Neighbor Contextual Information Learners for Joint Intent and Slot Prediction

Bharatram Natarajan[*], **Gaurav Mathur**[*] and **Sameer Jain**
research.samsung.com
{bharatram.n, gaurav.m4, sameer.jain}@samsung.com

Abstract

Intent Identification and Slot Identification are two important task for Natural Language Understanding (NLU). Exploration in this area have gained significance using networks like RNN, LSTM and GRU. However, models containing the above modules are sequential in nature, which consumes lot of resources like memory to train the model in cloud itself. With the advent of many voice assistants delivering offline solutions for many applications, there is a need for finding replacement for such sequential networks. Exploration in self-attention, CNN modules has gained pace in the recent times. Here we explore CNN based models like Trellis and modified the architecture to make it bi-directional with fusion techniques. In addition, we propose CNN with Self Attention network called Neighbor Contextual Information Projector using Multi Head Attention (NCIPMA) architecture. These architectures beat state of the art in open source datasets like ATIS, SNIPS.

1 Introduction

Intelligent Voice Assistant like Samsung Bixby, Google Assistant, Amazon Alexa and Microsoft Cortana are increasingly becoming popular. These assistants cater to the user request by extracting the user intention from the spoken utterance. NLU express the user intention in terms of intent label and slot tags. There is one intent label for entire utterance, which signifies unique action to execute. Whereas slots are tags given to tokens in utterance, which signifies extra information required to execute the unique action. Slot tags are denoted in IBO format. For example, consider the utterance "what flights are available from Denver to San Francisco". We denote Intent label as "atis_flight". We denote Slot information as "O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name". There is one slot tag for every token in the utterance, where "O" represents that token is not a slot and B-fromloc.city_name represents that token is "Beginning of slot fromloc.city_name" and "B-toloc.city_name I-toloc.city_name" represents that corresponding tokens are Beginning and Continuation of slot toloc.city_name respectively. We consider Intent Identification as Classification task and Slot tagging as sequence labelling task where we predict slot for each word.

Lots of work has happened in this area. Initially, research community explored both intent and slot tasks as independent task. Different techniques were explored for intent classification. Firdaus et al. (2018) created ensemble model by combining learnings of CNN, LSTM and GRU using multilayer perceptron to enhance intent classification. Kim et al. (2016) proposed enriched word embedding for making similar words together and dissimilar words farther, which aided intent detection better. Yolchuyeva et al. (2020) proposed use of self-attention for enhancing intent classification by capturing long-range and multi-scale dependency in data.

Similarly for slot tagging, Kurata et al. (2016) proposed use of label dependencies along with input sentence for enhancing slot learnings using LSTM. Mesnil et al. (2014) suggested use of Recurrent Neural Network for slot tagging task. Ngoc Thang Vu (2016) proposed novel CNN architecture for slot tagging task, which also used past and future words information.

These individual models approach resulted in pipelined design of NLU. Where intent model will predict intent and they use intent output in slot model to predict slot. This resulted in error propagation i.e. error in intent output resulted in slot tagging errors. In addition, the intent and slot learnings are not available for each other to enhance

[*]Authors contributed equally

Proceedings of 17th International Conference on Natural Language Processing: Workshop, pages 1–9
Patna, India, December 18 - 21, 2020.

each task learning.

To circumvent the above limitation, research community started exploring unified model where they identify both intent classification and slot tagging. Liu and Lane (2016) proposed attention as RNN encoder as input to Decoder for predicting slot and intent by the decoder. Tingting et al. (2019) proposed exploration of attention with Bi-LSTM for joint intent and slot prediction. Firdaus et al. (2019) suggested usage of CNN and RNN for contextual understanding of the utterance along with CRF for label dependency to predict intent and slot. Hardalov et al. (2020) proposed intent pooling attention along with word features on top of BERT for predicting intent and slot. The above approaches addresses both the task using single unified network where both the learnings are propagated to single network.

