INTRODUCING DEEP REINFORCEMENT LEARNING TO NLU RANKING TASKS

Ge Yu, Chengwei Su, Emre Barut

ABSTRACT

Natural language understanding (NLU) models in production systems rely heavily on human annotated data, which involves an expensive, time-consuming, and error-prone process. Moreover, the model release process requires offline supervised learning and human in-the-loop. Together, these factors prolong the model update cycle and result in sub-standard model performance in situations where usage behavior is non-stationary. In this paper, we address these issues with a deep reinforcement learning approach that ranks suggestions from multiple experts in an online fashion. Our proposed method removes the reliance on annotated data, and can effectively adapt to recent changes in the data distribution. The efficiency of the new approach is demonstrated through simulation experiments using logged data from voice-based virtual assistants. Our results show that our algorithm, without any reliance on annotation, outperforms offline supervised learning methods.

Index Terms— ranking, natural language understanding, reinforcement learning

1. INTRODUCTION

Voice-based virtual assistants such as Google Assistant and Amazon Alexa have gained popularity over the past few years, which brings great opportunities as well as challenges to natural language understanding (NLU) systems. The production systems of these devices generally rely heavily on annotated data and are typically trained through supervised learning. However, data annotation is expensive and often time-consuming. In addition, model updates through offline supervised learning can be lengthy and fail to capture trending requests.

The NLU model in the underlying system of voice-based virtual assistants usually classifies the user request into *hypotheses* for downstream applications to fulfill. A hypothesis consists of the user intention (intent) and the Named Entity Recognition (NER) tags. For example, the correct hypothesis for "play a Madonna song" is (PlaySong intent, ArtistName: Madonna) and for "turn on the TV" is (ControlAppliance intent, ApplianceName: TV). Our work considers a ranking task in an NLU system where hypotheses with their features are generated via fully independent domain experts, where a domain is an area of functionality, such as Music, Shopping or Weather [1]. These hypotheses are then ranked based on their scores, which are computed as a function of their features. Therefore, the ranker needs to calibrate features across domain experts and choose one hypothesis according to a policy.

We model the problem as a variant of the contextual multi-armed bandit, where arms represent the scores computed for hypotheses by domain experts, named *domain rankers*. Our model differs from the previous work in the literature [2, 3, 4, 5, 6, 7, 8] due to the independence of the arms: the features for hypotheses in a domain are not accessible by other domain rankers. Moreover, each domain ranker outputs scores for multiple hypotheses simultaneously. Thus, the setup requires a novel reinforcement learning approach with partially-observed contexts and multi-action estimates.

In this work, we utilize recent advances in deep reinforcement learning and propose an alternative online training method. Our policy gradient algorithm is a variant of REINFORCE [9] and uses the sample return as an estimate for the value of the state-action pair. Our proposed approach makes use of the implicit user feedback signals[1] to determine the rewards associated with samples in logged data, and does not depend on any human annotation. A positive reward is given if the chosen NLU hypothesis fulfills the customer's needs, and a negative reward otherwise. As the training method enables automatic model updates during runtime, the approach can swiftly pick up on trending requests. Simulation experiments show our proposed method outperforms offline supervised learning.

2. PROBLEM FORMULATION

For a domain ranker, the input hypotheses with their features are generated by the domain expert models: the domain classification score (denoted by DC $\in [0,1]$), the intent classification score (denoted by IC $\in [0,1]$) and the NER classification score (denoted by NER $\in [0,1]$). Therefore, the *context* for a request in domain d, denoted by X_d, is defined as the follow-

[1]Implicit user feedback signals are inferred from the interactions with users, for example, negative user feedback signals are inferred from users terminating a song when it starts playing. The signals can be classified into positive and negative user feedback signals, which correspond to the satisfaction and dissatisfaction of users, respectively.

ing:

$$X_d = \begin{bmatrix} DC_d^1 & DC_d^2 & ... & DC_d^n \\ IC_d^1 & IC_d^2 & ... & IC_d^n \\ NER_d^1 & NER_d^2 & ... & NER_d^n \end{bmatrix},$$

where DC_d^i, IC_d^i and NER_d^i are the DC, IC and NER scores for the i^{th} hypothesis in domain d. Note that $X_d \in \mathbb{R}^{m \times n}$, where m is the number of features for each hypothesis and n is the number of hypotheses (without specifications, $m = 3$ and $n = 5$ in this paper), and $X_{d_1} \neq X_{d_2}$ for $d_1 \neq d_2$. Each column of X_d represents the features of one hypothesis.

