Comprehensive Study of (Neural-Based) Joint Slot-Filling & Intent-Detection Models on Task-Oriented Dialogue

Frederick Zhang, Sam Lin, Leslie Zeng, and Jihyung Kil

Abstract

Joint Model of Slot Filling (SF) and Intent Detection (ID) is a popular trend in recent Spoken Language Understanding (SLU) community. Being able to make correct predictions on *_UNK* and differentiate tricky intents and tricky templates has been a problematic issue in SF and ID tasks in modern SLU community. Therefore, we propose two categories of systems, non-architecture (lemmatization, UNK-enhanced, and tricky dataset) and architecture (BERT-based), based on existing models to solve the problems mentioned above. Experiments on two real-world datasets exhibit efficiency of our solutions: our proposed integrated joint model has achieved 87.1% on semantic accuracy, which is relatively 1.9% higher than the SOTA model.

1 Introduction

Spoken language understanding (SLU) is an emerging field in between speech and language processing, investigating human-machine and human-human communication by leveraging technologies from signal processing, pattern recognition, machine learning and artificial intelligence (Tur et al., 2011). SLU system typically involves identifying speaker's intent and extracting semantic constituents from the natural language query, two tasks that are often referred to as intent detection and slot filling (Liu et al., 2016). Slot filling can be treated as a domain-specific sequence tagging task that maps a sequence of input words $x=(x^1, \ldots, x^T)$ to the corresponding slot label sequence $y^S=(y^{S1}, \ldots, y^{ST})$, and intent detection can be seen as a domain-specific classification problem to decide the intent label y^I given x. For example, the sentence *what are the flights from montreal to chicago* is a sample retrieved from ATIS.

In early days, the existing works treated slot filling and intent detection as two separate tasks, and these two tasks were modeled isolatedly. However, in recent days, the joint model for the two tasks is becoming a more prevailing trend in SLU field. In this paper, our aim is to provide a comprehensive and basic error analysis by conducting experiments on three baseline models Joint Seq (Hakkani-Tür et al., 2016), Atten.-Based (Liu and Lane, 2016), and Sloted-Gated (Goo et al., 2018), and two novel models, Joint Model (Haihong E et al., 2019) and Capsule Model (Chenwei Zhang et al., 2018). For this project, we mainly focus on the Bi-directional Interrelated Model since this model already has consummate architectures, in which slot F1 and intent accuracy almost attain Bayes Risk, and compared with early baseline models, semantic accuracy is improved a lot. Meanwhile, we also mainly focus on ATIS because we do not have sufficient GPU resources. In this paper, we analyze and improve the models by investigating the brittleness of current system from both non-architecture and architecture approaches. Based on the analysis, we develop novel models by leveraging intuitive and straightforward ideas to boost up the current models' performance on the semantic accuracy of this joint task on both datasets. It has been shown that our proposed integrated model has outperformed recent SOTA model by relatively 2%.

2 Preliminary Work

Datasets: We analyze the task by using two real-world datasets ATIS and SNIPS (see table 1) but we are more focused on ATIS due to its smaller vocab size and train dataset size.

Dataset	ATIS	SNIPS
Vocab Size	722	11241
Avg. sent. length	11.28	9.05
#slots	124	72
#intents	22	7
Train size	4478	13084
Dev size	500	700
Test size	893	700

Table 1: Dataset

Total	Error
unk	33.8%
Tricky intent	14.6%
Tricky template	21.5%
Short sentence	6.9%
Unknown label	3.8%
others	19.2%

Table 2: Error Analysis

Error Analysis: In this part, we conduct basic error analysis of Bidirectional Interrelated Model (Joint Model) on ATIS, unk contributes ⅓ wrong predictions in total errors. Tricky intent and tricky template contribute another ⅓ to the error. Tricky intent is the problem that when slot predictions are 100% correct, intent prediction is still mis-predicted. Tricky template is the problem that the model can be confused by certain cases such as from/to location, depart/return location, city/state name, and etc. Our ultimate goal is to improve the semantic accuracy, based on the error analysis (see table 2 above).

Based on our analysis, we propose two types of systems to handle the brittleness of existing models, which are non-

architecture (illustrated in section 3) and architecture (illustrated in section 4) approaches.

3 Our System 1 (Non-architecture approach)

In this section, we propose four ways to tackle the current issues standing in existing novel models. They are Lemmatization, UNK-enhanced, Tricky Dataset and Integrated Model.

Lemmatization: Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma (Manning et al., 2008). For example,

- am, is, are => be
- car, cars, car's, cars' => car.

