IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2021, accepted June 28, 2021, date of publication July 7, 2021, date of current version July 15, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3095416*

# Smart Stacking of Deep Learning Models for Granular Joint Intent-Slot Extraction for Multi-Intent SLU

**NIRAJ KUMAR AND BHIMAN KUMAR BAGHEL**
Samsung Research Institute, Bengaluru 560037, India
Corresponding author: Niraj Kumar (niraj.kumar@samsung.com)

**ABSTRACT** These days' multi-intent utterances have become very important for the spoken language understanding (SLU). The multi-intent systems and algorithms add more complexity (Compare to the single-intent-based system) to the SLU. As, it requires an accurate system, which can identify intents and slots at fine-grain (i.e., word/token) level and also able to handle the relation between intents and slots locally at utterance level. In this case, intents may belong to multiple domains and multiple different classes. Similarly, slots may also belong to multiple different classes, and slots of the same class may be related to multiple different intent classes. Unfortunately, very few works have been done till now to address these issues at the fine-grain level. To solve this problem, we propose a smart stacking-ensemble strategy. The first stage of this system uses a combination of three different types of powerful multitasking NLP models, developed on top of pre-trained BERT, XLNet, and Elmo. Finally, a stacking ensemble layer learns to predict the best possible results. We have evaluated our model on four publicly available datasets. The evaluation results on the state-of-the-art public datasets show that our devised system outperforms the existing multi-intent-based systems at token-level and sentence-level.

**INDEX TERMS** Artificial intelligence, artificial neural networks, machine learning, natural language processing, pattern analysis, pattern recognition.

## I. INTRODUCTION

Intent classification and slot tagging are very important parts of Spoken Language Understanding (SLU). However, most of the prior research work only focuses on a single intent scenario. Their models are trained based on the assumption that each utterance has only one single intent. These days, a lot of utterances contain multiple intents. The presence of multi-intent cases in spoken language texts may vary. Its percentage occurrences may vary from 2% (e.g., ATIS dataset [2]) to 52% (as reported in [1] with the Amazon internal dataset). The single intent-based SLU systems fail to extract all the required features from the multi-intent utterances. In the single intent setting, we generally extract intent and slot(s) from the text. However, in the multi-intent cases, we mostly focus on the following:

a) Identification of multiple intent classes.
b) Extracting all slots.
c) Identifying the relationship between intents and the corresponding slots.

Most of the classifiers of this category (like - [1], and [3]), consider intent classification as a multi-level classification task and slot classification as a sequence labeling task that maps an input word sequence $x = (x_1, x_2, \ldots, x_T)$ onto the corresponding slot tags sequence $y^{slot} = y_1^s, y_2^s, \ldots, y_T^s$ (with the help of Begin/In/Out (BIO) format). Finally, [1], and [3], propose deep learning based interaction mechanism to combine intents and corresponding slots. To achieve these goals, they have used multitasking deep learning architectures.

The paper [37] has claimed to identify the token level slots and intents using stack propagation, however, the entire system is designed for the "single intent" based system. The dataset used in the system[1] uses sentence-level mapping of intents and word/token level mapping of slots (using BIO format).

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

[1] https://github.com/LeePleased/StackPropagation-SLU

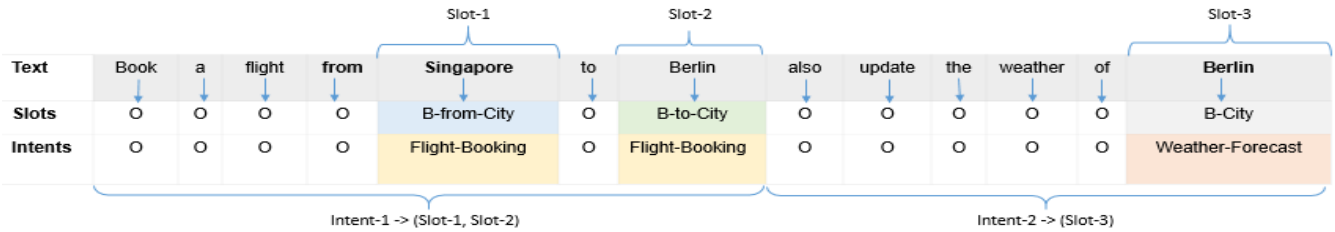


**FIGURE 1.** Word level intent, slot annotation and intent, slot relation extraction.

**TABLE 1.** Intent and slot annotation used in [3].

| |
|---|
| **BIO-Format slot tagging**: what O \n airlines O \n are O \n **ac B-airline_code** \n and O \n **as B-airline_code** \n , O \n please O \n show O \n me O \n all O \n airports O \n in O \n **denver B-city_name** \n and O \n also O \n how O \n many O \n flights O \n does O \n **twa B-airline_code** \n have O \n in O \n **business B-class_type** \n **class I-class_type** |
| **Intent Class (separated by '#')**: **atis_airline#atis_airport#atis_quantity** |

**Note**: '\n' represents the new line.

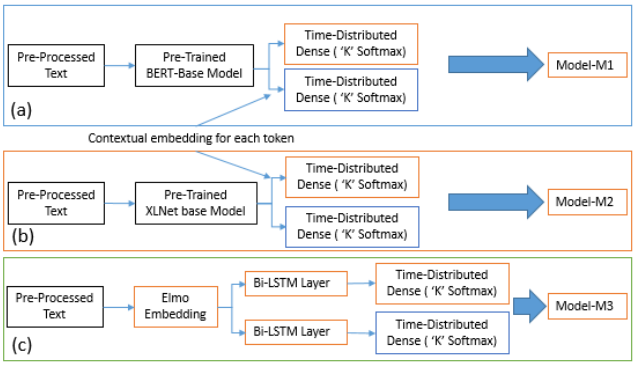


**FIGURE 2.** Architecture of the three different base learners. (a) Base algorithm-1, (b) Base algorithm-2, (c) Base algorithm-3.

