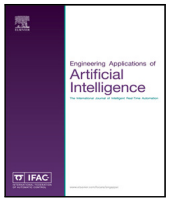
Engineering Applications of Artificial Intelligence 126 (2023) 107013

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



CTRAN: CNN-Transformer-based network for natural language understanding

Mehrdad Rafiepour, Javad Salimi Sartakhti *

Computer and Electrical Engineering Department, University of Kashan, Kashan, Iran

ARTICLE INFO

Keywords:
Natural language understanding
Slot-filling
Intent-detection
Transformers
CNN - Transformer encoder
Aligned transformer decoder
BERT

ABSTRACT

Intent-detection (ID) and slot-filling (SF) are fundamental tasks for natural language understanding. This study introduces a new encoder–decoder CNN-Transformer-based architecture (CTRAN) designed for ID and SF. The encoder integrates of BERT, followed by several convolutional layers with different kernel sizes. We propose using a kernel size of 1 to preserve the one-to-one correspondence between input tokens and output tags. Subsequently, we rearrange the output of the convolutional layer using the window feature sequence and apply stacked Transformer encoders. The ID decoder leverages self-attention and a linear layer, while the SF decoder employs an aligned Transformer decoder with a zero diagonal mask, facilitating alignment between output tags and input tokens. We evaluate our model on ATIS and SNIPS datasets, achieving 98.46% and 98.30% F1 score for the SF task, respectively. These results surpass the previous state-of-the-art by 0.64% and 0.99%. Moreover, we compare two strategies: using a language model as the encoder or as word embedding. We find out that the latter strategy yields better results.

1. Introduction

With the rapid growth of smartphones, digital assistants are becoming more involved in our life. Dialogue systems are the backbone of digital assistants. Moreover, goal-driven dialogue systems are used in banking, website customer support, voice assistants in automotive, and travel guidance. Natural language understanding (NLU) is a critical part of a dialogue system. NLU has many real applications in various domains and scenarios, such as (1) Conversational systems (Liu et al., 2021): NLU enables systems to interact with users in natural language, such as chatbots, virtual assistants, voice assistants, and dialogue systems. NLU can help these systems to understand the user's request, query, feedback, or command and provide appropriate responses or actions. (2)Information extraction (Witte and Cimiano, 2022): NLU can help these systems to identify entities, relations, events, facts, opinions, sentiments, etc. (3) Question answering (Namazifar et al., 2021): NLU can help these systems to analyze the question type, identify the keywords and constraints, retrieve relevant information from various sources, and generate natural language answers. (4) Machine translation (Samant et al., 2022): NLU can help these systems to understand the source language, preserve the meaning and context, and generate natural language output in the target language. (5) Text summarization (Reshamwala et al., 2013): NLU assists these systems in recognizing the main points and themes of the text source and producing natural language summaries that reflect the core of the original text. NLU comprises two tasks. The first is to understand the intent that the user has in mind, and the second is to extract the semantic information from the sentence. The first task is called intent-detection (ID), and the latter is referred to as slot-filling (SF). ID can be defined as a classification problem, and SF can be considered a sequence labeling problem. Table 1 shows an example of the airline travel information system (ATIS) dataset containing the user's question, intent, and target labels.

We spotted three significant issues affecting the performance of the previous models. The first problem is that most of the literature has adopted Long short-term memory (LSTM) in their proposed networks (E et al., 2019; Kane et al., 2020), even though LSTM fails to capture the deep bidirectional meaning of a sentence (Devlin et al., 2019). Furthermore, LSTM-dependent models are prone to gradient vanishing and have a slow training process (Vaswani et al., 2017). The second issue is that previous works used convolutional layers with kernel sizes of 2 or more (Wang et al., 2018; Zhou et al., 2015). Kernel size determines the span of the words that are fused to create a single vector, representing all of the words in that span. We argue that while the output of the convolutional layer may still have the embedding of the word present, the one-to-one relation of input to output tokens needs to be explicit. The third issue is that Bidirectional Encoder Representation from Transformers (BERT) is mainly used as

* Corresponding author.
E-mail addresses: mehrdad.rafie.p@gmail.com (M. Rafiepour), salimi@kashanu.ac.ir (J.S. Sartakhti).

https://doi.org/10.1016/j.engappai.2023.107013
Received 29 September 2022; Received in revised form 16 August 2023; Accepted 17 August 2023
Available online 4 September 2023

Table 1
Example input, target label and intent in ATIS dataset.

user's question	what	is	the	abbreviation	d10
target label	O	O	O	O	B-aircraft_code
target intent	atis_abbreviation				

an encoder, and the embeddings provided by BERT are not further encoded in most existing NLU models. However, a task-specific encoder may benefit the model's efficacy. In this paper, we propose a novel CNN-Transformer-based (CTRAN) encoder–decoder network. For the encoder, we experiment with both BERT and ELMo as our word embedding. We use CNN on word embeddings and restructure its output using the window feature sequence. The final part of the encoder is stacked Transformer encoders. The decoder uses self-attention and a linear layer to classify the user's intent. We propose the aligned Transformer decoder for the SF task, followed by a fully connected layer. To address the first issue in the literature, we avoided LSTM in our network. For the second issue, we introduced the kernel size of 1 into our CNN and proposed alignment in the Transformer decoder. To address the third issue, we investigated the best strategy for using a language model in the architecture of an NLU model. In short, our contribution has four folds.

1. We propose a novel joint intent-detection and slot-filling architecture for natural language understanding, achieving state-of-the-art for slot-filling on both ATIS and SNIPS.
2. We introduce alignment in the Transformer decoder to keep the one-to-one relationship between input tokens to output tags.
3. We propose convolutional layers with the kernel size of 1 to preserve the original one-to-one relationship while fusing the word embedding with adjacent words.
4. We employ the language model solely as word embeddings, demonstrating the superior performance of this approach compared to utilizing the language model as an encoder.

The rest of this paper is ordered as follows: We discuss related works in Section 2. Section 3 describes our proposed network. In Section 4, we specify our experiment setup. Section 4.3 contains our results and analysis, and in Section 5 , we conclude our study and discuss future work.

2. Related work

Although there are several statistical models, neural networks have proven to be more accurate. In some of the literature, ID and SF tasks are carried out separately (Yang et al., 2023; Trewhela and Figueroa, 2023). However, recent studies have shown that the joint training of two tasks yields a better result (Wang et al., 2018; Goo et al., 2018; Zhang, 2018; Qin et al., 2019).

