Information Retrieval from Surgical Reports using Data Programming

Hardie Cate
Computer Science Department
Stanford University
Stanford, CA 94305
ccate@stanford.edu

Elliott Chartock
Computer Science Department
Stanford University
Stanford, CA 94305
elboy@stanford.edu

Zeshan Hussain
Computer Science Department
Stanford University
Stanford, CA 94305
zeshanmh@stanford.edu

Abstract

Natural language processing (NLP) has many potential uses, but perhaps some of the most beneficial applications are for medical purposes. In this work, we perform information retrieval (IR) on medical procedure reports for patients suffering from deep vein thrombosis (DVT). We extract information related to two key domain-specific entities. Since the dataset is largely unlabeled, we use a relatively novel framework called data programming. We build upon this approach by using several different deep learning architectures as end extraction models. Our best model on one of our extraction targets attains results that outperform a regular expression baseline and other non-deep learning approaches, with an F1 score of 0.77 on the test set. These initial results are promising first steps in showing the effectiveness of our methodology for information extraction on unlabeled medical reports.

1 Introduction

The field of NLP has recently made great progress in a wide variety of tasks and benchmarks with the help of deep neural networks. However, one drawback of these networks is that they usually require large amounts of labeled data to train effectively. The process of acquiring labeled training data is a critical bottleneck in machine learning, and for many real-world applications, such hand-labeled datasets do not exist. This is a problem especially for applications of NLP in the medical sphere. In many situations, vast amounts of information are contained as unstructured data in procedure reports and images that require complex domain-specific knowledge to interpret. As such, the task of creating labeled datasets is prohibitively expensive or impossible. Without proper machine learning systems that leverage this unlabeled data, a doctor who wishes to make an informed decision for a patient based on past data would need to read through hundreds, if not thousands, of medical procedure reports. The difficulty of that task along with the critical importance of medical decision-making necessitate an effective NLP machine learning system for unstructured data.

In this work, we perform IR on a data set provided by colleagues in the Department of Radiology at the Stanford School of Medicine. The dataset consists of medical procedure reports for patients suffering from DVT, which is the formation of a blood clot within a deep vein, most commonly occurring in the legs. We design models that extract information in the reports about stents, which are

support tubes inserted into veins that aid healing and relieve obstructions. We search for mentions of two key extraction targets, namely *stent brand* and *stent diameter*, within the unlabeled set. To address the lack of labeled data, we use the data programming paradigm outlined in Ratner et al. [7], which employs weak supervision in the form of heuristics that can easily be applied programmatically. We then build discriminative deep learning models, most notably a bidirectional LSTM, that learn from this data in a noise-aware fashion.

In Section 2, we discuss background and related work, including a brief overview of Snorkel, a prominent implementation of the data programming framework. In Section 3, we outline our technical approach which includes our data programming pipeline and our deep learning end extraction models. In Section 4, we discuss the dataset and our experiments. Finally, in Section 5, we review our results, and in Section 6, we conclude with our overall findings and future work.

2 Related Work

Many hospital reports currently reside in the form of free text as dark data - unstructured data that cannot be processed by existing software. In the current age of big data [3], curating and understanding patient data is an increasingly prevalent problem in the field of health informatics. The application of accurate information extraction from biomedical reports is widespread, including clinical decision support, treatment assistance, and pharmacovigilance [1, 2].

Semi-supervised learning techniques such as bootstrapping and transfer learning have proven to be a good start in the task of biomedical information extraction [4]. Xu and Wang use bootstrapping techniques to identify drug-gene pairs in MEDLINE articles [9]. Their model, which starts with just one labeled relation, outperforms previous dictionary-based approaches to the same dataset. While semi-supervised learning is an improvement over non-machine learning extractors, other weak supervision approaches have yielded better results, while avoiding the need for even a single labelled extraction target.

Oberkampf, et al. use a distant-learning model to extract measurement-entity relations from radiology reports [6]. Their knowledge-based approach uses an annotator to perform measurement extraction and named entity recognition of ontology concepts, and then further test the extracted relations on a knowledge model that contains common size specifications for the extracted relation. In this project the authors discover that high F1 can be achieved with a large enough ontology, irrespective of knowledge base size. Many distant-learning approaches, like the work of [6], hinge on the existence of large and accurate ontology databases, which will not be accessible in all domains.