Further, exploration of parallel learning networks, using common base module, with fusing of intent and slot learnings have gained momentum. Goo et al. (2018) suggested slot gate mechanism to fuse the intent attention learning with slot attention learning to predict slot along with intent. E et al. (2019) proposed a novel iteration mechanism to fuse the meanings of intent and slot subnet for predicting intent and slot.

Inspired by the work on parallel learning networks, where we address the intent and slot tasks learnings by each parallel network, we are proposing a novel architecture, Neighbor Contextual Importance Projector (NCIP) for learning the word importance in its immediate vicinity to boost its importance learning in the overall utterance. We use parallel Multi head attention on top of NCIP to project importance phrases for each task, to predict intent and slot. We are also exploring Trellis network, which learns immediate neighbors in a layer and entire utterance in multiple such layers. In addition, the weights of the network is shared in both temporal direction as well as across layers. Hence, we are proposing a bi-directional Trellis network with different fusion techniques, like Linear Fusion, Concatenation, to predict intent and slot to mimic bi-directional LSTM or GRU.

We organize rest of the section as follows. Section 2 describes the Proposed Approaches. Section 3 describes the experimental setup including dataset, metrics used followed by results. Finally, we conclude and suggest future work and extensions.

2 Proposed Approaches

2.1 Trellis Network Based Architectures

Bai et al. (2018) proposed a novel architecture containing special Temporal Convolutional Neural Networks (T-CNN) for language modeling (LM) task. In this model, they share weights across all layers and they inject the input to deep layers.

As Trellis network has structural and algorithmic elements from both LSTM & CNN, it achieved state of art in various LM tasks. Therefore, we are exploring extension of the same in intent determination and slot tagging.

Original Trellis Network process the input sequence in forward direction only. We propose bidirectional network with two parallel Trellis network encoders, one for forward pass and one for backward pass, for intent determination and slot tagging.

2.1.1 Utterance Pre-Processing

We tokenize the input utterance into words. If the number of words is less than seq_len (seq_len is obtained by taking maximum of the count of words for each utterance in the dataset), we pad that utterance with "PAD" word. We convert words to index using dictionary of unique training words to incremental index. If we do not find any word in the dictionary, than assign the index of "unk" (specially added token into dictionary to handle unseen words). We convert every utterance of training batch into list of indices to generate training data of shape [bs, seq_len]. Where 'bs' is batch size.

For bidirectional Trellis network, we first reverse the training utterances than apply same preprocessing as on original utterances.

2.1.2 Unidirectional Trellis Network Based Architecture

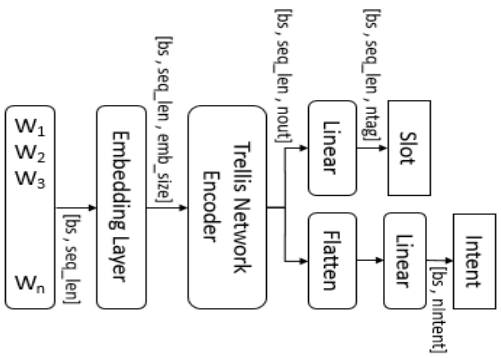

Figure 1: Unidirectional Trellis flow for joint intent and slot prediction

Figure 1 shows unidirectional Trellis encoder based architecture. We pass input data of shape [bs, seq_len] through embedding layer to represent every word with embedding size (emb_size) vector. Trellis Network is used as an encoder to represent input data in [bs, seq_len, nout] dimension. Where nout is a hyper parameter of Trellis network. For slot prediction, output of Trellis network is passed by a linear layer to produce output of shape [bs, seq_len, ntag], where ntag is number of tags.

For intent determination, we first flatten the Trellis output than pass to output linear layer. It produces output of shape [bs, nIntent], where nIntent is number of intent labels.

