# A Neural Framework for Joint Prediction on Intent Identification and Slot Filling

Jiawei Shan[1], Huayun Xu[1], Zeyang Gong[1], Hanchen Su[1], Xu Han[2], and Binyang Li[1](✉)

[1] School of Information Science and Technology, University of International Relations, Beijing, China
{jwshan, hyxu, zygong, hcsu, byli}@uir.edu.cn

[2] College of Information Engineering, Capital Normal University, Beijing, China
hanxu@cnu.edu.cn

**Abstract.** In task-oriented dialog systems, understanding of users' queries (expressed in natural language) is a process of parsing users' queries and converting them into some structure that machine can handle. The understanding usually consists of two parts, namely intent identification and slot filling. To address this problem, we propose a neural framework, named SI-LSTM, that combines two tasks and integrates CRF into LSTM network, where the slot information is extracted by using CRF, and the intent will be identified by using LSTM. In our approach, the slot information is used for determining the intent, while the intent type is used to rectify the slot filling deviation. Based on the dataset provided by NLPCC 2018, SI-LSTM achieved 90.71% on intent identification, slot filling and error correction in terms of accuracy.

**Keywords:** Deep learning · Intent identification · Slot filling

## 1 Introduction

In task-oriented dialog systems, understanding of users' queries (expressed in natural language) is a process of parsing users' queries and converting them into some structure that machine can handle. The understanding usually consists of two parts, namely intent identification and slot filling. The textual strings, fed into a dialog system as input, are mostly the transcripts translated from spoken language by ASR (Automatic Speech Recognition) and thus subject to recognition errors.

Intent identification is to recognize the behavioral goals of the queries or sentences presented by users, such as playing music or booking tickets. Slot filling targets on extracting the semantic slot information related to the specific intents from conversation. Take the task of playing music as an example. When user intents to play music, the information of singer and song are the slots that are required to be extracted and filled. The former is considered as a classification problem, while the latter is a sequential labeling problem. Generally, as to the spoken language understanding in task-oriented dialog systems, intent identification and slot filling aim at converting

R. Xu et al. (Eds.): ICCC 2019, LNCS 11518, pp. 12–25, 2019.
https://doi.org/10.1007/978-3-030-23407-2_2

human natural language expressions into structured information. NLPCC 2018 [22] provides an annotated dataset which covers three domains, namely music, navigation and phone call.

There are much research on these two issues. Traditional approaches toward intent identification include rules matching and machine learning algorithms. These approaches was proved effective in some specific domains. However, as the size of the corpora grows dramatically, it is difficult to extract characteristics from general domain. Under this state, some deep learning techniques were applied, including convolutional neural networks (CNN) [19] and recurrent neural network (RNN) [20], especially long-short time memory (LSTM) [21], and so on. Similar situations occurred in slot filling task [14–16].

Our team tries to conduct a model to handle both tasks in spoken language understanding in task-oriented dialogue systems efficiently. We propose a neural framework, named SI-LSTM model, for tackling the intent identification and slot filling. SI-LSTM integrates CRF into an LSTM network, where the CRF will segment the words into different parts of entities and generate the sequential labels for slot filling; and the LSTM will maintain the semantics of each sentence for intent identification. SI-LSTM combines intent identification and slot filling together, and the slot information is used for determining the intent while each type of the intent is used to rectify the slot filling deviation.

Based on the dataset provided by NLPCC 2018, SI-LSTM successfully improves the accuracy of each task, and achieved 90.71% on intent identification, slot filling and error correction in terms of accuracy.

## 2 Related Works

In this paper, there are two main tasks that are intent identification and slot filling. It is required to firstly identify the intent and then fill the corresponding slot with respect to the specific intent. Intent identification is usually considered as a classification problem, which is to classify each query into a corresponding intent category, while slot filling is regarded as a sequential labeling issue. In this section, we will introduce the current studies on these two tasks.

