A Human-in-the-Loop Method for Developing Machine Learning Applications

Conference Paper · January 2020

DOI: 10.1109/ICSAI48974.2019.9010163

6 authors, including:

Yang Li
Institute of Software, Chinese Academey of Sciences
7 PUBLICATIONS **31** CITATIONS

A Human-in-the-Loop Method for Developing Machine Learning Applications

Li Yang*†‡, Menghan Li†‡,Jianlong Ren†‡, Chun Zuo†‡, Jiajia Ma†, and Weiyi Kong†‡

*Laboratory of Blockchain Technology and Application, Institute of Software, Chinese Academy of Sciences
†Institute of Software, Chinese Academy of Sciences
‡University of Chinese Academy of Sciences
Beijing, China

***Abstract*—An intelligent system is usually built based on a machine-learning model trained over offline datasets. However, such a system is difficult to adapt to new patterns or new data in the online environment, i.e. offline model has relatively poor online generalization. Moreover, understanding and solving errors taking place in the online system is also hard because errors arise in offline training pipelines and could propagate. We propose a novel systematic method, including design, architecture, and implementation to mix people's experience and intelligence with a relatively low cost. The method includes three negative-feedback loops and one data loop, supporting iterative and incremental developing procedure. Based on the method, we implemented a general system architecture and conducted two case studies. The results illustrate the effectiveness of the method.**

***Index Terms*—machine learning system, negative-feedback, human-in-the-loop**

I. Introduction

Currently, the amount of data resources and distributed parallel high-performance computing capabilities have been benefiting from the continuous improvement of the technology of data collection and storage. Hence the algorithms, usually referring to machine learning algorithms, to calculate and probe values from the data are extremely sufficient.

Traditionally in data mining, data is first collected and then processed in an offline mode. For instance, predictive models are trained using historical data given as a set of pairs (input, output). Models trained in such a way can be afterwards applied for predicting the output for new unseen input data. However, streaming data cannot be treated similarly because the data is constantly changing over time and may never end. It is impractical to accommodate such data in the main memory of the machine and is generally not feasible. Hence, only an online processing is suitable[1].

Generally the development of machine learning applications are divided into three parts: First, the data level, including feature cleaning, feature conversion, feature selection, and other feature engineering; second, the model training level, including model algorithm, model tuning, model evaluation, etc; and third, the application level, including model deployment and online service. Machine learning is a costly and high-threshold work, requiring experienced machine learning experts to carry out the appropriate machine learning task. In essence, however, machine learning is complicated and redundant. Developing a machine learning application with high quality efficiently remains ongoing challenges in the current industry.

There exist following problems when developing a machine learning application:

1) There are several gaps between different stages of machine learning, leading to incoherence.
2) The cost and threshold of machine learning are relatively high, and the experience and wisdom of algorithm practitioners are needed in the specific development process.
3) Machine learning process itself is a stage of continuous training, learning and optimization, which demands continuous improvement.

Therefore, when designing a method or system for developing machine learning applications, it is necessary to consider the connection of components, the experience and wisdom of algorithm experts, and the iteration and persistence of the development process. Plus, some processes need to be converted into automation process to reduce costs.

The existing open source machine learning platforms (e.g. TensorFlow[2], PyTorch[3], and Mxnet[4]) are always designed for specific algorithm and programming languages, failing to consider the process and coherence of developing machine learning applications as a whole, and also have problems such as the cost of platform migration. In the industry, Alibaba proposed MaxCompute computing platform based on self-developed storage and computing technology, connecting the whole process in series. However, the experience and wisdom of the algorithm experts have not been integrated into the whole process, and there is no design of sustainable and iterative loop. It is still inconvenient for application iterations and upgrades, and existing platforms do not adequately support multiple machine learning frameworks and programming languages. Currently there are a lot of researches about improving the effectiveness of machine learning applications through the human-machine cooperation[5-14], but most of them have flaws such as the lacking of generality.

In this paper we present a human-in-the-loop method for developing machine learning applications. The main contributions of this paper are:

1) A general developing method connecting the various development stages that are scattered and independent in series.

2) A system integrating human experience and wisdom into the developing loop.