2.1.3 Bidirectional Trellis Network Based Architecture

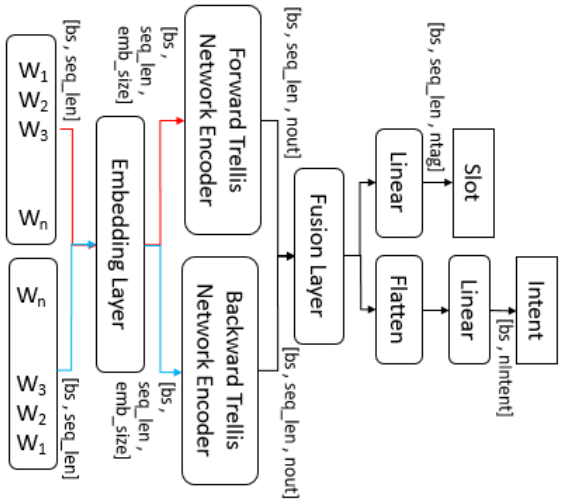

Figure 2: Bidirectional Trellis Flow for joint intent and slot prediction

Figure 2 shows Bi-directional Trellis network. In this, we used two Trellis network encoders, first to process the input sequence as-is and the second to process a reversed copy of the input sequence. Both Trellis encoders generates outputs of dimension [bs , seq_len , nout] . Then we pass both the Trellis network outputs through fusion layer. Fusion layer determines the importance of each Trellis output by selectively propagating the learnings from both the outputs. Then we predict intent by flattening the fused output and passing through dense layer with nintent [distinct intents] as final labels. We also predict slot by passing through dense layer with ntag [distinct tags] as output label. We explore two fusion techniques in Section 2.1.4 and 2.1.5 and Trellis network in 2.1.6.

2.1.4 Masked Flip and Concatenation

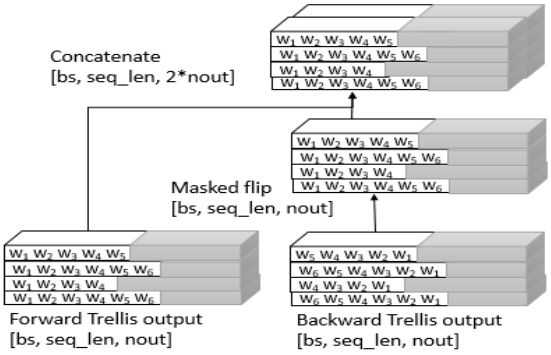

Figure 3: Masked flip and concatenation

Masked flip is implemented by Gardner et al. (2017) in AllenNLP platform. As shown in Figure 3, we first flip the output of backward Trellis (it brings representation of "pad" at beginning) than slice every example of batch on original unpadded length of that example. It gives representation of pad and actual words separately, than again concatenate pad representation at end of word representation. After masked flipping of backward Trellis output, concatenate it with forward Trellis output.

2.1.5 Linear Fusion

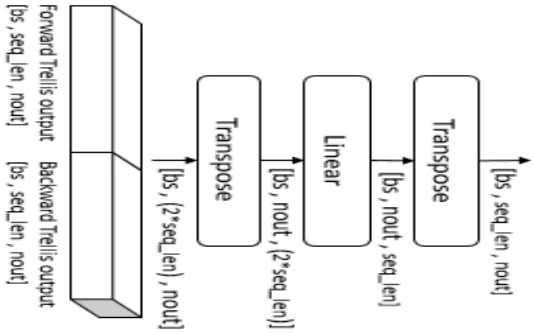

Figure 4: Linear Fusion

As shown in Figure 4, we are using a linear layer to learn joint representation of forward and backward Trellis outputs. For it we first concatenate both outputs at axis 1, then transpose it to bring word representation as final axis and pass by linear layer to bring the final axis to seq_len. We then transpose the same to get seq_len in first axis.

