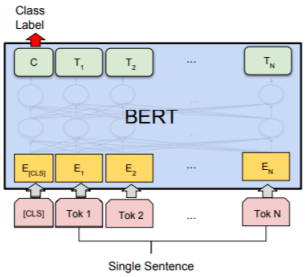

Figure 1: Fine-tuning Bert Architecture

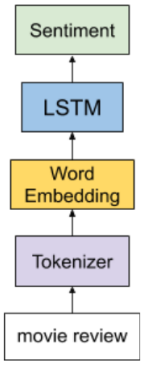

Figure 2: Sequential LSTM Architecture

Although conventionally binary classification task uses the empirical loss for logistic regression as the loss function, one of our datasets contains probabilistic labels, which renders the logistic regression loss function non-applicable. We therefore modeled our training as a regression task and used mean squared error as our loss function, where the output was a probabilistic prediction and casted to a binary prediction when we computed accuracy.

In Table 5 and Figures 3, 4, and 5, we present the results obtained from evaluating the trained models on the TEST set.

Train set	OG (25K)	SD12800	SD6400	SD3200	SD1600	SD800	WD-S	WD-N
Test AUC w/ BERT	0.974	0.971	0.966	0.962	0.953	0.946	0.932	0.807
Test AUC w/ LSTM	0.939	0.911	0.886	0.867	0.818	0.800	0.903	0.782

Table 2: Evaluation results of BERT and LSTM trained on each sampled dataset.

6 Discussion

We first compare the AUC score of the Snorkel labels compared with gold labels in the **train** set with the **test** AUC score of the models trained on WD-S, the Snorkel-generated dataset. The AUC score of the Snorkel labels compared with the golden training labels is 0.88. When trained on WD-S, LSTM achieved a test AUC score of 0.90, and BERT achieved a test AUC-score of 0.93, which are both higher than 0.88. This indicates that by feeding weakly-supervised probabilistic labels into deep learning models, the deep learning models were able to denoise the data (though to varying extents depending on model architecture) and generalize.

In the scenario where no transfer learning was involved, we observe that with weak supervision, the LSTM model was able to achieve an AUC score on par with that obtained with 12800 manually labeled examples. Recall that WD-S was generated with a dev set of size 800, with the assumption that the developer would only collect 800 manual labels. This means that the model achieved similar performance while decreasing the labeling efforts by a factor of 16. In addition, compared with the performance of LSTM trained directly on 800 manual labels (i.e. on the dataset SD-800), the weak supervision dataset generated by Snorkel achieved a 10 point increase in AUC score.

With transfer learning, the Bert model trained on the SD-800 dataset was able to accomplish a higher AUC score than that of a LSTM model trained on OG, the full training set. The SD-800 Bert model was able to achieve an astounding AUC score of 0.95, only 2.8 points lower than BERT trained on OG. Meanwhile, BERT trained on WD-S achieved an AUC score of 0.93.

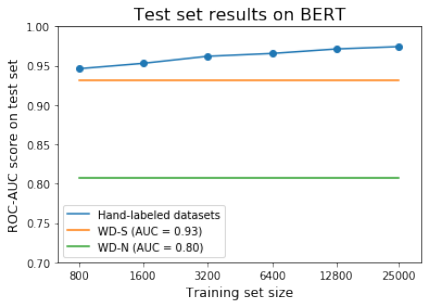

Figure 3: Evaluation results of BERT trained on each sampled dataset.

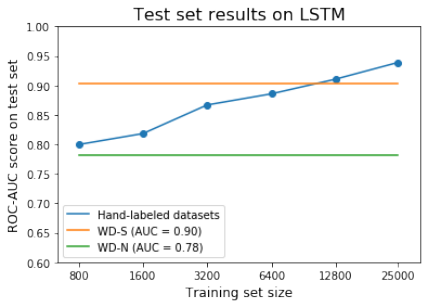

Figure 4: Evaluation results of LSTM trained on each sampled dataset.