The article is organized as follows. Section II discusses the advantages and drawbacks of current machine learning platforms and develop methodologies. In Section III we present the machine learning developing system. Section IV demonstrates the system performance with two case studies. Section V concludes the paper .

II. Related Works

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. It uses dataflow graphs to represent computation, shared state, and the operations that mutate that state[2]. It could adapt to plenty of optimizations, but lacks flexibility and is not convenient for rapid prototype deployment in design and use. PyTorch[3], another competitive machine learning system, features in the mix of high-level and low-level APIs, but is difficult to support large-scale applications. As a promising deep learning framework, Mxnet[4] has the advantages of fast speed, good flexibility, and supporting multiple languages. However, it doesn't consider the coherence of developing machine learning applications, and has the problem of migration across platforms.

None of the above three platforms take into account the experience and wisdom of the algorithm experts. At the same time, a sustainable and iterative loop is not designed. These limitations hinder the development of machine learning applications. Based on the disadvantages, plenty of application developing methods have been proposed.

Human computation has traditionally been an integral part of artificial intelligence systems as a means of generating labeled training data[5]. There are number of applications based on crowd-sourcing architecture, such as real-time captioning[6], person re-id[7], and labeling datasets[8]. Ece Kamar et al.[9] constructed a set of Bayesian predictive models from data and described how the models operated within an overall crowd-sourcing architecture that combines the efforts of people and machine vision. Additionally, in classifier-debugging[10][11] and dialogue systems[12], researchers have developed techniques for a domain expert to interact with the machine learning process. However, concentrating on a specific domain, they usually lack generality. More generally, in modular component-based systems, Nushi et al.[13] proposed a human-in-the-loop method for troubleshooting and error-fixing. Plus, they also presented Pandora[14], a hybrid human-machine approach to analyzing and explaining failure in component-based machine learning systems.

In general, current human-machine cooperation programs always have a lack of versatility due to professionalism and focus on a specific developing stage, without considering all stages of application development as a whole. Inspired by the negative feedback in cybernetics, we designed the method demonstrated in this paper, which informs the workers with a negative feedback when the output result does not reach the preset threshold, and then the workers supplement and correct the data and model (rather than simply solve a specific problem). When the result reaches the threshold, it is automatically operated by the machine again. By this means, developers could provide seamless service to users and ultimately achieve the effect of minimizing labor consumption.

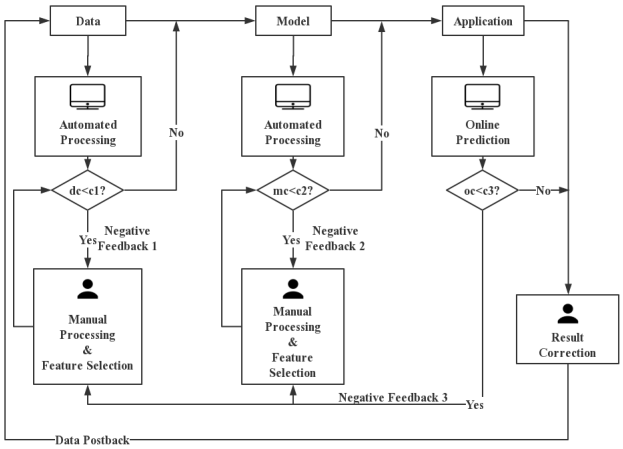


Fig. 1: The overall process of the development method

III. Machine Learning Method with Human in the Loop

The basic idea of the method is that the process of developing machine learning is divided into two stages, namely, offline and online, and an artificial negative feedback loop is constructed to perform iterative and incremental development. Figure 1 illustrates the overall process of the method.

In the offline data processing stage, when the result of the data quality analysis d_c is lower than the preset threshold c_1, negative feedback 1 is initiated to enter the manual data processing loop. When $d_c \geq c_1$, the data stream gets into the model development stage. Simultaneously, the unlabelled data is marked by both active learning and manual work to improve the quality and usability of data.

In the offline model development stage, when the automated training result m_c is less than the threshold c_2 of the set evaluation index, the negative feedback 2 is initiated to enter the artificial model development loop. The problem analysis, algorithm design and selection, and hyperparameter adjustment are performed manually. When $m_c \geq c_2$, the process gets into the online prediction stage.