2.1.6 Trellis Network Decoded

As we are using Trellis Network to learn a token representation from its neighbor, let us discuss

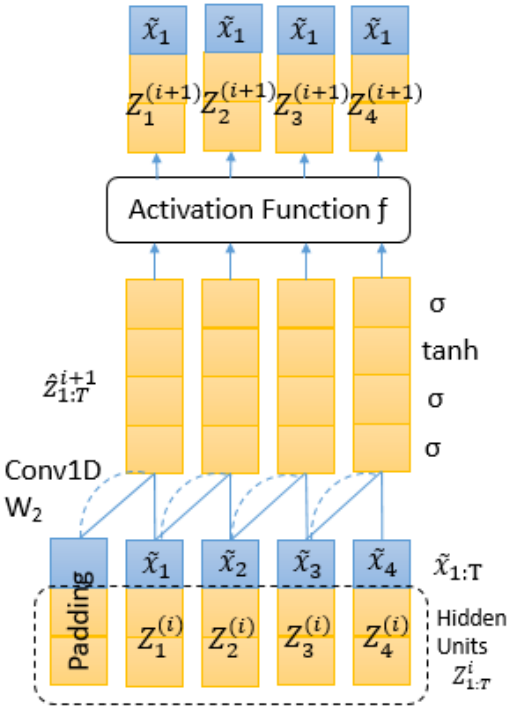

Figure 5: Inter layer transformation of Trellis

about its working. Trellis network is a special form of temporal convolutional neural network (T-CNN) with special structure. It share weights across depth and provide input matrix to all the layers. Figure 5 explains the working of Trellis in intermediate layers. We denote $x_t \epsilon \mathbb{R}_p$ as input embedding vectors at time step t. We denote $Z_1^{T_i} \epsilon \mathbb{R}_q$ as hidden output vector for all-time steps 1 to T and layer i. Hyper parameters of Trellis networks are nhid is hidden size , nout is output size and hsize = nhid + nout

Hidden output $Z_{1:T}^{i+1}$ at for layer i+1 is computed by three steps

First, precompute linear transformation on input x, result will be directly passed to all layers as shown in Equation 1.

$$\tilde{x}_{1:T} = Conv1D(x_{1:T}, W_1) \tag{1}$$

Shape of input x is (bs x emb_size x seq_len) which is transpose of embedding layer output. Shape of W1 is [4* hsize, emb_size, kernel size). Kernel size is fixed to 2. nhid is hidden layer size. Padding is done to keep output as same seq_len as input, $\tilde{x}_{1:T}$ will be of shape [bs, (4* hsize), seq_len]

After computing Pre-activation output $\hat{z}_{1:T}^{i+1}$, it will be divided into four equal parts to apply LSTM style activation function.$\hat{z}_{1:T}^{i+1}$ is kind of considered as concatenation of input gate, output gate, cell state and forget gates. This is the reason number of filters are kept 4*hsize in convolution operations. Conv1D computes Pre-activation output $\hat{z}_{1:T}^{i+1}$ as shown in Equation 2.

$$\hat{z}_{1:T}^{i+1} = Conv1D(z_{1:T}^{i}, W_2) + \ddot{(}x)_{1:T} \tag{2}$$

Shape of previous layer hidden output $z_{1:T}^{i}$ is [bs , hsize , seq_len], which can be initialized all zero for first layer. Shape of W_2 is [4* hsize, hsize , kernel size]. Kernel size is fixed to 2. Padding is done to keep output as same seq_len as input. Pre-activation output $\hat{z}_{1:T}^{i+1}$ will be of shape [bs, 4*hsize , seq_len].

Lastly, we produce Output $z_{1:T}^{i+1}$ by non-linear activation function as shown in Equation 3.

$$z_{1:T}^{i+1} = f(\hat{z}_{1:T}^{i+1}, z_{1:T-1}^{i}) \tag{3}$$

As we discussed, the nonlinear activation is based on LSTM cell equations. The pre-activation output is equally divided in four parts of shape [bs , hsize , seq_len] as shown in Equation 4

$$\hat{z}_{1:T}^{i+1} = [\hat{z}_{1:T,1}^{i+1}, \hat{z}_{1:T,2}^{i+1}, \hat{z}_{1:T,3}^{i+1}, \hat{z}_{1:T,4}^{i+1}] \tag{4}$$