Traditional models mostly used pre-trained fixed vector embeddings such as GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013) for the word representation. For example, CoBiC used CNN to process the embeddings of the input tokens (Kane et al., 2020). Furthermore, The output of the CNN is fed to a bi-directional LSTM, and then a conditional random field (CRF) layer is used to generate the output labels. Lastly, CoBiC used the hidden state of the last unit to produce the intent probabilities. E et al. (2019) proposed Reinforce Vector as a solution for intent-slot integration. They utilized a shared Bi-LSTM as the encoder. Moreover, they concatenated the last hidden state of the encoder with their novel Intent Reinforce Vector, followed by softmax for intent prediction. Furthermore, Slot Reinforce Vector is concatenated with the encoder's hidden state, which is then fed to a CRF layer for slot generation.

In recent years, pre-trained language models have become progressively adopted as they have proven beneficial for many downstream tasks (Qiu et al., 2020). NLU models either utilize the language model to obtain word embeddings or employ them as their encoder. In cases of language model used as a word embedding, the network utilizes the language model solely to acquire contextualized word embeddings and defines a task-specific encoder. Hence, the role of the language model, in these cases, is to provide robust word embeddings and help with the out-of-vocabulary words (Fukuda et al., 2020). For example, Huang et al. (2020) introduced the multi-view encoder, consisting of BERT, followed by several encoders, namely position-wise encoder, local encoder, global encoder, and time-series encoder. Furthermore, they proposed federated learning, meaning the encoder shares parameters between 4 tasks (ID, SF for SNIPS, and ATIS). Hao et al. (2023) utilized the heterogeneous attention mechanism to integrate intent information into word embeddings. Their proposed model comprised BERT alongside a shared encoder, an intent detection module, an intent-slot integration module, and a CRF for slot generation. Although utilizing the attention mechanism in every module of their network, they used LSTM in the shared encoder, exposing the network to the well-known shortcomings of LSTM. Qin et al. (2021) restructured the encoder–decoder into encoder - Co-Interactive module - decoder to build a bidirectional connection between SF and ID. They also used BERT as a word embedding, followed by additional encoding. In cases of the language model as an encoder, the model does not introduce additional encoding and relies on the structure of the language model to capture the logical or hidden relations inside the sentence. Consequently, these types of models rely primarily on fine-tuning the language model. As a result, they are limited to the versatility and prowess of the language model to adapt to a niche task. For example, Chen et al. (2019) used BERT as an encoder, harnessed [CLS] token with a softmax layer for ID, and produced the target slots with a CRF layer. Alternatively, Han et al. (2021) proposed a bidirectional NLU layer, introducing a multi-stage hierarchical process to integrate intent probabilities to slots and vice-versa. The proposed process concatenated initial output probabilities generated by BERT for intent and slots, combining them in a later stage where secondary intent and slot probabilities are generated. However, In the proposed decoder, a simple feed-forward network is used despite the findings of previous research, which showed the positive effect of structured prediction networks (Wang et al., 2020; Chen et al., 2019). Tu et al. (2023) also introduced a bidirectional model where a soft intent is first concatenated into slot vectors. Output probabilities for slots in this model were generated with a biaffine classifier. The final output probabilities of intents were also generated using the predicted slots and the initial soft intent. Wang et al. (2020) utilized BERT and applied the slot-gate mechanism to the outputs of BERT. In this mechanism, the output vector corresponding with the [CLS] token of BERT is integrated into each slot. They also incorporated the self-attention mechanism to contextualize the slot-gate output further and utilized CRF to produce final labels for each slot. However, from a linguistic point of view, concatenating a fixed intent vector with word embeddings does not generate a new meaning for a word, and therefore, performing self-attention does not add additional context to the embeddings. This is backed by their ablation analysis, where removing the slot-gate or attention mechanism creates only a meager difference in performance. Huang et al. (2022) also introduced a simplified, fast attention network that used BERT as an encoder. They introduced label attention, integrating intent information to slots and vice-versa in a bidirectional manner. However, they also did not use a structured prediction network in their decoder and resorted to a feed-forward network. Yang et al. (2021) utilized BERT as an encoder and introduced a novel masked attention-based decoder for the SF task. They calculated intent output probabilities using multihead attention on BERT embedding outputs. Their proposed decoder incorporated masked multiheaded attention, generating keys and values from a concatenation of intent probability output and previously generated tokens while the query was generated from the corresponding position in the encoder.

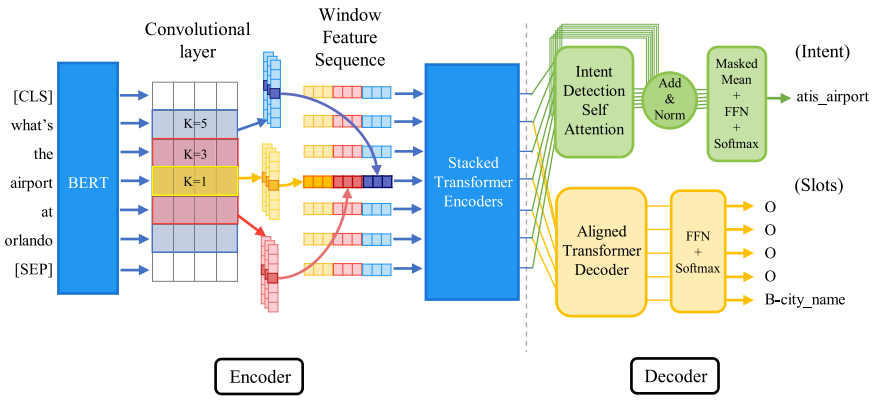


Fig. 1. CTRAN's architecture.

3. CTRAN

In this section, we describe the components of our proposed joint network. Fig. 1 shows a detailed overview of our proposed encoder–decoder architecture. CTRAN comprises three main components: A shared encoder, an SF decoder, and an ID decoder. SF and ID tasks use the same encoder, meaning both tasks use the vectors produced by the encoder, and the subsequent losses are summed before calculating the gradients. Hence, CTRAN benefits from joint learning.

3.1. Shared encoder

The proposed encoder comprises a pre-trained language model, CNN, window feature sequence structure, and stacked Transformer encoders. First, as depicted in Fig. 1, tokens are fed to BERT, which outputs a contextualized embedding. Then, a one-dimensional convolutional operation is performed on the embedded words using various kernel sizes. In this operation, a kernel slides over the word embeddings in one dimension, extracting features from them. Multiple kernel sizes are employed, each with different initial values, resulting in a new representation for the word embeddings. It is usual to use operations like max-pooling after convolutions. However, intending to conserve the original order of tokens, we use window feature sequence, in which we transpose and concatenate the features extracted with different kernels related to each word into a single vector. The final piece is stacked Transformer encoders, which we employ to capture the new representation provided by the window feature sequence structure. Following subsections describe the different parts of our shared encoder.

