PRELIMINARY VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference

Encoding Syntactic Knowledge in Transformer Encoder for Intent Detection and Slot Filling

Jixuan Wang[1*], **Kai Wei**[2†], **Martin Radfar**[2], **Weiwei Zhang**[2], **Clement Chung**[2]
[1] University of Toronto, Vector Institute, [2] Amazon Alexa
jixuan@cs.toronto.edu, {kaiwe, radfarmr, wwzhang, chungcle}@amazon.com

Abstract

We propose a novel Transformer encoder-based architecture with syntactical knowledge encoded for intent detection and slot filling. Specifically, we encode syntactic knowledge into the Transformer encoder by jointly training it to predict syntactic parse ancestors and part-of-speech of each token via multi-task learning. Our model is based on self-attention and feed-forward layers and does not require external syntactic information to be available at inference time. Experiments show that on two benchmark datasets, our models with only two Transformer encoder layers achieve state-of-the-art results. Compared to the previously best performed model without pre-training, our models achieve absolute F1 score and accuracy improvement of 1.59% and 0.85% for slot filling and intent detection on the SNIPS dataset, respectively. Our models also achieve absolute F1 score and accuracy improvement of 0.1% and 0.34% for slot filling and intent detection on the ATIS dataset, respectively, over the previously best performed model. Furthermore, the visualization of the self-attention weights illustrates the benefits of incorporating syntactic information during training.

Introduction

Recent years have seen great success in applying deep learning approaches to enhance the capabilities of virtual assistants (VAs) such as Amazon Alexa, Google Home and Apple Siri. One of the challenges for building these systems is mapping the meaning of users' utterances, which are expressed in natural language, to machine comprehensible language (Allen 1995). An example is illustrated in Figure 1. In this utterance "*Show the cheapest flight from Toronto to St. Louis*", the machine needs to map this utterance to an intent *Airfare* (intent detection) and to slots such as *Toronto: FromLocation* (slot filling). In this work, we focus on intent detection and slot filling and refer to these as Natural Language Understanding (NLU) tasks.

Previous works show that a simple deep neural architecture delivers better performance on NLU tasks when compared to traditional models such as Conditional Random Fields (Collobert et al. 2011). Since then, deep neural architectures, predominantly recurrent neural networks,

Slots	O	O	O	O	B-fromloc	O	B-toloc	I-toloc
Utterance	*Show*	*the*	*flights*	*from*	*Toronto*	*to*	*St.*	*Louis*
Intent	SearchFlight							

Figure 1: An example of NLU tasks.

have become an indispensable part of building NLU systems (Zhang and Wang 2016; Goo et al. 2018; E et al. 2019). Transformer-based architectures, as introduced more recently by (Vaswani et al. 2017), have shown significant improvement over previous works on NLU tasks (Chen, Zhuo, and Wang 2019; Qin et al. 2019). Recent studies show that although the Transformer model can learn syntactic knowledge purely by seeing examples, explicitly feeding this knowledge to such models can significantly enhance their performance on tasks such as neural machine translation (Sundararaman et al. 2019) and semantic role labeling (Strubell et al. 2018). While incorporating syntactic knowledge has been shown to improve performance for NLU tasks (Tur et al. 2011; Chen et al. 2016), both of these assume syntactic knowledge is provided by external models during training and inference time.

In this paper, we introduce a novel Transformer encoder-based architecture for NLU tasks with syntactic knowledge encoded that does not require syntactic information to be available during inference time. This is accomplished, first, by training one attention head to predict syntactic ancestors of each token. The dependency relationship between each token is obtained from syntactic dependency trees, where each word in a sentence is assigned a syntactic head that is either another word in the sentence or an artificial root symbol (Dozat and Manning 2016). Adding the objective of dependency relationship prediction allows a given token to attend more to its syntactically relevant parent and ancestors. In addition to dependency parsing knowledge, we encode part of speech (POS) information in Transformer encoders because previous research shows that the POS information can help dependency parsing (Nguyen and Verspoor 2018). The closest work to ours is (Strubell et al. 2018). However, they focused on semantic role labeling and trained one attention head to predict the direct parent instead of all ancestors.

We compare our models with several state-of-the-art neu-

[*] Work done during author's internship at Amazon Alexa.
[†] Corresponding author.

ral NLU models on two publicly available benchmarking datasets: the ATIS (Hemphill, Godfrey, and Doddington 1990) and SNIPS (Coucke et al. 2018) datasets. The results show that our models outperform previous works. To examine the effects of adding syntactic information, we conduct an ablation study and visualize the self-attention weights in the Transformer encoder.

Problem Definition

We define intent detection (ID) and slot filling (SF) as an utterance-level and token-level multi-class classification task, respectively. Given an input utterance with T tokens, we predict an intent $y^{int.}$ and a sequence of slots, one per token, $\{y_1^{slot}, y_2^{slot}, \ldots, y_T^{slot}\}$ as outputs. We add an empty slot denoted by "O" to represent words containing no labels. The goal is to maximize the likelihood of correct the intents and slots given input utterances.

Proposed Model

We jointly train our model for NLU tasks (*i.e.*, ID and SF), syntactic dependency prediction and POS tagging via multi-task learning (Caruana 1993), as shown in Figure 2. For dependency prediction, we insert a syntactically-informed Transformer encoder layer after the $(x+y)$th layer. In this encoder layer, one attention head is trained to predict the full ancestry for each token on the dependency parsing tree. For POS tagging, we add a POS tagging model that shares the first x Transformer encoder layers with the NLU model. We describe the details of our proposed architecture below.