It uses the intent information through stack propagation to enhance the slot prediction accuracy and finally map slots to the intent of the sentence. The entire scenario has been discussed for the single intent cases. In the case of a multi-intent scenario, there may be multiple intents related to different domains or classes. The current strategy of mapping of all intents (maybe related to a different domain/class) to all the word/token level slots may or may not help in increasing the quality of slot prediction. Also, it is tough to identify the relationship between intents and slots beforehand in the multi-intent environment to apply the stack propagation (as discussed in [37]).

Based on the above experiences, we decided to use a fine-grain annotation of intents and slots and solve the problem at the word/token level. So, the main research problem is – "how to achieve the high accuracy at word/token level intents and slots extraction and mapping?". To solve these issues, we have made the following contributions.

### 1) CONTRIBUTION-1

First of all, we change the way to annotate the data. Reference [3], has extracted the relation between intents and slots, but the way to annotate the data do not serve our purpose to extract and map intents and slots at word/token level. For example, a sample taken from [3][2] shows the following annotation (see TABLE 1).

This annotation gives separate sentence-level intent annotation (see the second row of TABLE 1, two intents are separated by '#') and slot annotation (in BIO format). Due to the absence of direct annotation of word/token level intents, it becomes tough to map the word level intents and slots (like, we mapped in Fig. 1).

To solve this issue, we have re-annotated ATIS [2], and SNIPS [40] dataset. After annotation, we have applied a similar strategy (as applied by [3] to evaluate multi-intent system) to prepare MixATIS and MixSNIPS dataset.

### 2) CONTRIBUTION-2

The next contribution is - applying "multitask deep learning systems" to get word level intents and slots with proper mapping (identifying relation) between them. For this, we have considered the word level intent and slot classification tasks as multitasking sequence labeling tasks that maps an input word sequence $x = (x_1, x_2, \ldots, x_T)$ onto the corresponding slot tags sequence $y^{slot} = y_1^s, y_2^s, \ldots, y_T^s, where\ s \in [0, N^s]$ (here $N^s$ represents the slot count) and intent tags sequence $y^{Intent} = y_1^i, y_2^i, \ldots, y_T^i, where\ i \in [0, N^i]$ (here $N^i$ represents the Intent count). This converts the three tasks, (i.e., (a) Intent extraction, (b) slot tagging, and (c) relating slots to the extracted intents) into just two tasks, and results in very good accuracy. We have used three diverse set of learners – (a) BERT [7], (b) XLNet [5], and (c) Elmo [30] to develop the multitasking models. (See Fig. 2). The following contains a brief overview of these models.

The BERT [7] model uses a multi-layer bidirectional Transformer encoder based on the original Transformer model [8]. The input representation of BERT uses (a) a

[2]https://raw.githubusercontent.com/LooperXX/AGIF/master/data/MixATIS/train.txt

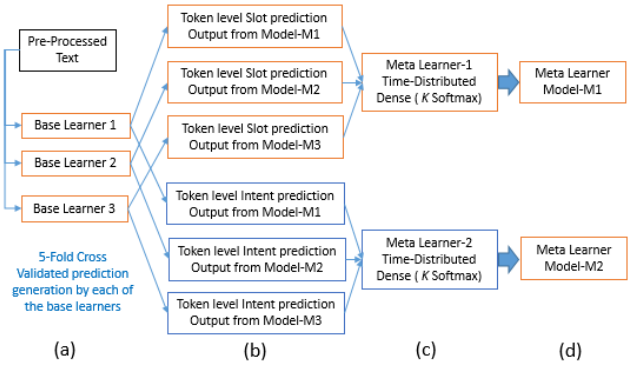


**FIGURE 3.** Training meta learners. (a) shows the training using base learners, (b) shows validated prediction output for word level intents and slots, we also add ground truth as class label to train the meta learners, (c) shows meta learners for word level intent and slot predictions, (d) Trained meta learning models.

concatenation of Word Piece embeddings [10], (b) positional embeddings, and (c) segment embedding. A special classification embedding ([CLS]) is inserted as the first token and a special token ([SEP]) is added as the final token.

XLNet [5] uses the permutation language model based system with two-stream attention. It uses the Transformer-XL model for training. This architecture has provided it an extra advantage and thus outperforms on several NLP tasks [11].

Elmo [30] is a pre-trained contextual word embedding model. It is a bidirectional LSTM (BiLSTM) language model which can generate context-dependent word embeddings. The prediction process of the BiLSTM language model is to maximize the log-likelihood of the probability of token from both directions.

#### 3) CONTRIBUTION-3

The next contribution is the use of the stacked generalization based ensemble technique [36] also known as stacking or stacking ensemble. Stacking (also called Super Learning [33]), is a class of algorithms that involves training a second-level ''meta-learner'' to find the optimal combination of the base learners. Unlike bagging and boosting, the goal in stacking is to ensemble strong, diverse sets of learners together. Although the concept of stacking was originally developed in 1992 [36], the theoretical guarantees for stacking were not proven until the publication of a paper titled, ''Super Learner'', in 2007 [33]. In paper [33], it was shown that the Super Learner ensemble represents an asymptotically optimal system for learning. Following the concept of the selection of diverse learner, as discussed in [33], we have used BERT, XLNet, and Elmo based models as a base learner. Finally, we have used 'TimeDistributedDense[3]' networks as meta-learners for word/token level intents and slots. (See Fig. 3).