3.1.1. Pre-trained language model

Pre-trained language models such as ELMo (Peters et al., 2018) and BERT have proven to be effective in improving the performance of the downstream NLP tasks (Ethayarajh, 2019). We employ both language models in our architecture and compare the results. BERT consists of several bi-directional Transformer encoders. $BERT_{base}$ has 12 and $BERT_{large}$ is made of 24 layers. Input to the BERT is the sum of WordPiece token embeddings (Wu et al., 2016), positional embedding, and segment embedding. The latter does not affect our case since we only provide one sentence to the model. For the given input sentence $X = [x_1, x_2, x_3, \dots, x_L]$, BERT provides contextualized embeddings $H = [h_1, h_2, h_3, \dots, h_L]$. Note that WordPiece tokenization treats punctuations as separate tokens, breaking words containing punctuation marks into several tokens. This issue causes token clutter and out-of-order tokens. We solved this issue by removing all punctuation marks. ELMo is comprised of CNN for character embedding and bi-directional LSTM, which are trained on a large corpus. The main idea behind ELMo is that word embeddings are a function of other words in the sentence, meaning the embedding of a word depends on the context of the sentence in which it is used. ELMo is a sequence-to-sequence model that uses non-contextual character-based word representation. The benefit of such representation is the ability to deal with word misspellings and unseen words.

3.1.2. Convolutional layer

In Wang et al. (2018) and Zhou et al. (2015), CNN was used to cover the shortcomings of LSTM. We use Transformer encoder instead of an LSTM network. While Transformer rectifies said flaws, we argue that CNN on Transformer can still be beneficial. We use CNN to extract features from a token span, which includes extracting features from a single token. Moreover, utilizing CNN also helps to put an emphasis on the meaning of adjacent tokens, creating a more inclusive local word representation. We utilize several convolutional layers with different kernel sizes and initial values to fuse the embedding of each token with neighboring tokens. Assume that d represents embedding dimension. If the input sentence has L tokens, the input sentence would be defined as $x \in \mathbb{R}^{L\times d}$, and the embedding vector for the ith token is shown by $x_i \in \mathbb{R}^d$. Let k be a filter size and $f \in \mathbb{R}^{k\times d}$ denote a filter-map. Then for each position i in the sentence, there is a window w_i containing k token vectors as $w_i = [x_i, x_{i+1}, x_{i+2}, \dots, x_{i+k-1}]$. $\odot$ is element-wise multiplication, b is bias and A is an activation function. Hence, the convolution operation c_i for each window w_i is given by:

$$c_i = A\left(w_i \odot f + b\right) \tag{1}$$

This operation is performed over all available indexes i, so the result of f convolving over x is represented by $C = [c_1, c_2, c_3, \dots, c_i, \dots, c_L]$. Note that we use zero padding, so the convolutional layer's input and output always have the same length.

3.1.3. Window feature sequence

It is usual to use pooling operations after a convolutional layer, but the discontinuous sampling will lose some of the information, and subsequently, such operations will destroy sequential data. We use window feature sequence to retain the original order of words.

Let V be the total number of filters with different kernel sizes and initial values for the convolutional layer, and j denotes the index referring to the jth filter. R_i defined in Eq. (2), shows final embedding vector of the word i and C_i^j indicates ith element of the jth filter,

$$R_i = C_i^1 \oplus C_i^2 \oplus C_i^3 \oplus \cdots \oplus C_i^j \oplus \cdots \oplus C_i^V \tag{2}$$

Where $\oplus$ represents concatenation, therefore R_i concatenates all of the convolutional values belonging to index i for all of the filter-maps. This operation is done over all indexes until all the word vectors are produced for the input words. These vectors are then fed to stacked Transformer encoders to produce the final word representations.

3.1.4. Transformer encoder

Directly accessing all of the positions in the sentence helps the Transformer encoder overcome information loss. Stacked Transformer encoders consist of several layers where each layer's output is used as the next layer's input. We utilize stacked Transformer encoders to generate a new representation for the words that incorporates an emphasis on the neighboring words. We use 2 layers of Transformer encoders which is smaller relative to BERT, in the light that this layer of the network only has one objective, and the general meaning of the word is provided through BERT. Transformer encoder is the final part of our shared encoder architecture. The vector provided by this layer will be used in both ID and SF decoders.

3.2. Intent-detection decoder

As shown in Fig. 1, ID decoder consists of a self-attention layer with residual connection and layer normalization, followed by a linear feed-forward network with softmax applied for classification. Accordingly, the output for the ID self-attention layer is determined by following:

$$S = e + LayerNorm(MultiHead(e)) \tag{3}$$

Where $e \in \mathbb{R}^{L \times d_{model}}$ is encoder output, and $LayerNorm$ denotes Layer-Normalization. Moreover, $MultiHead(e)$ denotes the $MultiHead$ attention function defined by Vaswani et al. (2017) where Q, K, V are transformed from e. Furthermore, S represents the output of the operation. Finally, probability distribution between intents is computed with Eq. (4).

$$P_{intent} = softmax(W \cdot S + b) \tag{4}$$

Note that b and W are parameters to be learned at training time.

3.3. Slot-filling decoder

The proposed SF decoder is made of two parts: an aligned Transformer decoder and a linear layer to bring down the dimension to the number of tags. Since the input to output tags have a one-to-one relationship, The Transformer decoder needs to be aligned. The proposed alignment happens in the cross-attention segment. In this section, keys and values are transformed from the encoder's output which we call memory. A matrix with the size of (T, S) is needed for masking the memory, where T and S are the target and source length. Since input and output have the same shape, the mask will be in the shape of (S, S). We aligned the Transformer decoder by applying a zero diagonal mask for the memory. Hence, every position in the memory except for $t = s$ is masked out where s is the row index, and t is the column index. Thus, all the keys related to positions other than the position of generating token would change to zero. Consequently, the Transformer decoder only considers the corresponding embedding vector for each predicting token. In order for masking to be applied on $MultiHead$, it is altered to Eq. (5). In this equation, $Q \in \mathbb{R}^{S \times d_k}, K \in \mathbb{R}^{S \times d_k}, V \in \mathbb{R}^{S \times d_v}$ and $M \in \mathbb{R}^{S \times S}$ represent query, key, value and mask matrices, and d denotes dimension.

$$MultiHead_{Masked}(Q, K, V, M) = softmax(\frac{Q.K^T}{\sqrt{d_k}} + M).V \tag{5}$$

Similar to the original Transformer decoder, the aligned Transformer decoder can be divided into three segments based on functionality. Considering Query, Key, and Value transformed from target token embeddings $D \in \mathbb{R}^{S \times d_{model}}$, first segment which utilizes self-attention to perceive the relations between previously generated tokens is defined as following:

$$C^d = D + LayerNorm(MultiHead_{Masked}(Q, K, V, M_{upper})) \tag{6}$$

Note that M_{upper} is a strictly upper diagonal matrix where all entries above the main diagonal are $-inf$. This mask changes the attention score for positions ahead to zero. Thus, the attention mechanism will not be dependent on future tokens.