**Table 1.** The samples of intent type and the slot information.

| Intent | Slot |
|---|---|
| music.play | song, singer, theme, style, age, toplist, emotion, language, instrument, scene |
| navigation.navigation | destination, origin, custom_destination |
| phone_call.make_a_phone_call | phone_num, contact_name |

### 2.1 Intent Identification

There are many previous researches on intent identification task, and most of them can be divided into three categories.

[1] presented a rule-based method and designed different templates for tackling this problem. For different intent categories, various thesauruses were constructed to facilitate to identify the intent in the specific domain. More specifically, based on the previous collected records, a list of intents will be listed that are computed by probability distribution. However, this method required too much professional knowledge to construct all the rules, and it is rather time-consuming.

[2, 3] proposed machine learning approaches for intent identification, and Support Vector Machine (SVM), Naive Bayesian, and Decision Trees (DT) [3]. Most of the approaches focused on either feature construction or model selection. These methods have been proved to be effective in the accuracy improvement on the identification. Yet since corpora have become more and more random, it is difficult to find uniform features to figure out all intents.

More recently, researchers attempted to utilize deep learning technology to classify the intent, since the attributes of each sentence can be represented well. For instance, CNN [4], RNN [20], LSTM [5] were widely applied. Also, user profiling tried to guide the intent identification and perform the determination [6]. It was proved that these neural network models worked better on intent identification and speeded up the training process.

### 2.2 Slot Filling

Since our model also targets at the slot filling, we review the related works on slot filling from the following three aspects as well.

Firstly, rule-based methods are proposed. Based on linguistic rules, slot filling was usually accomplished by matching different rules. Despite a high accuracy, much time and abundant knowledge in specific field are required. After that, many statistical models were applied to solve the problem, which was proved efficient and effective. Hidden Markov Model (HMM) [7] and CFG [8] and Conditional Random Field (CRF) [9] are widely used for the sequential tagging problems. Then, researchers today begin to apply RNN [10] into slot filling, and many advantages are explored, including faster training process, a flexible architecture, effective performance, and so on. More importantly, these neural network models can be integrated by the sequential tagging model, such as HMM, CRF, which will further improve the performance.

Our work is inspired by the studies above, and we design a SI-LSTM model for intent identification and slot filling, which will be described in the following section.

## 3 Joint Prediction on Intent Identification and Slot Filling

In task-oriented dialog systems, the comprehension of users' queries is a process of parsing users' queries and converting them into some structures that machine can handle. This comprehension usually consists of two parts, namely intent identification and slot filling.

It is obvious that there is a strong correlation between intent identification and slot filling, and both of them can be mapped into a certain scope, especially in the dataset provided by NLPCC [22]. For example, some specific slot categories such as *song*, *language*, *toplist*, etc. will only appear in the sentences about *music.play*, rather than *navigation.navigation*. The correlations between the intention and the slot are shown in Table 1.

Besides, the dataset can be seen as a stream of user queries ordered by time stamp. The stream is further split into a series of segments according to the gaps of time stamps between queries and each segment is denoted as a 'session'. Instead of being separated contexts, the contexts within a session is correlated with previous ones. For example, given the input text "张三 (Zhang San)" (a singer's name), only when the latest intention is *phone_call.make_a_phone_call*, the current intent will be recognized as *phone_call.make_a_phone_call*, and the slot information will be "<contact_-name> 张三 <contact_name>". Otherwise, the intent will be classified into *OTHERS* and the slot information is empty.

