# ADAPTING PRETRAINED TRANSFORMER TO LATTICES FOR SPOKEN LANGUAGE UNDERSTANDING

*Chao-Wei Huang and Yun-Nung Chen*

National Taiwan University, Taipei, Taiwan
r07922069@ntu.edu.tw y.v.chen@ieee.org

## ABSTRACT

Lattices are compact representations that encode multiple hypotheses, such as speech recognition results or different word segmentations. It is shown that encoding lattices as opposed to 1-best results generated by automatic speech recognizer (ASR) boosts the performance of spoken language understanding (SLU). Recently, pre-trained language models with the transformer architecture have achieved the state-of-the-art results on natural language understanding, but their ability of encoding lattices has not been explored. Therefore, this paper aims at adapting pre-trained transformers to lattice inputs in order to perform understanding tasks specifically for spoken language. Our experiments on the benchmark ATIS dataset show that fine-tuning pre-trained transformers with lattice inputs yields clear improvement over fine-tuning with 1-best results. Further evaluation demonstrates the effectiveness of our methods under different acoustic conditions[1].

***Index Terms***— Transformer, lattice, spoken language understanding (SLU)

## 1. INTRODUCTION

Spoken language understanding (SLU) aims at parsing spoken utterances into corresponding structured semantic concepts. It plays an important role in spoken dialogue systems, because the whole system may easily fail with the incorrect SLU results. Typically, SLU includes intent detection and slot prediction. For example, a movie-related utterance "*find comedies by James Cameron*" has an intent find_movie and two slot-value pairs (genre, comedy) and (director, James Cameron).

An SLU component is usually implemented in a pipeline manner, where spoken utterances are first transcribed by an automatic speech recognizer (ASR), then the transcripts are parsed by a natural language understanding (NLU) system. One drawback of this approach is that ASR systems may introduce recognition errors, so the following NLU may not be able to capture the correct semantic meaning given the transcribed results. In addition, the incorrect understanding results from ASR errors may be propagated into later stages of the dialogue system, resulting in undesired responses.

To mitigate this problem, previous work tried to design tighter integration of ASR and NLU systems beyond 1-best results. The prior work proposed to utilize word confusion networks (WCNs) as input to NLU systems to preserve information in possible hypotheses [1, 2, 3, 4]. Yaman et al. leveraged n-best lists with similar spirit [5]. With the recent advance in deep learning methods for SLU [6, 7, 8, 9], several solutions were proposed to tackle ASR errors. Masumura et al. examined spoken utterance classification using WCNs with neural networks [10]. Inspired by earlier work that extended recurrent neural networks (RNNs) to tree structures [11, 12, 13], Ladhak et al. proposed a generalized RNN, LatticeRNN, that can process word lattices and achieve better performance for SLU [14]. However, due to the inherently sequential nature of RNNs, the training and inference speed of LatticeRNN is dramatically slower than traditional RNNs.

Recently, language models trained with large generic corpora have shown their ability to transfer knowledge from language modeling to various classification tasks either by providing contextualized features [15] or by fine-tuning pre-trained weights on downstream tasks [16, 17]. Fine-tuning a pre-trained transformer [18] language model with a small amount of parameters yields state-of-the-art results on a variety of language understanding tasks, such as sentiment classification, textual entailment, and question answering.

Although pre-trained transformers have demonstrated their effectiveness of classifying sequential inputs, it is not clear whether they can encode uncertain inputs in the lattice structure effectively. In this work, we aim at exploring the ability of pre-trained transformers to encode and classify lattice inputs. Specifically, our main contributions are 3-fold:

- This paper is the first attempt that extends the pre-trained transformer models to take lattice inputs for the spoken language understanding task.
- This paper conducts experiments on the benchmark ATIS dataset [19, 20, 21] and demonstrates clear improvement over baselines using 1-best transcripts.
- This paper examines the effectiveness of the proposed method under various acoustic conditions and shows that our method yields consistent improvement.