Considering $K^e \in \mathbb{R}^{S \times d_k}$, $V^e \in \mathbb{R}^{S \times d_v}$ as key and value transformed from memory, the second segment utilizes cross-attention to use information from the encoder is given by:

$$F^d = C^d + LayerNorm(MultiHead_{Masked}(C^d, K^e, V^e, M_{zerodiag})) \tag{7}$$

Note that $M_{zerodiag}$ is a zero diagonal matrix in which the main diagonal is zero and all other entries are $-inf$. When the softmax function is applied, the off-diagonal entries would change to zero. Consequently, only the value vector for corresponding positions in the memory is added to the C^d matrix.

With FFN representing the position-wise feed-forward network, the output of the aligned Transformer decoder is $O^d = LayerNorm(FFN(F^d)) + F^d$. Finally, the tag probability distribution is calculated by $P = softmax(W \cdot O^d + b)$ where both W and b are trainable parameters.

4. Experiments and setup

In this section, the datasets used to evaluate CTRAN are introduced, and then, the hyper-parameters in our experiments are specified. Moreover, performance and runtime of the proposed model is compared with recently published models and an ablation analysis is provided for the proposed model. Finally, time complexity and model robustness are explained.

4.1. Datasets

We conduct experiments on two well-known datasets, namely ATIS and SNIPS, to measure the performance of our proposed network. We train the network on the train set, tune it on the dev set and report the final results on the test set. Some of the tags in the dev and test sets may not appear during training phase. In that case, we have assigned them the 'Unknown' class.

ATIS benchmark (Hemphill et al., 1990) which is widely adopted in the evaluation of NLU models, is a dataset regarding people asking questions about flight information and reservations. It contains 4478, 500, and 893 for train, dev, and test set, respectively. Moreover, ATIS has 127 slot types and 21 intents. Out of the 21 intents, 4 are not present in the training phase and are labeled as 'Unknown' in the test and validation phases. These 4 intents account for 5 instances in the test dataset. Since the network does not see any examples of the 'Unknown' class during training, it cannot learn to recognize it. Therefore the highest possible accuracy for intent detection is 99.44%.

SNIPS benchmark (Coucke et al., 2018) is a dataset derived from SNIPS personal assistant containing 13084, 700, and 700 utterances for training, test, and dev sets, respectively. Furthermore, it encompasses 7 types of intents and 72 types of slots. Importantly, all intents present in the test and validation phases are adequately represented during the training phase. In contrast to ATIS, SNIPS is more complicated, containing several domains and a more extensive vocabulary.

4.2. Experimental settings

Hyper-parameters have an essential role in the performance of a neural network model. Since each component of CTRAN uses a distinct kind of neural network, we used different learning rates α with AdamW optimizer (Loshchilov and Hutter, 2019) for each layer. We adjust the learning rate with the step learning rate scheduler, which reduces the learning rate of each parameter by the decrease rate γ.

In this study, we used five methods to make the model more robust: dropout (Srivastava et al., 2014), gradient clipping (Pascanu et al., 2013), layer normalization, layer removal, and noise addition. Dropout ρ prevents the model from relying on specific features and approximates an ensemble of networks. Gradient clipping is a technique to prevent

Table 2
Hyper-parameters used in training phase. Learning-rate α, scheduler rate γ and dropout probability ρ used for each layer while training our model.

Layer\Parameter	α	γ	ρ
BERT *base*	10^{-4}	0.96	0.1
BERT *large*	10^{-5}	0.96	0.1
ELMo	10^{-4}	0.96	0.5
CNN & Window Feature Sequence	10^{-3}	0.96	0.0
Transformer Encoder	10^{-4}	0.96	0.1
ID Decoder	10^{-4}	0.96	0.5
SF Decoder	10^{-4}	0.96	0.5

exploding gradients in very deep networks by re-scaling gradients so that their norm is at most a specified value. Layer normalization stabilizes the hidden state dynamics and allows faster training. Layer removal eliminates redundant features and shows that the model can perform well with fewer layers. Noise addition shows that the model can handle noisy inputs without losing performance.

Table 2 shows the α, γ and ρ for each layer. We also used gradient clipping with a threshold of 0.5, the optimal value in our experiment, preventing exploding gradients and improving the network's generalization.

We tried 1, 2, 3, [1, 3], [1, 3, 5], [2, 3, 5], and [1, 2, 3, 5] as kernel sizes where brackets denote multiple kernel sizes used. The cumulative filter count is always 512, and the number of filters is evenly spread throughout different kernels.

We ran our model 10 times, each having 50 epochs, and recorded the best results for each task. We reported the median value as the result for each experiment. We chose 50 epochs for training based on the validation performance of the network. We observed that the network reached a plateau after 50 epochs and did not improve significantly with more epochs. Therefore, we decided to use 50 epochs as a reasonable trade-off between accuracy and efficiency.

4.3. Results and analysis

As the models approach near 100% accuracy, improvements become smaller and more difficult to achieve. Table 3 compares our results with current well-known best-performing models on ATIS and SNIPS datasets. CTRAN+BERT$_{\text{large}}$ outperforms the current state-of-the-art model on ATIS with 0.64% improvement and sets a new record in the SF task. Furthermore, for the ID task, our proposed structure showed improvement over Wang et al. (2018), which also used the CNN-window feature sequence.

To assess the generalization capacity of our model, we also applied CTRAN to SNIPS. We improve nearly 1% upon the previous state-of-the-art for the SF task on SNIPS. Additionally, our ID accuracy is comparable to the current state-of-the-art, although it does not surpass it. The overall performance comparison between CTRAN and the literature can be viewed in Fig. 2. The following subsections will delve into the individual contributions of each idea to the enhancement of CTRAN.