We have faced the following research challenges in the path of achieving the goal.

[3]https://keras.io/api/layers/recurrent_layers/time_distributed/

#### 4) RESEARCH CHALLENGES

The token-level annotation of intents and slots results in a very high increase in the count and complexity of the relation between intents and slots. Several times, the different terms having the same slot class may be related to the two or more different intent classes (maybe related to different domains/sub-domains). Thus, the total count of such relations becomes very high (compare to the actual count of intent or slot labels in the dataset). Solving such problems requires a robust effort, that's why we have used a smart stacking ensemble with three different types of top-performing and diverse nature NLP models.

#### 5) PAPER ORGANIZATION

The organization of the rest of the paper can be described as follows. Section II discusses the related work in this area. Section III discusses the algorithmic flow of this system. Section IV discusses the experimental evaluation of the system, and finally, Section V concludes the paper.

## II. RELATED WORK

We can summarize the work in this area under the following categories.

### A. JOINT MULTI-INTENT SLOT EXTRACTION

Authors of [1] have investigated an attention-based neural network model that performs multi-label classification for identifying multiple intents and produces labels for both intents and slots at the token level. They tested their system on ATIS dataset [2], SNIPS[4] dataset, and one internal dataset. This paper identifies the relation between slots and intents. Authors of [3], have proposed an Adaptive Graph-Interactive Framework (AGIF) for joint multiple intent detection and slot filling, where they introduce an intent-slot graph interaction layer to model the strong correlation between the slot and intents.

### B. JOINT MODEL FOR INTENT (SINGLE/MULTI-INTENT) AND SLOT EXTRACTION

To consider the high correlation between intent and slots, many joint models [4], [19]–[21], and [22] are proposed to solve two tasks (i.e., intent extraction and slot tagging). References [19], [20], and [23] propose to utilize the intent information to guide the slot filling. Reference [22] further utilize a stack-propagation framework for better leveraging intent semantic information to guide the slot filling, which achieves the state-of-the-art performance. References [25] and [24] consider the cross-impact between the slot and intents. However, these modes, do not identify the relationship between intents and slots.

[4] https://github.com/snipsco/nlubenchmark/tree/master/2017-06-custom-intent-engines

### C. INTENT DETECTION

Single Intent detection is formulated as an utterance classification problem. Different classification methods, such as support vector machine (SVM) and RNN [12], and [15], have been proposed to solve it. References [16] adopts a capsule-based neural network with self-attention for intent detection. However, the above models mainly focus on the single intent scenario, which cannot handle the complex multiple intent scenario. References [17] and [18] explore the complex scenario, where multiple intents are assigned to a user's utterance. Reference [17] use log-linear models to achieve this.

### D. SLOT FILLING

Slot filling can be treated as a sequence labeling task. The popular approaches are conditional random fields (CRF) [26] and recurrent neural networks (RNN) [17]; [27]. Recently, [28] and [29] introduce the self-attention mechanism for CRF-free sequential labeling. Reference [33] have introduced the use of the ''stack propagation'' based concept to get the benefit from an intent class in word level slot prediction. Finally, it presented the mapping of intent for extracted word level slots for the single intent cases.

## III. METHODOLOGY

In this section, we describe our system architecture. First of all, we use the ''truecase'' [31] to correct the case of the alphabet of the given text. We use this corrected text in the entire training and test operations. The next important thing is the use of a 'Stacking Ensemble' based system. The following contains the description of ''STACKING ENSEMBLE LEARNING'' (see Section III-A), its base learners like: ''BASE ALGORITM-1, 2, and 3 (see Sections 'III-B', III-C', and 'III-D'), and ''META-LEARNING ALGORITHM'' (see Section III-E).

### A. STACKING ENSEMBLE LEARNING

The following contains the steps applied in the ''Stacking Ensemble Learning''.

1. Set up the ensemble.
   - 1.1. *We specify a list of L base algorithms (with a specific set of model parameters). In this algorithm, we have used $L = 3$ base learners. They are (a) BERT, (b) XLNet and (c) Elmo based multitasking system for word level intent and slot classifications.*
   - 1.2. *We specify a meta -learning algorithm. Our meta-learning algorithm is a 'TimeDistributed-Dense' used to learn the correct prediction for word level intents and slots.*
2. Train the ensemble.
   - 2.1. *We perform k'-fold cross-validation on each of these learners and collect the cross-validated predicted values from each of the L algorithms. In the current system, we use k' = 5. (**Note:** Scalability is the main reason behind the selection of 5-fold cross validation, instead of using 10-fold or higher.). By this way, we generate $N'$ cross validated predicted values. As, in 5-fold cross validation, each time one-fold will be used for prediction and the rest of the four-folds will be used for training. ($N'$ = the number of rows in the training set.)*
   - 2.2. *The $N'$ cross-validated predicted values from each of the L algorithms can be combined to form new $N' \times L$ matrixes for word level intents and slots respectively. These matrices, along with the original response vectors, are called the "level-one" data.*
   - 2.3. *We train the meta-learning algorithm on the "level-one" data. Our meta-learning algorithm contains two separate TimeDistributedDense networks (for one each for intents and slots). This results in separate meta-learning models for intents and slots.*
3. Predict on new data.
   - 3.1. *To generate ensemble predictions, we first generate predictions from the base learners. To generate the test result in the k'-fold cross validation, we use the strategy as suggested in [38]. I.e., instead of taking all $k'$ models (here $k' = 5$ is taken) from each of the base learners ($L = 3$ is taken), which can generate scalability issues, we have used the whole training dataset to generate a single model for each of the base learners. Thus, instead of 15, we got three base learner models to generate predictions on new data. We name it as "base learner models for test". One question, may come here, - "why we didn't use the best model from k'-fold cross validation?". The answer is − as per observations given in [39], such selection, will either result in overfitting or it may require additional data to test the model (which may be tough in the current case and it may also result in high data dependencies). The proposed changes in the preparation of "base learner models for test" have made our models scalable. Now, it is as scalable as model averaging type ensembles, and we effectively utilize the entire training data to train our meta-learners (i.e., achieved by using k'-fold cross validation). (See Fig. 4)*
   - 3.2. *We feed those predictions (obtained from step 3.1) into the meta-learner to generate the ensemble prediction for intents and slots. (See Fig. 4)*