In the online prediction stage, the prediction result of the online model is evaluated manually. If the evaluated result o_c is lower than the preset threshold c_3, the negative feedback 3 is initiated to activate the offline manual data processing and model training. Afterwards, the corrected data is post back to the offline through the loop for secondary development.

The above three steps are repeated to iteratively develop and continuous improve the model.

We also provide a cloud server that deploys a human-in-the-loop machine learning application development system as described above. It provides a variety of software runtime environment for machine learning model development and training. Various software runtime environment and locale

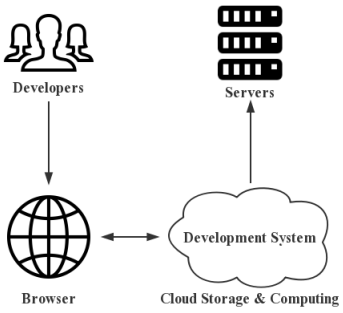


Fig. 2: The system layout

include the underlying Linux operating system - centos7, GPU accelerated computing environment, Java development library - JDK8, services Containers - Tomcat, C++ compiler - GNU Compiler Collection, python development management package - Anaconda, these environments are configured on the server side, providing web interface web access on the desktop. Figure 2 depicts the system layout. Users could access the human-in-the-loop machine learning application system through browsers on various devices (mainly personal computers). The storage and calculation are performed on the cloud server, to which users upload data, files, and codes through browsers to perform operations.

We will explain the offline data processing and model training in detail separately in the following two subsections.

A. *Offline Data Processing*

As shown in Figure 3, the data resources consist of the basic data and data calculated in the online loop. The data processing mainly includes two parts:

1) Automated data processing and quality analysis: Specifically, the automated information cleaning and processing methods include:

1) Missing value processing. Discard or meaning fill (fill with the record mean of the dimension in which the missing field is located) by default.
2) Noise processing. The noise data is processed in a smooth manner by sub-boxing, that is, the data is sorted according to the principle of equal depth, and then replaced by the median in the box;
3) Deduplication for the duplicate data;
4) Balance for the samplings with unbalanced distribution of categories. The small samples are randomly resampled, that is, oversampled, and the large samples are randomly sampled, that is, undersampled.

The quality analysis of the data is set as automated quantitative analysis, including the following four aspects (the total number of data records is set as n):

1) Deficiency analysis, including the absence of a record of the absence of a field in the record. It is quantified as a percentage of the missing records ($nLoss$) to the total number of records $q_1 = nLoss/n$;

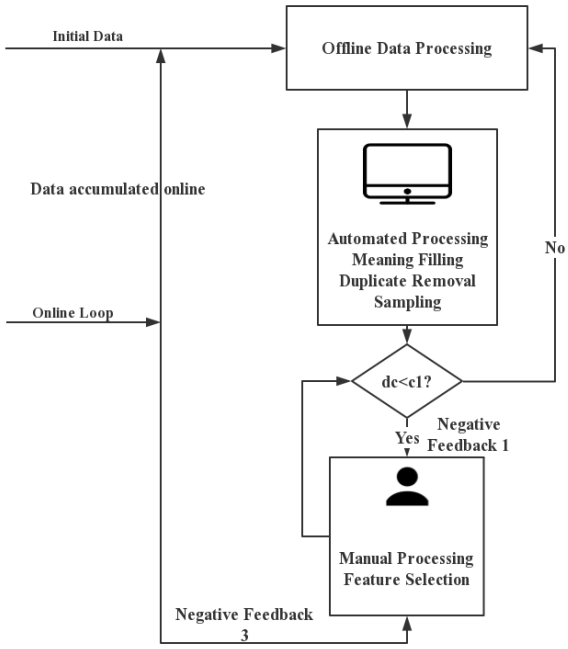


Fig. 3: The Schematic diagram of offline data processing

2) noise and anomalies detection to check whether the data has input errors or contains unreasonable data. It is quantified as a percentage of the odd records ($nOdd$) to the total number of records $q_2 = nOdd/n$;
3) Repetitive detection to detect the repeated content in the data. It is quantified as a percentage of the repeated records ($nRepeat$) to the total number of records $q_3 = nRepeat/n$;
4) Data category distribution analysis to check whether the data samples are evenly distributed into each class. If the data category is x, the result of the equilibrium distribution should be n/x for each type of record. And the result of the analysis is quantized as the ratio of the number of records in each category to twice the number of records in the equilibrium distribution $q_4 = \sum_{i=1}^{x} \mid n_i - n/x \mid /2n$.