4.3.1. The effect of convolutional layer with transformer encoder

Since previous models used CNN to treat the inherent deficiency of the LSTM, the adoption of Transformer encoder may alleviate the importance of the convolutional layer. Our goal for incorporating CNN-window feature sequence with Transformer encoder was to fuse the adjacent token embeddings. Table 4 compares the performance of a Transformer encoder with our proposed CNN-window feature sequence-Transformer Encoder. For both ATIS and SNIPS the SF f1 and ID accuracy improved with CNN-window feature sequence-Transformer encoder architecture. The results corroborate our intuition about the importance of adjacent tokens.

Figs. 3 and 4 show the class activation map (Zhou et al., 2016) for the ATIS and SNIPS datasets, respectively. A class activation map shows

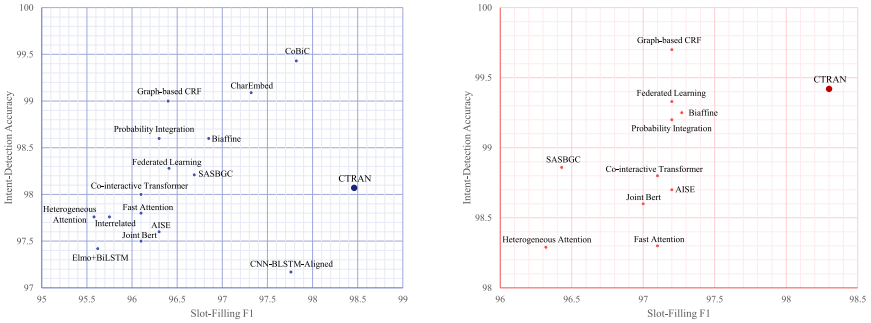
Table 3
A comparison between other well-known models and the proposed model on ATIS and SNIPS datasets. β denotes BERT used.

Model	Dataset			
	ATIS		SNIPS	
	ID acc	SF f1	ID acc	SF f1
SASBGCβ (Wang et al., 2020)	98.21	96.69	98.86	96.43
Joint Bert+CRFβ (Chen et al., 2019)	97.50	96.10	98.60	97.00
CNN-BLSTM-Aligned (Wang et al., 2018)	97.17	97.76	–	–
CharEmbed+CNN-LSTM-CRF (Firdaus et al., 2019)	99.09	97.32	98.24	94.38
Elmo+BiLSTM+CRF (Siddhant et al., 2019)	97.42	95.62	99.29	93.90
Bi-directional Interrelated (E et al., 2019)	97.76	95.75	97.43	92.23
AISEβ (Yang et al., 2021)	97.60	96.30	98.70	97.20
Masked Graph-based CRFβ (Tang et al., 2020)	99.00	96.40	99.70	97.20
Co-interactive Transformerβ (Qin et al., 2021)	98.00	96.10	98.80	97.10
Federated Learningβ (Huang et al., 2020)	98.28	96.41	99.33	97.20
CoBiC (Kane et al., 2020)	99.43	97.82	–	–
Soft intent+Biaffineβ (Tu et al., 2023)	98.60	96.85	99.25	97.27
Fast Attentionβ (Huang et al., 2022)	97.80	96.10	98.3	97.1
Heterogeneous Attentionβ (Hao et al., 2023)	97.76	95.58	98.29	96.32
Bidirectional Probability Integrationβ (Han et al., 2021)	98.60	96.30	99.20	97.20
CTRANβ	**98.07**	**98.46**	**99.42**	**98.30**

Table 4
A comparison between CNN-window feature sequence-Transformer encoder and Transformer encoder only. Experiments were done on $BERT_{base}$.

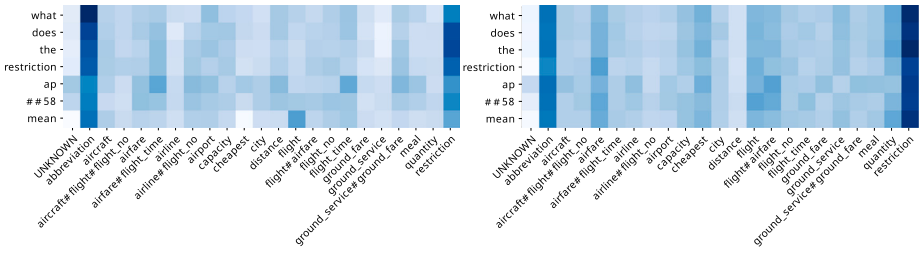
Model	Dataset			
	ATIS		SNIPS	
	ID acc	SF f1	ID acc	SF f1
CNN Window Feature Sequence Transformer Encoder	97.95	98.39	99.42	98.21
Transformer Encoder Only	97.88	98.36	99.01	97.91

the intent distribution for each input word and the contribution of each word towards the prediction of an intent. Notably, within the class activation map, areas of darker shading correspond to elevated values, indicative of the stronger proclivity of a word towards a specific intent. In Fig. 3, Transformer Encoder only architecture errantly predicts the intent as "restriction", while the ground truth is "abbreviation". This can be due to instances present in the training set, namely, the sentence "what is restriction ap 57", which belongs to "restriction". Moreover, the word "mean" in the example sentence changes the intent to "abbreviation", a dependency that the Transformer encoder alone could not capture due to the similarity of the mentioned example and the sentence present in Fig. 3. In contrast, the proposed CNN-



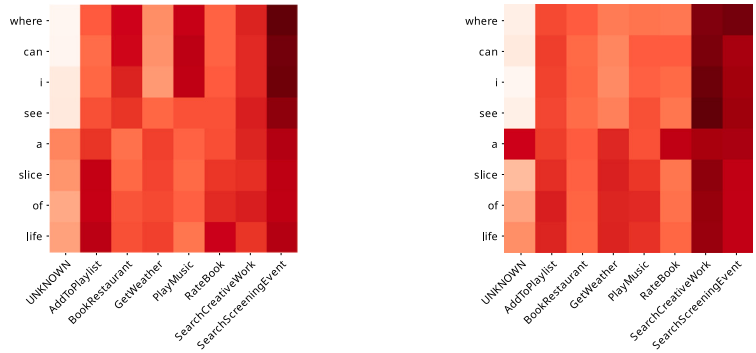
(a) ATIS (b) SNIPS

Fig. 2. A graphical performance comparison between CTRAN and other well-known models.



(a) CNN-Window Feature Sequence-Transformer (b) Transformer Encoder Only

Fig. 3. Class activation map for the ATIS dataset. This figure shows the effect of the proposed CNN-Transformer architecture for the encoder. The input sentence in this example is "what does the restriction ap58 mean". The correct intent for the sentence is "abbreviation". Darker areas represent elevated values.