The outputs from a domain ranker are the scores for the input hypotheses, denoted by $s_d = f_d(X_d)$, where $s_d = [s_{d1}, s_{d2}, ..., s_{dn}]$ and $s_{di} \in (0, 1)$ is the final confidence score for the i^{th} hypothesis in domain d. The function f_d approximates the relationship between the hypothesis features and their scores, with the parameters learned in the training process.

We model a group of domain rankers as a variant of the contextual multi-armed bandit model, with each domain's ranker as an arm. The final hypothesis is chosen by gathering hypotheses with their scores from all domain rankers. When exploration is desired, the hypothesis is chosen by sampling from a policy, and when exploitation is desired, the hypothesis with the highest score is chosen. Finally, a reward $r \in \{-1, 1\}$(-1 representing dissatisfaction and 1 representing satisfaction from the user) is observed and used to update the parameters of the domain ranker. The objective is to maximize the cumulative reward over time.

3. APPROACHES FOR RANKING

We first consider linear relationships between the hypothesis features and scores by using standard contextual multi-armed bandit algorithm in Section 3.1. Then, we generalize to nonlinear relationships by using policy gradient algorithms in Section 3.2 and Q-learning algorithms in Section 3.3. We note that all of the domain rankers utilize the same input features, given by X_d for each domain d.

3.1. Contextual multi-armed bandit with linear arms

We consider two classical algorithms for linear arms: LinUCB [10] and Thompson Sampling [4, 11]. Note that the context in our problem setting is a matrix with each column as an available action, whereas the input context in classical multi-armed bandit problems is a vector for a single action [10, 4, 11].

LinUCB algorithm and Thompson Sampling algorithm use a linear model to estimate the expected reward for a context using the following formula,

$$f_d(X_d) = \theta_d^T X_d, \tag{1}$$

where $\theta_d \in \mathbf{R}^m$ is the parameter for the ranker in domain d.

3.2. Policy-gradient rankers

We design domain rankers that use non-linear functions to estimate scores for hypotheses and use REINFORCE algorithm [9] to update their parameters. For domain d, let π_{θ_d} denote the policy for choosing hypothesis. The value of policy π_{θ_d} is

$$V(\pi_{\theta_d}) = \mathbf{E}_{\pi(\theta_d)}\left[\sum_{j=1}^{N} r(t_j)\right] \approx \frac{1}{N}\sum_{j=1}^{N} r(t_j), \tag{2}$$

where t_j is the j_{th} sample trajectory (a trajectory is the chosen hypothesis for a user request). The gradient of the value function can be computed using policy gradient theorem [12] as

$$\begin{aligned}\nabla_{\theta_d} V(\pi_{\theta_d}) &= \mathbf{E}_{\pi_{\theta_d}}\left[\nabla_{\theta_d} \log \pi(\theta_d) r(t)\right] \\ &\approx \frac{1}{N}\sum_{j=1}^{N} \nabla_{\theta_d} \log \pi(\theta_d) r(t_j).\end{aligned}$$

3.2.1. *Policy-gradient Sigmoid-policy Rankers*

The policy-gradient sigmoid-policy rankers treat the multiple hypotheses in a domain independently. The ranker in domain d outputs a binary policy for each hypothesis in the current context X_d. That is, the probability of choosing the i^{th} hypothesis is given by

$$\pi_{\theta_d}(X_d^i) = sigmoid(\theta_d^T X_d^i) = \frac{\exp(\theta_d^T X_d^i)}{\exp(\theta_d^T X_d^i) + 1}, \tag{3}$$

where X_d^i is the feature for the i^{th} hypothesis in the current context X_d. For policy-gradient sigmoid-policy rankers, the exploration for hypotheses is conducted through an *exploration-depth* ϵ and a "greedy" strategy, as described in Algorithm 1.