The final result of data quality analysis is $d_c = 0.3 \times q_1 + 0.3 \times q_2 + 0.3 \times q_3 + 0.1 \times q_4$.

2) Manual data processing loop based on the negative feedbacks: when the result of the data quality analysis d_c is lower than the preset threshold c_1, negative feedback 1 is initiated to enter the manual data processing loop until $d_c \geq c_1$, and the data stream is propagated to the model development stage. Additionally, when the online model evaluation result o_c is smaller than the preset target threshold c_3, negative feedback 3 is initiated, and the manual processing data loop is entered to iteratively develop the application.

Besides, the original data (i.e. unlabeled data) is marked by active learning assisted by manual learning. The active learning algorithm is divided into two stages:

1) The initial stage. Randomly a small part is selected from

TABLE I: automated machine learning AutoML algorithm selection and super-parameter configuration

Algorithm	Type	Number of parameters	Description of Parameters	Range
Random Forests	Classification, Regression	2	rf_max_depth: max depth of the tree	loguniform(np.log(2), np.log(10))
			rf_n_estimators: number of the decision tree	randint(2, 35)*10
Gradient Boosting Tree	Classification, Regression	3	gbt_max_depth: max depth of the tree	loguniform(np.log(2), np.log(10))
			gbt_n_estimators: number of the decision tree	randint(2, 35)*10
			gbt_learning_rate: learning rate of every tree	loguniform(np.log(0.05), np.log(0.5))
Support Vector Machine	Classification	3	svm_regType: regularization type	pchoice([0.5,' |1 '], [0.5,' |2 ']
			svm_regParam: regression parameter coefficient	loguniform(np.log(1e-4), np.log(1e-2))
			svm_step:gradient descent learning rate	loguniform(np.log(1e-5), np.log(1))
Logistic Regression	Classification	3	lr_regType: regularization type	pchoice([0.5,' |1 '], [0.5,' |2 ']
			lr_regParam: regression parameter coefficient	loguniform(np.log(1e-4), np.log(1e-2))
			lr_step:gradient descent learning rate	loguniform(np.log(1e-5), np.log(1))
Logistic Regression	Regression	1	lm_step: gradient descent learning rate	loguniform(np.log(1e-5), np.log(1))

the unlabeled data U, labeled by supervisor S as the training set to establish the initial classifier model.

2) The looping query phase. The supervisor S, from the unlabeled sample set U, selects certain unlabeled samples for labeling according to a query criterion Q, adds them to the training sample set L, and retrains the classifier until the training stop criterion is reached.

Use the optimized distributed gradient boosting tree algorithm XGBoost as the active learning classifier model. The algorithm flow is as follows:

1) initiate the model as a constant model:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{1}$$

where x denotes the input, y_i denotes the label for sample i, γ denotes the residual parameter, and n denotes the number of samples;

2) Iteratively generates M base learners for m from 1 to M:

a) Calculate the value of the negative gradient of the loss function in the current model as a residual estimate:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{2}$$

where $F(x_i)$ denotes the predicted value of the sample i, $F(x)$ denotes the model function, and $F_{m-1}(x)$ denotes the current model function;

b) Estimate the regression tree node area $R_{jm}, j = 1, 2, ..., J_m$, to fit the approximate value of the residuals;

c) Estimate the value of leaf nodes area by linearly searching to minimize the loss function:

$$\begin{aligned}\gamma_{jm} = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x)) \\ + \frac{1}{2}\gamma \sum_{\gamma=1}^{T} \parallel w_j \parallel_2 + \gamma T\end{aligned} \tag{3}$$

where T denotes the number of leaf nodes, and w_j denotes the weight of leaf node j;

d) Update the regression tree model:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}) \tag{4}$$

where I denotes the indicator function. $I = 1$ if the value in parentheses is true, otherwise $I = 0$.