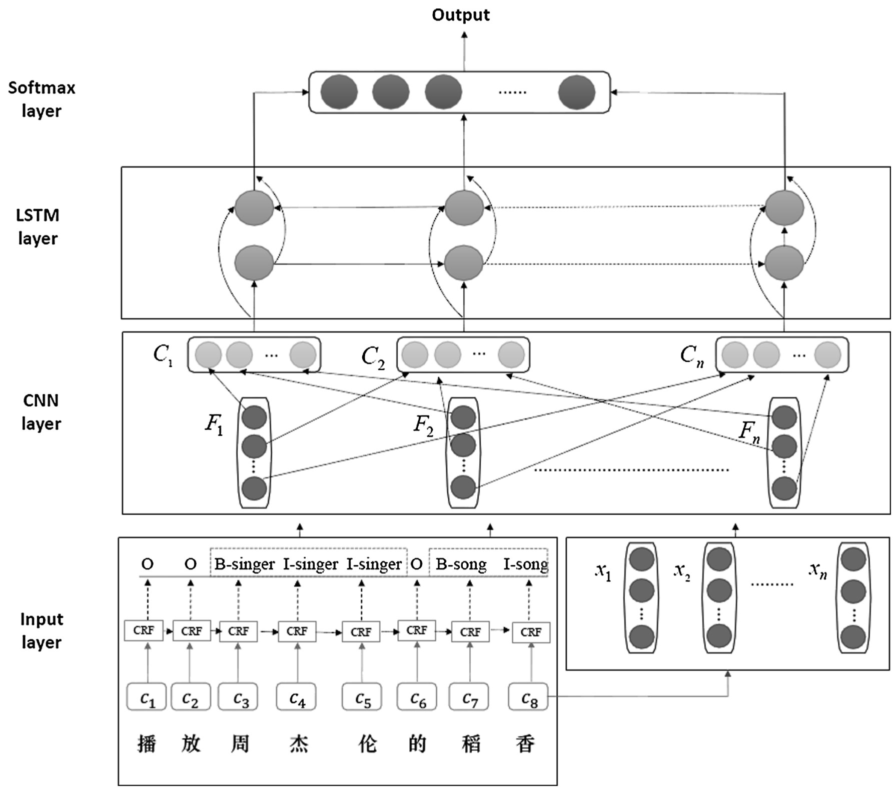

**Fig. 1.** The structure of SI-LSTM. (The Chinese sentence in the example means that play the song by Jay Chou.)

Based on the above observation, we design a neural framework to tackle two tasks at the same time, named SI-LSTM. The structure of the SI-LSTM is shown in Fig. 1. SI-LSTM is a four-layer neural framework, consisting of a CRF layer, a CNN layer, a LSTM layer and n fully connected layer Firstly, each word in a sentence is regarded as the input to the CRF layer to generate the sequential label. After the CRF layer, slot filling results are obtained. Then the output of CRF layer together with the vectorized representation is putted into the CNN layer to extract rich semantics feature. On top of that, the LSTM layer enhances the use of text word order and time information. In the end, the fully connected layer will output the prediction result of the intent identification. Since the contexts within a session are taken into, we set up a memory cell to store the latest user intent. In this way, SI-LSTM will take the memory cell as a reference when it outputs the final result of the classification.

## 3.1 Conditional Random Field Layer

In our Conditional Random Field (CRF) layer, we utilize the classic CRF [11] model for sequential labeling, which attempts to model the conditional probability distribution P(Y|C). Our model first uses CRF for named entity recognition and slot filling.

Based on the given sentence, the CRF layer will firstly segment the words into different parts of entities, and then classify each entity according to the type, such as person, organization, location, and so on. To avoid of the limitations of the data bias towards the states with few successor states, our CRF layer is designed to have the ability to relax strong independence assumptions made in other models.

Given the observation $C$, the model based on the assumption of first order Markov chain predicts the hidden sequence $Y$ represent the attributes of the entities.

Conditional distribution is computed by Eq. (1):

$$P(Y|C) = \frac{1}{Z(c)} e^{\sum_i \sum_j \lambda_j f_j (y_{i-1}, y_i, c, \mathrm{i})} \tag{1}$$

where $Z(c)$ is a normalizing constant, and $\lambda_j$ is the bias weight learned from the training data, and $f_j$ is the feature function.

In our CRF layer, the output will be considered as the input for the CNN layer, in this way both the sequential label and the semantics information will be maintained.