Input Embedding

The input embedding model maps a sequence of token representations $\{t_1, t_2, \ldots, t_T\}$ into a sequence of continuous embeddings $\{e_0, e_1, e_2, \ldots, e_T\}$, with e_0 being the embedding of a special start-of-sentence token, "[SOS]". The embeddings are then fed into the NLU model.

Transformer Encoder Layer

The Transformer encoder layers are originally proposed in (Vaswani et al. 2017). Each encoder layer consists of a multi-head self-attention layer and feed forward layers with layer normalization and residual connections. We stack multiple encoder layers to learn contextual embeddings of each token, each with H attention heads. Suppose the output embeddings of the encoder layer $j-1$ is $E^{(j-1)}$, each attention head h at layer j first calculates self-attention weights by the scaled dot product (1).

$$A_h^{(j)} = softmax(\frac{Q_h^{(j)} {K_h^{(j)}}^T}{\sqrt{d_k}}) \quad (1)$$

In (1), the query $Q_h^{(j)}$ and key $K_h^{(j)}$ are two different linear transformations of $E^{(j-1)}$, d_k is the dimension of the query and the key embeddings. The output of the attention head h is calculated by:

$$F_h^{(j)} = A_h^{(j)} V_h^{(j)} \quad (2)$$

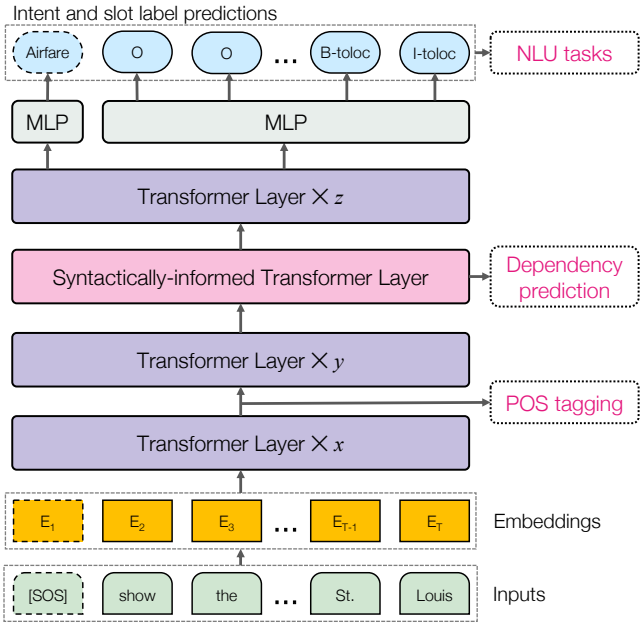

Figure 2: A high level overview of the proposed architecture. Note that x, y and z all refer to number of layers that can vary depending on implementation. "MLP" refers to a multi-layer perceptron (MLP).

in which the value $V_h^{(j)}$ is also a linear transformation of $E^{(j-1)}$. The outputs of H attention heads are concatenated as the self-attended token representations, followed by another linear transformation:

$$Multi^{(j)} = [F_1^{(j)}, F_2^{(j)}, \cdots, F_H^{(j)}]W^F \quad (3)$$

which is fed into the next feed forward layer. Residual connections and layer normalization are applied after the multi-head attention and feed forward layer, respectively.

Encoding Syntactic Dependency Knowledge

As shown in Figure 3, the syntactically-informed transformer encoder layer differs from the standard Transformer encoder layer by having one of the H attention heads trained to predict the full ancestry for each token, *i.e.*, parents, grandparents, great grandparents, *etc*. Different from (Strubell et al. 2018), we use full ancestry prediction instead of just direct parent prediction. Later we will demonstrate the benefits of our approach in the Results section.

Given an input sequence of length T, the output of a regular attention head is a $T \times T$ matrix, in which each row contains the attention weights that a token puts on all the tokens in the input sequence. The output of the syntactically-informed attention head is also a $T \times T$ matrix but this attention head is trained to assign weights only on the syntactic governors (*i.e.*, ancestors) of each token.

To train this attention head, we define a loss function by the difference between the output attention weight matrix of the syntactically-informed attention head and a predefined prior attention weight matrix. The prior attention weight ma-

