ONENET: JOINT DOMAIN, INTENT, SLOT PREDICTION FOR SPOKEN LANGUAGE UNDERSTANDING

Young-Bum Kim[*], *Sungjin Lee*[**], *Karl Stratos****

Amazon Alexa Brain, Seattle, WA[*],
Microsoft Research, Redmond, WA[**],
Toyota Technological Institute, Chicago, IL[***]

ABSTRACT

In practice, most spoken language understanding systems process user input in a pipelined manner; first domain is predicted, then intent and semantic slots are inferred according to the semantic frames of the predicted domain. The pipeline approach, however, has some disadvantages: error propagation and lack of information sharing. To address these issues, we present a unified neural network that jointly performs domain, intent, and slot predictions. Our approach adopts a principled architecture for multitask learning to fold in the state-of-the-art models for each task. With a few more ingredients, e.g. orthography-sensitive input encoding and curriculum training, our model delivered significant improvements in all three tasks across all domains over strong baselines, including one using oracle prediction for domain detection, on real user data of a commercial personal assistant.

Index Terms— Natural language understanding, Multi-task learning, Joint modeling

1. INTRODUCTION

Major personal assistants, e.g., Amazon's Alexa, Apple Siri and Microsoft Cortana, support task-oriented dialog for multiple domains. A typical way to handle multiple domains for the spoken language understanding (SLU) task [1] is to perform domain prediction first and then carry out intent prediction [2, 3, 4] and slot tagging [5, 6, 7, 8, 9, 10, 3, 4] (Figure 1). However, this approach has critical disadvantages. First, the error made in domain prediction propagates to downstream tasks - intent prediction and slot tagging. Second, the domain prediction task cannot benefit from the downstream prediction results. Third, it is hard to share domain-invariant features such as common or similar intents and slots across different domains.

Despite such disadvantages, there were a few reasons that such a pipelined approach was widely adopted before the deep neural networks (DNNs) era has come. For instance, in graphical models, jointly modeling multiple tasks often entails an unwieldy model with a high computational complexity [11]. Also, with a discrete representation, a unified model that covers all domains usually results in severe data sparseness problems given the relatively small amount of labeled data. Last but not least, one can use domain-specific resources, e.g. domain gazetteers, that often results in a significant performance increase. Domain-specific resources, however, are costly to build in new domains.

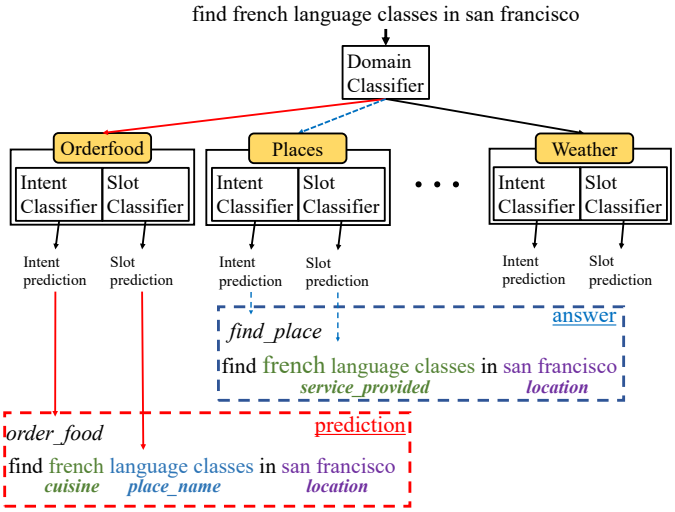

Fig. 1. Illustration of the pipelined procedure. If domain classifier predicted a wrong domain `Orderfood`, subsequent tasks – intent detection and slot tagging – would also make errors consequently.

With recent advances, DNNs provide various affordances that allow us to address most issues described above: Multi-task learning becomes as easy as dropping in additional loss terms [12]. Continuous representation learning addresses the data sparseness problem [13]. Unsupervised learning helps us tap into unlimited data sources, obviating demands for domain-specific resources [14].

In this paper, we present a unified neural architecture, *OneNet*, that performs domain, intent and slot predictions all

together. Our approach adopts a principled architecture for multitask learning to fold in the state-of-the-art models for each task. With a few more ingredients, e.g. orthography-sensitive input encoding and curriculum training, our model delivered significant improvements in all three tasks across all domains over strong baselines, including one using oracle prediction for domain detection, on real user data of a commercial personal assistant.

2. RELATED WORK

There has been an extensive line of prior studies for jointly modeling intent and slot predictions: a triangular conditional random field (CRF) [11], a convolutional neural networks-based triangular CRF [15], recursive neural networks learning hierarchical representations of input text [16], several variants of recurrent neural networks [17, 18, 19].