[1]The code is available at https://github.com/MiuLab/Lattice-SLU

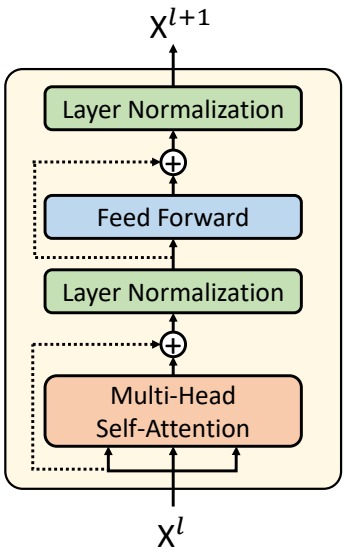



**Fig. 1**. A transformer encoder block

## 2. LANGUAGE MODEL PRE-TRAINING

Language model pre-training has achieved a great success among language understanding tasks with different model architectures. Because training language models requires a large amount of text data, and it is relatively difficult to acquire a lot of lattices, this work focuses on first pre-training language models with the transformer architecture [18] and then adapts the model to support lattice inputs. The pre-training from natural language texts is described below.

### 2.1. Transformer Encoder

We first introduce the transformer encoder model [18], which is the backbone model of our method. The transformer encoder is a stack of $N$ transformer encoder blocks. The $l$-th block takes a sequence of hidden representations $X^l = \{X_1^l, \cdots, X_n^l\}$ as the input and outputs an encoded sequence $X^{l+1} = \{X_1^{l+1}, \cdots, X_n^{l+1}\}$. A transformer encoder block consists of a multi-head self-attention layer and a position-wise fully connected feed-forward layer. A residual connection [22] is employed around each of the two layers followed by layer normalization [23]. An illustration of a transformer encoder block is presented in Figure 1. The detailed components are described as follows.

#### *2.1.1. Positional Encoding*

Because the transformer model relies on a self-attention mechanism with no recurrence, the model is unaware of the sequential order of inputs. To provide the model with positional information, positional encodings are applied to the input token embeddings

$$X_i^1 = \text{embed}_{\text{token}}[w_i] + \text{embed}_{\text{pos}}[i], \tag{1}$$

where $w_i$ denotes the $i$-th input token, $\text{embed}_{\text{token}}$ and $\text{embed}_{\text{pos}}$ denote a learned token embedding matrix and a learned positional embedding matrix respectively.

#### *2.1.2. Multi-Head Self-attention*

An attention function can be described as mapping a query to an output with a set of key-value pairs. The output is a weighted sum of values. We denote queries, keys and values as $Q$, $K$ and $V$, respectively. Following the original implementation [18], a scaled dot-product attention is employed as the attention function. Hence, the output can be calculated as

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2}$$

where $d_k$ denotes the dimension of key vectors.

The idea of multi-head attention is to compute multiple independent attention heads in parallel, and then concatenate the results and project again. The multi-head self-attention in the $l$-th block can be calculated as

$$\text{MultiHead}(X^l) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)W^O, \tag{3}$$

$$\text{head}_i = \text{Attention}(X^l W_i^Q, X^l W_i^K, X^l W_i^V), \tag{4}$$

where $X^l$ denotes the input sequence of the $l$-th block, $h$ denotes the number of heads, $W_i^Q$, $W_i^K$, $W_i^V$ and $W^O$ are parameter matrices.

#### *2.1.3. Position-Wise Feed-Forward Layer*

The second sublayer in a block is a position-wise feed-forward layer, which is applied to each position separately and independently. The output of this layer can be calculated as

$$\text{FFN}(x) = \max(0, x \cdot W_1 + b_1)W_2 + b_2, \tag{5}$$

where $W_1$ and $W_2$ are parameter matrices, $b_1$ and $b_2$ are parameter biases.