Ratner et al. solve this problem with the data programming paradigm [7]. This approach allows users to specify weak supervision strategies in the form of heuristics, or *labeling functions*, which are used to programmatically generate training sets. These labeling functions are often noisy and may conflict, but by modeling the process as a generative model, it is possible to "denoise" this training set in many cases. A "noise-aware" discriminative end model is then created, which is where the user can apply conventional deep learning architectures. Ratner et al. show that given certain conditions on the labeling functions, their method achieves the same asymptotic scaling as supervised learning methods. In other words, with just $O(1)$ labeling functions, data programming is capable of achieving the same guarantees for a desired error bound ϵ as a labeled dataset with the canonical $O(\epsilon^{-2})$ training examples. Ratner et al. apply data programming to unlabeled datasets in tasks including genomics and pharmacogenomics, in which they outperform baseline distant supervision approaches by an average 2.34 points for F1 score.

3 Technical Approach

3.1 Problem Statement

Our inputs in this problem are unstructured reports of surgical procedures done on patients with Deep Vein Thrombosis. Given an input report, we aim to train a model that finds mentions of an extraction target. We focus on two extraction targets: *stent brand* and *stent diameter*. More concretely, our method can be divided into two phases. The first phase entails applying the data programming paradigm to the reports, where we extract candidates and noisily label them using our

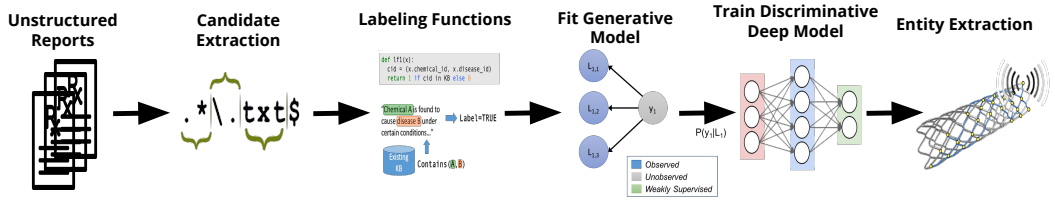

Figure 1: A diagram of our full workflow, starting with unstructured raw text to a fully structured training set on which we train several discriminative models.

labeling functions. In the second phase, we analyze and experiment with several end deep learning models that perform the classification. Furthermore, for each candidate, the end extraction model classifies whether that candidate is a mention of the extraction target (i.e. either *stent brand* or *stent diameter*). A more detailed explanation of both phases is given in the following sections.

3.2 Data Programming with Deep Learning

Our implementation of the data programming pipeline leverages and continues the existing pipeline created by Ratner et al. (entitled Snorkel [8]). To get the pipeline off the ground, we run the procedure reports through a preprocessing step that splits the document into sentences and tokens and provides such annotations as part-of-speech tags, dependency parse structure, named entities, etc. using Stanford CoreNLP [5].

3.2.1 Candidate Extraction

After parsing and preprocessing the text, the first major step in the data programming paradigm is candidate extraction. In this stage, we extract words or tokens that represent possible mentions for our extraction targets, *stent diameter* and *stent brand*. Our candidate extractor for *stent diameter* considers any token containing a numerical character, while the corresponding one for *stent brand* retrieves all n-grams for $n = 1, 2, 3$. We choose a maximum value of $n = 3$ because according to our colleagues in Stanford Interventional Radiology, the vast majority of stent brands/types are expressed in three words or fewer. The objective of candidate extractors is to maximize recall at the potential expense of precision. They aim to capture all possible mentions of the extraction targets, which are then passed on to the labeling function stage.

3.2.2 Labeling Functions

Labeling functions are the next step in the pipeline that make weak supervision possible. Formally, a labeling function is a mapping $\lambda : \chi \to \{-1, 0, 1\}$ on the set of candidate extractions χ that provides a non-zero label for some subset of the objects. These functions can encode domain heuristics or incorporate information from external knowledge bases. Since they are intended to predict heuristically and potentially naively, individual functions need not have perfect precision or recall; rather labeling functions can (and should) overlap and conflict. In the next step, we describe the generative model that learns the dependencies among labeling functions.

3.2.3 Generative Model

Once we have enough reasonable labeling functions (on the order of 5-10 in most cases) with sufficient overlaps, conflicts, and coverage, we use these functions to train a generative model. The model learns, under certain reasonable conditions, a parameterization of the interactions between labeling functions by applying maximum likelihood estimation and minimizing a noise-aware risk measure. Once the learning is complete, the model takes as input a candidate with surrounding context and generates a noisy label. Rather than hard labels consisting of strictly 0 or 1, these noisy labels are real-values between 0 and 1 that represent an estimate of the probability of the occurrence of a true label based on the labeling functions applied to the candidate. At this point, we can apply a noise-aware discriminative end model that learns using these generated labels.