Output $z_{1:T}^{i+1}$ has two parts, computed as shown in Equation 5

$$\begin{aligned} z_{1:T,1}^{i+1} =& \sigma(\hat{z}_{1:T,1}^{i+1}) \odot z_{0:T-1,1}^{i} + \sigma(\hat{z}_{1:T,2}^{i+1}) \\ & \odot tanh(\hat{z}_{1:T,3}^{i+1}) \\ z_{1:T,1}^{i+1} =& \sigma(\hat{z}_{1:T,4}^{i+1}) \odot tanh(z_{1:T,1}^{i+1}) \end{aligned} \tag{5}$$

Therefore, in multiple such layers, Trellis network is learning short and long distance relation in input sequence. From last layer' $z_{1:T}^{i}$, last "nout" representations for every time stamp are passed as final output.

2.2 NCIPMA Network

Figure 6 shows Neighbor Contextual Information Projector with Multi Head Attention. (NCIPMA) architecture. First, we pass the utterance through Utterance Pre-Processing stage explained in 2.2.1.

2.2.1 Utterance Pre-Processing

The input utterance is broken down into chunk of words. If the number of words is less than required maximum number (obtained by taking maximum of the count of words for each utterance in the training data, denoted as max_words), we pad the rest of the words with "PAD" word. We convert words to index using sorted dictionary of unique

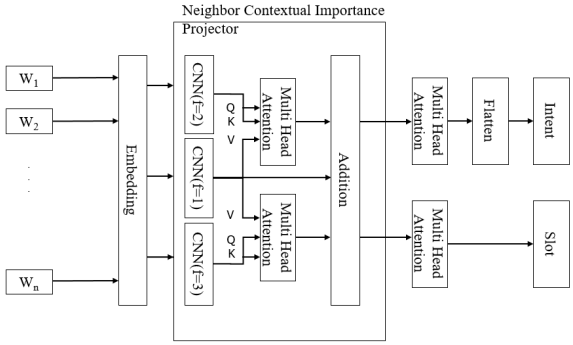

Figure 6: NCIPMA Architecture. Neighbor Contextual Importance Projector enhances the word importance learning by adding the unigram, bi-gram and tri-gram word importance learnings using Multi- Head Attention. Here f=1 means filter size as 1 representing unigram, f=2 means filter size as 2 representing bi-gram and f=3 means filter size as 3 representing tri-gram. MHA means Multi-Head Attention

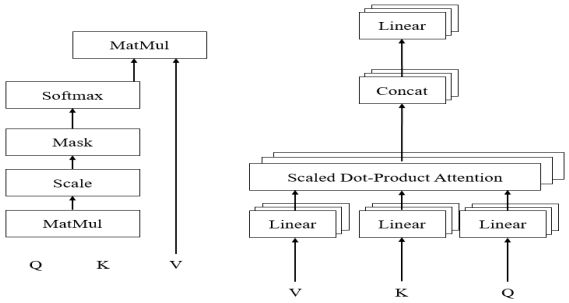

Figure 7: (Left) Scaled Dot-Product Attention. (Right) Multi-Head Attention consist of many scaled dot product attention in parallel.

training words to incremental index. If we do not find the word in the dictionary, then we assign the index of "unkword" (specially added into sorted dictionary to handle unseen words).We map sorted words in dictionary to index from 1 to n (number of words in the dictionary). We use index 0 for "PAD" word. This is the way we convert utterance to list of indices.

We pass the converted list of indices to Embedding layer. During training time, we have trained Embedding layer by masking zero index, passing weight-embedding matrix and making the matrix as trainable. We used unique training words, from sorted dictionary, to construct weight-embedding matrix. We did this by taking word index from sorting dictionary and 300-dimension word embedding from Glove Embedding for each word in unique training words. Hence, Embedding layer provides trained word embedding vector for each word index in the utterance during test time. This creates 3-d matrix of size (1, max_words, 300).

2.2.2 Neighbor Contextual Information Projector(NCIP) Module

In this module, we pass the 3-d matrix (output of utterance pre-processing module) through three parallel CNN layers. Each CNN layer uses filter size as one, two and three respectively capturing unigram, bi-gram and tri-gram word information. To learn each word information importance over other, we keep the word length same by making padding "same".