## 3.2 CNN Layer

Inspired by the good feature extraction capabilities of CNN [12], we also use a CNN layer to extract features from texts. Meanwhile in this layer we will combine the slot filling categories with intent identification for the further processing.

Without the loss of generality, for each sentence $s$, it can be formulated as a word sequence $s = \{w_1, w_2, \ldots, w_L\}$, where $L$ denotes the length of s. The objective of intent identification is produce $y_i$ for each $s_i$, and $y_i$ is belonging one of the intent type. Then in the CNN layer, an $n$-dimensional vector is obtained by the combination of the whole words in the sentence shown in Eq. (2).

$$\mathrm{x}_{1:\mathrm{n}} = \mathrm{x}_1 \oplus \mathrm{x}_2 \oplus \mathrm{x}_3 \oplus \ldots \oplus \mathrm{x}_\mathrm{n} \tag{2}$$

The convolutional layer is mainly used to capture the local information between words based on a sliding window. In our CNN layer, the length is denoted by $h$ and the size is denoted by $\omega \in \mathbb{R}^{hk}$, which means that from $i_{th}$ word to the $i + h - 1_{th}$ word will be covered. Then the convolution kernel obtains a characteristic representation by Eq. (3).

$$f_i = f(\omega_i \cdot x_{i:i+h-1} + b) \tag{3}$$

The convolution kernel sequentially convolves all the windows in the sentence to get a feature map $F \in \mathbb{R}^{n-h+1}$ shown in Eq. (4).

$$F = [f_1, f_2, \ldots, f_{n-h+1}] \tag{4}$$

A max pooling is used to get the max dimension from each feature map as the final feature and retains only the most representative features in the feature vector.

The *Softmax* function is used to output the results which can be seen as the conditional probability distribution shown in Eq. (5), and it help us determine the most likely intent shown in Eq. (6).

$$P_\theta\left(y_j | h_{s_i}\right) = softmax(h_{s_i}\omega + b) \tag{5}$$

$$Y_{pred} = argmaxP_\theta \tag{6}$$

In our CNN layer, by using the classic CNN model, the input vector will be converted into a new fixed-length global vector which contains the most representative feature. Meanwhile, the training process is accelerated.

## 3.3 Long Short-Term Memory Layer

In LSTM layer, our model will encode the information from the vector converted by CNN into a fixed-length vector representation. In our task, the dialogue is a hierarchical sequence of data: each sentence is a sequence of words, and each session is a list of sentences.

The long and short term memory layer consists of several repeated cells, and each of them receives the output of the hidden layer $h_{t-1}$ at the previous time and the current input $h_t$. Each cell is made up of an input gate $i_t$, an oblivion gate $f_t$ and an output gate $o_t$. For every neuron in LSTM, the whole working process is as follows:

$$i_t = \sigma\left(w_t^i \cdot x_t + w_t^i \cdot h_{t-1} + b_t\right) \tag{7}$$

$$f_t = \sigma\left(w_t^f \cdot x_t + w_t^f \cdot h_{t-1} + b_t\right) \tag{8}$$

$$q_t = tanh\left(w_t^q \cdot x_t + w_t^q \cdot h_{t-1} + b_q\right) \tag{9}$$

$$o_t = \sigma(w_o \cdot x_t + w_o \cdot h_{t-1} + b_o) \tag{10}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{11}$$

$$h_t = o_t \odot tanh(c_t) \tag{12}$$

Each vector of the feature combination layer is obtained by sequentially connecting the elements corresponding to the $i_{th}$ dimension of each convolutional feature map shown in the Eq. (13).

$$C_i = F_1^i \oplus F_2^i \oplus F_3^i \oplus \ldots \oplus F_{n-h+1}^i \tag{13}$$

We will get $n - h + 1$ combination vectors so that each vector is recombined according to the convolution order to ensure the temporality of the text. Then the vectors are sequentially sent to the LSTM cell. The output of the last hidden layer is obtained as the final sentence representation and finally, the prediction result would be identified by the Softmax layer.