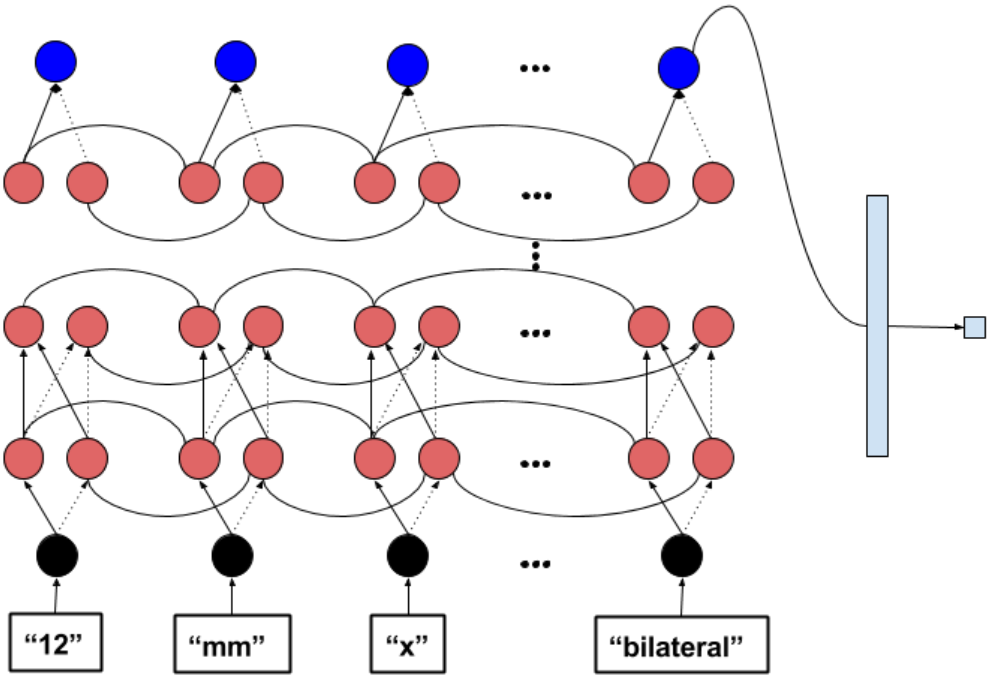

Figure 2: Diagram of our full multilayer bi-LSTM model.

3.2.4 Deep Discriminative End Models

We utilize three baseline models. The first baseline simply finds all mentions of the extraction target using a regular expression. For *stent diameter*, the regex that we use takes advantage of the fact that the diameter is usually the first dimension in the following measurement format for a stent: A mm x B cm. Similarly, the regex that we use for stent brand simply checks whether there is a number plus a dimension immediately on the left as well as the word "stent" on the right.

Our second baseline model is a vanilla logistic regression. We run this model on the features extracted from the candidate and the candidate context, including its span text as well as dependency path. These features are part-of-speech tags, dependency parse structure, lemmatized word forms, and named entities. We pass this feature matrix, F, into the logistic regression model as follows,

$$\begin{aligned} \hat{y} &= \sigma(Fw + b) \\ loss &= \mathbf{CE}(\hat{y}, y_{marginal}) \end{aligned} \tag{1}$$

Note that instead of using strict 0 or 1 labels in the training set, we use the probabilistic labels (training marginals) generated by the generative model in the data programming phase. The third baseline model that we use is a fully connected neural network with one hidden layer. We use the same featurized matrix that was used as input for the logistic regression model.

Our final model is a multilayer bidirectional LSTM (bi-LSTM). The input to the bi-LSTM is the sentence that contains the candidate. We process the input slightly by tagging the candidate with an open and closing brace. For example, suppose the original sentence containing the candidate is "The 13 mm x 14 cm stent was inserted into the common iliac vein," with "13" being the candidate. Then the new input sentence is, "The –[[13]]– mm x 14 cm stent was inserted into the common iliac vein." The reasoning behind tagging the candidate in the input sentence is to insert a local attention mechanism for the bi-LSTM. Intuitively, the brackets indicate to the bi-LSTM where the candidate is in the sentence, allowing it to pay more attention to that specific part of the sentence. Our base model is a one layer bi-LSTM that consists of a recurrent neural network in both the forward and backward directions and one output layer. As input to the output layer, we concatenate the outputs that are