We capture the importance of uni-gram, bi-gram and tri-gram word information on the uni-gram word information using Multi-Head Attention, as shown in Figure 7. Multi-Head Attention aid in capturing multiple phrases importance in the provided input by passing the same input as query, key and value and calculating the importance as shown in Figure 6. Here, we pass n-gram (uni-gram, bi-gram or tri-gram) word information as query and key. We pass value as unigram word information.

Finally, we add the outputs of all the three Multi-Head Attention Module with unigram output.

This module aids in capturing

- Multi-phrase importance for each word.
- Multi-phrase importance for each phrase made from two to three words as well.

Addition of the above information projects that a word is important even as a part of the phrase and not only as a single word. We pass this information to two parallel Multi-Head Attention Modules.

These parallel Multi-Head Attention (MHA) modules try to learn the importance of phrases for each task in the provided input by passing the input as Query, Key and Value as shown in Figure 7. We predict intent through one MHA module by first flattening the 3 dimensional output, then by passing through dense layer with distinct intent as final hidden dimension. We predict slot through another MHA module, by passing through dense layer with distinct slot as final hidden dimension.

2.3 NCIPMA With CRF

Figure 8 shows the architecture of NCIPMA with CRF. First, we pass the utterance through Utterance Pre-Processing module. We pass the output of the module to NCIP module. We use the output of NCIP module to predict intent and slot.

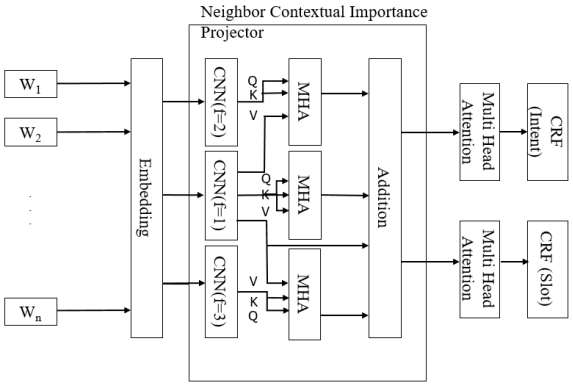

Figure 8: NCIPMA with CRF

For intent, we pass through MHA module to enhance the learning important for intent. Then we pass through Conditional Random Field (CRF) module.

We use linear chain CRF. Linear Chain CRF implements sequential dependencies during prediction. It is a generalization of Hidden Markov Model (HMM) where it solves the chain graph problem. CRF predicts for each word, the most probable intent possible. We then check if we predict the same intent for all the words and consider the intent as pass if we do so otherwise fail.

Similarly, for slot, we pass through MHA module to enhance the learning important for slot. Then we pass through Conditional Random Field (CRF) module. We use same linear chain CRF module. This module now predicts the most probable slot, for each word.

2.4 RASA DIET

RASA has developed a separate neural network framework and **??** proposed DIET architecture for intent classification and slot tagging. Impressed by the framework, we conducted experiments on RASA's DIET architecture. We use a typical Rasa pipeline for our experiments on the DIET classifier, which consists of: 1) Tokenization, 2) Featurization, and 3) Entity Recognition/Intent Classification. The Rasa framework allows for a modular approach in creating a model pipeline. We use a Whitespace tokenizer, followed by a set of supervised embedding featurizers, followed by the DIET components. The DIET architecture has two components: intent classification and slot tagging. For intent classification, it captures a representation of the entire utterance by combining individual token representations and passing the result through a transformer layer. For slot tagging, individual token representations obtained from the transformer layer are further fed into a conditional random field (CRF) layer. Finally, the model optimizes on the total loss obtained by combining intent loss and slot loss.

3 Experiments

We evaluate proposed models on two open source dataset ATIS[1] and SNIPS [1]

3.1 Data

Dataset	Train Data	Valid Data	Test Data	Intent	Slot
ATIS	4478	500	893	21	120
SNIPS	13084	700	700	7	72

Table 1: Dataset Information.

Table 1 shows ATIS and SNIPS dataset information.

