# A Multitask Active Learning Framework for Natural Language Understanding

**Hua Zhu**[*], **Wu Ye**[*], **Sihan Luo**[*], **Xidong Zhang**[*]
vivo Mobile Communication Co., Ltd
{zhuhua.bjai, yewu.aiyjy, sihan.luo, xidong.zhang}@vivo.com

## Abstract

Natural language understanding (NLU) aims at identifying user intent and extracting semantic slots. This requires sufficient annotating data to get considerable performance in real-world situations. Active learning (AL) has been well-studied to decrease the needed amount of the annotating data and successfully applied to NLU. However, no research has been done on investigating how the relation information between intents and slots can improve the efficiency of AL algorithms.

In this paper, we propose a multitask AL framework for NLU. Our framework enables pool-based AL algorithms to make use of the relation information between sub-tasks provided by a joint model, and we propose an efficient computation for the entropy of a joint model. Experimental results show our framework can achieve competitive performance with less training data than baseline methods on all datasets. We also demonstrate that when using the entropy as the query strategy, the model with complete relation information can perform better than those with partial information. Additionally, we demonstrate that the efficiency of these active learning algorithms in our framework is still effective when incorporate with the Bidirectional Encoder Representations from Transformers (BERT).

## 1 Introduction

Voice assistant becomes increasingly intelligent on the mobile phone. Given an utterance from the user, the task of natural language understanding (NLU) aims at identifying the user's intent and extracting semantic information. This can help a voice assistant to convert the utterance to an executable instruction. The task can be separated as two sub-tasks called intent detection (ID) and slot filling (SF) (Tur and De Mori, 2011). Table 1 shows an example of ID and SF using query "*Flights from Ontario to Orlando*".

| Query | Flights | from | Ontario | to | Orlando |
|---|---|---|---|---|---|
| Slots | O | O | B-fromloc | O | B-toloc |
| Intent | atis_flight | | | | |

Table 1: example from user query to semantic frame

Traditional pipeline models solved the task individually (Haffner et al., 2003; Raymond and Riccardi, 2007; Yao et al., 2014). Considering possible error propagation effect caused by pipeline method, the joint model has been proposed to exploit the relation information between ID and SF (Liu and Lane, 2016; Goo et al., 2018; Chen et al., 2019a; Qin et al., 2019).

Sufficient data becomes another bottleneck in real-world situations. Both ID and SF need a considerable amount of training data to get a satisfying performance. Acquiring annotated data is trivial and expensive. The Active Learning (AL) method offers a promising solution to deal with this bottleneck (Settles and Craven, 2008; Settles, 2009). It allows the model to query certain unlabeled samples to annotate for training. For multitask framework, researchers proposed methods to select samples considering



[*]Equal contributions.



*Proceedings of the 28th International Conference on Computational Linguistics*, pages 4900–4914
Barcelona, Spain (Online), December 8-13, 2020

task priority or dependency (Reichart et al., 2008; Zhang, 2010; Fang and Tao, 2015). In recent years, researchers tried to apply AL algorithms into NLU task. Peshterliev et al. (2019) proposed a CRF-based ensemble AL method for detecting samples from a new domain. Chen et al. (2019b) focused on ID and proposed a method based both on QBC (Seung et al., 1992) and uncertainty sampling (Lewis and Gale, 1994). Dimovski et al. (2018) introduced the submodularity-inspired selection method for SF. However, to the best of our knowledge, most existing researches of AL for NLU did not make use of the relation information between ID and SF from a joint model. This motivates us to build a multitask AL framework for NLU to select samples that can benefit ID and SF jointly. Our contributions in this paper are listed as follows:

- We propose a multitask AL framework for NLU. We employ a joint model in our framework to model the relation information between ID and SF, and to provide necessary marginal distribution for pool-based AL algorithms.
- We implement representative pool-based AL algorithms including Least Confidence (Settles, 2009), Margin Sampling (Scheffer et al., 2001) and Entropy (Shannon, 1948) under the multitask scenario. We also propose an efficient computation for the entropy of a joint-model by DP.
- Through simulated experiments, we show that our AL framework can achieve competitive performance with less training data than baseline methods on all datasets.
- We demonstrate that these AL algorithms keep efficient when the joint model is changed to the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019).

## 2 Related Work

**Active learning (AL):** AL is a method to improve the performance of a model using as few labeled samples as possible (Settles, 2009). A famous scenario is pool-based AL, which uses different sampling strategies to evaluate informativeness over a large pool of unlabeled data.

The main categories are summarized in (Settles, 2009), which introduced many different sampling methods developed for pool-based AL, along with their application on various tasks. Expected Gradient Length (EGL) (Settles et al., 2008) and Expected Error Reduction (EER) (Roy and McCallum, 2001) are pool-based AL frameworks based on decision-theoretic methods. EGL aims to select utterances that cause the largest change in new training gradient, and EER tries to select samples which can reduce the expected future error maximally. However, these two methods are computationally expensive in NLU scenario because of a large feature space. Seung et al. (1992) studied another sample selection framework: Query-By-Committee (QBC). It determines the query uncertainty by a committee of models and selects informative samples with most disagreement in the committee. Since it needs a committee of models that are all trained on the labeled set, this will take a lot of time to do training and also need extra memory. Our work mainly focuses on estimating uncertainty sampling which is the simplest and most commonly query strategy algorithm in NLU.

**Natural language understanding (NLU):** According to whether ID and SF are separated in the model, NLU models can be categorized as independent modeling methods and joint modeling methods. Independent methods include hierarchical attention network (Yang et al., 2016), adversarial multitask learning framework (Liu et al., 2017) which mainly focused on ID. Others for SF include encoder-labeler deep LSTM (Kurata et al., 2016), joint pointer and attention model (Zhao and Feng, 2018).