Recall the example we showed before, *what are the flights from montreal to chicago* will be converted to *what be the flight from montreal to chicago*.

Consequence of adding Lemmatization layer can help reduce the train vocab size from 722 to 660, meaning that 62 words in the original vocab list are simply the variants of their lemmas. More importantly, lemmatization also gets rid of some noisy words which are not encountered in training but the lemma of noisy words does show up in training. For example, (snack, snacks) pair:

- Training dataset: what is the earliest flight from boston to bwi that serves a snack
- Testing dataset: are snacks served on tower air

Last thing is that this method will not change the semantics but will make machine understand the text more easily.

UNK-enhanced:

The motivation of implementing the UNK-enhanced technique is due to the high error contribution of *_UNK* to mispredictions. The most intuitive way we come up with to resolve this issue is to enforce the model to directly learn how to handle *_UNK* during training phase. Inspired by the smart mask tricks applied by BERT (Devlin et al., 2018), we imitate what they did and extend their ideas to our UNK-enhanced technique.

The methodology is essentially similar to BERT. Before word sequences are fed into the model, they will first go through a UNK-enhanced layer, which will replace candidate tokens to *_UNK* token with pre-defined flip ratio. The model then will be enforced to predict the slot for the *_UNK* token by looking at its surrounding non-UNK words.

By introducing this extra layer, we also introduce three more parameters associated with this layer for fine-tune purpose.

- unk_ratio: the ratio of a candidate word to be flipped to *_UNK*.
- unk_threshold: the maximum number of UNK that can appear in one sentence.
- unk_candidate: the type of candidate tokens that can be flipped to *_UNK* (with three options)
    - Full: Every word can be candidate
    - Outside: Outside-tag word can be candidate
    - Entity: Entity-tag word can be candidate

sentence	what	be	the	flight	from	montreal	to	chicago
unk-full	what	be	the	flight	from	montreal	to	chicago
unk-entity	what	be	the	flight	from	_UNK	to	chicago
unk-outside	what	_UNK	the	_UNK	from	montreal	to	chicago
Slots	O	O	O	O	O	B-fromloc.city_name	O	B-toloc.city_name
intent	atis_flight							

Table 3: Examples of UNKed sentences under diff. options

Since the UNK-enhanced layer is fully stochastic/random, it is possible that the output sentence of the UNK-enhanced layer is the same as the original one, as you can see from the unk-full case in Table 3. One more example is the unk-entity option, so only *montreal* and *Chicago* can be flipped to be *_UNK*, and *montreal* that gets flipped in our case.

Tricky Dataset: We construct a tricky dataset and add it to ATIS training dataset in order to improve the model's performance. We build the dataset by duplicating each template by a factor of 6 and using a set of predefined entities to fill missing words. For example,

List me flights to Columbus

where *List me flights to* ___ is the template and *Columbus* is an entity word.

The first template is inspired by two papers PAWS dataset (Zhang et al., 2019) and the Cybercrime with slot filling/ NER (Durrett et al., 2017), which is designed to increase the model's ability to identify task-oriented entities in the input sentences. We exploit PAWS dataset idea of detecting severely-underrepresented patterns in the training dataset so we add those patterns to develop a more well-rounded training dataset. We detect that in the Cybercrime dataset, multiple products might be listed in the same sentence but only one of them is of their interest. In this case, they will only extract products that have been traded. Since ATIS does not contain this template, we create 10 templates containing noisy reference information in order to enforce the model to distinguish between relevant and irrelevant slots in an input sentence.

While all other flights from A to B arrive before 9 pm, why will my flight arrive after 10 pm?

In terms of information extraction, it would be more reasonable to only extract *after-10-pm flight* information though *before-9-pm flight* information is also present in the sentence.

The second template aims to reduce imbalance in the existing training data. Hence, we randomly select 500 samples from ATIS and extract the underrepresented templates such as *at the Chicago airport, are there any taxi services available*. Some sentences also contain misleading

intents such as *find me a list of airports in New York that has good ground transportation after midnight*. The real intent asks for a list of airports, but the model may be confused by the phrase *good ground transportation*.
The third template focuses on testing the model's ability to predict real-world inquiries made by travelers online. We extract templates from questions difficult to predict by searching websites including Expedia, Yelp, and more. After collecting 36 templates, we utilize template-based approach to generate 216 samples. All punctuations and capitalizations are removed to maintain the style of ATIS.
We prove the complexity of our tricky dataset by evaluating the model on the tricky dataset, and the evaluation score is 33.7% so the model does not well generalize to real-use cases. Then we test the model on tricky dataset combined with ATIS training dataset, and the semantic accuracy is increased to 86.9 which is higher than 85.5 (ATIS).