(a) CNN-Window Feature Sequence-Transformer (b) Transformer Encoder Only

Fig. 4. Class activation map for the SNIPS dataset. This figure shows the effect of the proposed CNN-Transformer architecture for the encoder. The input sentence in this instance is "where can i see a slice of life". The ground truth is "SearchScreeningEvent". Darker areas signify higher values.

Transformer encoder accurately predicts the intent as "abbreviation". The class activation map exhibits significant augmentation of words in the proposed encoder, indicating that the model can predict the correct class based on the sentence structure and the presence of the word "mean".

Similarly, in Fig. 4, the proposed model benefits from a CNN Transformer combination, which can accurately predict "SearchScreening Event" as the intent. Meanwhile, the Transformer encoder seems to have overfitted for the "SearchCreativeWork" across all words.

4.3.2. Kernel size

Kernel size specifies the width in which local semantic information is extracted. Furthermore, kernel size in convolutions can be viewed as n-grams, in that a size of 2 is similar to bigram. To the best of our knowledge, the kernel size of 1 was not used in similar papers which adopted convolutional layers for NLU. We propose 1, to explicitly keep the token embeddings of each word while bringing down the embedding dimension to match the other kernel sizes. This helps the model fuse adjacent word embeddings while keeping explicit embeddings of each word. Table 5 shows the effect of different kernel sizes on the performance of CTRAN. Among single kernel sizes, the proposed kernel size of 1 performs the best for the SF task. The reason is that a kernel size of 1 saves the unigram embeddings, which does not interfere with the one-to-one relation of the input to output tags. Moreover, comparing [2, 3, 5] to [1, 2, 3, 5] confirms our previous statement. Based on our results, we can conclude that using multiple kernel sizes is better than a single kernel size. At the endnote, the proposed kernel [1, 2, 3, 5] achieves the best accuracy in SF and ID tasks.

4.3.3. The transformer decoder alignment

A regular Transformer decoder does not use any key-maskings other than the padding masks in cross-attention segment; meaning, in the

Table 5
The result of different kernel sizes in the convolutional layer. Bracket denotes multiple kernel sizes.

Kernel Size	Dataset			
	ATIS		SNIPS	
	ID acc	SF f1	ID acc	SF f1
1	97.61	98.39	98.69	98.14
2	97.73	98.37	98.84	98.13
3	97.84	98.35	98.84	98.13
[1, 3]	97.84	98.40	98.98	98.15
[1, 3, 5]	97.95	98.43	99.13	98.25
[1, 2, 3, 5]	98.07	98.46	99.13	98.30
[2, 3, 5]	98.07	98.38	98.98	98.20

Table 6
The efficacy of Transformer decoder alignment. Numbers show the SF f1 score of each model. Experiments were conducted on $BERT_{base}$.

Model	Dataset	
	ATIS	SNIPS
Aligned LSTM with attention	98.30	98.10
Regular Transformer decoder	97.42	97.37
Aligned Transformer decoder	98.40	98.21

generation of each target token, the key vector transformed from all context positions are used in the computation of the cross-attention. In contrast, the proposed aligned Transformer decoder, only the value generated from the context corresponding to each predicting token appears in the output of the cross-attention. In order to corroborate the effectiveness of the proposed alignment, we also combined CTRAN's encoder with a regular Transformer decoder, and in another experiment, with an aligned LSTM with the attention mechanism as described by Wang et al. (2018). Table 6 compares the SF f1 of the three experiments in which the proposed alignment shows an average of 0.9% improvement over the regular Transformer decoder. Furthermore, comparing the proposed aligned Transformer decoder with the Wang et al. (2018) SF decoder shows 0.1% improvement, stating it is overall a better architecture for the SF task.

4.3.4. Language model incorporation strategy

Table 7 compares two strategies: language model as an encoder and language model as word embeddings. In the former, we only use CTRAN's decoder after the language model. In the latter, we use both the encoder and decoder of the CTRAN. Also, we experiment with two different language models for each strategy. Using ELMo as the encoder with CTRAN's decoder already surpasses previous SF state-of-the-art on ATIS. We observe an increase in SF f1 and ID accuracy when ELMo is used as word embeddings instead of an encoder. Using $\text{BERT}_{\text{base}}$ as word embeddings with CTRAN, had a better performance on SNIPS when compared to $\text{BERT}_{\text{base}}$ as encoder. In contrast, the language model as word embedding did not do well on ATIS as it did not cause any improvement in results. It may be because ATIS is a smaller dataset; thus, having additional network layers can cause the final model to overfit. For $\text{BERT}_{\text{large}}$ Language model as word embeddings did well on both datasets. Furthermore, our architecture with $\text{BERT}_{\text{large}}$ achieved maximum performance. Although our results indicate that the complete architecture shows superior performance, it introduces extra computational costs. See Appendix for details. Our experiments show that although BERT performs better related to ELMo in all cases, ELMo can be used for domain-specified datasets with nearly the same accuracy while having lower computational complexity and training time. This case may be due to goal-driven datasets not having a diverse vocabulary, thus making the presence of a pre-trained language model less significant. Also $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ do not have an advantage over each other in our implementation for ID task. Their difference is shown in SF, where $\text{BERT}_{\text{large}}$ shows superiority over $\text{BERT}_{\text{base}}$ for all datasets.

4.4. Time complexity analysis

The proposed model has an encoder–decoder architecture. Hence, the time complexity is calculated in encoding and decoding phases, each of which consists of several layers. CTRAN's layers are based on self-attention, Transformer, CNN, and softmax that time complexity of them are represented in Table 8. Since all the layers are placed sequentially, the time complexity of the model will be equal to sum of the time complexity of all the layers.

5. Conclusion and future work

In this paper, we proposed CTRAN a novel CNN-Transformer-based architecture for joint intent-detection and slot-filling. The proposed model uses the encoder–decoder architecture. We use BERT as word embeddings and apply CNN with the window feature sequence structure to fuse local semantic information. Next, stacked Transformer encoders are used to provide the final encoder output. For the intent-detection task, we used self-attention followed by a fully connected layer. Additionally, We used a stack of aligned Transformer decoders for the slot-filling decoder. We compared our network with the well-known models, and the results show that our proposed CNN-Transformer model achieves state-of-the-art on slot-filling task. We also show that using language models as word embeddings is a better strategy than incorporating them into the structure. Using additional encoding after the language model introduces new parameters to the network, which comes with a small computational cost at training time and negligible inference delay. In the future, possible solutions to directly integrate predicted intent to slots and vice-versa can be explored in our architecture. Furthermore, we used pre-trained embeddings in a CNN-Transformer architecture. Future research can train a CNN-Transformer language model from scratch and remove the additional encoder altogether. Also, a more thorough language model comparison might be necessary.