emitted from the final hidden states of both the forward LSTM and the backward LSTM. Formally, if the outputs from the forward LSTM and the backward LSTM are h_T^f and h_T^b, respectively, then the input to the final output layer is, $h_T = [h_T^f; h_T^b]$. We then apply dropout to h_T to help overfitting issues and then run it through the following affine transformation: $out = Wh_T + b$. Finally, we compute a logistic loss during train time as shown in Equation 1 and compute a prediction during test time by running out through a sigmoid layer. Our full architecture is shown in Figure 2.

4 Experiments

4.1 Dataset

The dataset consists of 561 procedure reports as free-form text, sixty-six of which are labeled for our two extraction targets. We divide these labeled reports into development and test sets. Each label consists of a character span within a document marking the location of a mention of an extraction target. Thus, given an extraction target and one of these 66 documents, each of the labeled spans has a label of 1 and all other spans in that document have an implicit label of 0. Reports are labeled such that these spans correspond precisely to mentions in the text rather than to individual stents. For instance, a single stent might have references in several places in the text, each of which corresponds to a mention in the labeling scheme. See Figure 3 for an example excerpt from one of the procedure reports.

> "...the IVC and bilateral iliac veins 20. Simultaneous bilateral common iliac vein 12 mm x 8 cm SMART stent deployment 21. Balloon venoplasty bilateral..."

Figure 3: An excerpt from the dataset containing a mention of both the *stent brand* and *stent diameter* extraction targets. The diameter mention is shown in blue and the brand is in red.

4.2 Experiments and Evaluation

We run several experiments for our entity extraction tasks. First, we run regex baselines on our test set; essentially, this experiment entails writing a regex (as part of a labeling function, for example) and predicting 1 for a candidate if it matches the regex exactly and -1 if it does not match the regex.

Next, we run the logistic regression and 1-layer neural network models on our dev and test set. For both models, we first run a random hyperparameter search on the dev set. Specifically, we search the space of learning rates, L1 regularization, and L2 regularization parameters between 10^{-2} and 10^{-6}. Then, we run our best model found during this random search on the test set and report our results.

Our final experiments are with the multilayer bi-LSTM model. We run extensive experiments on the characteristics of the bi-LSTM model to analyze its behavior. In particular, we see how performance of the bi-LSTM is affected by varying the embedding dimension size of the input vectors, the number of hidden layers, the rebalance parameter, and the learning rate. Note that the rebalance parameter refers to the ratio of positive to negative examples in our training set. Because our dataset is unbalanced towards negative examples, we specify a rebalance parameter in order to dynamically rebalance our dataset before train time by randomly sampling from the negative examples. The results of these experiments, all of which are run on the dev set, are shown in the next section. Finally, we take the best parameters from our prior experiments and run a final model with those parameters on the test set. To measure the performance of our models, we use precision, recall, and F1, standard metrics in NLP and information retrieval.

5 Results and Analysis

In Figure 4 and Figure 5, we report our findings on the behavior of the bi-LSTM for *stent diameter* when varying its hyperparameters. In Figure 6 (see Appendix), similar results can be found for the behavior of the bi-LSTM on stent brand entity extraction. Note that we did not run as many