3.2.2. *Policy-gradient MDP-policy Rankers*

The policy-gradient MDP-policy rankers work in the same way as the policy-gradient sigmoid-policy rankers but use a different formula to compute the scores by considering the dependence between the features. The features for a hypothesis in domain d with intent IT and NER labels ST are interpreted as follows:

$$\begin{aligned} DC &= \mathbb{P}(\text{the reference domain } D_r = d), \\ IC &= \mathbb{P}(\text{the reference intent is } IT | D_r = d), \\ NER &= \mathbb{P}(\text{the reference NER labels are } ST | D_r = d). \end{aligned}$$

Finally, the probability for the j^{th} hypothesis in the i^{th} domain to be the correct hypothesis is given by

$$\mathbb{P}(a_{ij}) = DC_{ij}^{\theta_1} IC_{ij}^{\theta_2} NER_{ij}^{\theta_3}, \tag{4}$$

where DC_{ij}, IC_{ij} and NER_{ij} are the features for this hypothesis, and $\{\theta_i\}$ are the calibration parameters across domain rankers.

Algorithm 1 REINFORCE Algorithm for Rankers

```
Input $\alpha_d \in \mathbf{R}$ as a learning rate for domain $d$. Input $\epsilon_{depth}$ as the exploration depth.
Initialize $\theta_d$ as all ones for all $d$.
for $t = 1, 2, \ldots, T$ do
    Compute policies $\pi_{\theta_d}(X_d^t)$ for each domain $d$
    Aggregate all policies and pick the action using the following strategy:
    $\epsilon = \epsilon_{depth}$
    while $\epsilon > 0$ do
        $d^*, i^* = \arg\max_{d,i} \pi_{\theta_d}(X_d^t)$
        sample action according to policy $\pi_{\theta_{d*}}(X_{d*}^{i*})$
        if action $(d^*, i^*)$ is picked then
            break
        else
            $\epsilon = \epsilon - 1$
        end if
    end while
    Observe the feedback $r_t$
    Update domain $\theta_{d^*}$
        $\theta_{d^*} = \theta_{d^*} + \alpha_{d*} r_t \bigtriangledown \log \pi_{\theta_{d^*}}(X_d^{i,t})$
end for
```

3.2.3. Policy-gradient Softmax-policy Rankers

Policy-gradient softmax-policy rankers use a linear function to compute scores for hypotheses in the current context X_d. Then, a softmax function is applied on top of the scores for hypotheses from all domain rankers. The result of this softmax is used as the policy to sample hypothesis as the final output. In detail,

$$\pi_\theta(a_{ij}|\{X_i\}_i) = \frac{\exp(f_{\theta_i}(a_{ij}))}{\sum_i \sum_j \exp(f_{\theta_i}(a_{ij}))}, \tag{5}$$

where $\theta = [\theta_1, \theta_2, \ldots, \theta_k]$ and k is the total number of domains. X_i is the context for the i^{th} domain, a_{ij} is the j^{th} hypothesis in the i^{th} domain and $f_{\theta_i}(a_{ij})$ is the score by the i^{th} domain ranker for hypothesis a_{ij}.