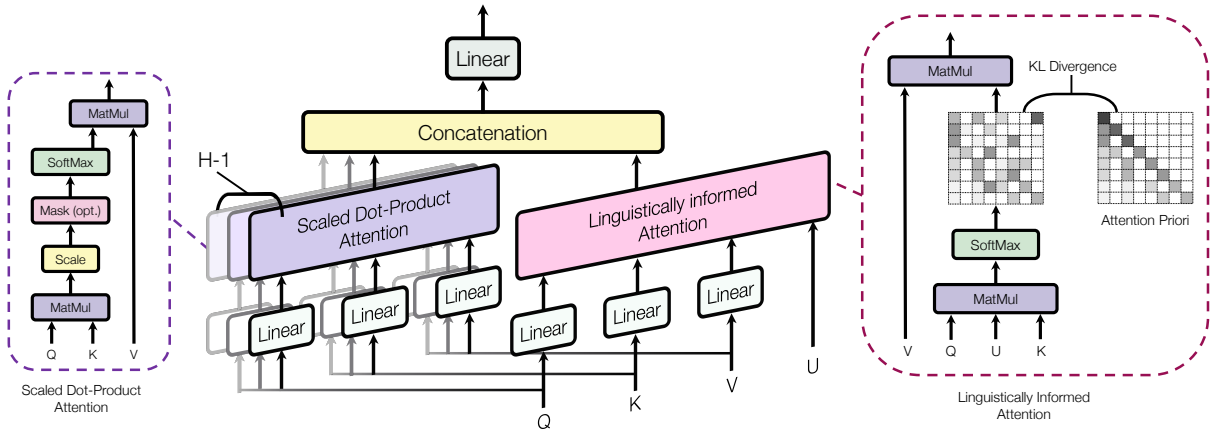

Figure 3: Overview of the syntactically-informed Transformer layer. One out of the H attention heads is trained for predicting syntactic parse ancestors of each token. For each token, this attention head outputs a distribution over all positions in the sentence, which corresponds to the probability of each token being the ancestor of this token. The loss function is defined as the mean Kullback–Leibler (KL) divergence between the output distributions of all tokens and the corresponding prior distributions.

trix contains the prior knowledge that each token should attend to its syntactic parse ancestors, with attention weights being higher on ancestors that are closer to that token. During training, we obtain the prior attention weights based on the outputs of a pre-trained dependency parser.

For example, in the utterance "*list flights arriving in Toronto on March first*", the syntactic parse ancestors of word "*first*" are "*March*", "*arriving*" and "*flights*" which are 1, 2 and 3 hops away on the dependency tree, respectively, as shown in Figure 4. The ancestors are syntactically meaningful for the determination of the slot of "*first*", which is "*arrive date, day number*" in this case.

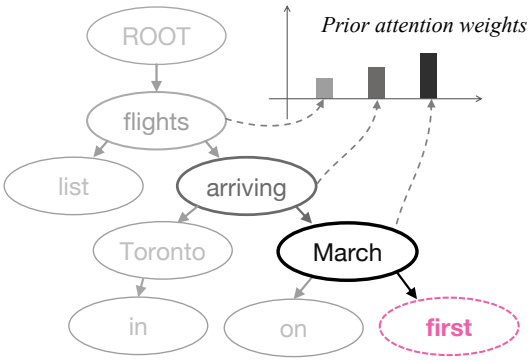

Figure 4: Syntactic dependency tree of "*list flights arriving in Toronto on March first*" and prior attention weights of word "*first*".

To train the attention head to assign higher weights on the ancestors of each token, we define prior attention weights of each token based on the distance between the token and its ancestors. Formally, the prior attention weights of token i are defined as:

$$w_{i,j}^{prior} = \begin{cases} softmax(-d_{i,j}/\tau) & \text{if } j \in ancestors(i) \\ 0 & \text{if } j \notin ancestors(i) \end{cases} \tag{4}$$

in which $d_{i,j}$ is the distance between token i and j, $softmax$ is the Softmax function, τ is the temperature of the Softmax function controlling the variance of the attention weights over all the ancestors and $i, j \in \{1, 2, \ldots, T\}$. The stack of prior attention weights of all T tokens is a $T \times T$ matrix, denoted by W^{prior}. We train our model to decrease the difference between W^{prior} and the attention matrix $W_h^{(s)}$ output by the attention head h at the sth layer. The difference is measured by the mean of row-wise KL divergence between these two matrices, which is used as an additional loss function besides the NLU loss functions. We refer to this loss as dependency loss denoted by $\mathcal{L}^{dep.}$, formally:

$$\mathcal{L}^{dep.} = \frac{1}{T}\sum_{i=1}^{T} D_{KL}(W_i^{prior} \parallel W_{h,i}^{(s)}) \tag{5}$$

$$W_h^{(s)} = softmax(Q^{(s)} U^{dep.} (K^{(s)})^T) \tag{6}$$

in which $D_{KL}(\cdot)$ denotes the KL divergence, $Q^{(s)}$ and $K^{(s)}$ are linear transformations of $E^{(s-1)}$, $W_{h,i}^{(s)}$ is the ith row of $W_h^{(s)}$, and $U^{dep.}$ is a parameter matrix. In (6) we use the biaffine attention instead of the scaled dot product attention, which has been shown to be effective for dependency parsing (Dozat and Manning 2016).

We treat τ as a hyperparameter and tune it on the validation set. With $\tau \to 0^+$, attention head h will be trained to only pay attention to the direct parent of each token, a special case used by (Strubell et al. 2018). Thus, our method is a more general approach compared to (Strubell et al. 2018).

Encoding Part-of-Speech Knowledge

Part-of-Speech (POS) information is important for disambiguating words with multiple meanings (Alva and Hegde 2016). This is because an ambiguous word carries a specific POS in a particular context (Pal, Munshi, and Saha 2015). For instance, the word "*May*" could be either a verb or a noun. Being aware of its POS tag is beneficial for downstream tasks, such as predicting the slots in the utterance "*book a flight on May 1st*". Furthermore, previous studies have shown that while models trained for a sufficiently large number of steps can potentially learn underlying patterns of POS, the knowledge is imperfect (Jawahar, Sagot, and Seddah 2019; Sundararaman et al. 2019). For these reasons, we explicitly train our model to perform POS tagging using the POS tags generated by a pretrained POS tagger.

