HANDLING LONG-TAIL QUERIES WITH SLICE-AWARE CONVERSATIONAL SYSTEMS

Cheng Wang, Sun Kim, Taiwoo Park, Sajal Choudhary
Amazon Alexa AI
{cwngam, kimzs, parktaiw, sajalc}@amazon.com

Sunghyun Park, Young-Bum Kim, Ruhi Sarikaya, Sungjin Lee
Amazon Alexa AI
{sunghyu, youngbum, rsarikay, sungjinl}@amazon.com

ABSTRACT

We have been witnessing the usefulness of conversational AI systems such as Siri and Alexa, directly impacting our daily lives. These systems normally rely on machine learning models evolving over time to provide quality user experience. However, the development and improvement of the models are challenging because they need to support both high (head) and low (tail) usage scenarios, requiring fine-grained modeling strategies for specific data subsets or slices. In this paper, we explore the recent concept of slice-based learning (SBL) (Chen et al., 2019) to improve our baseline conversational skill routing system on the tail yet critical query traffic. We first define a set of labeling functions to generate weak supervision data for the tail intents. We then extend the baseline model towards a slice-aware architecture, which monitors and improves the model performance on the selected tail intents. Applied to de-identified live traffic from a commercial conversational AI system, our experiments show that the slice-aware model is beneficial in improving model performance for the tail intents while maintaining the overall performance.

1 INTRODUCTION

Conversational AI systems such as Google Assistant, Amazon Alexa, Apple Siri and Microsoft Cortana have become more prevalent in recent years (Sarikaya, 2017). One of the key techniques in those systems is to employ machine learning (ML) models to route a user's spoken utterance to the most appropriate skill that can fulfill the request. This requires the models to first capture the semantic meaning of the request, which typically involves assigning the utterance query to the candidate domain, intent, and slots (El-Kahky et al., 2014). For example, "Play Frozen" can be interpreted with *Music* as the domain, *Play Music* as the intent, and *Album Name:Frozen* as the slot key and value. Then, the models can route the request to a specific skill, which is an application that actually executes to deliver an experience (Li et al., 2021). For commercial conversational AI systems, there usually exists a large-scale dataset of user requests with ground-truth semantic interpretations and skills (e.g., through manual annotations and hand-crafted rules or heuristics). Along with various contextual signals, it is possible to train ML models (e.g., deep neural networks) with high predictive accuracy in routing a user request to the most appropriate skill, which then can continue to optimize towards better user experience through implicit or explicit user feedback (Park et al., 2020).

Nevertheless, developing such ML models or improving existing ones towards better user experience is still challenging. One hurdle is the imbalance in the distribution of the user queries with a long tail in terms of traffic volume. This often makes it difficult for the ML models to learn the patterns from the long-tail queries, some of which could be for critical features. Several approaches have been proposed to address such imbalance issue (Smith et al., 2014; He et al., 2008; Chawla et al., 2002). However, they are mainly based on applying reverse-discriminative sampling strategies, for example, over-sampling minority and/or under-sampling majority. The sampling methods are usually insufficient in inspecting and improving model performance on pre-defined data subgroups.

In this work, we focus on the problem of imbalanced queries, specifically on tail but critical intents, in the context of the recently proposed slice-based learning (SBL) (Chen et al., 2019). SBL is a novel programming model that sits on top of ML systems. The approach first inspects particular data subsets (Ratner et al., 2019), which are called slices, and it improves the ML model performance on those slices. While the capability of monitoring specific slices is added to a pre-trained ML model (which is termed the *backbone* model), the approach has shown that overall performance across the whole traffic is comparable to those without SBL. Motivated by this idea, we propose to adopt the SBL concept to our baseline skill routing approach (we term the baseline model as $\mathbf{P}$; please refer to Sec. 3.1 for details) to improve its performance on tail yet critical intent queries while keeping the overall performance intact. First, we define slice functions (i.e., labeling functions) to specify the intents that we want to monitor. A pre-trained $\mathbf{P}$ is used as a backbone model for extracting the representation for each query. Then, we extend $\mathbf{P}$ to a slice-aware architecture, which learns to attend to the tail intent slices of interest.