3.3. Q-learning rankers

Q-learning rankers use a sigmoid function on top of a linear function of the features to compute Q-values (i.e., the scores) for hypotheses. For the i^{th} domain, let $Q_{\theta_i}(X_i, a_{ij})$ denote the the Q-value for hypothesis a_{ij} for context X_i, where a_{ij} is the j^{th} hypothesis in the i^{th} domain, then

$$Q_{\theta_i}(X_i, a_{ij}) = sigmoid(\theta_i^T a_{ij}). \tag{6}$$

There are two differences between Q-learning rankers and policy-gradient rankers. Firstly, the hypothesis is sampled using an ϵ-greedy strategy: with probability $1 - \epsilon$, the hypothesis with the highest Q-value is chosen, and with probability ϵ, a hypothesis is chosen randomly uniformly from all of the available hypotheses. Secondly, the optimization for the parameters for Q-learning rankers is done via minimizing mean squared errors of the hypotheses' Q-values.

Table 1. Relative performance of rankers compared to the baseline on annotated data. Negative values imply reduction in error rate.

Metrics	UCB	TS	Q	Sigmoid	Softmax	MDP
ICER	+11.0%	+8.9%	+1.6%	+0.9%	-2.4%	**-4.1%**
IRER	+2.6%	+1.6%	-1.1%	-1.9%	-0.7%	**-2.8%**

UCB: Linear UCB, **TS**: Thompson Sampling, **Q**: Q-learning, **Sigmoid, Softmax, MDP**: policy-gradient with sigmoid, softmax and MDP, respectively.

4. EXPERIMENTS

We show simulation experiment results using two sets of data: (a) annotated data, whose expected response is the ground-truth and determined by human annotators, (b) unannotated logged historical data, whose ground-truth is unknown but the expected response (considered ground-truth) is determined by the implicit user feedback (when there is no negative user feedback signals). [2] We evaluate our algorithms using previously randomly recorded data following similar ideas as in [2]. We report the IRER (exact-match hypothesis error rate) and ICER (intent classification error rate) metrics for these experiments.

4.1. Experiments with annotated data

We utilize internal data set. The training and testing data contain 503,000 and 1,154,000 utterances, respectively. In the testing, the parameters for each domain ranker are initialized from the training results and remain fixed throughout testing. The baseline model is the production model where domain rankers compute the scores for hypotheses using equation (4) and are trained through offline supervised learning on the same annotated training data. For detailed designs, see [1].

Table 1 provides the relative performance of different rankers when trained on annotated data and tested on annotated data, compared to the baseline model. From Table 1, the non-linear rankers perform better than linear rankers. In terms of IRER, policy-gradient rankers with sigmoid-policy and MDP-policy have the best performances, with a relative IRER reduction of 1.87% and 2.77%, respectively. The results in Table 1 show that our training scheme works better than the traditional supervised learning with the same annotated data.

[2]The data we used comply with our commitment to users. All data have been collected under authorization and have been anonymized.

4.2. Experiments with positive unannotated logged data

In the next set of experiments, we sample from historical unannotated data where we use implicit user feedback signals to filter out *positive turns*, interactions that have no negative user feedback signals. For these turns, the production model output[3] is treated as the true hypothesis of the user request. We subsample from historical data to build a dataset with more than 800,000 utterances. The baseline model is the same model as in Section 4.1 but is trained with the unannotated training data in a semi-supervised fashion where we treat the given response as the ground-truth label.

Table 2 provides the relative performance of different rankers, compared to the baseline model, when trained and tested on unannotated testing data. We report two sets of evaluation metrics for this situation: the overall metrics on all the testing data (in rows 2 to 4), and the *stable* metrics on testing data excluding the first 10,000 samples (in last two rows), which measure the model performance after the learned parameters converge. From Table 2, the three policy-gradient rankers perform better than the baseline model after learning from 10,000 samples, and the policy-gradient MDP-policy rankers perform the best with a relative 3.87% improvement in IRER. We conjecture that this due to the multiplicative treatment of the scores in the MDP-policy framework, as given in equation (4). This allows correct hypotheses with lower scores to “bubble up” relatively more easily more during exploration, and thus helps the model to perform better on less frequent utterances due to more effective online learning.

Table 2. Relative performance of rankers on unannotated data with only positive turns. Negative values imply reduction in error rate.