As for the joint model, recent papers mainly construct neural network (NN) models to make use of the relation information between ID and SF, which include the slot-gated model with attention (Goo et al., 2018), joint BERT (Chen et al., 2019a), bi-directional interrelated model (E et al., 2019) and stack-propagation method (Qin et al., 2019). Additionally, Jeong and Lee (2006) developed a joint NLU model with a triangular-chain conditional random field (TriCRF), which is a unified probabilistic model combining two related problems. Xu and Sarikaya (2013) proposed a CNN version of the TriCRF model

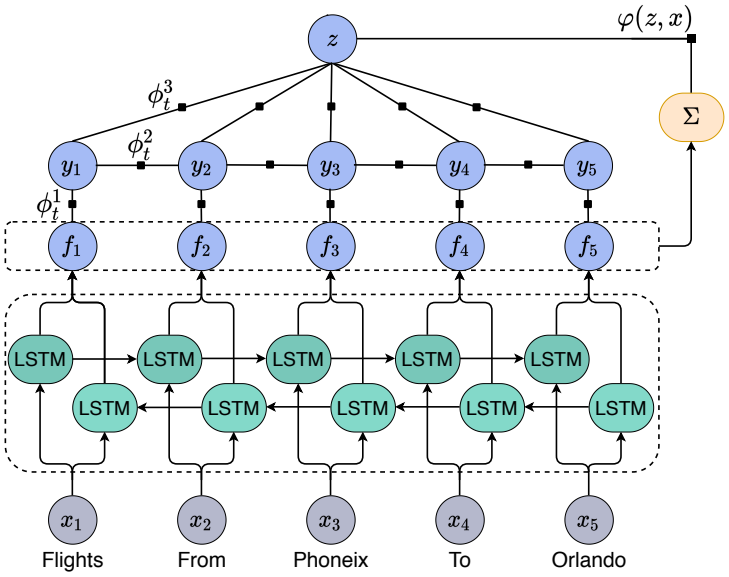


Figure 1: BiLSTM-TriCRF model

to extract features and share with ID and SF.

**Multitask Active Learning (MTAL):** MTAL is proposed to select samples with more information to annotate for all tasks involved. Reichart et al. (2008) focused on using rank combination protocol to rank tasks and then choose the best samples for the chosen task, which can keep the performance for a set of tasks close to single selection for every task. In multitask active learning setting, this method did not leverage inter-task association for selecting instances.

In recent years, several works have been done to evaluate the possibility of applying active learning methods on the NLU task. Chen et al. (2019b) proposed an AL method for automatically selecting the most informative labeling, which mainly focused on intent identification, and Dimovski et al. (2018) introduced the submodularity-inspired selection AL method and apply this selection criteria to the problem of slot filling. Peshterliev et al. (2019) targeted AL for new domains in NLU and explored the majority-CRF algorithm to select live utterances which achieved a statistically significant improvement compared to random sampling and traditional AL methods. Although their method selected samples considering the information from ID and SF at the same time, it did not utilize the inter-task relationship between two tasks.

## 3 Multitask Active Learning for NLU

Our main goal is to enable AL algorithms to select samples for both tasks simultaneously. In order to achieve this, the model should be able to directly provide the joint probability of intents and slots given the input sequence. This joint probability contains the aforementioned relation information for ID and SF. Any model that can provide above components can directly fit into our framework.

For the above reasons, we build our joint model from (Jeong and Lee, 2008) and (Huang et al., 2015) as shown in Figure 1. This section briefly introduce the necessary components in our joint model, and show how to apply AL algorithms by these components.

**BiLSTM-TriCRF joint model**: Jeong and Lee (2008) introduced Triangular Chain Conditional Random Field (TriCRF) for NLU joint model. TriCRF jointly represents the slots and intents in probabilistic graphic model, which can encode their dependencies and uncertainty.

In TriCRF, the potential of input sequence $\mathbf{x} = (x_1, ..., x_T)$, slot sequence $\mathbf{y} = (y_1, ..., y_T)$ and intent $z$ is described as

$$\phi_t(z, y_t, y_{t-1}, \mathbf{x}) = \phi_t^1(y_t, \mathbf{x}) \cdot \phi_t^2(y_t, y_{t-1}) \cdot \phi_t^3(z, y_t) \quad (1)$$

where $\phi_t^1(y_t, \mathbf{x})$, $\phi_t^2(y_t, y_{t-1})$, $\phi_t^3(z, y_t)$ respectively denotes the local potentials of input sequence $\mathbf{x}$ and

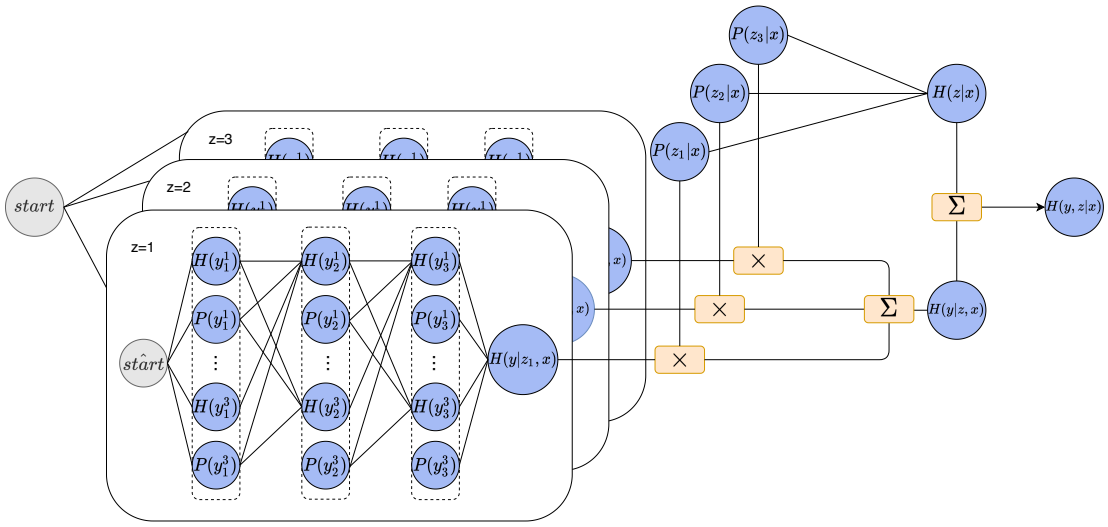


Figure 2: Structure for the recursive computation of Joint Entropy $H(\mathbf{y}, z|\mathbf{x})$, the intermediate entropies $H(y_i^j)$ represents $H(y_{i-1}|y_i = j, z = k, \mathbf{x})$, if $i = 1$, $y_{i-1} = starttag$.

slot $y_t$, state transition $y_t$ and $y_{t-1}$, and indent $z$ and slot $y_t$. The conditional distribution $p_\theta(z, \mathbf{y}|\mathbf{x})$ is defined as

$$p_\theta(z, \mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Big( \phi_t(z, y_t, y_{t-1}, \mathbf{x}) \Big) \cdot \varphi(z, \mathbf{x}) \tag{2}$$

where $Z(x)$ is defined to ensure that the target distribution is normalized, as $Z(x) \triangleq \sum_{y,z} \prod_{t=1}^{T} \big(\phi_t(z, y_t, y_{t-1}, \mathbf{x})\big) \cdot \varphi(z, \mathbf{x})$, and $\varphi(z, \mathbf{x})$ is the potential of input sequence $\mathbf{x}$ with intent $z$.