Similar to slot filling, we simplify POS tagging as a token-level classification problem. We apply a MLP-based classifier on the output embeddings of the rth transformer encoder layer and use cross entropy as the loss function:

$$p_{i,o}^{pos} = softmax(MLP(e_{r,i})) \tag{7}$$

$$\mathcal{L}^{pos} = -\sum_{i=1}^{T}\sum_{o=1}^{O} y_{i,o}^{pos} \log p_{i,o}^{pos} \tag{8}$$

in which $p_{i,o}^{pos}$ is the predicted probability of the ith token's POS label being the oth label in the POS label space, O is the total number of POS labels, $y_{i,o}^{pos}$ is the one-hot representation of the groundtruth POS label.

Intent Detection and Slot Filling

Intent detection: We apply a linear classifier on the embedding of the "[SOS]" token, $e_{L,0}$, which is output by the last Transformer encoder layer L. Cross entropy loss is used for intent detection. The loss on one utterance is defined as:

$$p_n^{int.} = softmax(W^{int.} e_{L,0}^T + b^{int.}) \tag{9}$$

$$\mathcal{L}^{int.} = -\sum_{n=1}^{N} y_n^{int.} \log p_n^{int.} \tag{10}$$

in which $W^{int.}$ and $b^{int.}$ are the parameters of the linear classifier, $y^{int.}$ is the one-hot representation of the ground truth intent label, N is the total number of intent labels and $p_n^{int.}$ is the predicted probability of this utterance's intent label being the nth label in the intent label space.

Slot filling: We apply a MLP-based classifier on the embeddings output by the last Transformer encoder layer using cross entropy as the loss function. The loss on one utterance is defined as follow:

$$p_{i,s}^{slot} = softmax(MLP(e_{L,i})) \tag{11}$$

$$\mathcal{L}^{slot} = -\sum_{i=1}^{T}\sum_{s=1}^{S} y_{i,s}^{slot} \log p_{i,s}^{slot} \tag{12}$$

in which $p_{i,s}^{slot}$ is the predicted probability of the ith token's slot being the sth label in the slot space, S is the total number of slots, and $y_{i,s}^{slot}$ is the one-hot representations of the ground truth slot label.

Multi-task Learning

We train our model via multi-task learning (Caruana 1993). Our loss function is defined as:

$$\mathcal{L} = \mathcal{L}^{NLU} + c^{dep} \cdot \mathcal{L}^{dep} + c^{pos} \cdot \mathcal{L}^{pos} \tag{13}$$

where $\mathcal{L}^{NLU}$ equals to $\mathcal{L}^{slot}$ for slot filling, $\mathcal{L}^{int.}$ for intent detection or $\mathcal{L}^{slot} + \mathcal{L}^{int.}$ for joint training, and c^{dep} and c^{pos} are coefficients of the dependency prediction loss and the POS tagging loss, respectively. c^{dep} and c^{pos} are treated as hyperparameters and selected based on validation performance.

Experiments

Datasets

We conducted experiments on two benchmark datasets: the Airline Travel Information Systems (ATIS) (Hemphill, Godfrey, and Doddington 1990) and SNIPS (Coucke et al. 2018) datasets. The ATIS dataset has a focus on airline information and has been used as benchmark on NLU tasks. We used the same version as (Goo et al. 2018; E et al. 2019) that contains 4,478 utterances for training, 500 for validation and 893 for testing. The SNIPS dataset has a focus on personal assistant commands, with a larger vocabulary size and more diverse intents and slots. It contains 13,084 utterances for training, 700 for validation and 700 for testing.

Evaluation Metrics

We use classification accuracy for intent detection and the F1 score for slot filling, which is the harmonic mean of precision and recall. For the SNIPS dataset, we use the same version and evaluation method as pervious works (Zhang et al. 2020). For the ATIS dataset, we find that previous works use two different evaluation methods for intent detection on utterances with multiple labels. The first method counts a prediction as correct if it is equal to one of the ground truth labels of the utterance (Liu and Lane 2016). We refer this method as the single label matching method (ID-S). The second method counts a prediction as correct only if it matches all labels of the utterance (Goo et al. 2018; E et al. 2019). We refer this method as the multiple label matching method (ID-M). We report both in our results.

Implementation Details

Our experiments are implemented in PyTorch (Paszke et al. 2017). The hyperparameters are selected based on the performance on the validation set. We use the Adam optimizer (Kingma and Ba 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$ and the weight decay fix as described in (Loshchilov and Hutter 2017). Our learning rate schedule first increases the learning rate linearly from 0 to 0.0005 (warming up) and then decreases it to 0 following the values of the cosine function. We use warming up steps $\approx 20\%$ of the total training steps. The specific number of warming up