### B. BASE ALGORITHM-1 - BERT BASED MULTITASKING SEQUENCE LABELING MODEL

Authors of [32], have used BERT for joint Intent and slot classification. For this, they have used the pooled output and sequence output of the pre-trained BERT model and jointly learn intents and slots. For intent, authors

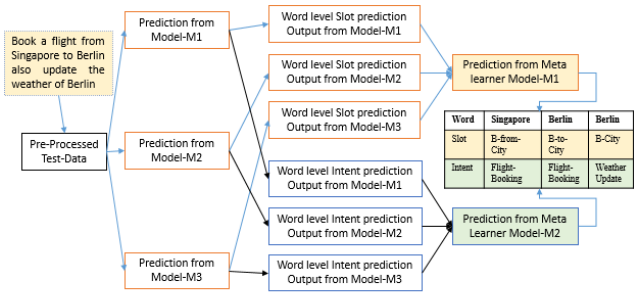


**FIGURE 4.** Test option. We pass the test data to the base models - 'Model-M1', 'Model-M2', and 'Model-M3'. The outputs of these models are passed to the meta learner models, i.e., 'meta learner Model-M1', and meta learner Model-M2, which finally gives the intents and slots at the word level.

have used multi-level classification and for slots, they have used sequence classification. We have borrowed the idea of slot tagging from [32] and utilize it in developing the joint sequence classification strategy for slots and word level intents (See Fig. 2). For this, we have considered the sequence output only. Let us represent utterance $S_i = [w_1, w_2, \ldots, w_i, \ldots, w_N]$ where $N$ is maximum sequence length (in the terms of count of maximum number of tokens) obtained after trucasing. Let $H_i = [h_1, h_2, \ldots, h_l, \ldots, h_N]$, *where* $h_l \in R^d$ is a D-dimensional vector for each of the token $w_i$ obtained as BERT sequence output.

We have used two different TimeDistributedDense layers as the last layers to achieve the multitasking outputs.

The first part of the multitasking system generates the Softmax intent class output for each of the $N$ words/tokens. The TimeDistributedDense layer used for this task uses the BERT sequence output $H_i$ as input and generates the following output for each word and map it to the output intent classes.

$$out_i = Softmax\left(W_i H_i + b_i\right) \tag{1}$$

Each output (i.e., $out_i$, *where* $i \in [1, N]$) of the Softmax classification layer consists of $K_I$ outputs, which indicates the probability of each of the $K_I$ intent classes. Here $W_i$ is the trainable weight matrix of the Dense layer and $b_i$ is the associated bias.

Similarly, the second part of the multitasking system uses TimeDistributedDense layer to generate the Softmax slot class output for each of the $N$ words/tokens. For example,

$$out_j = Softmax\left(W_j H_j + b_j\right) \tag{2}$$

Each output (i.e., $out_j$, *where* $j \in [1, N]$) of the Softmax classification layer consists of $K_S$ outputs, which indicates the probability of each of the $K_S$ slot classes. Here $W_j$ is the trainable weight matrix of the Dense layer and $b_j$ is the associated bias.

Let $y^I$ and $y^S$ represent the sequence classification task for word level intents and slots. To jointly model both tasks, the learning objectives can be formulated as:

$$P\left(y^I, y^S|w\right) = \prod_{n=1}^{N} P\left(y_n^I|w\right) \prod_{n=1}^{N} P\left(y_n^S|w\right) \tag{3}$$

The learning objective is to maximize the conditional probability $P\left(y^I, y^S|w\right)$ (at token/word level, represented as $w$ and for the text sequence of length $N$). The model is fine-tuned end to end via minimizing the cross-entropy.

### C. BASE ALGORITHM-2 - XLNet BASED MULTITASKING SEQUENCE LABELING MODEL

Similar to the use of the BERT (see previous section), we have used the sequence output from pre-trained XLNet model,[5][6] (See Fig. 2). Let us represent utterance $S_i = [w_1, w_2, \ldots, w_i, \ldots, w_N]$ where $N$ is maximum sequence length (in the terms of count of maximum number of tokens) obtained after trucasing. Let $H_i = [h_1, h_2, \ldots, h_l, \ldots, h_N]$, *where* $h_l \in R^d$ is a D-dimensional vector for each of the token $w_i$ obtained as XLNet sequence output. We feed it to two separate TimeDistributedDense layers dedicated to classify word level intents and slots (see "(1), and "(2)", like we did with BERT in the previous section). We keep the learning objective and classification outputs same as discussed above for BERT (See "(3)", in the previous section).