## 4 Spelling Correction

Among the process of slot filling, we need to handle the Chinese typos in the slot, thus the work of spelling correction is needed. Spelling correction has two requirements: to recognize the typos and to find the correct answers. For this consideration, we should design our method to calculate the string similarity such as Edit Distance and determine whether the lot is a typo by using the corresponding thesaurus and find the correct answer for each typo.

### 4.1 Preprocessing

Since the training corpus and test corpus in this task both contain multiple languages and some other characteristics may impose obstacles to the result, data preprocessing is supposed to be necessary.

There are a few obstacles. First of all, given the fact that Arabic numbers and Chinese numbers are considered as completely different character in this research, the conversion between Arabic numbers and Chinese numbers is necessary. When it comes to the expression of exact years such as ‘2002’, the conversion should follow the Chinese traditional expression.

Then the multiple languages could be another problem not only in the process of comparing the similarity between strings, but also in the process of getting each word’s Chinese Pinyin. In order to reduce the negative influence from the foreign languages on comparing Chinese Pinyin, all consecutive foreign characters in each slot are regarded as one Pinyin.

### 4.2 Congruent Length in String Order

We design our approach to compute the string similarity, especially when the string is missing one or two characters. Since the lengths of two strings are crucial for the similarity computation, we use the following equation to compute the similarity when the congruent length($Cl$) is not less than a predefined value.

$$Sim = 0.8 + 0.01 * Cl \quad \text{where} \quad Cl \geq k \tag{14}$$

The parameter $k$ is the congruent length, which is defined as follows.

$$k = \begin{cases} 2 \quad 2 \leq slotlength \leq 4 \\ slotlength - [0.4 * slotlength] \quad 5 \leq slotlength \leq 7 \\ 6 \quad slotlength \geq 8 \end{cases} \tag{15}$$

## 4.3 Edit Distance

We also treat the edit distance [13] as another measurement to calculate similarity between two strings. We regard the edit distance as the minimum number of times needed to edit a single character (such as modify, insert, delete) when changing from one string to another. The smaller the edit distance is, the higher the similarity will be. This time complexity of this algorithm is $\mathrm{O}(m * n)$ and space complexity is $\mathrm{O}(m * n)$. $m$ and $n$ stand for the length of string $a$ and $b$.

## 4.4 Spelling Correction

In this task, the content and the category of input slots are the two variables. At first, the content will be matched to the thesaurus in accordance with the specific category. If the content is appeared in the corresponding thesaurus, this content is not a spelling error. On the contrary, the algorithm will calculate its possibility of becoming a spelling error. Then the remaining slots will be compared with all the strings comparing to the thesaurus and then we obtain the highest similarity.

In our model, we design three ways to calculate the similarity of two strings. The first measurement is congruent length in string order. If there are qualified strings in corresponding thesaurus we will achieve the first similarity. The second measurement is edit distance($lev$). This could be used to calculate the similarity in this way.

$$sim = 1 - \frac{lev}{slotlength} \tag{17}$$

The third measurement is the edit distance($lev$) between Chinese Pinyins. We define a parameter $m$ to measure the similarity of the conversion between the character to Pinyin. Then the similarity can be computed by the following equation.

$$sim = 1 - \frac{m * lev}{slotlength} \tag{18}$$

In our experiment, based on the performance on training dataset, we set $m = 1.8$.

With the incensement of the similarity, the corresponding slot is more likely a typo. On the contrary, this means that this slot is not similar to any strings in the corpus and is not likely to be a typo. We set a variable $p = 0.55$ to help us determine whether the slot is a typo. We take the biggest measurement among these three to represent the overall possibility. When the biggest measurement is bigger than $p$, then this slot is a spelling error and the corresponding string is the correct answer.