Referring to (Huang et al., 2015)'s work, the output of BiLSTM $F = (f_1, ...f_T)$ contains the transition from input to slot at time step $t$. A transition score matrix $[A]_{i,j}$ means transition from i-th slot to j-th slot for a pair of consecutive time steps. We use an intent-slot transition matrix $[B]_{i,j}$ to contain the transition from i-th intent to j-th slot. Therefore, $\phi_t^1(y_t, \mathbf{x})$, $\phi_t^2(y_t, y_{t-1})$ and $\phi_t^3(z, y_t)$ were computed as:

$$\begin{aligned} \phi_t^1(y_t, \mathbf{x}) &= e^{\lambda f_t} \\ \phi_t^2(y_t, y_{t-1}) &= e^{[A]_{i,j}} \\ \phi_t^3(z, y_t) &= e^{[B]_{i,j}} \end{aligned} \tag{3}$$

where $\lambda$ is the parameter to be learned. In our model, the feature $I$ is computed as $I = \sum_i^T f_i$ to provide the intent information. Using $I$, $\varphi(z, \mathbf{x})$ is computed as

$$\varphi(z, \mathbf{x}) = e^{\lambda^1 I} \tag{4}$$

where $\lambda^1$ is the parameter to be learned. See Jeong and Lee (2008), Huang et al. (2015) for more details.

After getting the output $z$ and $y$ from equation(2), the loss of joint model is defined as

$$L_{joint} = -\ log \prod_{(\mathbf{x},\mathbf{y},z) \in trainset} p_\theta(z, \mathbf{y}|\mathbf{x}) \tag{5}$$

where $\theta = (\lambda, \lambda^1, A, B, W_{BiLSTM})$

**Query Strategies:** Our work mainly lies on the most common method-uncertainty sampling strategies. The best training data will have highest uncertainty in the learner's predictive distribution. We estimate the following general heuristics for NLU model. The first one is **Least Confidence(LC)**(Settles, 2009)

$$\mathbf{x}_{LC}^* = \underset{\mathbf{x}}{\text{argmax}} \Big( 1 - p_\theta(\mathbf{y}^*, z^*|\mathbf{x}) \Big) \tag{6}$$

Where $(\mathbf{y}^*, z^*) = \underset{\mathbf{y},z}{\text{argmax}}\ p_\theta(\mathbf{y}, z|\mathbf{x})$ is the most likely sequence slots and intent for joint probability distribution.

**Algorithm 1** Calculate margin sampling for NLU

**Input:** $V \leftarrow$Viterbi Algorithm
$V2 \leftarrow$Viterbi Top2 Algorithm
$z = z_1, z_2 ... z_n \leftarrow$ all intents
$Argmax2 \leftarrow$Return two largest values and indexs in the input
**Output:** $top1score - top2score$

1: **for** $z = z_1, z_2, ..z_n$ **do**
2: calculate the most likely state sequence $ints$ for every intent $z_i$ using Viterbi Algorithm
3: **end for**
4: find two largest values $(int1s, int2s)$ and their intent indexes $(int1id, int2id)$ in the $ints$ using $Argmax2$
5: find the second largest joint probability $int1s2$ for $int1id$ using Viterbi Top2 Algorithm
6: the best joint probability in all intents $top1score$ is $int1s$
7: compare $int1s2$ with $int2s$, and the bigger one is the second largest joint probability in all intents $top2score$
8: **return** $top1score - top2score$

Scheffer et al. (2001) proposed another method called **Margin Sampling(M)**:

$$\mathbf{x}_M^* = \underset{\mathbf{x}}{\operatorname{argmin}} \left( p_\theta(\mathbf{y}_1^*, z_1^* | \mathbf{x}) - p_\theta(\mathbf{y}_2^*, z_2^* | \mathbf{x}) \right) \tag{7}$$

Here, $(\mathbf{y}_1^*, z_1^*), (\mathbf{y}_2^*, z_2^*)$ are the first and second best sequence slots and intents under the current model respectively. This method aims to select the instance with the smallest margin between the posteriors of its top two most likely sequence slots and intents. This instance can be considered as the learner has much doubt in distinguishing two possible intents and slot paths. When we select best two joint probabilities on sequence slots and intents, we need to consider the joint probability over all intents rather than considering the former two joint probability on one most possible intent.

$$\begin{aligned} \underset{\mathbf{y},z}{TOP2} \left( p_\theta(\mathbf{y}, z \mid \mathbf{x}) \right) &= \underset{\mathbf{y},z}{TOP2} \left( p_\theta(\mathbf{y} \mid z, \mathbf{x}) p_\theta(z \mid \mathbf{x}) \right) \\ &= \underset{z}{TOP2} \left( p_\theta(z \mid \mathbf{x}) \underset{\mathbf{y}}{TOP2}\, p_\theta(\mathbf{y} \mid z, \mathbf{x}) \right) \end{aligned} \tag{8}$$

Considering the increasing number of intents and sequence slots paths, we adopt the Algorithm 1 to accelerate the speed of calculation.