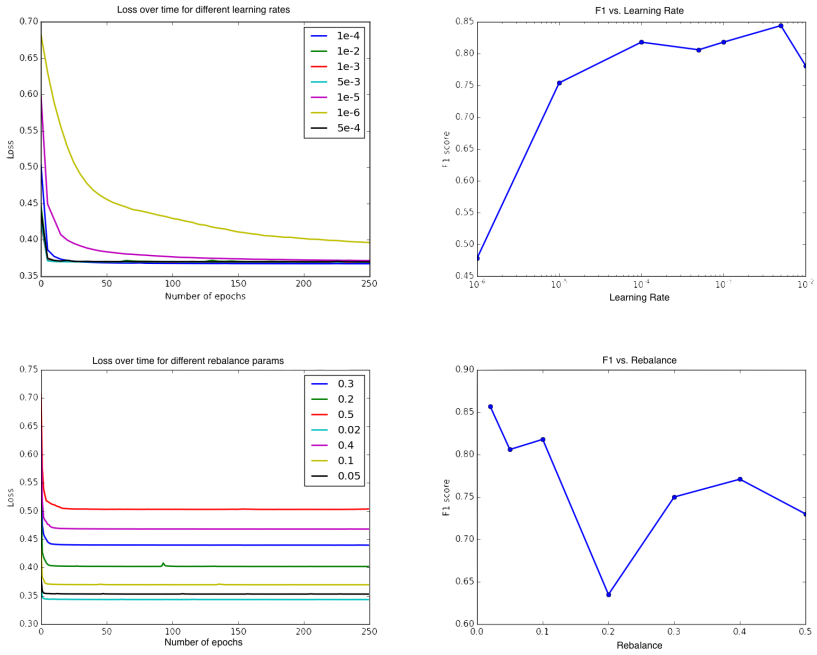

Figure 4: Learning rate experiments for bi-LSTM (top); Rebalance parameter experiments for bi-LSTM (bottom)

	Stent Diameter				**Stent Brand**			
Model	Regex	LR	FCN	Bi-LSTM	Regex	LR	FCN	Bi-LSTM
Precision	0.814	0.540	0.605	0.818	0.6	0.118	0.192	0.111
Recall	0.565	0.964	0.929	0.726	0.326	0.674	0.789	0.958
F1	0.667	0.692	0.732	**0.770**	0.423	0.201	0.309	0.200

Table 1: Final results on two extraction targets for various models

hyperparameter experiments for *stent brand* due to time constraints. The candidate training set for strent brand is an order of magnitude bigger than the candidate training set for stent diameter; consequently, the bi-LSTM takes much longer to train.

Analyzing the hyperparameter experiments done for *stent diameter*, we see how altering learning rate affects performance. Namely, there is a fairly smooth improvement in F1 score as the learning rate is increased, plateauing at around 10^{-2}. We observe an opposite trend with respect to rebalance parameter. As we increase the rebalance parameter, the F1 score tends to go down. This phenomenon makes sense, as a higher rebalance might lead to more overfitting on the positive examples. More positive examples, via more procedure reports, will help this problem significantly. Next, we see that a lower embedding dimension results in a higher F1 score, which can be explained with a similar overfitting argument. Namely, model complexity increases with higher dimensional embeddings, potentially leading to overfitting due to low bias and high variance. Finally, we increased the number of hidden layers in our bi-LSTM, with not much affect in performance. This result is most likely due to a lack of data to train deeper than 1-layer bi-LSTM models. A common trend in all our experiments is that adding more complexity to components in our model tends to either have little or negative effect on performance due to lack of data.

For *stent brand*, we look at behavior of the bi-LSTM model when varying learning rate and rebalance parameters. The effect of varying learning rate on bi-LSTM performance for *stent brand* is akin to the effect of this variation on performance for *stent diameter*. That is, higher learning rates in our

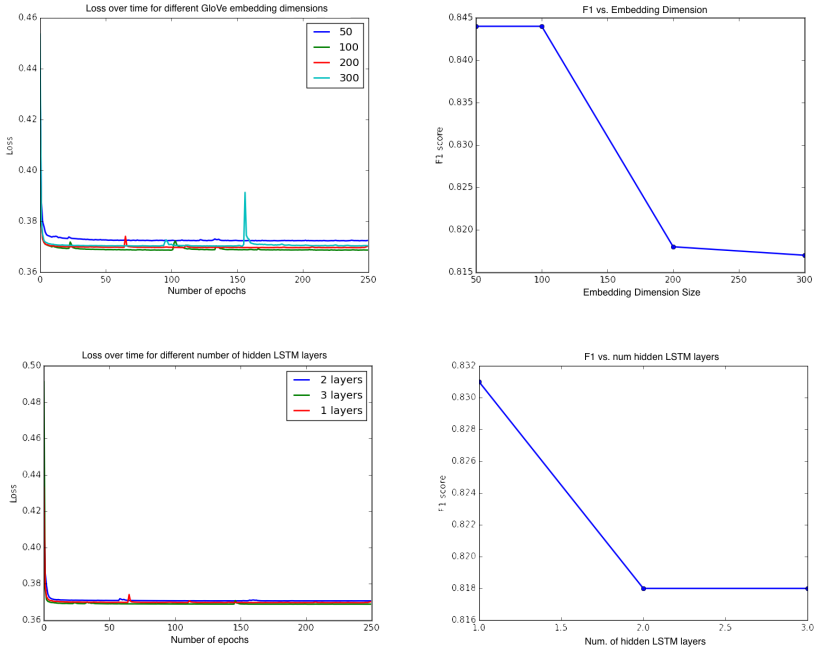

Figure 5: Bi-LSTM experiments varying number of embedding dimensions (top); Results for different number of bi-LSTM layers (bottom)

search space tend to lead to higher F1 scores. With regard to rebalance, we see a somewhat opposite effect when compared to *stent diameter*. Namely, a higher rebalance leads to a higher F1 score.