### D. BASE ALGORITHM-3 - ELMO-BiLSTM BASED JOINT SEQUENCE LABELING MODEL

In this architecture (See Fig. 2) we feed the utterance $S_i = [w_1, w_2, \ldots, w_i, \ldots, w_N]$ to pre-trained Elmo model, obtained after trucasing. Let the output of the Elmo embedding is $X_i = [x_1, x_2, \ldots, x_l, \ldots, x_N]$, *where* $x_l \in R^d$ is a D-dimensional word vector for word $w_j$. We feed the Elmo embedding output $X_i$ to (a) word/token level intent classification network and (b) word/token level slot classification network.

The (a) word/token level intent classification network, uses a BiLSTM, which uses the Elmo embedding output $X_i$ and pass the result to a Final TimeDistributedDense layer. The TimeDistributedDense layer gives the Softmax score for word/token level intent classification.

#### 1) DETAILS

The BiLSTM has been used to capture both past and future contexts. It provides $h_t$ (i.e., hidden state output at time '$t$') from both the directions (forward and backward). The forward LSTM takes the natural order of embedding output obtained from Elmo to obtain the hidden state at time $t$ $\overrightarrow{h_t}$, while the backward-LSTM takes the reverse order input to obtain $\overleftarrow{h_t}$, then the final hidden state at time $th_t$ can be calculated as:

$$h_t = \underset{h_t}{\rightarrow} \oplus \underset{h_t}{\leftarrow} \in R^{2L} \tag{4}$$

[5]https://github.com/zihangdai/xlnet/tree/4c83f2f688a59576eb3b479228a647e5ed3315a2

[6] https://github.com/YankeeMarco/xlnet_sequence_tagging

where, $\oplus$ is the concatenation and $L$ is the size for the one-dimensional LSTM. Thus, we can write the BiLSTM output as:

$$h_t = BiLSTM\left(\hat{x}_t, \hat{x}_{t-1}, \ldots, \hat{x}_1\right) \in R^{2L} \forall t \in [1 \ldots l] \quad (5)$$

where, $\hat{x}_t$ denotes the vector for word $x_t$ obtained from $X_i$. Here $h_t$ represents hidden states calculated over all time steps. We use $h_t$ as input for the TimeDistributedDense Layer, which generates the Softmax intent class output for each of the $N$ words/tokens. It generates the following output for each word and maps it to the output intent classes.

$$out_i = Softmax\left(W_i h_t + b_i\right) \quad (6)$$

Each output (i.e., $out_i$, *where* $i \in [1, N]$) of the Softmax classification layer consists of $K_I$ outputs, which indicates the probability of each of the $K_I$ intent classes. Here $W_i$ is the trainable weight matrix of the Dense layer and $b_i$ is the associated bias.

Similarly, the second part of the multitasking system uses the same network structure to generate the Softmax slot class output for each of the $N$ words/tokens. Other than target slot classes, the entire network is same as discussed in the first part of the multitasking system (see "(2)"). Thus, in this case each output (related to the TimeDistributedDense layer) of the Softmax classification layer consists of $K_S$ outputs, which indicates the probability of each of the $K_S$ slot classes.

Suppose $y^I$ and $y^S$ represent the sequence classification task for word level intents and slots. We apply the same strategy, to maximize the conditional probability and joint learning, as applied in Section III-A (see "(3)").

### E. META-LEARNING ALGORITHM

We have used the two separate TimeDistributedDense for meta-learning. (See Fig. 3), shows the flow of the algorithms. The TimeDistributedDense takes $N' \times L$ inputs for word level intent prediction. It uses $N$ dense (where $N$ is the max number of words in the text) with $K$ Softmax class levels (where, $K$ is the total number of intent class levels in the text). Finally, it generates $N \times K$ size output.

The structure of the second meta-learner (i.e., TimeDistributedDense) takes $N' \times L$ inputs for word level slot prediction. It uses $N$ dense (where $N$ is the max number of words in the text) with $K'$ Softmax class levels (where, $K'$ is the total number of slot' class levels in the text. Finally, it generates $N \times K'$ size output.

We apply argmax[7] on the corresponding TimeDistributed-Dense to get the word level intents and slots.

## IV. EXPERIMENTS

### A. DETAILS OF DATASET

We take two widely used public datasets, ATIS (Airline Travel Information System) [2] and SNIPS [40].

The ATIS dataset contains audio recordings of people requesting flight reservations, with 21 intent types and 120 slot labels. There are 4,478 utterances in the training set, 893 in the test set, and 500 utterances in the development set.

The SNIPS data was collected from the SNIPS personal voice assistant, with 7 intent types and 72 slot labels. The training set contains 13,084 utterances, the test set contains 700 utterances and the development set also contains 700 utterances.

#### 1) NOTE

The ATIS dataset contains few (around 2%) utterances with multiple intents, but to allow multi-class classification, it adds the combination of more than one intents through the '+' sign and considers it as one class. For example − "flight + airfare" is considered as one intent. With this arrangement, it behaves as single intent data with 21 different intent classes. But in the current experiment, we consider "flight" and "airfare" as two separate intents. With this consideration, the count of intent classes in ATIS dataset reduced to 18. The SNIPS is the single intent based dataset.