Another more common uncertainty-based method is **entropy(H)** (Shannon, 1948):

$$\mathbf{x}_H^* = \underset{\mathbf{x}}{\operatorname{argmax}} \left( -\sum_{\mathbf{y},z} p_\theta(\mathbf{y}, z | \mathbf{x}) log p_\theta(\mathbf{y}, z | \mathbf{x}) \right) \tag{9}$$

Where $(\mathbf{y}, z)$ ranges over all possible slots and intents for a sequence. This method represents the information needed to "encode" a joint distribution $(\mathbf{y}, z)$, and it can be thought as posteriors of our model over its slots and intents. In practice, since the number of possible labels increases sharply with the length of sequence, the quantity of candidate intents and labels combinations grows exponentially. Previous work by Kim et al. (2006) has employed **N-best Sequence Entropy (NSE)** that the entropy of the N parses with the highest probabilities. In our scenario, it can be rewritten as:

$$\begin{aligned} \mathbf{x}^* &= \underset{\mathbf{x}}{\operatorname{argmax}}\, H_{NSE}(\mathbf{y}, z \mid \mathbf{x}) \\ H_{NSE}(\mathbf{y}, z \mid \mathbf{x}) &= H(z \mid \mathbf{x}) + \sum_z p_\theta(z \mid \mathbf{x}) \cdot H_{NSE}(\mathbf{y} \mid z, \mathbf{x}) \\ H_{NSE}(\mathbf{y} \mid z, \mathbf{x}) &= -\sum_{\mathbf{y} \in N} p_\theta(\mathbf{y} | z, \mathbf{x}) log p_\theta(\mathbf{y} | z, \mathbf{x}) \end{aligned} \tag{10}$$

Where $N = (y_1^*, y_2^*, ...y_n^*)$, the set of the N-best state sequences in a certain intent. It is an approxima-

**Algorithm 2** Calculate joint entropy for NLU

**Input:** $z = z_1, z_2...z_n \leftarrow$ all intents
$n \leftarrow$the length of sequence
$PZ_Z \leftarrow$ marginal distribution for intent $z$ ; // The detail in Equation (15)
$PX_{z_i} \leftarrow$ partition function in one intent$z_i$ ; // The detail in Equation (16)
$TAGS = t_1, t_2....t_j, starttag, endtag \leftarrow$all possible tags value for **y**
**Output:** $H(\mathbf{y}, z \mid \mathbf{x})$
**for** $z = z_1, z_2, ..z_n$ **do**
  calculate intent entropy $intent\,entropy$ using $PZ_{z_i}$;
  **for** $TAGS = t_1, t_2....t_j$ **do**
    calculate $p_\theta(y_1 = t_k \mid z = z_i, \mathbf{x})$ and $H(\emptyset \mid y_1 = t_k, z = z_i, \mathbf{x})$;
  **for** $s = 1...n-1$ **do**
    **for** $t_m = TAGS = t_1, t_2....t_j$ **do**
      **for** $t_k = TAGS = t_1, t_2....t_j$ **do**
        $firstitem \mathrel{+}= H(y_s = t_k \mid y_{s+1} = t_m, z = z_i, \mathbf{x})$;
        $seconditem \mathrel{+}= p_\theta(y_s = t_k \mid y_{s+1} = t_m, z = z_i, \mathbf{x}) \cdot H(y_1...y_{s-1} \mid y_s = t_k, z = z_i, \mathbf{x})$;
      $H(y_1..y_s \mid y_{s+1} = t_m, z = z_i, \mathbf{x}) = firstitem + seconditem$;
    **for** $TAGS = t_1, t_2....t_j$ **do**
      $ynfirst \mathrel{+}= H_\theta(y_{s=n} = t_k \mid z = z_i, \mathbf{x})$;
      $ynsecond \mathrel{+}= p_\theta(y_{s=n} = t_k \mid z = z_i, \mathbf{x}) \cdot H(y_1...y_{s-1=n-1} \mid y_s = t_k, z = z_i, \mathbf{x})$;
    $H(y_1...y_{s=n} \mid z = z_i, \mathbf{x}) = ynfirst + ynsecond$;
    $sequence\,entropy \mathrel{+}= PZ_{z_i} \cdot H(y_1...y_{s=n} \mid z = z_i, \mathbf{x})$;
**return** $H(\boldsymbol{y}, z \mid \boldsymbol{x}) = sequence\,entropy + intent\,entropy$

tion to make computational entropy feasible. Mann and McCallum (2007) and Hernando et al. (2005) introduced an efficient calculation for true sequence entropy in linear-chain CRFs. We use this algorithm to derive Algorithm 2 to calculate the true Joint Entropy Equation (9) for our model:

$$\begin{aligned}
\mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} H(y_1, ..., y_T, z \mid \mathbf{x}) \\
H(y_1, ..., y_T, z \mid \mathbf{x}) &= H(z \mid \mathbf{x}) + \sum_z p_\theta(z \mid \mathbf{x}) \cdot H(y_1, ..., y_T \mid z, \mathbf{x}) \\
H(y_1, ..., y_T \mid z, \mathbf{x}) &= H(y_T \mid z, \mathbf{x}) + \sum_{y_T} p_\theta(y_T \mid z, \mathbf{x}) \cdot H(y_1, ...y_{T-1} \mid y_T, z, \mathbf{x}) \\
H(y_1, ..., y_{T-1} \mid y_T, z, \mathbf{x}) &= H(y_{T-1}|y_T, z, \mathbf{x}) + \sum_{y_{T-1}} p_\theta(y_{T-1} \mid y_T, z, \mathbf{x}) \cdot H(y_1, ..., y_{T-2} \mid y_{T-1}, z, \mathbf{x})
\end{aligned} \tag{11}$$

Using this decomposition, we can define a dynamic programming over entropy. We introduce the detailed computation and proof of Algorithm 2 in NLU to the Appendix A.