	SNIPS		ATIS		
	SF	ID	SF	ID-M	ID-S
Joint Seq (Hakkani-Tür et al. 2016)	87.30	96.90	94.30	92.60	-
Attention-based RNN (Liu and Lane 2016)	87.80	96.70	95.78	-	97.98
Slot-Gated (Goo et al. 2018)	89.27	96.86	95.42	95.41	-
SF-ID, SF first (E et al. 2019)	91.43	97.43	95.75	97.76	-
SF-ID, ID first (E et al. 2019)	92.23	97.29	95.80	97.09	-
Stack-Propagation (Qin et al. 2019)	94.20	98.00	95.90	96.90	-
Graph LSTM (Zhang et al. 2020)	95.30	98.29	95.91	97.20	-
TF	96.37	98.29	95.31	96.42	97.65
SyntacticTF (Independent)	96.56	98.71	95.94	**97.76**	98.10
SyntacticTF (Joint)	**96.89**	**99.14**	**96.01**	97.31	**98.32**
JointBERT (Chen, Zhuo, and Wang 2019)*	97.00	98.60	96.10	97.50	-

Table 1: SF and ID results on the ATIS and SNIPS dataset (%). **TF** refers to the Transformer encoder-based model trained without syntactic information. **SyntacticTF** refers to our model. **Independent** and **Joint** refer to independently and jointly training for SF and ID, respectively. **ID-M** refers to multiple label matching for intent detection evaluation and **ID-S** refers to single label matching. *This work relies on pretraining, which is not required by other works in the table.

steps is determined by validation performance. We use the implementation of the optimizer and learning rate scheduler of the Transformers library (Wolf et al. 2019).

We use Stanza (Qi et al. 2020) to generate training labels for POS tagging and dependency prediction. For the NLU model trained with both dependency prediction and POS tagging, c^{dep} and c^{pos} are both set to 1. For the NLU model trained with only dependency prediction, c^{dep} is set to 5. We used weight decay of 0.1 and dropout rate (Srivastava et al. 2014) of 0.1 and 0.3 for the SNIPS and ATIS dataset, respectively. We use batch size of 32 and train each model for 100 epochs. We report the testing results of the checkpoints achieving the best validation performance.

We use the concatenation of GloVe embeddings (Pennington, Socher, and Manning 2014) and character embeddings (Hashimoto et al. 2017) as token embeddings and keep them frozen during training. The hidden dimension of the Transformer encoder layer is 768 and the size of feed forward layer is 3072. Considering the small size of the two datasets, we only use two Transformer encoder layers in total (with $x = 1$, $y = 0$ and $z = 0$ as in Figure 2), each of which has 4 attention heads. The dimension of $Q^{(s)}$ and $K^{(s)}$ is 200. For slot filing, we apply the Viterbi decoding at test time. BIO is a standard format for slot filling annotation schema, as shown in Figure 1. The transition probabilities are manually set to ensure the output sequences of BIO labels to be valid, by simply specifying the probabilities of invalid transition to zero and the probabilities of valid transition to one.

Baseline Models

We compare our proposed model with the following baseline models:

- Joint Seq (Hakkani-Tür et al. 2016) is a joint model for intent detection and slot filling based on the bi-directional LSTM model.
- Attention-based RNN (Liu and Lane 2016) is a sequence-to-sequence model with the attention mechanism.
- Slot-Gated (Goo et al. 2018) utilizes intent information for slot filling through the gating mechanism.
- SF-ID (E et al. 2019) is an architecture that enables the interaction between intent detection and slot filling.
- Stack-Propagation (Qin et al. 2019) is a joint model based on the Stack-Propagation framework.
- Graph LSTM (Zhang et al. 2020) is based on the Graph LSTM model.
- JointBERT (Chen, Zhuo, and Wang 2019) is a joint NLU model fined tuned from the pretrained BERT model (Devlin et al. 2018).
- TF is the Transformer encoder-based model trained without syntactic information.

Results

Table 1 shows the performance of the baseline and proposed models for SF and ID on the SNIPS and ATIS dataset, respectively. Overall, our proposed models achieve the best performance on the two benchmarking datasets. On the SNIPS dataset, our proposed joint model achieves an absolute F1 score and accuracy improvement of 1.59% and 0.85% for SF and ID, respectively, compared to the best performed baseline model without pre-training (Zhang et al. 2020). On the ATIS dataset, our proposed joint model also achieves an absolute F1 score and accuracy improvement of 0.1% and 0.34% for SF and ID-S, compared to the best performed baseline model for SF (Zhang et al. 2020) and ID-S (Liu and Lane 2016), respectively. In addition, our proposed independent model achieves the same performance as the best performed baseline model on ID-M (E et al. 2019, SF-ID, SF first).

Besides, the model based on Transformer encoder without syntactic knowledge can achieve SOTA results on the SNIPS dataset and is slightly worse than the SOTA results on the ATIS dataset. This indicates the powerfulness of the Transformer encoder for SF and ID. Moreover, the further improvement of our models over the baseline models demon-

	SNIPS		ATIS		
	SF	ID	SF	ID-M	ID-S
TF	96.37	98.29	95.31	96.42	97.65
TF + D	96.31	98.43	**95.99**	96.53	**98.76**
TF + P	96.47	98.57	95.82	97.31	98.10
TF + D + P	**96.56**	**98.71**	95.94	**97.76**	98.10

Table 2: Results of ablation study. **TF** refers to the baseline models with two Transformer encoder layers. **D** and **P** refers to dependency prediction and POS tagging, respectively.

	ATIS		SNIPS	
	SF	ID	SF	ID-S
TF + Par.	96.20	98.29	95.58	98.10
TF + Anc.	96.31	98.43	95.99	98.76

Table 3: Intent detection and slot filling results of Transformer (**TF**) encoder-based models with dependency parent prediction (**Par.**) and dependency ancestor prediction (**Anc.**) on the ATIS and SNIPS dataset.

strates the benefits of incorporating syntactic knowledge. Additionally, compared to previous works with heterogenous model structures, our models are purely based on self-attention and feed forward layers.