Airline Travel Information System (ATIS) dataset contains audio recording of people making flight reservations. It contains 4478 training data, 500 validation data and 893 test data. ATIS dataset is highly skewed in nature. In addition, there are 120 slot labels and 21 intent types present. SNIPS dataset is collected from SNIPS personal voice assistant. It contains 13084 training data, 700 validation data and 700 test data. In addition, there are 7 intents and 72 slots. The complexity of SNIPS dataset is high due to large number of cross-domain intents.

3.2 Training Details

3.2.1 Trellis Network Based Experiments

Trellis network has many hyper-parameters, we majorly experimented with different values of number of layers, embedding size and hidden size. We first identified optimal value of number of layers. Bai et al. (2018) used 55 layers for Word-PTB and 70 layers for Word-WT103 datasets for LM task. For joint intent and slot experimentation, we varied the number of layers from 5 to 20. We found that the experimentation provided best accuracy for 11 layers and started decreasing after 11 layers. Hence, all our experimentation consisted of 11 layers.

We kept embedding size small as compare to original LM task, LM experiments were done with embedding size between 280 and 512, whereas we

[1]https://github.com/MiuLab/SlotGated-SLU/tree/master/data

are keeping embedding size 50 & 100 for different experiments. Hidden size of LM was 1000 to 2000, whereas we are keeping 100 or 120 for different experiments. For all experiments, we are keeping nout (output dimension of Trellis network) same as embedding size.

3.2.2 NCIPMA Network

We use glove embedding of 300 dimensional vector for each seen word and "unk" word embedding (randomly initialized 300 dimensional vector) for unseen words to construct weight matrix for Embedding Module. There are three parallel CNN networks. We use Conv1D module with filter size as 1, 2 and 3 respectively and hidden dimension as 256 with padding "same" feature. The inputs for Multi-Head Attention are having same hidden dimension namely 256. Hence, the output dimension is also 256. Addition module adds the output of the three Multi-Head Attention Module. Hence the dimension size is same as 256. We pass through parallel Multi-Head Attention Module, which does not change the hidden dimensional. Hence, the output dimension for each Multi-Head Attention module is 256. For intent, we flatten the matrix to 2D and pass through "Dense" layer, with intent size as hidden units and activation as "Softmax". For slot, we pass through "Dense" layer, with slot size as hidden units and activation as "Softmax".

We use "Keras" platform with optimizer as "Adam", loss as "categorical crossentropy", batch size as 64 and learning rate as 0.001.

3.2.3 NCIPMA with CRF

All the dimensions used for this experiment is same as NCIPMA network except for intent and slot prediction.

For intent prediction, we use CRF layer with output dimensions as intent size, with mode set to join mode. For slot prediction, we use CRF layer with output dimensions as slot size, with mode set to join mode.

We use "Keras" platform with optimizer as "Adam", loss as "crf_loss", accuracy as "crf_viterbi_accuracy", batch size as 64 and learning rate as 0.001.

3.2.4 RASA DIET

For the DIET model, we use the default architecture suggested by RASA for intent classification and slot tagging without pre-trained embeddings. It consists of two transformer layers of size 256, with 4 attention heads. The learning rate is set to 0.001, batch size to 4, and the dropout to 0.2.

4 Results and Analysis

4.1 Impact of embedding and hidden size on Trellis Network

This section explores the impact of embedding and hidden size on Uni-directional and Bi-directional Trellis network.

4.1.1 Unidirectional Trellis Network

Dataset	Emb Size	Hidden Size	Intent Acc	Slot F1 Score
ATIS	50	100	95.0	94.3
ATIS	100	120	**95.33**	**94.44**
SNIPS	50	100	96.89	81.45
SNIPS	100	120	**98.16**	**83.59**

Table 3: Trellis Network results with different model parameters for ATIS and SNIPS.

Table 3 shows results with ATIS and SNIPS data. On both datasets, Accuracy increases little with increase in embedding and hidden sizes.