We report a summary of our final baseline and bi-LSTM results in Table 1. For *stent diameter*, our bi-LSTM outperforms all baselines by at least 4 F1 points, with an F1 score of 0.77. We slightly overfit on the dev set, which is expected given the complexity of the bi-LSTM and the dearth of data. For *stent brand*, our logistic regression and 1-layer neural network models get F1 scores of 0.201 and 0.309, respectively, while our bi-LSTM produces an F1 score of 0.200. There are several reasons for why the bi-LSTM does not outperform the baselines for stent brand. The primary reason is that the candidate set for *stent brand* is too large, meaning that there are potential candidates that match some words in a common stent brand, but may not actually be a true mention. This will lead to many false positives, leading to low precision, which is exactly what we observe. It may also be the case that our labeling functions are too noisy. More time to iterate on the candidate extraction and labeling functions for *stent brand* would certainly lead to higher performance.

6 Conclusion

In this work, we apply the data programming paradigm to weakly supervise unlabelled surgical reports. We then apply noise aware deep learning models to the generated labels to perform binary classification on certain extraction targets. The bidirectional LSTM has an F1 score of 0.77 for stent diameter, significantly outperforming other tested models. The relatively small dataset available for this project limits the complexity of deep learning models that we can train. For future work, we will acquire a larger dataset and experiment with a deeper LSTM network with attention. We will also make better use of the domain expertise of our friends at the Stanford Hospital to refine our labeling functions.

An important takeaway from our work is that despite our limited knowledge of DVT, we were able to train an accurate extraction model. We have successfully demonstrated a proof of concept for synthesizing weak supervision with deep learning models to effectively extract information from unlabeled surgical reports.

Acknowledgements

We would like to thank Alex Ratner, Nishith Khandwala, and Henry Ehrenberg for their help and guidance with this project. Finally, we would also like to thank Chris, Richard, and the rest of the 224N staff for teaching the course.

References

[1] Carol Friedman et al. "Representing information in patient reports using natural language processing and the extensible markup language". In: *Journal of the American Medical Informatics Association* 6.1 (1999), pp. 76–87.

[2] Rave Harpaz et al. "Text mining for adverse drug events: the promise, challenges, and state of the art". In: *Drug safety* 37.10 (2014), pp. 777–790.

[3] Steve Lohr. "The age of big data". In: *New York Times* 11.2012 (2012).

[4] Xinbo Lv, Yi Guan, and Benyang Deng. "Transfer learning based clinical concept extraction on data from multiple sources". In: *Journal of biomedical informatics* 52 (2014), pp. 55–64.

[5] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[6] Heiner Oberkampf et al. "Knowledge-based extraction of measurement-entity relations from german radiology reports". In: *Healthcare Informatics (ICHI), 2014 IEEE International Conference on*. IEEE. 2014, pp. 149–154.

[7] Alexander J Ratner et al. "Data Programming: Creating Large Training Sets, Quickly". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3567–3575.

[8] Henry Ratner Alex; Ehrenberg. *Snorkel*. https://github.com/HazyResearch/snorkel.git. 2017.

[9] Rong Xu and QuanQiu Wang. "A semi-supervised approach to extract pharmacogenomics-specific drug–gene pairs from biomedical literature for personalized medicine". In: *Journal of biomedical informatics* 46.4 (2013), pp. 585–593.

Appendix

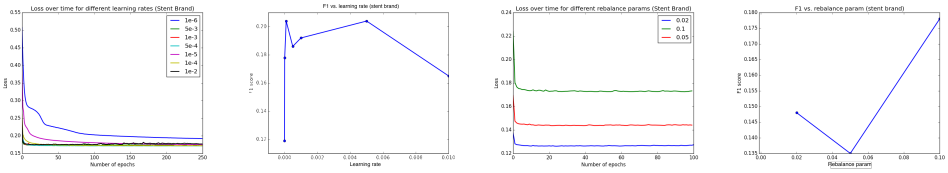

Figure 6: Stent brand experiments for learning rate (left); Experiments for rebalance parameter (right)