CRediT authorship contribution statement

Mehrdad Rafiepour: Conceptualization, Writing – original draft, Visualization, Software, Validation, Writing – review & editing. **Javad Salimi Sartakhti:** Supervision, Conceptualization, Methodology, Validation, Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Data availability

Data will be made available on request.

Appendix. Computational burden

The computational burden of additional encoding after the language model is a valid concern since more parameters are introduced to the network. Noting that we used a single RTX 3080 10 GB for our computations, Table A.9 shows training time before and after using additional encoding. Our measurements show that between 6%–14% extra time is needed to converge the model. For this calculation, we ran each model 10 times for 10 epochs and reported the median value. Table A.10 shows the inference time of the model. For this experiment, we ran the model for the entirety of the test dataset and reported the mean value for the inference time. Albeit a small gain in training time, the increase in inference time is negligible. Comparing before and after using additional encoding, we observe between 0.7–1.7% extra delay in inference. We anticipated this increase in training and inference time, since it is reasonable for a model with more parameters to have an extra delay.

Table 7
Two strategies used for finding the most suitable model. We used F-score for SF and accuracy for the ID task.

Strategy	Model	Dataset			
		ATIS		SNIPS	
		ID acc	SF f1	ID acc	SF f1
Language model as encoder	ELMo + CTRAN's decoder	97.54	98.17	97.25	96.01
	BERT base + CTRAN's decoder	97.99	98.44	98.86	98.00
	BERT large + CTRAN's decoder	97.99	98.43	98.86	98.18
Language model as word embeddings	CTRAN + ELMo	97.88	98.25	97.73	96.68
	CTRAN + BERT base	97.95	98.40	99.42	98.21
	CTRAN + BERT large	98.07	98.46	99.13	98.30

Table 8
Time complexity of CTRAN's layers. N is sequence length, d is representation dimension, l_b, number of layers in BERT, N_{uf} is number of filter types, N_f number of each filter type, N_I is number of Intents, N_S is number of different slots in dataset.

Layer	Time Complexity
BERT	$O(l_b \times N^2 \times d)$
CNN	$O(N \times d \times N_{uf} \times N_f)$
Window Feature Sequence	$O(1)$
Stacked Transformer	$O(l_{ST} \times N^2 \times (N_{uf} \times N_f))$
ID Self-Attention	$O(N^2 \times (N_{uf} \times N_f))$
Add & Norm	$O(N \times (N_{uf} \times N_f))$
Masked Mean + FFN + Softmax	$O((N_{uf} \times N_f) \times N_I)$
Aligned Transformer Decoder	$O(N^2 \times (N_{uf} \times N_f))$
FFN + Softmax	$O(N \times N_s \times (N_{uf} \times N_f))$

Table A.9
Additional training time caused by the proposed encoder (En.) is applied for each language model (LM). Values show how many seconds it takes for the entire model to be trained for one epoch over the noted dataset. The value in parentheses indicates the percentage increase. LM is short for Language Model, and En. indicates encoding.

LM	Dataset			
	ATIS		SNIPS	
	LM as En.	Additional En.	LM as En.	Additional En.
ELMo	42.51	47.55 (11%)	92.37	103.51 (11%)
BERT base	42.40	46.71 (10%)	110.39	119.50 (8%)
BERT large	73.57	79.44 (7%)	182.42	194.24 (6%)

Table A.10
Comparing inference time before and after additional encoding (En.) is applied for each language model (LM). Values show how many milliseconds it takes for a single example to be inferred. The value in parentheses indicates the percentage increase.

LM	Dataset			
	ATIS		SNIPS	
	LM as En.	Additional En.	LM as En.	Additional En.
ELMo	13.6	13.8 (1.5%)	13.8	14.0 (1.4%)
BERT base	11.7	11.9 (1.7%)	12.7	12.9 (1.5%)
BERT large	12.0	12.2 (1.6%)	13.0	13.1 (0.7%)

References

Chen, Q., Zhuo, Z., Wang, W., 2019. BERT for joint intent classification and slot filling. arXiv:1902.10909.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., et al., 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. pp. 12–16, arXiv preprint arXiv:1805.10190.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186. http://dx.doi.org/10.18653/v1/N19-1423, URL: https://aclanthology.org/N19-1423.

E, H., Niu, P., Chen, Z., Song, M., 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. Association for Computational Linguistics, Florence, Italy, pp. 5467–5471. http://dx.doi.org/10.18653/v1/P19-1544, URL: https://aclanthology.org/P19-1544.

Ethayarajh, K., 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. EMNLP-IJCNLP, Association for Computational Linguistics, Hong Kong, China, pp. 55–65. http://dx.doi.org/10.18653/v1/D19-1006, URL: https://aclanthology.org/D19-1006.

Firdaus, M., Kumar, A., Ekbal, A., Bhattacharyya, P., 2019. A multi-task hierarchical approach for intent detection and slot filling. Knowl.-Based Syst. 183, 104846. http://dx.doi.org/10.1016/j.knosys.2019.07.017, URL: https://www.sciencedirect.com/science/article/pii/S0950705119303211.

Fukuda, N., Yoshinaga, N., Kitsuregawa, M., 2020. Robust backed-off estimation of out-of-vocabulary embeddings. In: Findings of the Association for Computational Linguistics: EMNLP 2020. Association for Computational Linguistics, Online, pp. 4827–4838. http://dx.doi.org/10.18653/v1/2020.findings-emnlp.434, URL: https://aclanthology.org/2020.findings-emnlp.434.

Goo, C.W., Gao, G., Hsu, Y.K., Huo, C.L., Chen, T.C., Hsu, K.W., Chen, Y.N., 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). Association for Computational Linguistics, New Orleans, Louisiana, pp. 753–757. http://dx.doi.org/10.18653/v1/N18-2118, URL: https://aclanthology.org/N18-2118.

Han, S.C., Long, S., Li, H., Weld, H., Poon, J., 2021. Bi-directional joint neural networks for intent classification and slot filling. In: Proceecings of Interspeech 2021. pp. 4743–4747. http://dx.doi.org/10.21437/Interspeech.2021-2044.

Hao, X., Wang, L., Zhu, H., Guo, X., 2023. Joint agricultural intent detection and slot filling based on enhanced heterogeneous attention mechanism. Comput. Electron. Agric. 207 (C), http://dx.doi.org/10.1016/j.compag.2023.107756.