We have also tested our system on two other mixed kinds of datasets - (a) MixSNIPS, and (b) MixATIS prepared by authors of [3]. However, we have re-constructed the dataset by using our word/token level intent and slot annotation scheme. In the MixSNIPS dataset, conjunctions, "and", are used to connect sentences with different intents and ensure that the ratio of sentences has 1-3 intents is (same as discussed in [3]). It contains a total count of 45,000 utterances for training, 2,500 utterances for validation, and 2500 utterances for testing. Similarly, another multi-intent SLU dataset, MixATIS, used in the experiment. It is prepared from the ATIS dataset by applying the same strategy, as used with MixSNIPS (as discussed by authors of [3]). It contains 18,000 utterances for training, 1,000 utterances for validation, and 1,000 utterances for testing. We have used "BERT based semantic similarity checking" to maintain the similarity with the dataset prepared by authors of [3].

To achieve the 5-fold cross validation (See Section III-A), we merge the training and development (also written as validation set) data and apply 100% random shuffling with all the above discussed datasets. We consider it as the final training dataset for stacked ensemble learning. For the rest of the training operations, we have used the same training and validation (development) sets as discussed in the actual dataset (including the test phase, as discussed in Section III-A).

### B. EVALUATION METRICS AND STRATEGY

Actually, authors of [1] have used the following strategy. For utterances that had only a single intent, they assigned this intent to all tokens that had a slot label (i.e., to slot labels that do not correspond to O). For utterances that had more than one intent, they assigned all intents to all tokens that had slot labels. After this process, if an utterance had two intents, intent1 and intent2, and if a token 'i' had a slot label, the token ended up with targets of the form (slot-i; intent-i-1; intent-i-2).

[7] https://numpy.org/doc/stable/reference/generated/numpy.argmax.html

However, we didn't use this measure, instead, we prefer to use a different measure to calculate the sentence-level accuracy. According to our strategy, if all intents, slots are correctly identified and the relation between intents and slots are correct, then we consider it as a correct instance. This measure is very important, as the sentence-level accuracy shows, how effectively, external utterances can be handled by the devised system. In all results tables, we use it as ''Sentence Level Accuracy''. Similarly, if the system identifies all intents correctly at the sentence level, then we consider it as the correct instance of ''Intent Accuracy Sentence level''. The 'F1' measure strategy, for intents and slots, is the same, as used in [1], and [3].

### C. MODEL DETAILS AND HYPER-PARAMETERS

We have used the BERT base cased model.[8] This model uses L = 12 hidden layers (i.e., Transformer blocks), a hidden size of H = 768, and A = 12 attention heads. For XLNet, we have used XLNet-Base, Cased model[9] having 12-layer, 768-hidden, and 12-heads. The selected batch size for both cases was 16. Adam optimizer is used for dense of TimeDistributedDense with the initial learning rate 5e-5. For model discussed in Section III-C, we have used Elmo model with dimension 1024, BiLSTM hidden size 256, Batch size 32, Adam optimizer with initial learning rate 0.001 is used for dense of TimeDistributedDense. The maximum sequence length (as used in Section III-B, III-C, and III-D) N is data-dependent and calculated separately for each dataset. The sequence length is the count of tokens in a single utterance; thus, the max sequence length is the count of the maximum number of tokens in any of the utterances of the given dataset. We have used the early-stopping [41] based on minimizing the loss with callback[10] and patience = 2, for the best model selection (in all cases).

We use model name M1 to represent the BERT based model (Section III-B), M2 to represent XLNet based model (Section III-C), and M3 to represent Elmo-BiLSTM base model (Section III-D) in all the result tables. We use Avg(M1, M2, M3) to show the result when considering the weightage model averaging based ensemble ( [6]). Similarly, Avg(M1, M2), Avg(M1, M3), Avg(M2, M3) shows the weightage averaging ensemble ( [6]) taking two models at a time. Finally, 'Our-Model' shows the result of our Stacked Ensemble model. We use the grid search to get weights for the weightage ensembles.

### D. RESULT AND DISCUSSION

We compare our approach against the latest state-of-the-art approaches that have shown the best performance in previous work. For this, we have used the published results by [1] and [3]. Table 2, 3, 4, and 5 show the comparative results obtained by the model investigated in this paper.

[8] https://tfhub.dev/tensorflow/bert_en_cased_L-12_H-768_A-12/3
[9] https://github.com/zihangdai/xlnet
[10] https://keras.io/api/callbacks/

**TABLE 2.** Evaluation results with atis dataset.

| Model | Slot (F1) | Intent (F1) | Intent Acc Sentence level | Sentence Level Accuracy |
|---|---|---|---|---|
| [1] | 94.22 | 95.82 | 95.39 | NA |
| [3] | 96.0 | 80.2 | 97.1 | 87.2 |
| M1 | 96.98 | 97.99 | 97.29 | 89.01 |
| M2 | 96.33 | 97.80 | 97.30 | 87.30 |
| M3 | 95.90 | 97.83 | 97.00 | 86.88 |
| Avg(M1,M2) | 97.00 | 98.02 | 97.40 | 89.21 |
| Avg(M1, M3) | 97.00 | 98.00 | 97.38 | 89.20 |
| Avg(M2, M3) | 96.84 | 98.00 | 97.40 | 88.00 |
| Avg(M1, M2, M3) | 96.99 | 98.02 | 97.40 | 89.23 |
| **Our-Model** | **97.33** | **98.38** | **97.77** | **90.25** |