We perform two experiments using a large-scale dataset with de-identified customer queries. First, we examine the attention mechanism in the extended model $\mathbf{P}$ with SBL. In particular, we test two attention weight functions with different temperature parameters in computing the probability distribution over tail intent slices. Second, we compare SBL to an upsampling method in $\mathbf{P}$ for handling tail intents. Our experiments demonstrate that SBL is able to effectively improve the ML model performance on tail intent slices as compared to the upsampling approach, while maintaining the overall performance.

We describe the related work in Section 2. In Section 3, we explain the baseline skill routing model, $\mathbf{P}$, and then elaborate how to extend it to a slice-aware architecture. The experiment results are reported in Section 4, and in Section 5, we discuss the advantages and potential limitations of applying SBL in our use case. We conclude this work in Section 6.

2 Related Work

2.1 Slice-Based Learning

Slice-based learning (SBL) (Chen et al., 2019) is a novel programming model that is proposed to improve ML models on critical data slices without hurting overall performance. A core idea of SBL is to represent a sample differently depending on the data subset or slice to which it belongs. It defines and leverages slice functions, i.e., pre-defined labeling functions, to generate weak supervision data for learning slice-aware representations. For instance, in computer vision (CV) applications, a developer can define object detection functions to detect whether an image contains a bicycle or not. In natural language understanding (NLP) applications, a developer can define intent-specific labeling functions such as for *Play Music* intent. SBL exhibits better performance than a mixture of experts (Jacobs et al., 1991) and multi-task learning (Caruana, 1997), with reduced run-time cost and parameters (Chen et al., 2019). Recently Gustavo et al. (Penha & Hauff, 2020) have employed the concept of SBL to understand failures of ranking models and identify difficult instances in order to improve ranking performance. Our work applies the idea to improve skill routing performance on low traffic but critical intents in conversational AI systems.

2.2 Weakly Supervised Learning

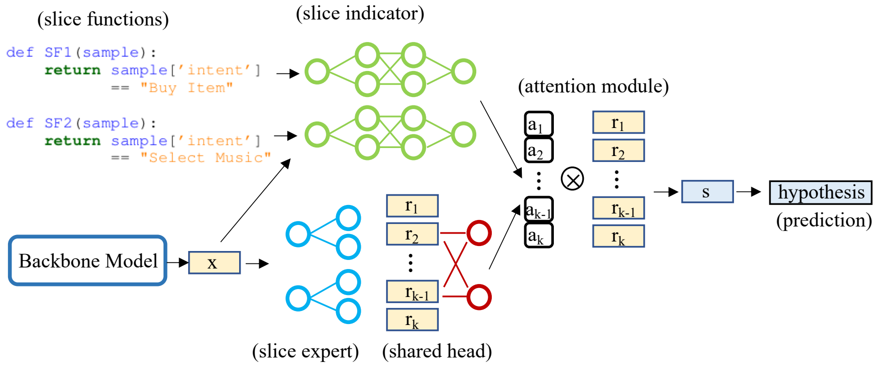
Weakly supervised learning attempts to learn predictive models with noisy and weak supervision data. Typically, there are three types of weak supervision: incomplete supervision, inexact supervision, and inaccurate supervision (Zhou, 2018). Various weakly supervised ML models are developed in NLP (Medlock & Briscoe, 2007; Huang et al., 2014; Wang & Manning, 2014) and in CV (Prest et al., 2011; Oquab et al., 2015; Peyre et al., 2017). Recently, promising approaches have been proposed to generate weak supervision data by programming training data (Ratner et al., 2016). In a large-scale industry setting, weak supervision data are highly desired given that human annotations are costly and time-consuming. Our work relates to weakly supervised learning in terms of inaccurate supervision. We split queries into different groups (slices) by defining labeling functions (slice functions). Each group is assigned with a group identity label. In practice, the slice functions may not perfectly assign labels to input data as mentioned in SBL (Chen et al., 2019; Cabannnes et al., 2020).