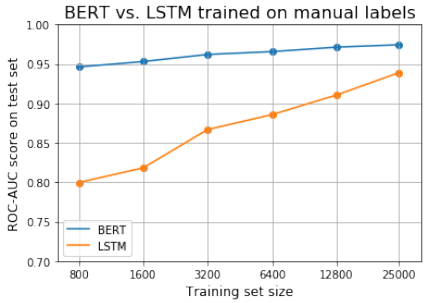

Figure 5: Comparison of Performance on BERT vs LSTM, trained on each hand-labeled dataset.

We hypothesize that BERT trained on Snorkel-generated labels did not provide a boost to performance likely due to the effectiveness of BERT's pre-training. As mentioned in Related section, BERT was pretrained on BookCorpus and English Wikipedia, which likely has very similar vocabulary distribution to that of the IMDB review dataset. The "transferring" of BERT's knowledge to this sentiment classification task is therefore highly relevant, causing Bert to achieve such a strong performance with a mere 800 fine-tuning training size. Given the small gap between SD-800 BERT and OG BERT to begin with, it was hard for a weakly labeled dataset to demonstrate a statistically significant impact. Nevertheless, if weak supervision had been applied to a task where the input vocabulary space has a drastically different distribution from BERT's pretrained data, e.g. classification of medical records, then it would take more "effort" (i.e., more manual labels) for BERT to transfer its pretrained knowledge to fine-tuning.

7 Conclusion & Future Work

We conclude from our ablation study that weakly-supervised data generation can compensate for data label scarcity to a certain extent. In traditional NLP, large amounts of data are required to train a performant model. In this scenario programmatic weak supervision can scalably produce large amounts data with adequate quality to significantly improve the model performance. On the contrary, in the transfer learning setting, as the applicability of the transferring of knowledge increases between the pre-trained model and the downstream task, we see a decreasing margin of benefit that weak supervision can provide.

We observe that transfer learning has indeed significantly reduced the lower bound of the size of a training set that would yield an effective model. Nevertheless, an interesting future exploration is to quantitatively determine this lower bound based on model and data attributes. This threshold can be informative in the decision-making process for various data-sensitive machine learning applications. Another potential route for future work is to apply the same study on tasks where the finetuning vocabulary space differs from that of Bert's pre-training data, and observe how the lower bound for train set size changes when the transferring of pretrained knowledge becomes less applicable.

8 Contributions

Milestone	Contributor
Setup Google Cloud Platform storage and virtual machine setup	Fei
Prepare and sample OG, SD, TEST datasets	Zihan
Write naive label generator and build WD-N dataset	Zihan
Build multiple label generating functions and work with Snorkel to build WD-S dataset	Fei
Implement data processor for reading datasets and convert into model-friendly formats	Fei & Zihan
Incorporate BERT training to work on our datasets, including automatic DEV set evaluation and early stopping, automatic TEST set evaluation and score reporting	Fei
Implement LSTM training to work on our datasets, including automatic DEV set and TEST set evaluation and score reporting	Zihan
Modify BERT and LSTM to work with soft labels, and modify training to regression task instead of binary classification task	Fei
Work on milestone, poster, report	Fei & Zihan

9 Source code

See `https://github.com/feifang24/cs229-project` for source code [1, 2, 10, 9, 5].

References

[1] 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, Dan 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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] François Chollet et al. Keras. `https://keras.io`, 2015.

[3] Jared Dunnmon, Alexander Ratner, Nishith Khandwala, Khaled Saab, Matthew Markert, Hersh Sagreiya, Roger E. Goldman, Christopher Lee-Messer, Matthew P. Lungren, Daniel L. Rubin, and Christopher Ré. Cross-modal data programming enables rapid medical machine learning. *CoRR*, abs/1903.11101, 2019.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[8] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 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*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[9] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. [Online; accessed 2019-12-13].

[10] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel. *Proceedings of the VLDB Endowment*, 11(3):269–282, Nov 2017.

[12] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision, 2018.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[14] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 721–729, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[15] Z. Weng, P. Varma, A. Masalov, J. Ota, and C. Ré. Utilizing weak supervision to infer complex objects and situations in autonomous driving data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 119–125, June 2019.

[16] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.