We also find that our proposed models can outperform the JointBERT model with pre-training (Chen, Zhuo, and Wang 2019) for intent detection tasks. Compared to the JointBERT model, our proposed joint model achieves an absolute accuracy improvement of 0.54% for ID on the SNIPS dataset; and our proposed independent model achieves an absolute accuracy improvement of 0.26% for ID-M on the ATIS dataset. While our proposed model does not outperform the JointBERT model for SF, the performance gap is relatively small (0.11% on SNIPS and 0.09% on ATIS). It should be noted that our model does not require pre-training and the size of our model is only one seventh of the JointBERT model (16 million vs. 110 million parameters).

Previous works have shown that models like BERT can learn syntactic knowledge by self-supervision (Clark et al. 2019; Manning et al. 2020). This can partially explain why the JointBERT can achieve very good results without being fed with syntactic knowledge explicitly.

Ablation Study

Table 2 shows the ablation study results of the effects of adding different syntactic information. A first observation is that the model trained with a singe syntactic task, either dependency prediction or POS tagging, outperforms the baseline Transformer encoder-based model without syntactic information. This gives us confidence that syntactic information can help improve model performance. Moreover, training a Transformer model with both the syntactic tasks achieves even better results than training with a single syntactic task. This could be because the POS tagging task improves the performance of the dependency prediction task (Nguyen and Verspoor 2018), which in turn improves the performance of SF and ID.

Interestingly, we observe that the addition of dependency prediction reduces the performance of slot filling on the SNIPS dataset (96.31%) when compared to the baseline Transformer encoder-based model (96.37%). There are several potential reasons. Firstly, the sentences in the SNIPS dataset are overall shorter than the ATIS dataset so that the syntactic dependency information might be less helpful. Secondly, previous work has shown that syntactic parsing performance often suffers when a named entity span has crossing brackets with the spans on the parse tree (Finkel and Manning 2009). Thus, the dependency prediction performance of our model might decrease due to the presence of many name entities in the SNIPS dataset, such as song names and movie names, which could introduce noisy dependency information into the attention weights and degrade the performance on the NLU tasks.

Qualitative Analysis

We qualitatively examined the errors made by the Transformer encoder-based models with and without syntactic information to understand in what ways syntactic information helps improve the performance. Our major findings are:

ID errors related to preposition with nouns: Prepositions, when appearing between nouns, are used to describe their relationship. For example, in the utterance "*kansas city to atlanta monday morning flights*", the preposition "*to*" denotes the direction from "*kansas city*" (departure location, noun) to "*atlanta*" (arrival location, noun). Without this knowledge, a model could misclassify the intent of this utterance as asking for city information rather than flight information. We found that about 50% of the errors made by the model without syntactic information contain this pattern, whereas less than 10% of the misclassified utterances contain this pattern for the model with syntactic information (See Appendix A for the full list).

SF errors due to POS confusion: A Word can have multiple meanings depending on context. For example, the same word "*may*" can be a verb expressing possibility, or as a noun referring to the fifth month of the year. We found that correctly recognizing the POS of words could potentially help reduce slot filling errors. For example, in this utterance "*May I have the movie schedules for Speakeasy Theaters*", the slot for "*May*" should be empty, but the model without syntactic information predicts it as "*Time Range*". By contrast, the model with syntactic information predicts correctly for this word, probably because the confusion of noun vs. verb for the word "*May*" is addressed by incorporating POS information. More examples are included in Appendix A.

Parent Prediction vs. Ancestor Prediction

We compare our approach of predicting all ancestors of each token with the approach described in (Strubell et al. 2018), which only predicts direct dependency parent of each token. Results in Table 3 show that the model with our approach can achieve better results for both ID and SF on the two datasets, which demonstrates that our approach is more beneficial to the NLU tasks. We hypothesize that incorporating syntactic ancestor prediction can better capture long-

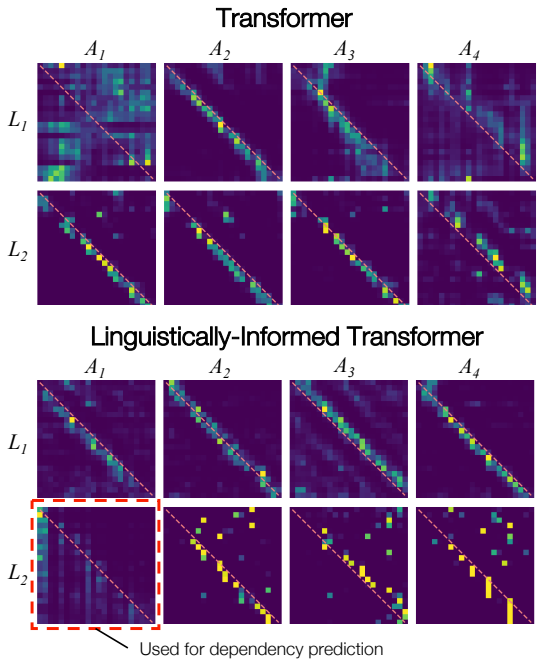

Figure 5: Visualization of the attention weights of the model with and without syntactic supervision for slot filling. L_i and A_j stands for the ith Transformer layer and jth attention head, respectively. The attention head inside the red-dotted box is trained for dependency prediction.

distance syntactic relationship. As shown in (Tur, Hakkani-Tür, and Heck 2010), long distance dependencies are important for slot filling. For example, in the utterance "*Find flights to LA arriving in no later than next Monday*", a 6-gram context is needed to figure out that "*Monday*" is the arrival date instead of the departure date.