2.3 Conversational Skill Routing Models

In conversational AI systems, a skill refers to the application that actually executes on a user query or request to deliver an experience, such as playing a song or answering a question. The skills often comprise both first-party and third-party applications (Li et al., 2021). The skill routing is a mechanism that maps users' queries, given contextual information such as semantic interpretations and device types, to an appropriate application. The routing decision is usually determined by an ML model that is separate from typical natural language understanding (NLU) models for domain, intent, slot parsing. Please refer to section 3.1 for more details.

3 Slice-Aware Conversational Skill Routing Models

This section explains our skill routing model (backbone model) and then explains how we extend the backbone model to a slice-aware architecture by adapting the concept from SBL (Chen et al., 2019).



```
def SF1(sample):
    return sample[’intent’]
        == "Buy Item"

def SF2(sample):
    return sample[’intent’]
        == "Select Music"
```

Figure 1: The slice-aware conversational skill routing model architecture for handling low traffic but critical intents. It consists of six components: (1) *slice functions* define tail intent slices that we want to monitor; (2) *backbone model* is our pre-trained skill routing model $\mathbf{P}$ that is used for feature extraction; (3) *slice indicators* are membership functions to predict if a sample query belongs to a tail slice; (4) *slice experts* aim to learn slice-specific representations; (5) *shared head* is the base task predictive layer across experts; (6) An *attention module* is used to re-weight the slice-specific representations $\mathbf{r}$ and form a slice-aware representation $\mathbf{s}$. Finally, the learned $\mathbf{s}$ is used to predict a final hypothesis (associated skill). The predicted hypothesis is used to serve a user query.

3.1 Backbone Model

We take our baseline skill routing approach ($\mathbf{P}$) as the backbone model and aim to make it a slice-aware architecture. $\mathbf{P}$ is a skill routing model, which takes in a list of routing candidates to select the most appropriate one. Each routing candidate is represented as a hypothesis with various contextual signals, such as utterance text, device type, semantic interpretation, and associated skill. While some contextual signals are common across all hypotheses, some are unique due to the presence of multiple competing semantic interpretations and skill-specific context. The core component of $\mathbf{P}$ consists of attention-based bi-directional LSTMs (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2005) with fully connected layers on top of it. Formally, Let X be the set of query signals (e.g., utterance text, semantic interpretations, device type, etc.), $H = \{h_1, ..., h_n\}$ be the hypothesis list and $h_g \in H, g = [1, n]$ be the ground-truth hypothesis. The learning objective is to minimize the binary cross entropy:

$$\zeta_{base} = \mathcal{L}_{bce}(\pi(\mathcal{M}(X, H)), h_g), \tag{1}$$

```
def intent_based_slice_function_1(sample):
    return sample[’intent’] == "Buy Item"

def intent_based_slice_function_2(sample):
    return sample[’intent’] == "Select Music"

def intent_based_slice_function_3(sample):
    return sample[’intent’] == "Buy Book"
```

Table 1: The slice functions (SFs) which split user queries into multiple data slices according to the pre-defined tail intents. The non-tail intents are in a base slice. Note SFs are only available at training stage for generating weak supervision labels. At inference stage, SFs will not be applied.

where π is a linear predictive layer which outputs a prediction over hypotheses $\hat{H} = \{\hat{h}_1, ..., \hat{h}_n\}$, and $\mathcal{M}$ is a set of multiple neural network layers, which extract the representation $\mathbf{x} \in \mathbb{R}^{n\times d}$ for a given (X, H) pair, i.e., $\mathbf{x} = \mathcal{M}(X, H)$.