## 4 Experiments

**Dataset:** Schuster et al. (2019) contributed a dataset which contains 43k labelled English utterances across the three domains: Facebook-Weather, Facebook-Alarm, and Facebook-Reminder[1]. In our experiment, each domain is treated as an independent dataset. ATIS dataset (Tur et al., 2010) is the most commonly used dataset for NLU task, which is consisted of spoken queries on flight-related information [2]. The training, development and test sets contain 4478, 500 and 893 queries respectively. Snips dataset (Coucke et al., 2018) is collected from the Snips personal voice assistant[3]. The training, development and test sets, consist of 13084, 700 and 700 utterances, respectively. The size of the dataset is called $D_i$,

[1]https://fb.me/multilingual_task_oriented_data
[2]https://www.kaggle.com/siddhadev/ms-cntk-atis
[3]https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

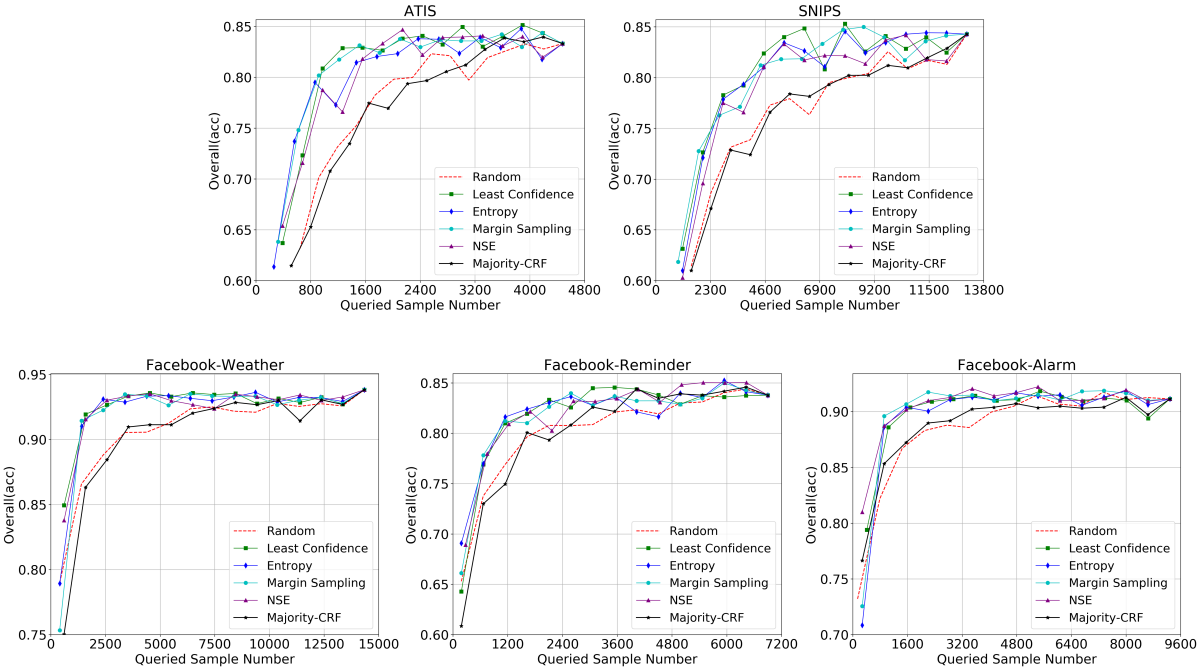


Figure 3: Overall accuracy of our framework and baseline framework on five datasets.

| Dataset | | ATIS | SNIPS | FB-Weather | FB-Reminder | FB-Alarm | Average |
|---|---|---|---|---|---|---|---|
| Size | | 4478 | 13084 | 14339 | 6900 | 9282 | - |
| Random | $Q_{needed}$ | 3843 | 11535 | 9803 | 4863 | 5403 | - |
| | Percentage | 85.8% | 88.2% | 68.4% | 70.5% | 58.2% | 74.2% |
| Majority-CRF | $Q_{needed}$ | 2691 | 9303 | 10403 | 4053 | 5808 | - |
| | Percentage | 60.1% | 71.1% | 72.6% | 58.7% | 62.6% | 65.0% |
| | Change | -25.7% | -17.1% | -4.2% | -11.8% | +4.4% | -9.20% |
| Least Confidence | $Q_{needed}$ | **1411** | **4281** | 2803 | **1983** | **1893** | - |
| | Percentage | **31.5%** | **32.7%** | 19.6% | **28.7%** | **20.4%** | **26.6%** |
| | Change | **-54.3%** | **-55.4%** | -48.8% | **-41.8%** | **-37.8%** | **-47.6%** |
| Entropy | $Q_{needed}$ | 2179 | 4839 | **2203** | 2523 | 2298 | - |
| | Percentage | 48.7% | 37.0% | **15.4%** | 36.6% | 24.8% | 32.5% |
| | Change | -37.1% | -51.2% | **-53.0%** | -33.9% | -33.4% | -41.7% |
| Margin | $Q_{needed}$ | 1859 | 4839 | 2603 | 2613 | 2163 | - |
| | Percentage | 41.5% | 37.0% | 18.2% | 37.9% | 23.3% | 31.6% |
| | Change | -44.3% | -51.2% | -50.2% | -32.6% | -34.9% | -42.6% |
| NSE | $Q_{needed}$ | 1859 | 5397 | 2603 | 2703 | 2298 | - |
| | Percentage | 41.5% | 41.3% | 18.2% | 39.2% | 24.8% | 33.0% |
| | Change | -44.3% | -46.7% | -50.2% | -31.3% | -33.4% | -41.2% |

Table 2: Detailed values for all query strategies on all evaluation dataset. For each dataset (column), the best result is shown in bold and the second best is shown underlined.

and the index $i$ is the name of the datasets.

**Framework Architecture** The hidden size of BiLSTM-TriCRF is 256 and the batch size is 512. Word embedding is implemented directly by PyTorch. We use Adam (Kingma and Ba, 2015) as the optimizer and the learning is set as 0.01. Every selected data consumes one annotating budget. We call the number of samples annotated every round as query size. For each dataset, the query size is set as $D_i/100$.

**Active Learning Baseline:** Two schemes are implemented and compared in our experiments: (a) pipeline model which does not consider relation information between ID and SF, and (b) passive random sampling, which randomly select samples from the unlabeled data pool.

We refer to Majority-CRF (Peshterliev et al., 2019) for building the pipeline model. Their framework uses Least Confidence as the basic query strategy and uses the QBC method to improve the quality of the query strategy. Additionally, they reported that their method can be reusable if the models are changed. We adjust several settings in this pipeline model to make it more suitable to compare in our experiment. The samples selected from the pipeline framework are then provided to the same NLU joint model in our framework.