#### *2.1.4. Residual Connection and Layer Normalization*

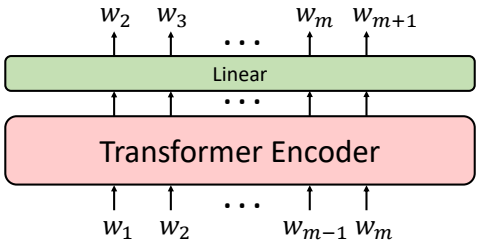
As shown in Figure 1, the residual connection is added around the two sublayers followed by layer normalization. The output of the $l$-th block can be calculated as

$$H^l = \text{LayerNorm}(\text{MultiHead}(X^l) + X^l), \tag{6}$$

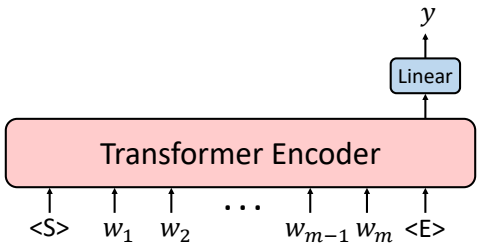
$$X^{l+1} = \text{LayerNorm}(\text{FFN}(H^l) + H^l). \tag{7}$$

### 2.2. Generative Pre-Training Model (GPT)

The generative pre-training (GPT) via a language model objective is shown to be effective for learning representations that capture syntactic and semantic information without supervision [15, 16, 17]. The GPT model proposed by Radford [16] employs the transformer encoder with 12 encoder




(a) The pre-training stage.




(b) The fine-tuning stage.

**Fig. 2**. Illustration of the two-stage method in the GPT model. <S> and <E> are special tokens introduced when fine-tuning.

blocks. It is pre-trained on a large generic corpus that covers a wide range of topics. The training objective is to minimize the negative log-likelihood:

$$\mathcal{L} = \sum_{t=1}^{T} -\log P(w_t \mid w_{<t}, \theta), \tag{8}$$

where $w_t$ denotes the $t$-th word in the sentence, $w_{<t}$ denotes all words prior to $w_t$, and $\theta$ is parameters of the transformer model.

To avoid seeing the future contexts, a masked self-attention is applied to the encoding process. In the masked self-attention, the attention function is modified into

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + M)V, \tag{9}$$

where $M$ is a matrix representing masks. $M_{ij} = -\infty$ indicates that the $j$-th token has no contribution to the output of the $i$-th token, so it is essentially "*masked out*" when encoding the $i$-th token. Therefore, by setting $M_{ij} = -\infty$ for all $j > i$, we can calculate all outputs simultaneously without looking at future contexts.

After the model is pre-trained with a language model objective, it can be fine-tuned on downstream tasks with supervised data. Given a sequence of input tokens $w_1, \cdots, w_m$ along with label $y$, the inputs are passed through the pre-trained encoder to obtain the output of the last token $X_m^{12}$, which is then fed into a linear layer with parameters $W_y$ to make predictions. Note that the linear layer is added in the fine-tuning stage. Figure 2 illustrates this two-stage approach. By fine-tuning on the target tasks, the GPT model has achieved the state-of-the-art performance on various supervised tasks including natural language inference, question answering, semantic similarity and linguistic acceptability [16].

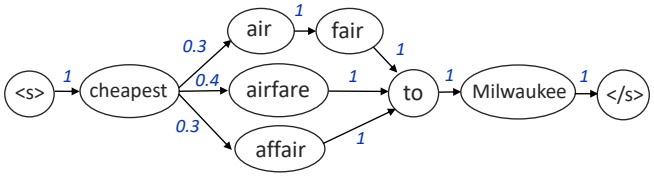



**Fig. 3**. An example lattice representation. Numbers in blue indicate the transition probabilities.

## 3. ADAPTING PRE-TRAINED TRANSFORMER TO LATTICES

As mentioned before, pre-training the language models from large lattice data is difficult. Therefore, our approach focuses on proposing an adaptation method that allows the pre-trained transformer to take lattices as its input. A simple approach of enabling the pre-trained transformers to consume lattice inputs is to flatten the lattice by its topological order and directly apply the sequential transformer model [24]. However, this approach ignores the graph structure entirely, and the resulting sequence may not form a meaningful sentence or may even express the different meaning. Sperber et al. [24] introduced *lattice reachability masks* and *lattice positional encoding* to encode lattices with self-attentional models. In the following subsections, we define the lattices and describe how we integrate the above techniques into the pre-trained transformer model in detail.