## 5 Experiment

In this section, we will report the performance of our proposed approach SI-LSTM based on the dataset provided by NLPCC 2018 [22].

### 5.1 Experiment Setup

NLPCC 2018 [22] provides a dialogue dataset focusing on three scenarios, namely music, navigation and telephone. The training dataset is consisted of 4,707 real annotated dialogue sessions, including 21,350 sentences and 11 intents. According to our statistics, there are 4.5 sentences in per session in average. As to the test dataset, 1,177 dialogue sessions containing 5,349 sentences are involved, and there are 4.5 sentences in per session on average as well. The training and test dataset information is shown in Table 2.

**Table 2.** The description of training and test dataset.

| | Sessions | Sentences | Intents | Avg # of sentences in per session |
|---|---|---|---|---|
| Training set | 4707 | 21350 | 11 | 4.5 |
| Test set | 1177 | 5349 | 11 | 4.5 |

To better demonstrate the dataset, we also list some statistics in Table 3. There are 11 types of intent in total, and the type of *music.play* contains the most sessions except for *OTHERS*. Note that, for some types of the intents, a corresponding thesaurus is also provided, which can help us to extract the slot information and spelling correction.

**Table 3.** The statistics of the dataset.

| Intent | Training set | Test set |
|---|---|---|
| music.play | 6,403 | 1,641 |
| music.pause | 298 | 75 |
| music.prev | 5 | 4 |
| music.next | 132 | 34 |
| navigation.navigation | 3,961 | 1,039 |
| navigation.open | 245 | 56 |
| navigation.start_navigation | 33 | 4 |
| navigation.cancel_navigation | 836 | 206 |
| phone_call.make_a_phone_call | 2,789 | 674 |
| phone_call.cancel | 22 | 18 |
| OTHERS | 6,628 | 1,641 |

In our training process, the dataset was randomly divided into 9:1, with training set (90%), validation set (10%), to train our model and tune some parameters.

In this task, $F1_{macro}$ is used as the evaluation metrics.

$$P_{macro} = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{\#\text{ of queries correctly predicted as intent } c_i}{\#\text{ of queries predicted as intent } c_i}$$

$$R_{macro} = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{\#\text{ of queries correctly predicted as intent } c_i}{\#\text{ of queries labelled as intent } c_i}$$

$$F1_{macro} = \frac{2}{\frac{1}{P_{macro}} + \frac{1}{R_{macro}}}$$

## 5.2 Experimental Results

We deal with all the three tasks, intent identification, slot filling and spelling correction, and we will report the results in this subsection. Moreover, other approaches, such as SVM, LSTM, etc., are also implemented for the comparison with SI-LSTM on the provided dataset.

**Intent Identification.** SI-LSTM was implemented based on the open source library Keras in Python which was backended by Tensorflow. In SI-LSTM, we trained our word embedding by using word2vec based on the whole dataset, and set the dimensionality as 50. 50 epochs were run in total, and the parameters were updated after each batch. We finally set the parameter with the value when the model achieved highest accuracy.

To better demonstrate the performance of our model, we also redesign some classic models for intent identification, including SVM [17], FastText [18], and LSTM. SVM is a classic supervised machine learning algorithm and LSTM is a representative deep learning algorithm. FastText is a classifier developed by Facebook that provides a simple and efficient way to represent textual information. These three models are widely used in NLP research and can provide basic support. Besides, since the slot information works as an important factor, we also integrate it into the three traditional models, and get S-SVM, S-FastText, S-LSTM.

The performance of each model on test set was shown in Table 4.

From Table 4, we can find that our proposed model SI-LSTM achieved the best run in both metrics, which proved the effectiveness of our model. Although SVM is non-deep learning model, the accuracy was 90.43% and the $F1_{macro}$ was 81.97%, that were comparable with other deep learning models. To the contrary, FastText performed poor in both accuracy and $F1_{macro}$. We can also find that compared with basic models, the $F1_{macro}$. The accuracy and $F1_{macro}$ were both improved when putting the slot information into the model. In Table 4.