Visualization of Attention Weights

We visualize the attention weights output by models trained with and without syntactic information to understand what the models have learned by incorporating syntactic information. We select the utterance "*show me the flights on american airlines which go from st. petersburg to ontario california by way of st. louis*" from the ATIS testing set. Only the model trained with syntactic information predicts the slot labels correctly. As shown in Figure 5, the model without syntactic information has simple attention patterns on both layers, such as looking backward and looking forward. Other attention heads seem to be random and less informative.

In contrast, the model with syntactic information has more informative attention patterns. On the first layer, all the attention heads present simple but diverse patterns. Besides looking forward and backwards, the second attention head looks at both directions for each token. On the second layer, however, we observe more complex patterns and long-distance attention which could account for more task-oriented operations. Therefore, it is possible that the Transformer encoder learns attention weights better with syntactic information supervision so that the encoder can leave more power for the end task.

Related Work

Research on intent detection and slot filling emerged in the $1990s$ from the call classification systems (Gorin, Riccardi, and Wright 1997) and the ATIS project (Price 1990). Early work has primarily focused on using a traditional machine learning classifier such as CRFs (Haffner, Tur, and Wright 2003). Recently, there has been an increasing application of neural models on NLU tasks. These approaches, primarily based on RNNs, have shown that neural approaches outperform traditional models (Mesnil et al. 2014; Tur et al. 2012; Zhang and Wang 2016; Goo et al. 2018; E et al. 2019). For example, Mesnil et al (2015) employed RNNs for slot filling and found an 2.3% relative improvement of F1 compared to CRF (Mesnil et al. 2014). Some works also explored Transformer encoder and graph LSTM-based neural architectures (Chen, Zhuo, and Wang 2019; Zhang et al. 2020).

Syntactic information has been shown to be beneficial to many tasks, such as neural machine translation (Akoury, Krishna, and Iyyer 2019), semantic role labeling (Strubell et al. 2018), and machine reading comprehension (Zhang et al. 2020). Research on NLU tasks has also shown that incorporating syntactic information into machine learning models can help improve the performance. Moschitti *et al.* (2007) used syntactic information for slot filling, where the authors used a tree kernel function to encode the structural information acquired by a syntactic parser. An extensive analysis on the ATIS dataset revealed that most NLU errors are caused by complex syntactic characteristics, such as prepositional phrases and long distance dependencies (Tur, Hakkani-Tür, and Heck 2010). Tur *et al.* (2011) proposed a rule-based dependency parsing based sentence simplification method to augment the input utterances based on the syntactic structure. Compared to previous works, our work is the first to encode syntactical knowledge into end-to-end neural models for intent detection and slot filling.

Conclusion

In this paper, we propose to encode syntactic knowledge into the Transformer encoder-based model for intent detection and slot filling. Experimental results indicate that a model with only two Transformer encoder layers can already match or even outperform the SOTA performance on two benchmark datasets. Moreover, we show that the performance of this baseline model can be further improved by incorporating syntactical supervision. The visualization of the attention weights also reveals that syntactical supervision can help the model to better learn syntactically-related patterns. For future work, we will evaluate our approach with larger model sizes on larger scale datasets containing more syntactically complex utterances. Furthermore, we will investigate incorporating syntactic knowledge into models pretrained by self-supervision and applying those models on the NLU tasks.

Acknowledgement

We would like to thank Siegfried Kunzmann, Nathan Susanj, Ross McGowan, and anonymous reviewers for their insightful feedback that greatly improved our paper.

References

Akoury, N.; Krishna, K.; and Iyyer, M. 2019. Syntactically supervised transformers for faster neural machine translation. *arXiv preprint arXiv:1906.02780* .

Allen, J. 1995. *Natural Language Understanding (2nd Ed.)*. USA: Benjamin-Cummings Publishing Co., Inc. ISBN 0805303340.

Alva, P.; and Hegde, V. 2016. Hidden markov model for pos tagging in word sense disambiguation. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 279–284. IEEE.

Caruana, R. 1993. Multitask learning: a knowledge-based source of inductive bias. In *ICML'93 Proceedings of the Tenth International Conference on International Conference on Machine Learning*, 41–48.

Chen, Q.; Zhuo, Z.; and Wang, W. 2019. BERT for Joint Intent Classification and Slot Filling. *arXiv preprint arXiv:1902.10909* .

Chen, Y.-N.; Hakanni-Tür, D.; Tur, G.; Celikyilmaz, A.; Guo, J.; and Deng, L. 2016. Syntax or semantics? knowledge-guided joint semantic frame parsing. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, 348–355. IEEE.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341* .

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12(ARTICLE): 2493–2537.

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. *arXiv preprint arXiv:1805.10190* .

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* .

Dozat, T.; and Manning, C. D. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* .