### 3.1. Lattices

Lattices represent multiple competing sequences in a compact structure as directed acyclic graphs (DAGs). There is one start node, labeled as <s>, and one end node, labeled as </s>. Each node is labeled as a token. In our case, lattices are outputs of ASR that store multiple decoded hypotheses with uncertainty, and each path from the start node to the end node indicates a possible hypothesis. An example lattice is illustrated in Figure 3.

More formally, let $G = (V, E)$ be a DAG with nodes $V$ and edges $E$. For a node $v \in V$, we denote the set of its predecessors as $\text{Pre}(v)$. For any pair of nodes $v_i$ and $v_j$, $\text{P}(v_j \in \text{Pre}(v_i) \mid v_i)$ represents the conditional probability that a path from start node to end node in $G$ contains $v_j$ as a predecessor of $v_i$, given that $v_i$ appears in the path.

### 3.2. Lattice Reachability Masks

Recall that lattices store multiple hypotheses in a DAG. If we perform self-attention with respect to all nodes in the lattice, the information from different hypotheses may be mixed in an undesired way, because the model is unaware of the global structure of lattices due to the fact that it is pre-trained with sequential data only.

Inspired by the concept about conditioning from TreeLSTM [13] where each node is conditioned on its predecessors, Sperber et al. proposed *lattice reachability masks* to prevent the self-attention mechanism from attending to nodes that are not predecessors of a given query node $v_i$ [24]. There are two variants of masks:

- **Binary masks** restrict the self-attention mechanism to only attend to the predecessors of the query node $v_i$, other nodes are masked:

$$M_{ij}^{bin} = \begin{cases} 0 & v_j \in \text{Pre}(v_i) \text{ or } v_i = v_j, \\ -\infty & \text{otherwise.} \end{cases} \quad (10)$$

- **Probabilistic masks** generalize the binary masks to a probabilistic form. Binary masks ignore the uncertainties in the lattice, so the model considers each node equally disregarding their confidence scores. Therefore, probabilistic masks are designed to bias the attention toward the nodes with higher confidence:

$$M_{ij}^{prob} = \begin{cases} \log P(v_j \in \text{Pre}(v_i) \mid v_i) & v_j \in \text{Pre}(v_i), \\ 0 & v_i = v_j, \\ -\infty & \text{otherwise.} \end{cases} \quad (11)$$

In the pre-training stage, a mask $M$ is used to prevent the tokens from attending to future contexts, where $M_{ij} = -\infty$ for all $j > i$. Notice that the masks $M^{bin}$ and $M^{prob}$ described above are strict generalization of $M$, considering that the successors of the query node is always masked. In order to leverage the information from lattices, this paper proposes to extend the masked self-attention to

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + M^{lattice})V, \quad (12)$$

where $M^{lattice}$ can be either $M^{bin}$ or $M^{prob}$. Therefore, the signal about acoustic confidence from lattices provides additional cues for the model, and it may perform better for spoken language tasks due to the lattice information.

### 3.3. Lattice Positional Encoding

Positional encoding is crucial for transformer models, since it is the only source providing token ordering information. Normally, transformers use either learned or fixed embeddings for each position. When dealing with lattice inputs, determining the position of each lattice node is not straight-forward. Sperber et al. [24] proposed to use the longest-path distance from the start node as a node's position, so the input of the transformer model becomes

$$X_i^1 = \text{embed}_{\text{token}}[w_i] + \text{embed}_{\text{pos}}[\text{ldist}(v_i)], \quad (13)$$

where $\text{ldist}(v_i)$ denotes the longest-path distance from the start node to the node $v_i$ and $w_i$ is the label of $v_i$.

This method that assigns the position to lattice nodes brings several benefits: 1) positions of each lattice path are strictly monotonically increasing, which complies with the setting in pre-training, 2) unnecessary jump between neighboring nodes are avoided, and 3) positions are no larger than the length of the longest hypothesis, so the sequence lengths do not differ much from that in pre-training.

The pre-trained GPT model uses byte-pair encoding to split words into subword units. To follow this tokenization scheme, we split a node in word lattice generated by ASR according to the tokenization of its label. For instance, a node labeled as Milwaukee is split into three nodes that are labeled as Mil, wau, kee respectively. An example of this processing method is shown in Figure 4. After node splitting, we assign positions to the resulting lattice using the *longest-path distance* method.