**Framework Performance:** For each of the datasets, we use all data to test the performance of different active learning algorithms. All algorithms start training from a seed-set by random selection. For each dataset, different AL algorithms should eventually achieve the same performance since the data and the training model are the same. We treat the final overall accuracy as a performance baseline and record the least quantity of samples needed for different AL algorithms to achieve this baseline. We call this quantity as $Q_{needed}$ in our experiment. The percentage of $Q_{needed}$ on different dataset is computed as $Q_{needed}/D_i$. For different AL method, we compute the percentage change comparing to the random sampling. We notice that the accuracy curve does not monotonically increase when the amount of annotations increases. Therefore using $Q_{needed}$ may introduces randomness into the results. To mitigate this impact, we also use the area under curve (AUC) as a metric. Figure 3 shows the results on 5 datasets and Table 2 shows detailed values for $Q_{needed}$ , percentage and percentage change. Table 3 shows the detailed AUC value for all datasets in Appendix B.

From experiment results, $Q_{needed}$ and AUC show similar tendency on five datasets. All AL algorithms including Majority-CRF perform better than the random sampling on most datasets. Majority-CRF achieves about 9% percentage decrease on average, while our methods can achieve more than 40%. This indicates that our framework with the relation information between ID and SF does have a positive impact on these pool-based AL algorithms. We observe that Majority-CRF does not work better than Random Sampling on Facebook-Alarm dataset for $Q_{needed}$. A possible reason is that Majority-CRF is originally designed for acquiring data from a new domain. It is more probable to be influenced by the distribution of the dataset. The result also shows that the exact computation of entropy does bring better performance than the approximation, although it also costs more computation time.

**Influence of Multitask:** This experiment aims at investigating what difference would be made when MTAL considers only single task rather than multitask. We use Entropy as the basic query strategy. AL strategies here only consider slots or intents information for computation, and provide these samples to the same joint-model in our framework. We call them **Intent Entropy** and **Slot Entropy** in our experiments. Detailed computation is shown in Appendix A. Figure 4 shows the results on 3 datasets from Facebook.

From the figure, the results of three computation of entropies show little difference at the early stage of training. As the number of queried sample grows, the Intent Entropy show stabler tendency earlier than the Entropy and the Slot Entropy. After a certain point, the Entropy and the Slot Entropy can firmly achieve higher overall accuracy than the Intent Entropy. We notice that the Entropy and the Slot Entropy show almost the same performance on these datasets. According to the equation (25), the Entropy is approximated as the sum of the Intent Entropy and the Slot Entropy. With the growth of the sequence length, the amount of Slot Entropy also grows while the Intent Entropy almost stays the same. Therefore the Slot Entropy becomes the decisive part of the Entropy and shows similar performance as the Entropy. With the help of the Intent Entropy, the Entropy can outperform the Slot Entropy on three of all datasets according to the results. These results indicate that the completeness of information is crucial for the complete NLU task.

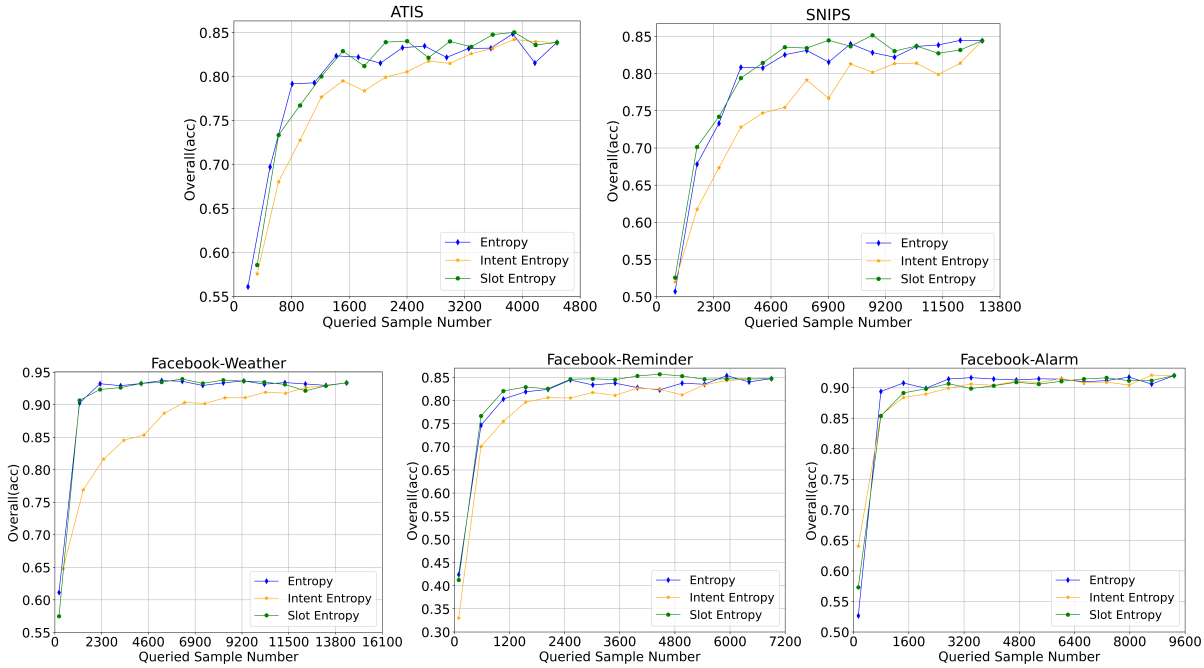


Figure 4: Overall accuracy of our framework with Intent Entropy, Slot Entropy and Entropy on five datasets.

**Influence of Different Base Model**: To demonstrate that our framework can still be effective when the base model is changed, we introduce BERT (Devlin et al., 2019) into our framework and conduct experiments on all datasets. The output of BiLSTM is replaced by the output of BERT and the feature I is now replaced by special embedding ($[CLS]$) to provide intent information. Other components keep the same as previous settings. The result is shown in Figure 5 in Appendix B.

From the figure, all AL methods still achieve $Q_{needed}$ earlier than the random sampling. Although BERT can provide richer semantic information and made the model get higher overall accuracy than previous model, it is still difficult for random sampling to keep the uptrend after the early stage. With the help of uncertainty sampling, the model can quickly get rid of this dilemma and achieve $Q_{needed}$. This indicates that our framework can still be effective when the base model is changed.