E, H.; Niu, P.; Chen, Z.; and Song, M. 2019. A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling. In *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, 5467–5471.

Finkel, J. R.; and Manning, C. D. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 326–334.

Goo, C.-W.; Gao, G.; Hsu, Y.-K.; Huo, C.-L.; Chen, T.-C.; Hsu, K.-W.; and Chen, Y.-N. 2018. Slot-gated Modeling for Joint Slot Filling and Intent Prediction. In *NAACL HLT 2018: 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, 753–757.

Gorin, A. L.; Riccardi, G.; and Wright, J. H. 1997. How may I help you? *Speech communication* 23(1-2): 113–127.

Haffner, P.; Tur, G.; and Wright, J. H. 2003. Optimizing SVMs for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, I–I. IEEE.

Hakkani-Tür, D.; Tür, G.; Çelikyilmaz, A.; Chen, Y.-N.; Gao, J.; Deng, L.; and Wang, Y.-Y. 2016. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM. In *Interspeech 2016*, 715–719.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1923–1933.

Hemphill, C. T.; Godfrey, J. J.; and Doddington, G. R. 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*.

Jawahar, G.; Sagot, B.; and Seddah, D. 2019. What does BERT learn about the structure of language?

Kingma, D. P.; and Ba, J. L. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*.

Liu, B.; and Lane, I. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *Interspeech 2016*, 685–689.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* .

Manning, C. D.; Clark, K.; Hewitt, J.; Khandelwal, U.; and Levy, O. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences* .

Mesnil, G.; Dauphin, Y.; Yao, K.; Bengio, Y.; Deng, L.; Hakkani-Tur, D.; He, X.; Heck, L.; Tur, G.; Yu, D.; 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.

Moschitti, A.; Riccardi, G.; and Raymond, C. 2007. Spoken language understanding with kernels for syntactic/semantic structures. In *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, 183–188. IEEE.

Nguyen, D. Q.; and Verspoor, K. 2018. An improved neural network model for joint POS tagging and dependency parsing. *arXiv preprint arXiv:1807.03955* .

Pal, A. R.; Munshi, A.; and Saha, D. 2015. An Approach to Speed-up the Word Sense Disambiguation Procedure through Sense Filtering. *arXiv preprint arXiv:1610.06601* .

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Price, P. 1990. Evaluation of spoken language systems: The ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; and Manning, C. D. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Qin, L.; Che, W.; Li, Y.; Wen, H.; and Liu, T. 2019. A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding. In *2019 Conference on Empirical Methods in Natural Language Processing*, 2078–2087.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1): 1929–1958.

Strubell, E.; Verga, P.; Andor, D.; Weiss, D.; and McCallum, A. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP 2018: 2018 Conference on Empirical Methods in Natural Language Processing*, 5027–5038.

Sundararaman, D.; Subramanian, V.; Wang, G.; Si, S.; Shen, D.; Wang, D.; and Carin, L. 2019. Syntax-Infused Transformer and BERT models for Machine Translation and Natural Language Understanding. *arXiv preprint arXiv:1911.06156* .

Tur, G.; Deng, L.; Hakkani-Tür, D.; and He, X. 2012. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5045–5048. IEEE.

Tur, G.; Hakkani-Tür, D.; and Heck, L. 2010. What is left to be understood in ATIS? In *2010 IEEE Spoken Language Technology Workshop*, 19–24. IEEE.

Tur, G.; Hakkani-Tur, D.; Heck, L.; and Parthasarathy, S. 2011. Sentence simplification for spoken language understanding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5628–5631.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5998–6008.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771.

Zhang, L.; Ma, D.; Zhang, X.; Yan, X.; and Wang, H. 2020. Graph LSTM with Context-Gated Mechanism for Spoken Language Understanding. *AAAI 2020 : The Thirty-Fourth AAAI Conference on Artificial Intelligence* 34(5): 9539–9546.

Zhang, X.; and Wang, H. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, volume 16, 2993–2999.

Zhang, Z.; Wu, Y.; Zhou, J.; Duan, S.; Zhao, H.; and Wang, R. 2020. SG-Net: Syntax-Guided Machine Reading Comprehension. In *AAAI*, 9636–9643.

Appendix A

Below lists examples of the intent detection errors made by the model without syntactic information that are related to one specific grammar pattern between prepositions and nouns.

- cleveland to kansas city arrive monday before 3 pm
- kansas city to atlanta monday morning flights
- new york city to las vegas and memphis to las vegas on Sunday

Below lists examples of the slot filling errors made by the model without syntactic information that contain POS confusion.

- cleveland to kansas city arrive monday before 3 pm
- new york city to las vegas and memphis to las vegas on Sunday
- baltimore to kansas city economy

The Transformer encoder-based model without syntactic information made mistakes on all these utterances. The model trained with POS tagging and the model trained with both POS tagging and dependency prediction fail on the last utterance in the list below. The model trained with dependency prediction does not make any mistakes on all these utterances. We underline the words that are assigned to wrong slots by the model without syntactic information.

- book a reservation for velma an a and rebecca for an american pizzeria at (correct: $B-TimeRange$; prediction: $B-RestaurantName$) 5 Am in MA
- Where is Belgium located (correct: $Other$; prediction: $B-PatialRelation$)
- May(correct: $Other$; prediction: $B-TimeRange$) I have the movie schedules for Speakeasy Theaters