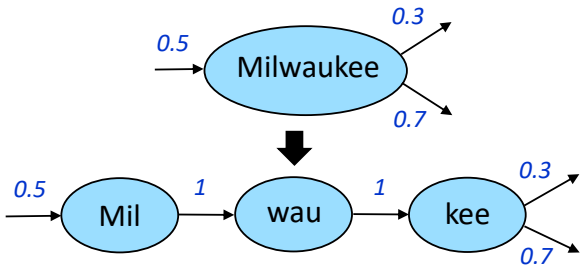



**Fig. 4**. The illustration of the node-splitting method, where the original node with the label `Milwaukee` is split into nodes with tokenized labels.

## 4. EXPERIMENTS

We conduct the experiments on a benchmark SLU task to examine the effectiveness of our method, where we fine-tune the pre-trained transformer model with lattice inputs.

### 4.1. Setup

ATIS (Airline Travel Information Systems) [19, 20, 21] is a widely used dataset for benchmarking SLU research. The dataset contains audio recordings of people making flight reservations or asking flight information with corresponding manual transcripts. The training set contains 4,478 utterances

**Table 1**. F1-scores for multi-label classification and accuracy for utterance-level performance on ATIS (%).

| | | Intent | | Slot | |
|---|---|---|---|---|---|
| | | **F-Measure** | **Accuracy** | **F-Measure** | **Accuracy** |
| Condition 1 (WER=15.5%) | 1-Best Baseline | 97.38 | 96.30 | 93.76 | 76.01 |
| | Lattice-Linearize | 97.98 | 96.90 | 94.56 | 79.21 |
| | Lattice-Probabilistic | 98.19 | **97.21** | **94.65** | 79.45 |
| | Lattice-Binary | **98.23** | 97.15 | **94.65** | **79.93** |
| Condition 2 (WER=26.3%) | 1-Best Baseline | 94.25 | 92.67 | 87.98 | 60.23 |
| | Lattice-Linearize | 94.87 | 93.32 | 88.59 | 62.14 |
| | Lattice-Probabilistic | 95.14 | 93.40 | 89.01 | 62.86 |
| | Lattice-Binary | **95.25** | **93.50** | **89.09** | **63.43** |
| Condition 3 (WER=38.7%) | 1-Best Baseline | 90.64 | 86.40 | 87.31 | 58.59 |
| | Lattice-Linearize | 91.87 | 88.14 | 88.36 | 59.58 |
| | Lattice-Probabilistic | **92.57** | **89.48** | **88.67** | **61.38** |
| | Lattice-Binary | 92.39 | 89.06 | 88.66 | 60.98 |
| Reference | | 98.90 | 98.08 | 95.96 | 87.14 |

and the test set contains 893 utterances. We treat both intent detection and slot prediction as multi-label classification problems, where slot prediction tries to predict what kind of slots appear in an utterance. There are 81 slot labels and 18 intents in the training set.

Our ASR is trained on WSJ [25] using the *s5* recipe from Kaldi [26]. We use the ASR system to recognize audio recordings in ATIS training set and extract lattices for fine-tuning. To simulate different acoustic conditions, we artificially corrupt the recordings with additive noises from MUSAN corpus [27] and simulated room impulse response [28, 29]. As a result, we have three copies of the dataset:

- *Condition 1*: The original dataset. The word error rate (WER) of the ASR results is 15.55% in the test set.
- *Condition 2*: A mildly corrupted version of the original dataset. The WER is 26.30% in the test set.
- *Condition 3*: A severely corrupted version of the original dataset. The WER is 38.69% in the test set.

### 4.2. Model and Training Details

The pre-trained weights of GPT from the original paper are adopted [16]. A linear classifier is placed on top of the outputs of transformers, and we use the output of the last token for prediction.

When fine-tuning GPT, we set the batch size to 8 and use `Adam` as the optimizer [30] with learning rate $6.25 \cdot 10^{-5}$. A linear warm-up schedule is adopted and the whole model is fine-tuned for 5 epochs. We train intent detection and slot prediction models separately.