Integrated System: The final proposed model is the integrated model. As name suggests, we put all previously proposed ideas together and stack it on the Joint model (Joint + Lemma + UNK-enhanced + Tricky dataset).
The results will be shown later in table 4 that this integrated system has achieved 87.1% accuracy on semantic evaluation metric, which significantly improves the semantic accuracy by 1.9% as compared with current SOTA model.

4 Our System 2 (Architecture approach)

Naive approach (Glove): We first try to use Glove to check if Glove_embeeding improves the performance of two models, Joint model (Haihong E et al., 2019) and Capsule model (Chenwei Zhang et al., 2018). As a result, with Glove, the performance of both models is not improved for slot-filling and intent classification tasks.

Approach based on BERT: BERT (Bidirectional Encoder Representations from Transformers) is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Devlin et al., 2018). It can be fine-tuned with one additional output layer to create state-of-the-art models for various tasks, such as question answering or language inference, without task-specific architecture modifications.
Based on (Chen et al., 2019), we fine-tune BERT on three tasks, 1) intent classification, 2) slot filing and 3) jointly intent classification and slot filling. For intent-classification, based on the hidden state of the special token ([CLS]), denoted h_1, the intent is predicted as:

$$y^i = \text{softmax}(W^i h_1 + b^i)$$

We follow BERT code from (Xiao et al, 2018) for this task.
For slot filing, we feed the final hidden states of other tokens $h_2, \ldots, h_T$ into a softmax layer to classify over the slot filling labels.

$$y_n^s = \text{softmax}(W^s h_n + b^s), n \in 1 \ldots N$$

where h_n is is the hidden state corresponding to the first sub-token of word x_n
We follow the code from github.com/kyzhouhzau/BERT-NER for the slot filing.
To jointly model intent classification and slot filling, the objective is formulated as:

$$p(y^i, y^s|x) = p(y^i|x) \prod_{n=1}^{N} p(y_n^s|x)$$

We maximize this conditional probability, equivalent to minimizing the cross-entropy loss. The code from (Chen et al., 2019) is used for this task.
For slot filling on ATIS dataset, BERT model trained on jointly slot filling and intent classification yields the best performance (96.5). For intent classification, Joint + Lemma + Unk-entity model has the highest accuracy (97.7). For sentence-level semantic accuracy, Joint + Tricky + Lemma + Unk-entity model reaches the best result (86.8). However, in (Chen et al., 2019), its model achieved the highest score 88.2.

Joint-BERT Model: Although fine-tuning BERT improves the result, we think that combining two models, BERT and Joint model may improve the result even more. We have not implemented this yet but the approach is to train two models simultaneously on both slot filling and intent classification tasks. The basic idea behind this approach is to employ the powerfulness of two models, 1) using iteration mechanism to enhance the bi-directional interrelated connections between slot-filling and intent classification tasks from joint model, 2) learning contextualized word representation from BERT.

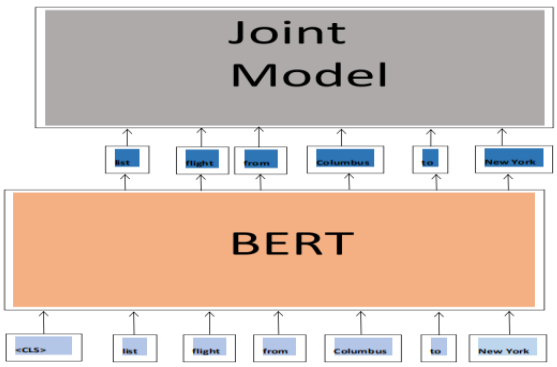

Figure 1: Our proposed Joint-BERT model (future work)

5 Experiments

Setup: The majority of experiments are run with the improved model, which is based upon Bi-directional Interrelated Model, on the pre-splitted ATIS datasets and ATIS with Tricky datasets.
All our models are trained on a single GPU in Linux Environment. In our experiments, we use Tesla P100-PCIE-16GB GPU with compute capability 6.0, Python 3.6 and different versions of Tensorflow (1.5: Capsule Model; 1.11: Joint Model; 1.14: BERT-based).