Metrics	Q	Sigmoid	Softmax	MDP
ICER	+27.10%	-6.57%	+12.32 %	**-12.94%**
IRER	+5.24%	-2.79%	+2.85 %	**-3.24%**
	Performance post 10k samples			
ICER	+26.69%	-6.98%	-11.50%	**-14.58%**
IRER	+5.18%	-2.90%	-2.90%	**-3.87%**

4.3. Experiments with negative unannotated logged data

In the final experiment, we follow the same data preparation as in Section 4.2, but we also include *negative turns*, which are turns with negative user feedback signals, in addition to the positive turns as. The training and testing strategy follows that for Table 2 with one difference: for the first 10,000 samples, the hypothesis chosen by the model is forced to be the same as the recorded response. Table 3 provides the relative metrics on testing data (same as in Section 4.2) after the first 10,000 samples. For policy-gradient MDP-policy rankers, which have the best performance across the board, there is a 4.44% reduction in IRER when learning with both positive and negative turns, compared to a 3.87% reduction in IRER when learning with only positive turns.

[3]The production model is a complicated system with multiple components functioning together. We focus on improving the performance of one statistical component of the system in this work.

Table 3. Relative performance of rankers on unannotated data with both positive and negative turns. Shown results are online performances after models have learned from 10,000 samples. Negative values imply reduction in error rate.

Metrics	Q	Sigmoid	Softmax	MDP
ICER	-5.95%	-4.52%	-11.70%	**-18.07%**
IRER	-2.50%	-2.45%	-2.16%	**-4.44%**

5. CONCLUSION

We introduce deep reinforcement learning techniques for NLU ranking, where independent NLU domain expert models generate hypotheses with their features, and a domain ranker computes the scores for hypotheses in the domain. We formulate NLU ranking as a contextual multi-armed bandit problem, and apply deep reinforcement learning techniques to build a completely online model that does not rely on annotated data. Experiment results show that the new technique yields 4% and 18% reductions in the exact-match hypothesis and intent classification error rates, respectively, compared to the baseline models.

In future work, we plan to utilize user friction in a more advanced manner, by accounting for the uncertainty in friction estimation in the reward process. We also seek to improve the exploration process during online learning, for instance by using look-ahead policies as in [13] and related work.

6. REFERENCES

[1] Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas, “A re-ranker scheme for integrating large scale NLU models,” *CoRR*, vol. abs/1809.09605, 2018.

[2] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” *Proceedings of the 19th international conference on World wide web - WWW ’10*, 2010.

[3] Olivier Chapelle and Lihong Li, “An empirical evaluation of thompson sampling,” in *Advances in neural information processing systems*, 2011, pp. 2249–2257.

[4] Shipra Agrawal and Navin Goyal, “Thompson sampling for contextual bandits with linear payoffs,” in *Interna-*

tional Conference on Machine Learning, 2013, pp. 127–135.

[5] Chu-Cheng Hsieh, James Neufeld, Tracy King, and Junghoo Cho, "Efficient approximate thompson sampling for search query recommendation," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 740–746.

[6] Dhruv Kumar Mahajan, Rajeev Rastogi, Charu Tiwari, and Adway Mitra, "Logucb: an explore-exploit algorithm for comments recommendation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 6–15.

[7] Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler, "Learning structured predictors from bandit feedback for interactive nlp," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1610–1620.

[8] Fabian Moerchen, Patrick Ernst, and Giovanni Zappella, "Personalizing natural language understanding using multi-armed bandits and implicit feedback," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2020, p. 2661–2668, Association for Computing Machinery.

[9] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[10] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 208–214.

[11] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li, "Online context-aware recommendation with time varying multi-armed bandit," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 2025–2034.

[12] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[13] Yingfei Wang, Chu Wang, and Warren Powell, "The knowledge gradient for sequential decision making with stochastic binary feedbacks," in *Proceedings of The 33rd International Conference on Machine Learning*, New York, New York, USA, 2016, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1138–1147, PMLR.