To evaluate the effectiveness of trained $\mathbf{P}$, we define offline evaluation metric called replication accuracy (RA):

$$RA(\mathcal{D}_{test}) = \sum\nolimits_{(X,H,h_g)\in\mathcal{D}_{test}} \frac{\mathbb{I}(\hat{h}_g = h_g)}{|\mathcal{D}_{test}|}. \tag{2}$$

The replication accuracy measures how effectively the trained model $\mathbf{P}$ replicates the current skill routing behavior in production which is a combination of ML model and rules. Though $\mathbf{P}$ achieves high performance, replicating most of heuristic patterns, it suffers from low RA in low-volume traffic, i.e., the tail user queries. We later introduce how we extend $\mathbf{P}$ with a slice-aware component.

3.2 Slice-Aware Architecture

As presented in Figure 1, a slice-aware architecture consists of several components.

Slice Function. We first define slice (or labeling) functions to slice user queries according to intent (e.g., “Buy Book”). The selected intents have a small number of query instances, making the model $\mathbf{P}$ difficult to learn data patterns from tail intents. Each sample is assigned a slice label $\gamma \in [0, 1]$ in $\{\gamma_1, \gamma_2, ..., \gamma_k\}$ for supervision. s_1 is the base slice, and s_2 to s_k are the tail slices.

Slice Indicator. For each tail intent slice, a slice indicator (membership function) is learned to indicate whether a sample belongs to this particular slice or not. For a given representation $\mathbf{x} \in \mathbb{R}^{n\times d}$ from the backbone model, we learn $u_i = f_i(\mathbf{x}; \mathbf{w}_i^f)$, $\mathbf{w}_i^f \in \mathcal{R}^{d\times 1}$, $i \in \{1, .., k\}$ that maps $\mathbf{x}$ to $\mathbf{u} = \{u_1, ..., u_k\}$. f_i is trained with $\{\mathbf{x}, \gamma\}$ pairs with the binary cross entropy $\zeta_{ind} = \sum_i^k \mathcal{L}_{bce}(\mathbf{u}_i, \gamma_i)$.

Slice Expert. For each tail intent slice, a slice expert $g_i(\mathbf{x}; \mathbf{w}_i^g)$, $\mathbf{w}_i^g \in \mathcal{R}^{d\times d}$ is used to learn a mapping from $\mathbf{x} \in \mathbb{R}^{n\times d}$ to a slice vector $r_i \in \mathcal{R}^d$ with the samples only belonging to the tail slice. Followed by a **shared head**, which is shared across all experts and maps r_i to a prediction $\hat{h} = \varphi(r_i; \mathbf{w}_s)$, g_i and φ are learned on the base (original) task with ground-truth label h_g by $\zeta_{exp} = \sum_i^k \gamma_i \mathcal{L}_{bce}(\hat{h}, h_g)$.

Attention Module. The attention module decides how to pay special attention to the monitored slices. The distribution over slices (or attention weights) are computed based on stacked k membership likelihood $P \in \mathbb{R}^k$ and stacked k experts' prediction confidence $Q \in \mathbb{R}^{k\times c}$ as described in (Chen et al., 2019):

$$a2 = \textsc{softmax}(P + |Q|). \tag{3}$$

Note, the above equation is used when $c = 1$ (i.e., binary classification). As our task is a multi-class classification task where $c \geq 2$, we use an additional linear layer to transform $Q \in \mathbb{R}^{k\times c}$ to $\phi(Q) \in \mathbb{R}^k$. Finally, we experiment with the following different ways to compute attention weights, i.e., slice distribution:

$$a1 = \textsc{softmax}(P/\tau) \tag{4}$$
$$a2 = \textsc{softmax}([P + |\phi(Q)|]/\tau). \tag{5}$$

In Eq. 4, we only use the output of the indicator function (membership likelihood) in computing attention weights. In Eq. 5 we use both the membership likelihood and the transformed experts' prediction scores. The τ is a temperature parameter. In principle, smaller τ can lead to a more confident slice distribution (Wang & Niepert, 2019; Wang et al., 2021), hence we aim to examine if a small τ helps improve the routing performance.