**TABLE 3.** Evaluation results with snips dataset.

| Model | Slot (F1) | Intent (F1) | Intent Acc Sentence level | Sentence Level Accuracy |
|---|---|---|---|---|
| [1] | 88.03 | 97.27 | 97.23 | NA |
| [3] | 94.8 | 98.3 | 98.1 | 87.3 |
| M1 | 96.31 | 98.81 | 98.20 | 91.07 |
| M2 | 96.02 | 98.38 | 97.71 | 91.11 |
| M3 | 95.58 | 97.80 | 97.00 | 89.70 |
| Avg(M1, M2) | 96.40 | 98.84 | 98.30 | 91.30 |
| Avg(M1, M3) | 96.40 | 98.84 | 98.27 | 91.18 |
| Avg(M2, M3) | 96.23 | 98.45 | 98.06 | 91.28 |
| Avg(M1,M2,M3) | 96.50 | 98.88 | 98.21 | 91.70 |
| **Our-Model** | **97.07** | **99.11** | **98.43** | **92.32** |

**TABLE 4.** Evaluation results with mixatis dataset.

| Model | Slot (F1) | Intent (F1) | Intent Acc Sentence level | Sentence Level Accuracy |
|---|---|---|---|---|
| [1] | NA | NA | NA | NA |
| [3] | 88.1 | 81.2 | 75.8 | 44.5 |
| M1 | 89.01 | 83.20 | 76.00 | 48.31 |
| M2 | 89.00 | 82.11 | 76.18 | 48.00 |
| M3 | 88.00 | 81.10 | 75.85 | 44.40 |
| Avg(M1, M2) | 90.01 | 83.00 | 76.11 | 48.52 |
| Avg(M1, M3) | 88.89 | 83.01 | 76.04 | 48.34 |
| Avg(M2, M3) | 88.00 | 82.25 | 76.20 | 48.21 |
| Avg(M1,M2,M3) | 90.11 | 83.27 | 76.28 | 48.43 |
| **Our-Model** | **91.63** | **85.46** | **77.51** | **52.00** |

**TABLE 5.** Evaluation results with mixsnips dataset.

| Model | Slot (F1) | Intent (F1) | Intent Acc Sentence level | Sentence Level Accuracy |
|---|---|---|---|---|
| [1] | NA | NA | NA | NA |
| [3] | 94.5 | 98.6 | 96.5 | 76.4 |
| M1 | 95.05 | 98.80 | 97.00 | 77.00 |
| M2 | 95.00 | 98.70 | 96.59 | 77.03 |
| M3 | 94.51 | 97.60 | 95.85 | 76.08 |
| Avg(M1, M2) | 95.31 | 98.91 | 97.00 | 77.08 |
| Avg(M1, M3) | 95.17 | 98.87 | 97.04 | 77.03 |
| Avg(M2, M3) | 95.11 | 98.81 | 97.00 | 77.03 |
| Avg(M1,M2,M3) | 95.31 | 98.91 | 97.03 | 77.17 |
| **Our-Model** | **95.64** | **99.00** | **97.54** | **78.33** |

The proposed model (denoted as **'Our-Model'**) shows a statistically significant improvement in sentence-level intent prediction and overall 'Sentence-level Accuracy' on ATIS when compared with other baselines. The Intent and slot accuracy at token level is also good (even if it is not

very high). This may be because, after this level, getting a little improvement becomes too tough and even big efforts give some little improvements. See the 'TABLE 2' for the details.

Similarly, 'TABLE 3', shows the result on SNIPS dataset. From the result given in 'TABLE 3', it is clear that, our ensemble model shows significant improvements over state-of-the-art methods.

TABLE 4 shows the result on the MIXATIS dataset. The experimental results of our system are better. However, the overall performance of all systems is low. We have investigated the cause of this problem at the data level. The mixing of two or more utterances with the conjunction 'and' (see the dataset details) has resulted in partial imbalance. This means, after combining, some intents are not getting the proper learning support or not participate in training. This is the major reason behind the little low accuracy of all systems (including baselines).

Similarly, from the results on MixSNIPS dataset, given in 'TABLE 5', it is clear that our system shows the better result. However, compared to the results in Table 3, the performance level is a little low. This is due to the imbalances that resulted after mixing two or more sentences in a slightly different ratio.

From all the above-discussed results, it is also clear that our strategy of using ''stacking ensemble'' based strategy shows better performance compared to the model averaging developed by taking two and three trained models on the full dataset.

## V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a ''stacking ensemble'' based strategy for multi-intent SLU. To prepare models for multi-intent cases, we have used a diverse nature of state-of-the-art NLP models, like - BERT, XLNeT, and Elmo. The use of the sequence labeling model at the token level makes the identification of the relation between intents and slots easy. Our strategy to apply stacking ensemble by using the diverse nature models is effective. The experimental results also prove this.

As future work, we are planning to investigate the chances of learning the workflow (order of intents and slots) from the local dataset as unified training models.

## REFERENCES

[1] R. Gangadharaiah and B. Narayanaswamy, ''Joint multiple intent detection and slot labeling for goal-oriented dialog,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 564–569.

[2] G. Tur, D. Hakkani-Tür, and L. Heck, ''What is left to be understood in ATIS?'' in *Proc. IEEE Spoken Lang. Technol. Workshop*, Dec. 2010, pp. 19–24.

[3] L. Qin, X. Xu, W. Che, and T. Liu, ''AGIF: An adaptive graph-interactive framework for joint multiple intent detection and slot filling,'' in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2020, pp. 1–10.