3) output the final model: $\hat{F}(x) = F_m(x)$

B. Offline Model Development

As shown in Figure 4, the offline model development is mainly divided into two parts:

1) The automated machine learning development method AutoML: This paper mainly focuses on the prediction problem, including classification and regression algorithms. The

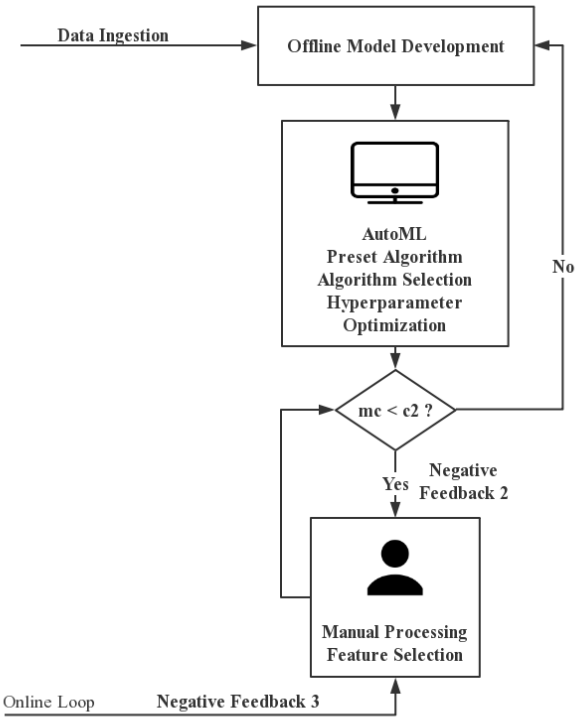


Fig. 4: The schematic diagram of offline model training.

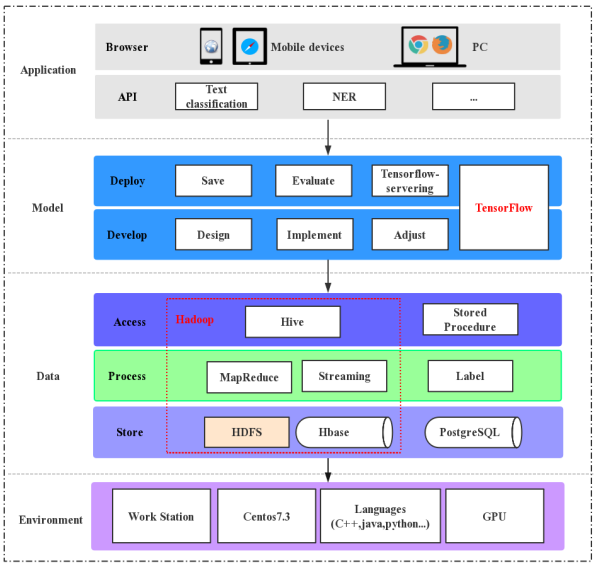


Fig. 5: System architecture for implementation of our proposed method

preset classification and regression algorithms and their hyperparameters are shown in TABLE I. For the classification algorithms, the default evaluation index is $f1$. Precision P and recall value R are also acceptable. For regression algorithms, the default evaluation index is mean-square error MSE, which can be adjusted according to the actual situation. When the index result m_c of AutoML is greater than or equal to the set threshold c_2, the model data stream feeds forward to the online prediction stage;

2) Manual data processing loop based on the negative feedbacks: When AutoML is used for model training, once the final index result does not reach the set threshold, ie $m_c < c_2$, negative feedback 2 is initiated, and the artificial development model loop is entered. The machine learning developer performs model design and training. When the actual evaluation index o_c of the loop is lower than the set target threshold c_3, the negative feedback 3 is initiated, and the loop of the artificial development model is entered during the iterative development.

IV. Demonstration

In order to demonstrate the effectiveness of our proposed method, we design and implement a system for developing machine learning applications according to human-in-the-loop based approach. As shown in Figure 5, the system architecture includes four layers: environment, data, model and application. The environment layer includes software and hardware configurations. The data layer provides the functions of storage, processing and access for data. The model layer supports implementing, training and deploying model. And finally, the application layer involves service interface-API and browser access. We conduct two case study: named entity recognition (NER) and text classification on our system based on human-in-the-loop method.