Result:

This section will show how our methods help to improve the two novel models as we discussed in section 3 and 4. Table 4 below shows the performance of Joint Model, our improved Joint-based models and BERT-based models on ATIS dataset.

		slot	intent	acc
Baseline Models	Joint seq	94.3	92.6	80.7
	Atten based	94.2	91.1	78.9
	Sloted-gated	95.4	95.4	83.7
Novel Models	Joint	95.4	96.7	85.5
	Capsule	94.5	94.3	81.4
Lemmatization	Joint+Lemma	95.7	97.0	86.2
unk-enhanced	Joint+Lemma+ unk-full	95.9	96.3	86.5
	Joint+Lemma+unk-entity	95.4	97.7	86.5
	Joint+Lemma+unk-outside	95.8	97.4	85.4
Tricky dataset	Joint+Lemma+ tricky	95.9	97.4	86.9
Integrated model	Joint+Lemma+ unk-entity	95.7	**97.9**	**87.1**
BERT-Based	BERT-Intent	/	97.4	/
	BERT-Slot	95.4	/	/
	BERT-Joint	**96.5**	96.2	79.2

Table 4: Results of Joint Model on ATIS dataset

Table 5 below shows the performance of Capsule Model and our improved Capsule-based models on ATIS and SNIPS datasets

		ATIS			SNIPS		
		slot	intent	acc	slot	intent	acc
Previous Models	Joint-seq	94.3	92.6	80.7	87.3	96.9	73.2
	Atten-based	94.2	91.1	78.9	87.8	96.7	74.1
	Sloted-gated	**95.4**	**95.4**	**83.7**	89.3	96.9	76.4
Novel Model	Capsule	94.5	94.3	81.4	91.9	97.4	82.9
Lemma -tization	Capsule+ Lemma	94.6	94.6	82.0	**92.3**	98.3	**83.1**
unk_ enhanced	Capsule+ Lemma+unk _entity	94.8	93.8	81.2	92.2	**99.0**	80.1

Table 5: Results of Capsule Model on ATIS/SNIPS dataset

The following table shows our suggested fine-tune hyperparameter setting for the UNK-enhanced layer (exclusively for Joint Model).

	Full	Entity	Outside
ratio	0.2	0.15	0.15
Threshold	5-6	4-5	4-5

Table 6: Fine-tune hyperparameter setting

Further analysis: Though we have made promising progress in terms of the better semantic accuracy, our proposed model still has a large room for improvement. In fact, our non-architecture approach implicitly and inherently has some limitations.

UNK-enhanced method we adopt cannot really balance Slot F1, Intent accuracy, and semantic accuracy (see table below). Additionally, unk_entity approach always does harm on Slot F1 score (see table 7 and 8). Table 7 shows the comparison between unk_full and unk_entity, and Table 8 shows the comparison between with and without unk_entity. The possible explanation can be that there are too many tags to learn (128 different slot tags in ATIS), and masking off some words does not really help learn the mapping between input words and huge slot tag space.

	slot	intent	acc
Joint+Lemma+ unk_full	**95.9**	96.3	**86.5**
Joint+Lemma+ unk_entity	95.4	**97.7**	**86.5**
Joint+Lemma+ unk_outside	95.8	97.4	85.4

Table 7: Comparison A

	slot	intent	acc
Joint+Lemma+ tricky	**95.9**	97.4	86.9
Joint+tricky+Lem-ma+unk_entity	95.7	**97.9**	**87.1**

Table 8: Comparison B

Tricky dataset also has some downsides since our annotation style is not professional because our annotators are not experienced. Also, it is a purely template-based approach which does not substantially increase language abundance.

6 Conclusion

In this paper, we comprehensively examine Bidirectional Interrelated Model (Joint Model) and Capsule Neutral Network Model (Capsule Model) on ATIS and SNIPS datasets and improve those models by using our implemented non-architecture and architecture methods. Overall, we think that our proposed approaches (lemmatization, UNK-enhanced, tricky dataset, and BERT-based) are effective in tested models, among which the integrated model attains the highest score, 87.1% on semantic accuracy evaluation metric.

In our future work, we need to consider how to utilize our non-architecture and architecture methods on SNIPS to help solve more complex Slot Filling and Intent Detection tasks. Also, we will think about how to effectively combine our Integrated model and BERT layer.

References

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. *arXiv preprint arXiv:1908.11828*.

Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Rebecca S. Portnoff, Sadia Afroz, Damon McCoy, Kirill Levchenko, and Vern Paxson. 2017. Identifying Products in Online Cybercrime Marketplaces: A Dataset for Fine-grained Domain Adaptation. *In Proceeding of EMNLP'17.*

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A Novel Bi-directional Interrelated Model for Joint Intent Detection. *arXiv preprint arXiv:1907.00390.*

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *CoRR*, abs/1812.09471.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and YunNung 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)*, volume 2, pages 753–757.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454.*

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

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and YeYi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.