4 Experiments

We evaluate the skill routing model $\mathbf{P}$ with slice-based learning (SBL) (Chen et al., 2019) (we term it as $\mathbf{S}$) by performing two groups of experiments. First, we test the attention module with different methods of computing the attention weights over slices. Second, we compare the effectiveness of SBL against upsampling – a commonly used method for handling tail data.

4.1 Experiment Setup and Implementation Details

We obtained live traffic from a commercial conversational AI system in production and processed the data so that individual users are not identifiable. We randomly sampled to create an adequately large data set for each training and test dataset. We further split the training set into training and validation sets with a ratio of 9:1. We used the replication accuracy (Eq. 2) to measure the model performance.

The existing production model $\mathbf{P}$ and its extension with SBL were implemented with Pytorch (Paszke et al., 2019). The hidden unit size for slice component was 128. All models were trained on AWS p3.8xlarge instances with Intel Xeon E5-2686 CPUs, 244 GB memory, and 4 NVIDIA Tesla V100 GPUs. We used Adam (Kingma & Ba, 2014) with a learning rate of 0.001 as the optimizer. Each model was trained with 10 epochs with the batch size of 256. We split the user queries into 21 data slices in total, one base slice and the rest for 20 tail intent slices. For each extracted query representation $\mathbf{x}$ for the tail intents, we add a Gaussian noise $\mathbf{x} = \mathbf{x} + \delta,\ \delta \sim \mathcal{N}(0, 0.005)$ to augment the tail queries.

4.2 Experiments on the Attention Mechanisms

Table 2 shows the absolute score difference in replication accuracy between the baseline model and its SBL extension, having the baseline model's all-intent accuracy as a reference. As shown in the table, the slice-based approaches maintain the baseline performance overall, but the RA performance is lifted on the monitored tail slices. The best attention mechanism outperforms the baseline by 0.1% in tail intents' replication accuracy[1]. Tuning the temperature parameter between $\tau = 0.1$ or $\tau = 1.0$ does not significantly improve model performance on the tail intents.

Attention Methods	All Intents (%)	Tail Intents (%)
$\mathbf{P}$ (baseline model)	>99	–1.45
SBL, Eq. (4), $\tau = 1.0$	+0.01	–1.35
SBL, Eq. (5), $\tau = 1.0$	+0.01	–1.36
SBL, Eq. (4), $\tau = 0.1$	+0.01	–1.34
SBL, Eq. (5), $\tau = 0.1$	+0.01	–1.38

Table 2: The performance comparison of the baseline model $\mathbf{P}$ and its SBL extension with different attention weights in replication accuracy. All data points denote the absolute difference from the baseline model's all intents accuracy value.

[1] Given the large volume of query traffic per day, 0.1% is still a significant improvement in our system.

4.3 Comparison between Slice-Based Learning and Upsampling

As upsampling is a widely used method to alleviate the tail data problem, we compare the performance between SBL and upsampling methods. Note SBL offers an additional advantage for inspecting particular tail data groups which are also critical. We denote the models as the following:

- $\mathbf{P}$ is the baseline model that is trained without applying upsampling.
- $\mathbf{S}$ is an extension of $\mathbf{P}$ (as a backbone model) to be a slice-aware model, which is trained with same training set as $\mathbf{P}$.
- $\mathbf{P}_{up}$ is the baseline model that is trained with applying upsampling.
- $\mathbf{S}_{up}$ is an extension of $\mathbf{P}_{up}$ to be a slice-aware model, which is trained with same training set as $\mathbf{P}_{up}$.