[4] S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson, ''The fourth dialog state tracking challenge,'' in *Dialogues With Social Robots*. Springer, 2017, pp. 435–449.

[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, ''XLNet: Generalized autoregressive pretraining for language understanding,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5753–5763.

[6] P. S. A. Krogh, ''Learning with ensembles: How overfitting can be useful,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 190–196.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ''BERT: Pre-training of deep bidirectional transformers for language understanding,'' 2018, *arXiv:1810.04805*. [Online]. Available: https://arxiv.org/abs/1810.04805

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ''Attention is all you need,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[9] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf

[10] D. Wu, L. Ding, F. Lu, and J. Xie, ''SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling,'' 2020, *arXiv:2010.02693*. [Online]. Available: https://arxiv.org/abs/2010.02693

[11] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, ''Deep learning based text classification: A comprehensive review,'' 2020, *arXiv:2004.03705*. [Online]. Available: https://arxiv.org/abs/2004.03705

[12] J. Howard and S. Ruder, ''Universal language model fine-tuning for text classification,'' in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, vol. 1, Jul. 2018, pp. 328–339, doi: 10.18653/v1/P18-1031.

[13] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, ''A latent semantic model with convolutional-pooling structure for information retrieval,'' in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 101–110.

[14] D. Bahdanau, K. Cho, and Y. Bengio, ''Neural machine translation by jointly learning to align and translate,'' 2014, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473

[15] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, ''Deep belief nets for natural language call-routing,'' in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 5680–5683.

[16] C. Xia, C. Zhang, X. Yan, Y. Chang, and P. Yu, ''Zero-shot user intent detection via capsule neural networks,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 3090–3099.

[17] P. Xu and R. Sarikaya, ''Exploiting shared information for multi-intent natural language sentence classification,'' in *Proc. INTERSPEECH*, 2013, pp. 3785–3789.

[18] B. Kim, S. Ryu, and G. G. Lee, ''Two-stage multi-intent detection for spoken language understanding,'' *Multimedia Tools Appl.*, vol. 76, no. 9, pp. 11377–11390, May 2017.

[19] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, ''Slot-gated modeling for joint slot filling and intent prediction,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, Jun. 2018, pp. 753–757.

[20] C. Li, L. Li, and J. Qi, ''A selfattentive model with gate mechanism for spoken language understanding,'' in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2018, pp. 3824–3833.

[21] Y. Liu, F. Meng, J. Zhang, J. Zhou, Y. Chen, and J. Xu, ''CM-Net: A novel collaborative memory network for spoken language understanding,'' 2019, *arXiv:1909.06937*. [Online]. Available: https://arxiv.org/abs/1909.06937

[22] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, ''A stack-propagation framework with token-level intent detection for spoken language understanding,'' in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2019, pp. 1–10.

[23] C. Zhang, Y. Li, N. Du, W. Fan, and P. Yu, ''Joint slot filling and intent detection via capsule neural networks,'' in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1–9.

[24] P. Niu, Z. Chen, and M. Song, ''A novel bi-directional interrelated model for joint intent detection and slot filling,'' in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5467–5471.

[25] Y. Wang, Y. Shen, and H. Jin, ''A bi-model based RNN semantic frame parsing model for intent detection and slot filling,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 1–6.

[26] C. Raymond and G. Riccardi, ''Generative and discriminative algorithms for spoken language understanding,'' in *Proc. 8th Annu. Conf. Int. Speech Commun. Assoc.*, 2007, pp. 1–5.

[27] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2014, pp. 189–194.
[28] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "DiSAN: Directional self-attention network for RNN/CNN-free language understanding," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 1–10.
[29] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[30] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, New Orleans, LA, USA, vol. 1, 2018, pp. 2227–2237,
[31] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, Jun. 2014, pp. 55–60.
[32] Q. Chen, Z. Zhuo, and W. Wang, "BERT for joint intent classification and slot filling," 2019, *arXiv:1902.10909*. [Online]. Available: https://arxiv.org/abs/1902.10909
[33] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, pp. 1–23, Jan. 2007.
[34] C. Ju, A. Bibaut, and M. V. D. Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *J. Appl. Statist.*, vol. 45, no. 15, pp. 2800–2818, 2017.
[35] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*. [Online]. Available: https://arxiv.org/abs/1909.11942
[36] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.
[37] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, "A stack-propagation framework with token-level intent detection for spoken language understanding," 2019, *arXiv:1909.02188*. [Online]. Available: https://arxiv.org/abs/1909.02188
[38] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.
[39] C. Beleites, U. Neugebauer, T. Bocklitz, C. Krafft, and J. Popp, "Sample size planning for classification models," *Analytica Chim. Acta*, vol. 760, pp. 25–33, Jan. 2013.
[40] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces," 2018, *arXiv:1805.10190*. [Online]. Available: https://arxiv.org/abs/1805.10190
[41] L. Prechelt and B. O. Geneviève, "Early stopping—But when?" in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), G. Montavon and K.-R. Müller, Eds. Berlin, Germany: Springer, 1998, pp. 53–67.

**NIRAJ KUMAR** received the Ph.D. degree in computer and has more than eight years of experience in industry research and development. His research interests include natural language processing, deep learning, machine learning, and advanced graph algorithms.

**BHIMAN KUMAR BAGHEL** received the master's degree in computer science, and is also working as a Senior Research Engineer with Samsung Research and Development Institute, Bengaluru.