## 5 Conclusion

In this paper, we propose a multitask active learning framework for NLU that focuses on making use of the relation information between ID and SF. We implement representative pool-based query strategies that include Least Confidence, Margin Sampling and Entropy in our framework. We also perform an efficient computation for the entropy of a joint-model. Experimental results show above query strategies with our framework can achieve competitive performance with less training data than baseline methods on all datasets. The results also demonstrate that making use of the relation information between tasks may achieve better performance rather than only consider intents. Additionally, our framework is still useful when the model is changed. These results suggest that the framework has the potential to be applied for industrial use.

## References


Qian Chen, Zhu Zhuo, and Wen Wang. 2019a. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Xi C. Chen, Adithya Sagar, Justine T. Kao, Tony Y. Li, Christopher Klein, Stephen Pulman, Ashish Garg, and Jason D. Williams. 2019b. Active learning for domain classification in a commercial spoken personal assistant. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*.

Mladen Dimovski, Claudiu Musat, Vladimir Ilievski, Andreea Hossmann, and Michael Baeriswyl. 2018. Submodularity-inspired data selection for goal-oriented chatbot training based on sentence embeddings. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*.

Meng Fang and Dacheng Tao. 2015. Active multi-task learning via bandits. In *Proceedings of the 2015 SIAM International Conference on Data Mining*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE.

Diego Hernando, Valentino Crespi, and George Cybenko. 2005. Efficient computation of the hidden markov model entropy for a given observation sequence. *Information Theory, IEEE Transactions on*, 51:2681 – 2685, 08.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Minwoo Jeong and Gary Geunbae Lee. 2006. Jointly predicting dialog act and named entity for spoken language understanding. In *2006 IEEE Spoken Language Technology Workshop*, pages 66–69. IEEE.

Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302.

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. 2006. Mmr-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 69–72.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder LSTM for semantic slot filling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2083.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 685–689.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

Gideon S Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE.

Stanislav Peshterliev, John Kearney, Abhyuday Jagannatha, Imre Kiss, and Spyros Matsoukas. 2019. Active learning for new domains in natural language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 90–96.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL-08: HLT*.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079.

Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*.

Claude E Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24. IEEE.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*. IEEE.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 189–194. IEEE.

Yi Zhang. 2010. Multi-task active learning with output constraints. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Lin Zhao and Zhe Feng. 2018. Improving slot filling in spoken language understanding with joint pointer and attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 426–431.

## Appendix A Joint Intent And Slot Filling Entropy

We aim to calculate the joint entropy $H(Y, Z \mid X)$, where $Y = (y_1, y_2, .....y_t)$ and $Z = (z_1, z_2...z_j)$ range over all possible sequence slots and intents for input sequence $X$, additionally, $|x| = |Y| = t$ and each slot $y_i$ has its different possible state. Note, in order to efficiently calculate all subsets, we can factor the entropy given a linear-chain CRF of Markov order 1, because $y_{i+2}$ is independent of $y_i$ given $y_{i+1}$ (Mann and McCallum, 2007). According to the model introduction of TriCRF (Jeong and Lee, 2008), the above independence still holds. Now, recall the decomposition formulas for entropy firstly:

$$H(X, Y) = H(X) + H(Y \mid X) \tag{12}$$

$$H(Y \mid X) = \sum_x p(X = x) H(Y \mid X = x) \tag{13}$$

Secondly, $\alpha_t(z, y_t)$ is the forward value and $\beta_t(z, y_t)$ is the backward value in forward–backward recursions for TriCRF. If $y_0 = start$, the base cases of the forward value are defined as $\alpha_0(z, y_0) = 0$. Similarly, if $y_{t+1} = end$, we define $\beta_{t+1}(z, y_{t+1}) = 1$. In forward-backward algorithm, it also can be shown $\alpha_{t+1}(z, end) = \sum_z \alpha_t(z, t_t)$. The detail of forward and backward algorithm for TriCRF can be seen in (Jeong and Lee, 2008). Using the above formulas, we can decompose $H(Y, Z \mid X)$ as follow:

$$H(Y, Z \mid X) = H(Z \mid X) + \sum_z p_\theta(Z = z \mid X) \cdot H(Y \mid Z = z, X) \tag{14}$$

In (Jeong and Lee, 2008),

$$p_\theta(Z = z \mid X) = \frac{Z(z, X)}{Z(X)} \tag{15}$$

$$Z(z, X) = \alpha_{t+1}(z, end) \cdot \varphi(z, X) \tag{16}$$

$$Z(X) = \sum_z \alpha_{t+1}(z, end) \cdot \varphi(z, X) \tag{17}$$

Equation (14) needs range all potential intents. Now, consider a certain intent $z$ and compute $H(Y \mid Z = z, X)$. We can define a dynamic program over the entropy by the decomposition formulas of Equation (12) and Equation (13).For explicitly, we will denote $H(Y \mid Z = z, X)$ as $H(y_1, y_2, .....y_t \mid Z = z, X)$;

$$\begin{aligned} H(y_1, y_2, .....y_t \mid Z = z, X) &= H(y_t | Z = z, X) \\ &+ \sum_{y_t} p_\theta(y_t \mid Z = z, X) \cdot H(y_1, y_2, ...y_{t-1} \mid y_t, Z = z, X) \end{aligned} \tag{18}$$

$$H(y_t | Z = z, X) = -\sum_{y_t} p_\theta(y_t \mid Z = z, X) \cdot log p_\theta(y_t \mid Z = z, X) \tag{19}$$

According to the formulas in (Jeong and Lee, 2008),$p_\theta(y_t | Z = z, X)$ will be

$$p_\theta(y_t \mid Z = z, X) = \frac{\alpha_t(z, y_t) \cdot \beta_t(z, y_t)}{Z(z, X)} \cdot \varphi(z, X) \tag{20}$$

We further use Equation (13) to decompose the second part of Equation(18):

$$\begin{aligned} H(y_1, y_2, ...y_{t-1} \mid y_t, Z = z, X) =& H(y_{t-1} | y_t, Z = z, X) \\ &+ \sum_{y_{t-1}} p_\theta(y_{t-1} \mid y_t, Z = z, X) \cdot H(y_1, y_2, ...y_{t-2} \mid y_{t-1}, Z = z, X) \end{aligned} \tag{21}$$