A. Text Classification Case

In a comprehensive system engineering project for information collection, processing, analysis, disposal, and prevention of Internet information, the discovery, screening, and disposal of harmful and sensitive information are difficult due to the strong concealment of online information. It is necessary to use machine learning algorithms to label network information

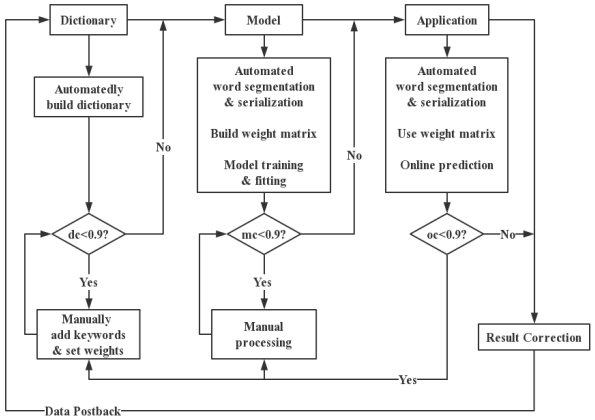


Fig. 6: The XGBoost execution process based on domain knowledge base with the human-in-the-loop method

and improve the correctness and timeliness of the harmful judgment of massive information. We chose XGBoost algorithm, which automatically utilizes the multithreading of the CPU for parallelism. Researches showed that XGBoost could achieve better classification results in the case of small samples. However, the traditional XGBoost algorithm can not solve the labeling problem with high domain-related content. Therefore, we propose an improved algorithm for configurable special keyword weights based on this method, whose developing process is illustrated in Figure 6.

Results before and after adopting the methodology are shown in TABLE II. The entries are from Internet social applications with some fuzzy processing. Categories of the three entries have turned from Military, which consists of both positive and negative corpus, to Harmful, which is more accurate. Apparently, the usage of the method we proposed could significantly benefit the effect of the classifier.

TABLE II: Simulation results of the classifier (weight added: 4 for Lake-Town, 4 for Thranduil, and 3 for Dale)

Tested corpus	Before	After
A riot in Lake-Town! So close to hurt people! I'm so scared!	Military	Harmful(Horror/Rumor)
At about 21: 00-22: 00 on the evening of yesterday, a large number of Goblin rioted near the northern secret forest. The number of casualties is unknown, and 36 dwellings are destroyed. Thranduil gives instructions for force suppression, trying to control the riot situation.	Military	Harmful(Horror/Rumor)
Lake-Town, Mt. Doom, and Dale are in chaos! Many Goblins are getting together. It is estimated that they will have a fight again. Something big is going to happen! Even the Elf King can't control it!	Military	Harmful(Horror/Rumor)

B. NER Case

NER, also known as entity identification and entity extraction, is a subtask of Natural Language Processing (NLP) , which seeks to locate and classify named entity mentions in text into pre-defined categories such as the person names, organizations, locations, date, etc. As one of lexical analysis methods, NER is a basic, key task in NLP and it is more difficult than other lexical analysis tasks, such as Chinese word segmentation and part of speech tagging. More importantly, it lays the basis for many NLP tasks, relation extraction, event extraction, knowledge graph, machine translation and dialogue system. etc. Nowadays, more and more unseen new entity mentions arise quickly with the development of economy, information technology and industry, but it is hard to recognize such new entities for offline trained model. Because present model is data-driven and bases on fixed pre-defined dataset, it is difficult to recognize new and unseen entities for such models.

We present a human-in-the-loop based approach to train a NER model aiming to recognize new and unseen entities. As

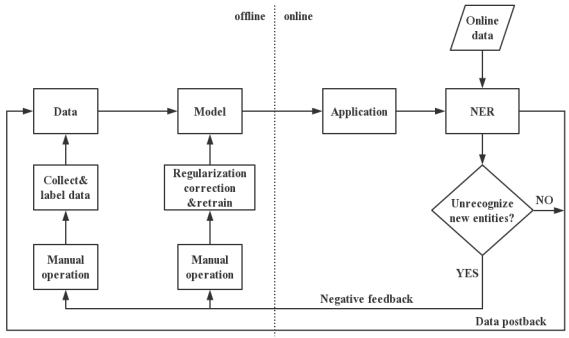
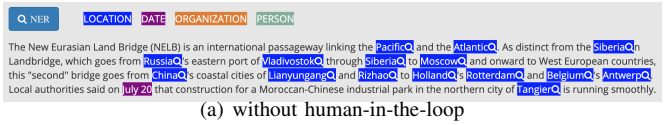
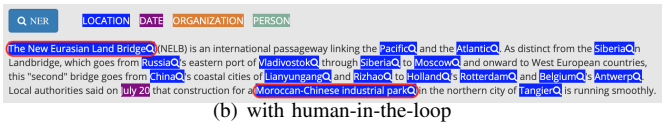