What's more, the combination of CNN and LSTM accelerates the training process greatly. Also, if we eliminate some minimum categories which contain only a few number of sentences, we are able to see a more surprising output from SI-LSTM.

**Table 4.** The comparison between different models.

| Model | Accuracy | $F1_{macro}$ |
|---|---|---|
| SVM | 90.43% | 81.97% |
| FastText | 87.90% | 82.27% |
| LSTM | 90.60% | 86.26% |
| S-SVM | 91.30% | 78.41% |
| S-FastText | 94.10% | 85.47% |
| S-LSTM | 93.64% | 87.49% |
| SI-LSTM | **94.52%** | **87.73%** |

In our experiment, we also list the accuracy and recall of each type of intent as shown in Table 5.

**Table 5.** The results of each type of intent.

| Intent | Accuracy | Recall |
|---|---|---|
| music.play | 94.30% | 98.78% |
| music.pause | 80% | 64% |
| music.prev | 100% | 75% |
| music.next | 91.18% | 91.18% |
| navigation.navigation | 95.11% | 99.13% |
| navigation.open | 100% | 91.07% |
| navigation.start_navigation | 100% | 100% |
| navigation.cancel_navigation | 89.89% | 86.41% |
| phone_call.make_a_phone_call | 87.65% | 97.92% |
| phone_call.cancel | 100% | 16.67% |

**Spelling Correction.** Since the typos only appear in the slots, the spelling correction is based on the result of slot filling. Besides, we will focus on the errors that are related to *music* and *navigation*, and other fields, e.g. *phone calls*, are not considered. The experimental results are shown in Tables 6 and 7.

**Table 6.** The confusion matrix of spelling correction on both training set and test set.

| Table head | | Predict | |
|---|---|---|---|
| | | Non-typos | Typos |
| Training data | Non-typos | 5,913 | 51 |
| | Typos | 46 | 252 |
| Test data | Non-typos | 1,485 | 13 |
| | Typos | 6 | 82 |

**Table 7.** The result of spelling correction.

| Types | # of typos | # of correction | Accuracy |
|---|---|---|---|
| Song | 71 | 63 | 88.73% |
| Singer | 11 | 7 | 63.63% |
| Total | 82 | 70 | 85.37% |

**Intent Identification and Slot Filling Results.** We evaluated the performance on both intent identification and slot filling (with spelling correction) on the SVM, FastText, basic LSTM and SI-LSTM, and the results are shown in Table 8.

**Table 8.** The comparison between different models on intent identification with slot filling.

| Model | Accuracy |
|---|---|
| SVM | 84.49% |
| FastText | 84.90% |
| LSTM | 84.56% |
| S-SVM | 72.67% |
| S-FastText | 89.25% |
| S-LSTM | 89.08% |
| SI-LSTM | **90.71%** |

It is obviously that the accuracy of model for the joint prediction on slot filling and intent recognition achieved the best performance.

## 5.3 Discussion

Based on the above experimental results, we summarized some characteristics of our approach and made some error analysis.

SI-LSTM achieved a high accuracy in intent identification, but performed not as good in the metric of $F1_{macro}$. In Table 5, we can see the recall of our model is very low under '*phone_call.cancel*' intent which in turn imposes negative effect on $F1_{macro}$, although the size of that type is quite small, i.e. 18. In fact, many contents under this type '*phone_call.cancel*' only express the instruction of cancel or stop, but few mention the specific objection of the instruction. So the classification of this specific intent needs to account for the previous content and it is difficult for the model to distinguish between '*music.pause*', '*navigation.cancel_navigation*' and '*phone_call.cancel*'. Regarding that there are only 22 data in training set and 18 data in test set under the intent '*phone_call.cancel*', the volume of data does greatly affect the final result of the model.