### 4.3. Results

Table 1 presents the results of intent detection and slot prediction under three acoustic conditions. For each number in the table, we perform ten runs with different random seeds and calculate the average after discarding the best and the worst runs. In each condition, we compare our methods with two baseline systems: **1-best** that takes 1-best transcripts as the input and **lattice-linearize** that takes lattices linearized with topological order.

The results show that fine-tuning pre-trained transformers with lattices significantly outperforms 1-best baseline on both tasks, yielding 32.4%, 15.4% and 20.6% relative error reduction under different acoustic conditions. The two variants of masks perform similarly across acoustic conditions. These results demonstrate that fine-tuning pre-trained transformers with lattices yields clear and consistent improvement for SLU.

In addition, using linearized lattices also improves the performance over the 1-best baseline. This is probably that ATIS is relatively simple, where keywords play an important role for understanding. Therefore, even though the linearized lattices do not form meaningful sentences, they still provide the model with more possible words and achieve the improved performance. Nevertheless, the best performance in the experiments comes from our proposed methods.

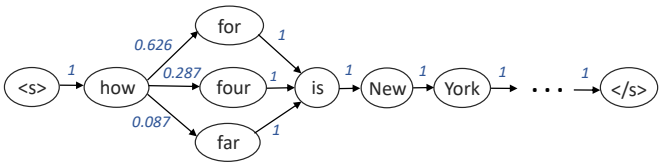
## 5. DISCUSSION AND ANALYSIS

We investigate the sample efficiency and ASR impact of our model and perform qualitative analysis here.

### 5.1. Sample Efficiency

Although lattices provide richer information than 1-best transcripts so the model can reach better performance, it is possible that the model requires more examples to learn the proper mapping between utterances and labels due to the noisy input

**Table 2**. A testing sample in the condition 3.

| | Input | SLU Result |
|---|---|---|
| **Reference** | how far is new york 's la guardia from downtown | *distance* |
| **ASR 1-best** | how for is new york 's la guardia from downtown | None |
| **ASR Lattice** |  | *distance* |



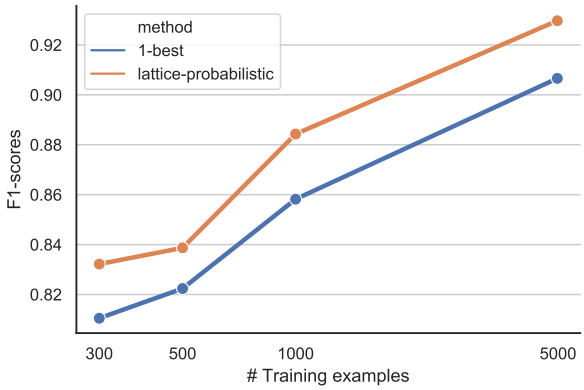



**Fig. 5**. F1 with respect to size of training data.

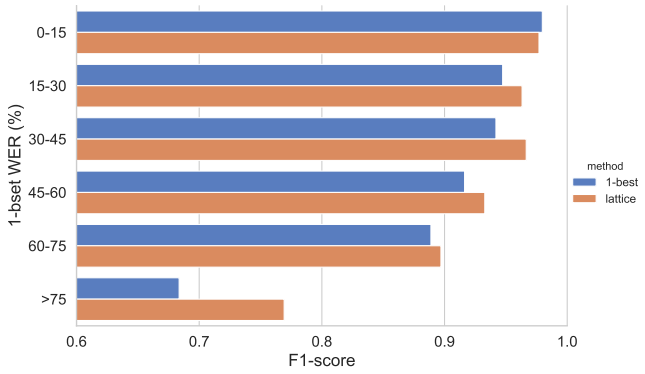



**Fig. 6**. F1 with respect to utterance-level word error rates.

lattices. We conduct experiments with different numbers of training examples to test the sample efficiency of fine-tuning with lattices. We sample $n$ examples randomly from the original training set and use this reduced set to fine-tune our models. All examples are drawn from the condition 3.