Fig. 7: The NER execution pipeline with human-in-the-loop

depicted in Figure 7, when unseen and new entities arise in the online application, the developers or users call negative-feedback, and then two operations will be conducted including two aspects: data and model. At the same time, the new corpus will be corrected and added to the dataset for iterative training. Specifically, negative-feedback will be committed to offline data and model when new entities arise, then human operations will be conducted. Firstly, to collect and label corpus about new entities, going to next stage after data processing: $d_c >= 0.9$; Secondly, the model will be retrained on the new dataset. At very first, the model is combined training with regularization correction, i.e. sentences including new entities will be identified via regular match. Finally, the model training will only depend on data with the amount of data increasing. We tested the system with an article on the Internet[1]. The experimental results are depicted in Figure 8, it could be seen that two new entities (The New Eurasian Land Bridge and Moroccan-Chinese industrial park added due to the Belt and Road Initiative) which were not recognized by the old model are marked by the new model with human-in-the-loop.



(a) without human-in-the-loop



(b) with human-in-the-loop

Fig. 8: A comparison of NER based on human-in-the-loop

Besides the case studies above, we believe more applications could achieve better results through the human-in-the-loop method we proposed in this paper.

V. Conclusion and Future Work

We present a developing method for machine learning applications, which covers the entire process and details from raw data to final application, and is more complete and

[1]http://www.xinhuanet.com//english/2017-05/09/c_136268314.htm

comprehensive. Based on the principle of negative feedback regulation in cybernetics, it integrates human wisdom and experience into a loop-based process at a small cost, and performs iterative and incremental development, effectively connecting all stages of application development in series. The experiments in Section IV illustrate the effectiveness and efficiency of the method. Future work directions include the expansion of solvable problems such as clustering and anomaly detection. We also expect to enable non-expert workers to provide relatively effective services by upgrading the method.

ACKNOWLEDGMENT

This work was supported by the Strategy Priority Research Program of Chinese Academy of Sciences (No. XDA20080200), the National Key Research and Development Program of China (No. 2018YFB1005002), the National Natural Science Foundation of China (No. 61572479), the National Natural Science Foundation of China together with the National Research Foundation of Singapore (No. 61661146002).

REFERENCES

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[3] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.(2017)," *URL https://github. com/pytorch/pytorch*, 2017.

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[6] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham, "Real-time captioning by groups of non-experts," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 2012, pp. 23–34.

[7] H. Wang, S. Gong, X. Zhu, and T. Xiang, "Human-in-the-loop person re-identification," in *European conference on computer vision*. Springer, 2016, pp. 405–422.

[8] J. C. Chang, S. Amershi, and E. Kamar, "Revolt: Collaborative crowdsourcing for labeling machine learning datasets," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 2334–2346.

[9] E. Kamar, S. Hacker, and E. Horvitz, "Combining human and machine intelligence in large-scale crowdsourcing," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 467–474.

[10] J. M. Attenberg, P. G. Ipeirotis, and F. Provost, "Beat the machine: Challenging workers to find the unknown unknowns," in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[11] A. Chakarov, A. Nori, S. Rajamani, S. Sen, and D. Vijaykeerthy, "Debugging machine learning tasks," *arXiv preprint arXiv:1603.07292*, 2016.

[12] M. Singh, P. Agarwal, A. Chaudhary, G. Shroff, P. Khurana, M. Patidar, V. Bisht, R. Bansal, P. Sachan, and R. Kumar, "Knadia: Enterprise knowledge assisted dialogue systems using deep learning," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1423–1434.

[13] B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, "On human intellect and machine failures: Troubleshooting integrative machine learning systems," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[14] B. Nushi, E. Kamar, and E. Horvitz, "Towards accountable ai: Hybrid human-machine analyses for characterizing system failure," in *Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.