Besides, in the task of spelling correction, multiple languages occur in a session. Due to limited resources, the result of typo detection is susceptible by the volume and quality of thesauruses. Therefore, for the slots beyond the thesaurus, it is difficult for our model to distinguish the intent accurately.

## 6 Conclusions

In this paper, we propose a neural framework, named SI-LSTM, for intent identification and slot filling. SI-LSTM combines two tasks and integrates CRF into a LSTM network, where the slot information is extracted by using CRF, while the intent will be identified by using LSTM. In our approach, the slot information is used for determining the intent, and the intent type is used for slot filling. Based on the dataset provided by NLPCC 2018, SI-LSTM achieved 90.71% on intent identification, slot filling and error correction in terms of accuracy.

**Acknowledgement.** This work is partially funded by the National Natural Science Foundation of China (61602326, U1636103, 61672361, and U1536207) and the Fundamental Research Fund for the Central Universities (3262019T29 and 3262019T54).

## References

1. De, A., Kopparapu, S.K.: A Rule-Based Short Query Intent Identification System. Submitted on 25 Mar 2015
2. Song, X., Zheng, Y., Cao, H.: Research on driver's lane change intention recognition based on HMM and SVM. J. Electron. Meas. Instrum. **30**(1), 58–65 (2016)
3. Worachartcheewan, A., Nantasenamat, C.: Identification of metabolic syndrome using decision tree analysis. Diabetes Res. Clin. Prac. **90**(1), e15–e18 (2010)
4. Turra, G., Arrigoni, S., Signoroni, A.: CNN-based identification of hyperspectral bacterial signatures for digital microbiology. In: Battiato, S., Gallo, G., Schettini, R., Stanco, F. (eds.) ICIAP 2017. LNCS, vol. 10485, pp. 500–510. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68548-9_46
5. Ren, J., et al.: Look, Listen and Learn - A Multimodal LSTM for Speaker Identification. Accessed 13 Feb 2016
6. Rezaei, B.A., Roychowdhury, V., Ghate, S., Khajehnouri, N., Boscolo, R., Mracek, J.: Concept-level user intent profile extraction and applications. Accessed 03 June 2014
7. Rabiner, L.R., Juang, B.H.: An introduction to HMMs. IEEE ASSP Mag. **3**(1), 4–16 (1986)
8. Wang, Y.-Y., Acero, A.: Combination of CFG and n-gram modeling in semantic grammar learning. In: European Conference on Speech Communication & Technology, pp. 2809–2812 (2003)
9. SUN, X., WANG, H.: Intent determination and slot filling in question answering. J. Chin. Inf. Process. **31**(6), 132–139 (2017)
10. Pollack, J.B.F.: Recursive distributed representations. Artif. Intell. **46**(1–2), 77–105 (1990)
11. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings 18th International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann (2001)
12. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. ICML 2008, pp.160–167. ACM, New York, 01 January 2008
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
14. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF Models for Sequence Tagging. Computer Science (2015)

15. Mesnil, G., Dauphin, Y., Yao, K.: Using recurrent neural networks for slot filling in spoken language understanding. IEEE/ACM Trans. Audio Speech Lang. Process. **23**(3), 530–539 (2015)
16. Yao, K., Peng, B., Zweig, G., Yu, D., Li, X., Gao, F.: Recurrent conditional random fields for language understanding. In: ICASSP (2014)
17. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
18. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759 (2016)
19. Lecun, Y., Boser, B., Denker, J.S., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
20. Graves, A., Mohamed, A., Hinton, G.: Speech Recognition with Deep Recurrent Neural Networks. arxiv (2013)
21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735. PMID 9377276
22. Zhao, X., Cao, Y.: Overview of the NLPCC 2018 shared task: spoken language understanding in task-oriented dialog systems. In: Zhang, M., Ng, V., Zhao, D., Li, S., Zan, H. (eds.) NLPCC 2018. LNCS (LNAI), vol. 11109, pp. 468–478. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99501-4_46