All the trained models are evaluated on the same test set. Among the aforementioned attention method choices, Eq. 4 with $\tau = 1.0$ is employed for $\mathbf{S}$ and $\mathbf{S}_{up}$. Our primary goal is to see whether $\mathbf{S}$ can improve $\mathbf{P}_{up}$.

Table 3 shows the performance comparison. When comparing $\mathbf{P}_{up}$ and $\mathbf{S}$, we can see $\mathbf{S}$ achieves slightly better performance for all intents. For the monitored tail intents, $\mathbf{S}$ achieves a slightly higher score as compared to $\mathbf{P}_{up}$.

Models	All Intents (%)	Tail Intents (%)
$\mathbf{P}$	>99	–1.41
$\mathbf{P}_{up}$	0.00	–1.47
$\mathbf{S}$	+0.01	–1.30
$\mathbf{S}_{up}$	+0.01	–1.37

Table 3: Performance comparison between the baseline model and its slice-aware architecture. $\mathbf{P}$ is the baseline model without upsampling, $\mathbf{P}_{up}$ is $\mathbf{P}$ with upsampling. $\mathbf{S}$ is the slice learning model with $\mathbf{P}$ as the backbone model, and $\mathbf{S}_{up}$ is the slice learning model with $\mathbf{P}_{up}$ as the backbone model. All data points are absolute score difference from the baseline model's all intent accuracy value.

Table 4 presents the absolute RA difference between the baseline and slice-aware models for the monitored 20 tail intents. Comparing $\mathbf{S}$ and $\mathbf{P}_{up}$, $\mathbf{S}$ improves the model performance on 14 tail intents. Compared to $\mathbf{P}_{up}$, $\mathbf{S}$ shows the comparable performance lift while effectively suppressing performance drops, for example, intent IDs 2, 3, 6, 15, and 20. As a result, $\mathbf{P}_{up}$ shows lower performance on 12 intents out of 20 (–2.41% on average), while $\mathbf{S}$ did on only 5 intents (–0.21% on average). This suggests the capability of slice-based learning in treating target intents through the slice-aware representation.

5 Discussion

In our experiments, we have shown the effectiveness of SBL in terms of improving model performance on tail intent slices. It is beneficial to have ML models which are slice-aware, particularly when we want to inspect some specific and critical but low-traffic instances. Although the overall performance gain of slice-aware approach compared to the upsampling was marginal, it is worthwhile to note that the slice-aware approach was able to lift up the replication accuracy for more number of tail intents while minimizing unexpected performance degradation that was more noticeable in the upsampling approach. This result implies that the slice-aware approach has more potential in stably and evenly supporting tail intents.

On the other hand, we also note a potential limitation of SBL in the case of addressing tail intents in the industry setting. As we increase the number of tail intents, for instance to 200 intents, the model's complexity increases as well, given that an indicator function and an expert head are needed for each slice. However, this does not necessarily diminish the value of the slice-aware architecture, as the upsampling method offers no chance for us to inspect and analyze model failures on particular slices.

Tail Intent ID	$\mathbf{P}$	$\mathbf{P}_{up}$	$\mathbf{S}$	$\mathbf{S}_{up}$	Sample Size
1	>99	–0.03	0.00	–0.02	Over 10K
2	>96	–0.4	+0.04	–0.21	Over 10K
3	>96	–0.19	+0.09	–0.18	Over 10K
4	>72	+0.07	+1.98	+0.96	Over 10K
5	>99	+0.01	–0.01	0.00	Over 10K
6	>96	–0.09	+0.02	–0.11	Over 10K
7	>96	+0.03	+0.02	–0.04	Over 10K
8	>99	+0.15	+0.01	+0.19	Over 10K
9	>96	–0.24	+0.06	+0.07	Over 10K
10	>96	+0.08	–0.03	0.00	Between 1K - 10K
11	>99	0.00	0.00	0.00	Between 1K - 10K
12	>96	+0.55	–0.13	+0.46	Between 1K - 10K
13	>96	+0.36	+0.42	+0.53	Between 1K - 10K
14	>93	–0.39	–0.14	–0.42	Between 1K - 10K
15	>93	–3.29	–0.73	–1.46	Between 1K - 10K
16	>96	–1.2	0.00	–0.93	Below 1K
17	>96	–0.71	0.00	–0.71	Below 1K
18	>99	–0.16	0.00	–0.16	Below 1K
19	>99	–0.96	0.00	–1.15	Below 1K
20	>96	–21.21	0.00	–18.18	Below 1K