Figure 5 plots the results, where fine-tuning with lattices consistently outperforms the 1-best baseline by 2% while the number of training examples varies from 300 to 5000, indicating that our method does not require more training examples to achieve such improvement.

### 5.2. Impact of Transcription Quality

We analyze the SLU performance with respect to WER of ASR. We group the utterances according to their utterance-level WER and present the performance of each group in Figure 6. Aligning with our intuition, using 1-best results achieves slightly better performance when lower WER (<15%). In all other cases, using lattices as inputs outperforms using 1-best, demonstrating that our model is especially suitable for the scenarios with poor ASR results.

### 5.3. Qualitative Analysis

In order to further analyze how lattices help SLU, we sample an example from the test set of the condition 3 shown in Table 2. In this example, the ASR misrecognizes *far* as *for*, so the SLU is unable to understand the intention behind this utterance. It is clear that the correct word, *far*, is in the lattice, so SLU is able to correctly understand that the user is asking about the distance, even though *far* has the lowest probability among alternatives. It justifies the effectiveness of our model.

## 6. CONCLUSION

This paper extends the pre-trained transformer to lattice inputs in order to perform understanding on lattices generated by ASR systems. We leverage lattice reachability masks and lattice positional encoding into the pre-trained transformer model, enabling it to consume lattice inputs during fine-tuning. The experiments on benchmark SLU data demonstrate the effectiveness of our methods under various acoustic conditions[2].

[2]This work was financially supported from the Young Scholar Fellowship Program by MOST in Taiwan, under Grant 108-2636-E002-003.

## 7. REFERENCES


[1] Gokhan Tur, Jerry Wright, Allen Gorin, Giuseppe Riccardi, and Dilek Hakkani-Tür, "Improving spoken language understanding using word confusion networks," in *Seventh International Conference on Spoken Language Processing*, 2002.

[2] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi, and Gokhan Tur, "Beyond asr 1-best: Using word confusion networks in spoken language understanding," *Computer Speech & Language*, vol. 20, no. 4, pp. 495–514, 2006.

[3] Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young, "Discriminative spoken language understanding using word confusion networks," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 176–181.

[4] Gökhan Tür, Anoop Deoras, and Dilek Z. Hakkani-Tür, "Semantic parsing using word confusion networks with conditional random fields," in *INTERSPEECH*, 2013.

[5] Sibel Yaman, Li Deng, Dong Yu, Ye-Yi Wang, and Alex Acero, "An integrative and discriminative technique for spoken utterance classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1207–1214, 2008.

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

[7] Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 554–559.

[8] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al., "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2014.

[9] Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen, "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)*, New Orleans, Louisiana, June 2018, pp. 753–757, Association for Computational Linguistics.

[10] Ryo Masumura, Yusuke Ijima, Taichi Asami, Hirokazu Masataki, and Ryuichiro Higashinaka, "Neural confnet classification: Fully neural network based spoken utterance classification using word confusion networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6039–6043.

[11] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[12] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo, "Long short-term memory over recursive structures," in *International Conference on Machine Learning*, 2015, pp. 1604–1612.

[13] Kai Sheng Tai, Richard Socher, and Christopher D Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1556–1566.

[14] Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister, "Latticernn: Recurrent neural networks over lattices," *Interspeech 2016*, pp. 695–699, 2016.

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[16] Alec Radford, "Improving language understanding by generative pre-training," 2018.

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

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, pp. 6000–6010.

[19] Charles T. Hemphill, John J. Godfrey, and George R. Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, 1990.

[20] Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg, "Expanding the scope of the atis task: The atis-3 corpus," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 43–48.

[21] Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck, "What is left to be understood in ATIS?," in *Proceedings of 2010 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2010, pp. 19–24.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *stat*, vol. 1050, pp. 21, 2016.

[24] Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel, "Self-attentional models for lattice inputs," 2019.

[25] Douglas B. Paul and Janet M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the Workshop on Speech and Natural Language*, Stroudsburg, PA, USA, 1992, HLT '91, pp. 357–362, Association for Computational Linguistics.

[26] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," Tech. Rep., IEEE Signal Processing Society, 2011.

[27] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[28] Jont B Allen and David A Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[29] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[30] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.