Hemphill, C., Godfrey, J., Doddington, G., 1990. 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, URL: https://aclanthology.org/H90-1021.

Huang, L., Liang, S., Ye, F., Gao, N., 2022. A fast attention network for joint intent detection and slot filling on edge devices. arXiv:2205.07646.

Huang, Z., Liu, F., Zou, Y., 2020. Federated learning for spoken language understanding. In: Proceedings of the 28th International Conference on Computational Linguistics. International Committee on Computational Linguistics, Barcelona, Spain (Online), pp. 3467–3478. http://dx.doi.org/10.18653/v1/2020.coling-main.310, URL: https://aclanthology.org/2020.coling-main.310.

Kane, B., Rossi, F., Guinaudeau, O., Chiesa, V., Quénel, I., Chau, S., 2020. Joint intent detection and slot filling via CNN-LSTM-CRF. In: 2020 6th IEEE Congress on Information Science and Technology. CiSt, pp. 342–347. http://dx.doi.org/10.1109/CiSt49399.2021.9357183.

Liu, X., Eshghi, A., Swietojanski, P., Rieser, V., 2021. Benchmarking natural language understanding services for building conversational agents. In: Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems. Springer, pp. 165–183.

Loshchilov, I., Hutter, F., 2019. Decoupled weight decay regularization. arXiv:1711.05101.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Namazifar, M., Papangelis, A., Tur, G., Hakkani-Tür, D., 2021. Language model is all you need: Natural language understanding as question answering. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 7803–7807.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. ICML '13, JMLR.org, pp. III–1310–III–1318.

Pennington, J., Socher, R., Manning, C., 2014. Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. EMNLP, Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543. http://dx.doi.org/10.3115/v1/D14-1162, URL: https://aclanthology.org/D14-1162.

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L., 2018. Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp. 2227–2237. http://dx.doi.org/10.18653/v1/N18-1202, URL: https://aclanthology.org/N18-1202.

Qin, L., Che, W., Li, Y., Wen, H., Liu, T., 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. arXiv:1909.02188.

Qin, L., Liu, T., Che, W., Kang, B., Zhao, S., Liu, T., 2021. A co-interactive transformer for joint slot filling and intent detection. In: 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 8193–8197. http://dx.doi.org/10.1109/ICASSP39728.2021.9414110.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X., 2020. Pre-trained models for natural language processing: A survey. Sci. China Technol. Sci. 63 (10), 1872–1897. http://dx.doi.org/10.1007/s11431-020-1647-3.

Reshamwala, A., Mishra, D., Pawar, P., 2013. Review on natural language processing. IRACST Eng. Sci. Technol. Int. J. (ESTIJ) 3 (1), 113–116.

Samant, R.M., Bachute, M.R., Gite, S., Kotecha, K., 2022. Framework for deep learning-based language models using multi-task learning in natural language understanding: A systematic literature review and future directions. IEEE Access 10, 17078–17097.

Siddhant, A., Goyal, A., Metallinou, A., 2019. Unsupervised transfer learning for spoken language understanding in intelligent agents. Proc. Assoc. Adv. Artif. Intell. Conf. Artif. Intell. 33 (01), 4959–4966. http://dx.doi.org/10.1609/aaai.v33i01.33014959, URL: https://ojs.aaai.org/index.php/AAAI/article/view/4426.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Tang, H., Ji, D., Zhou, Q., 2020. End-to-end masked graph-based CRF for joint slot filling and intent detection. Neurocomputing 413, 348–359. http://dx.doi.org/10.1016/j.neucom.2020.06.113, URL: https://www.sciencedirect.com/science/article/pii/S0925231220311024.

Trewhela, A., Figueroa, A., 2023. Text-based neural networks for question intent recognition. Eng. Appl. Artif. Intell. 121, 105933. http://dx.doi.org/10.1016/j.engappai.2023.105933, URL: https://www.sciencedirect.com/science/article/pii/S0952197623001173.

Tu, N.A., Xuan Hieu, D., Phuong, T.M., Xuan Bach, N., 2023. A bidirectional joint model for spoken language understanding. In: IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 1–5. http://dx.doi.org/10.1109/ICASSP49357.2023.10096195.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS '17, Curran Associates Inc., Red Hook, NY, USA, pp. 6000–6010.

Wang, C., Huang, Z., Hu, M., 2020. SASGBC: Improving sequence labeling performance for joint learning of slot filling and intent detection. In: Proceedings of 2020 the 6th International Conference on Computing and Data Engineering. In: ICCDE 2020, Association for Computing Machinery, New York, NY, USA, pp. 29–33. http://dx.doi.org/10.1145/3379247.3379266.

Wang, Y., Tang, L., He, T., 2018. Attention-based CNN-BLSTM networks for joint intent detection and slot filling. In: Sun, M., Liu, T., Wang, X., Liu, Z., Liu, Y. (Eds.), Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. Springer International Publishing, Cham, pp. 250–261. http://dx.doi.org/10.1007/978-3-030-01716-3_21.

Witte, C., Cimiano, P., 2022. Intra-template entity compatibility based slot-filling for clinical trial information extraction. In: Proceedings of the 21st Workshop on Biomedical Language Processing. Association for Computational Linguistics, Dublin, Ireland, pp. 178–192. http://dx.doi.org/10.18653/v1/2022.bionlp-1.18, URL: https://aclanthology.org/2022.bionlp-1.18.

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. http://dx.doi.org/10.48550/ARXIV.1609.08144, URL: https://arxiv.org/abs/1609.08144.

Yang, X., Bekoulis, G., Deligiannis, N., 2023. Traffic event detection as a slot filling problem. Eng. Appl. Artif. Intell. 123, 106202. http://dx.doi.org/10.1016/j.engappai.2023.106202, URL: https://www.sciencedirect.com/science/article/pii/S095219762300386X.

Yang, P., Ji, D., Ai, C., Li, B., 2021. AISE: Attending to intent and slots explicitly for better spoken language understanding. Knowl.-Based Syst. 211, 106537. http://dx.doi.org/10.1016/j.knosys.2020.106537, URL: https://www.sciencedirect.com/science/article/pii/S0950705120306663.

Zhang, Y., 2018. Joint models for NLP. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts. Association for Computational Linguistics, Melbourne, Australia, URL: https://aclanthology.org/D18-3001.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A., 2016. Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR.

Zhou, C., Sun, C., Liu, Z., Lau, F.C.M., 2015. A C-LSTM neural network for text classification. arXiv:1511.08630.