Table 4: Score (in %) differences in RA between the baseline and slice-aware approaches at the intent level. The baseline model's accuracy scores are rounded down to the nearest multiple of 3 percent, while the other models' are absolute score differences from the baseline ones. We denote each intent with their IDs. Sample Size is the number of random instances used for testing.

Further studies are necessary to employ and fine-tune the slice-based approach to serve tail traffic in a cost-effective way.

6 Conclusion

In this work, we applied and implemented the concept of slice-based learning to our skill routing model for a large-scale commercial conversational AI system. To enable the existing model to pay extra attention to selected tail intents, we tested different ways of computing slice distribution by using membership likelihood and experts' prediction confidence scores. Our experiments show that the slice-based learning can effectively and evenly improve model performance on tail intents while maintaining overall performance. We also compared the slice learning method against upsampling in terms of handling tail intents. The results suggest that slice-based learning outperforms upsampling by a small margin, while more evenly uplifting tail intents' performance. A potential future work would be to explore how to adapt SBL to monitor a large number of slices with minimum model and runtime complexity.

REFERENCES

Vivien Cabannnes, Alessandro Rudi, and Francis Bach. Structured prediction with partial labelling through the infimum loss. In *International Conference on Machine Learning*, pp. 1230–1239. PMLR, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. Slice-based learning: A programming model for residual learning in critical data slices. In *Advances in neural information processing systems*, pp. 9397–9407, 2019.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4067–4071. IEEE, 2014.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328. IEEE, 2008.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 40(1):85–120, 2014.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Han Li, Sunghyun Park, Aswarth Dara, Jinseok Nam, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. Neural model robustness for skill routing in large-scale conversational ai systems: A design choice exploration. *arXiv preprint arXiv:2103.03373*, 2021.

Ben Medlock and Ted Briscoe. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 992–999, 2007.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 685–694, 2015.

Sunghyun Park, Han Li, Ameen Patel, Sidharth Mudgal, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. A scalable framework for learning from implicit user feedback to improve natural language understanding in large-scale conversational ai systems. *arXiv preprint arXiv:2010.12251*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Gustavo Penha and Claudia Hauff. Slice-aware neural ranking. In *Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI)*, pp. 1–6, 2020.

Julia Peyre, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Weakly-supervised learning of visual relations. In *Proceedings of the ieee international conference on computer vision*, pp. 5179–5188, 2017.

Alessandro Prest, Cordelia Schmid, and Vittorio Ferrari. Weakly supervised learning of interactions between humans and objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):601–614, 2011.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567–3575, 2016.

Alexander J Ratner, Braden Hancock, and Christopher Ré. The role of massively multi-task and weak supervision in software 2.0. In *CIDR*, 2019.

Ruhi Sarikaya. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81, 2017.

Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine learning*, 95(2):225–256, 2014.

Cheng Wang and Mathias Niepert. State-regularized recurrent neural networks. In *International Conference on Machine Learning*, pp. 6596–6606, 2019.

Cheng Wang, Carolin Lawrence, and Mathias Niepert. Uncertainty estimation and calibration with finite-state probabilistic {rnn}s. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9EKHN1jOlA`.

Mengqiu Wang and Christopher D Manning. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2: 55–66, 2014.

Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1): 44–53, 2018.