4.2 Bidirectional Trellis network

Fusion	Emb Size	Hidden Size	Intent Acc	Slot F1 Score
Linear	50	100	**97.88**	**90.01**
Linear	100	120	97.88	88.57
Concat	50	100	96.89	88.75
Concat	100	120	97.31	89.43

Table 4: Bi Directional Trellis Network results for ATIS and SNIPS.

Table 4 shows results of Bidirectional Trellis with SNIPS data set, slot F1 improves a lot as compare to unidirectional model. But with increase in number of parameters, Bi-directional models are not improving well. This might because of limitations of fusion block. Therefore, there is need for trying different fusion techniques with it.

4.3 NCIPMA Architecture Result and Comparison with State of Art

We evaluate all the proposed architectures on open source dataset like ATIS and SNIPS. Table 2 shows the accuracy comparison of the proposed models

	ATIS		SNIPS	
Architecture	**Intent**	**Slot(f1)**	**Intent**	**Slot(f1)**
NCIPMA	97.87	95.42	**98.57**	91.55
NCIPMA with CRF	**97.87**	**96.25**	98.14	92.35
Unidirectional Trellis Network Based Model	95.33	94.44	98.16	83.59
Bi-directional Trellis Network Based Model	95.11	95.70	97.88	90.01
RASA	95.88	94.47	97.56	**92.91**
Goo et al. (2018)	94.1	95.2	97.0	88.8
E et al. (2019)	97.76	95.75	97.29	92.23

Table 2: Accuracy Comparison with State of the Art.

with each other in addition to state of the art models like Slot gated model (Goo et al., 2018) and Bi-directional Interrelated model (E et al., 2019). From the table, we are able to infer that NCIPMA with CRF model is able to surpass state of the art architecture and other architectures for ATIS and SNIPS. For ATIS, intent accuracy improved by 0.11% and slot accuracy improved by 0.5%. For SNIPS, the intent accuracy improved by 0.85% and the slot accuracy improved by 0.12%. NCIPMA architecture without CRF is able to perform better intent detection for SNIPS by 1.28%. We are able to infer that CRF has boosted the accuracy of NCIPMA architecture because final slot prediction is based on previous labels and current word, which aided in improving the slot prediction. In addition, word level intent prediction for ATIS aides in maintaining the accuracy for intent for ATIS and degraded for slot by 0.43%. This indicates the word-level intent evaluation by CRF aids in maintaining the intent accuracy without much degradation. We are also able to infer that CNN with Self-attention architecture is able to beat models with sequential models like GRU, LSTM. RASA for SNIPS slot is performing the best by beating state of the art by 0.68%.

5 Conclusion

We are able to find a replacement for sequential learning models like LSTM, GRU and RNN by using CNN with self-attention. We are able to see that NCIP module is able to project the importance of uni-gram, bi-gram and tri-gram well. Trellis network based models worked well but further research is required with them to improve on intent classification and slot tagging tasks.

Future scopes are exploration of unified model to predict domain, intent and slot for the said task. Exploration of impact of shared weights across layers for CNN with Self-attention is a needed task to reduce size without impact in accuracy.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471.

Mauajama Firdaus, Shobhit Bhatnagar, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Intent detection for spoken language understanding using a deep ensemble model. In *Pacific Rim international conference on artificial intelligence*, pages 629–642. Springer.

Mauajama Firdaus, Ankit Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2019. A multi-task hierarchical approach for intent detection and slot filling. *Knowledge-Based Systems*, 183:104846.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson HS Liu, Matthew Peters, Michael Schmitz, and Luke S Zettlemoyer. 2017. A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

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

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2020. Enriched pre-trained transformers for joint slot filling and intent detection. *arXiv preprint arXiv:2004.14848*.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 414–419. IEEE.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

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

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Chen Tingting, Lin Min, and Li Yanling. 2019. Joint intention detection and semantic slot filling based on blstm and attention. In *2019 IEEE 4th international conference on cloud computing and big data analysis (ICCCBDA)*, pages 690–694. IEEE.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2020. Self-attention networks for intent detection. *arXiv preprint arXiv:2006.15585*.