We can do dynamic programming by iterating this formula, and the base case for this formula is $H(\emptyset \mid y_1, Z = z, X) = -\sum_{y_1} p_\theta(y_1 \mid Z = z, X) log\ p_\theta(y_1 \mid Z = z, X)$.The conditional probabilities $p_\theta(y_{t-1} \mid y_t, Z = z, X)$ can be factorized into the following

$$p_\theta(y_{t-1} \mid y_t, Z = z, X) = \frac{p_\theta(y_{t-1}, y_t, Z = z \mid X) \cdot p_\theta(X)}{p_\theta(y_t, Z = z \mid X) \cdot p_\theta(X)} \tag{22}$$

In(Jeong and Lee, 2008) work:

$$p_\theta\left(y_t, y_{t-1}, Z = z \mid\ X\right) = \frac{\alpha_{t-1}\left(z, y_{t-1}\right) \cdot \beta_t\left(z, y_t\right)}{Z(X)} \cdot \phi_t(z, y_t, y_{t-1}, X) \cdot \varphi(z, X) \tag{23}$$

Where $\phi$ and$\varphi$ are the potentials over triangular-chain graphs.
Therefore,$p_\theta(y_{t-1} \mid y_t, Z = z, X)$can be rewrited as:

$$p_\theta(y_{t-1} \mid y_t, Z = z, X) = \frac{\alpha_{t-1}\left(z, y_{t-1}\right)}{\alpha_t\left(z, y_t\right)} \cdot \phi_t(z, y_t, y_{t-1}, X) \tag{24}$$

Now, We can calculate Equation (18) by using Equation (24) to iteratively compute Equation (21).

In Experiment **Influence of Multitask**, the single entropy only consider slots or intents information compared to our joint entropy. We can directly compute single Intent Entropy $H(Z|X)$ with Equation (15) and single Slot Entropy $H(Y|X)$ can be computed using similar decomposition with $H(Y \mid Z = z, X)$ and Equation (24). For explicitly, we will denote $H(Y \mid X)$ as $H(y_1, y_2, .....y_t \mid X)$ :

$$\begin{aligned}
H(y_1, y_2, .....y_t \mid X) &= H(y_t \mid X) + \sum_{y_t} p_\theta(y_t \mid X) \cdot H(y_1, y_2, .....y_{t-1} \mid y_t, X) \\
H(y_1, y_2, .....y_{t-1} \mid y_t, X) &= H(y_{t-1} \mid y_t, X) + \sum_{y_{t-1}} p_\theta(y_{t-1} \mid y_t, X) \cdot H(y_1, y_2, ...y_{t-2} \mid y_{t-1}, X) \\
p_\theta(y_{t-1} \mid y_t, X) &= \sum_z p_\theta(Z = z \mid X) \cdot p_\theta(y_{t-1} \mid y_t, Z = z, X)
\end{aligned} \tag{25}$$

## Appendix B Experiment Results

| Dataset | ATIS | SNIPS | FB-Weather | FB-Reminder | FB-Alarm |
|---|---|---|---|---|---|
| Random | 0.875 | 0.872 | 0.920 | 0.934 | 0.961 |
| Majority-CRF | 0.876 | 0.876 | 0.921 | 0.940 | 0.964 |
| Least Confidence | <u>0.928</u> | **0.915** | **0.937** | **0.958** | 0.970 |
| Entropy | 0.927 | 0.907 | <u>0.936</u> | <u>0.957</u> | 0.973 |
| Margin | **0.929** | <u>0.910</u> | **0.937** | <u>0.957</u> | <u>0.975</u> |
| NSE | 0.917 | 0.902 | <u>0.936</u> | 0.949 | **0.978** |

Table 3: Detailed AUC values for all query strategies on all evaluation dataset. For each dataset (column), the best result is shown in bold and the second best is shown underlined.

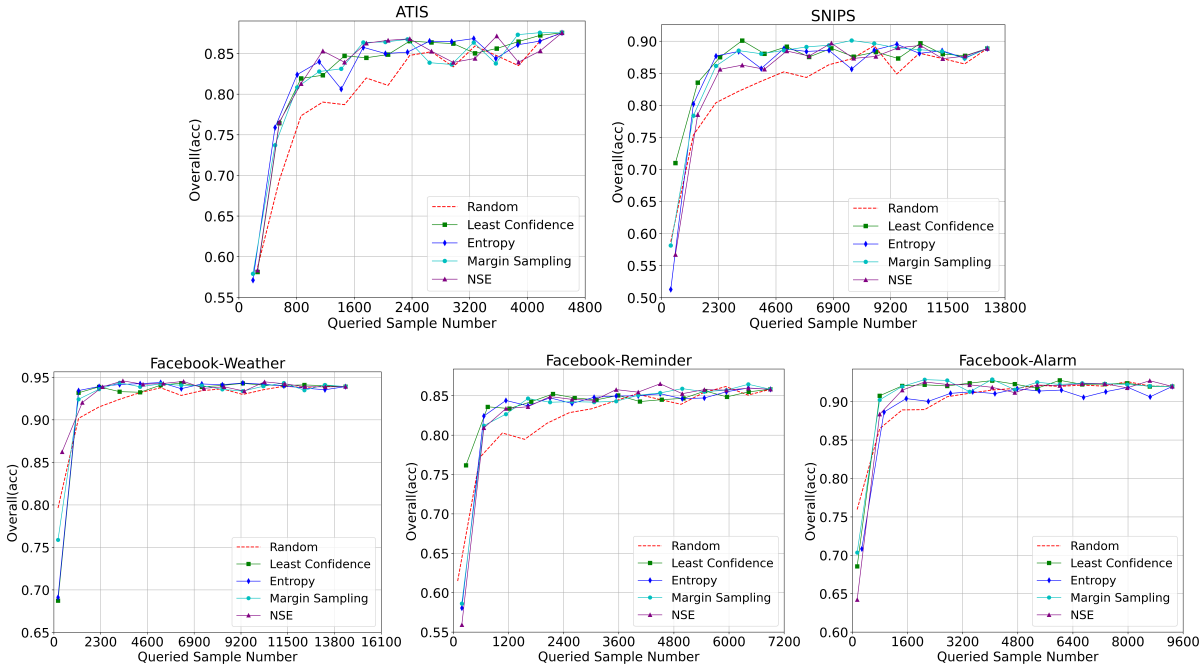


Figure 5: Overall accuracy of our framework with BERT on five datasets.