There are also a line of prior works on multi-domain learning to leverage existing domains: [20] proposed a constrained decoding method with a universal slot tagging model. [21] proposed K domain-specific feedforward layers with a shared word-level Long Short-Term Memory (LSTM) [22] layer across domains; [23] instead employed $K + 1$ LSTMs. Finally, there is a few approaches performing multi-domain/multitask altogether with a single network. [24], however, requires a separate schema prediction to construct decoding constraints. [25] proposed to employ a sequence-to-sequence model by introducing a fictitious symbol at the end of an utterance of which tag represents the corresponding domain and intent. In comparison to the prior models, not only does our approach afford a more straightforward integration of multiple tasks but also it delivers a higher performance.

3. ONENET

At a high level, our model consists of builds on three ingredients (Figure 2): a shared orthography-sensitive word embedding layer; a shared bidirectional LSTM (BiLSTM) layer that induces contextual vector representations for the words in a given utterance; three separate output layers performing domain, intent and slot prediction, respectively, which are rooted in the shared BiLSTM layer. We combine the losses of these output layers which will be minimized jointly. Crucially, the joint optimization updates the shared layers to be simultaneously suitable for all three tasks, allowing us to not only avoid the error propagation problem but also improve the performance of individual tasks by sharing task-invariant information.

3.1. Embedding layer

In order to capture character-level patterns, we construct a orthography-sensitive word embedding layer following [26]. Let $\mathcal{C}$ denote the set of characters and $\mathcal{W}$ the set of words. Let $\oplus$ denote the vector concatenation operation. We use an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. The model parameters associated with this layer are:

$$\textbf{Char embedding: } e_c \in \mathbb{R}^{25} \text{ for each } c \in \mathcal{C}$$
$$\textbf{Char LSTMs: } \phi^{\mathcal{C}}_f, \phi^{\mathcal{C}}_b : \mathbb{R}^{25} \times \mathbb{R}^{25} \rightarrow \mathbb{R}^{25}$$
$$\textbf{Word embedding: } e_w \in \mathbb{R}^{100} \text{ for each } w \in \mathcal{W}$$

Let $w_1 \dots w_n \in \mathcal{W}$ denote a word sequence where word w_i has character $w_i(j) \in \mathcal{C}$ at position j. This layer computes a orthography-sensitive word representation $v_i \in \mathbb{R}^{150}$ as

$$\begin{aligned} f^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_f \left(e_{w_i(j)}, f^{\mathcal{C}}_{j-1} \right) && \forall j = 1 \dots |w_i| \\ b^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_b \left(e_{w_i(j)}, b^{\mathcal{C}}_{j+1} \right) && \forall j = |w_i| \dots 1 \\ v_i &= f^{\mathcal{C}}_{|w_i|} \oplus b^{\mathcal{C}}_1 \oplus e_{w_i} \end{aligned}$$

3.2. BiLSTM layer

A widely successful architecture for encoding a sentence $(w_1 \dots w_n) \in \mathcal{W}^n$ is given by BiLSTMs [27, 28]:

$$\textbf{Word LSTMs: } \phi^{\mathcal{W}}_f, \phi^{\mathcal{W}}_b : \mathbb{R}^{150} \times \mathbb{R}^{100} \rightarrow \mathbb{R}^{100}$$

for each $i = 1 \dots n$.[1] Next, the model computes

$$\begin{aligned} f^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_f \left(v_i, f^{\mathcal{W}}_{i-1} \right) && \forall i = 1 \dots n \\ b^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_b \left(v_i, b^{\mathcal{W}}_{i+1} \right) && \forall i = n \dots 1 \end{aligned}$$

and induces a context-sensitive word representation $h_i \in \mathbb{R}^{200}$ as

$$h_i = f^{\mathcal{W}}_i \oplus b^{\mathcal{W}}_i \tag{1}$$

for each $i = 1 \dots n$. These vectors are used to define the domain/intent classification loss and the slot tagging loss below. Note that the model parameters associated with both shared layers are collectively denoted as Θ.

3.3. Domain classification

We predict the domain of an utterance using $(h_1 \dots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{D}$ denote the set of domain types. We introduce a single-layer feedforward network $g^d : \mathbb{R}^{200} \rightarrow \mathbb{R}^{|\mathcal{D}|}$ whose parameters are denoted by Θ^d. We compute a $|\mathcal{D}|$-dimensional vector

$$\mu^d = g \left(\sum_{i=1}^{n} h_i \right)$$

[1] For simplicity, we assume some random initial state vectors such as $f^{\mathcal{C}}_0$ and $b^{\mathcal{C}}_{|w_i|+1}$ when we describe LSTMs.

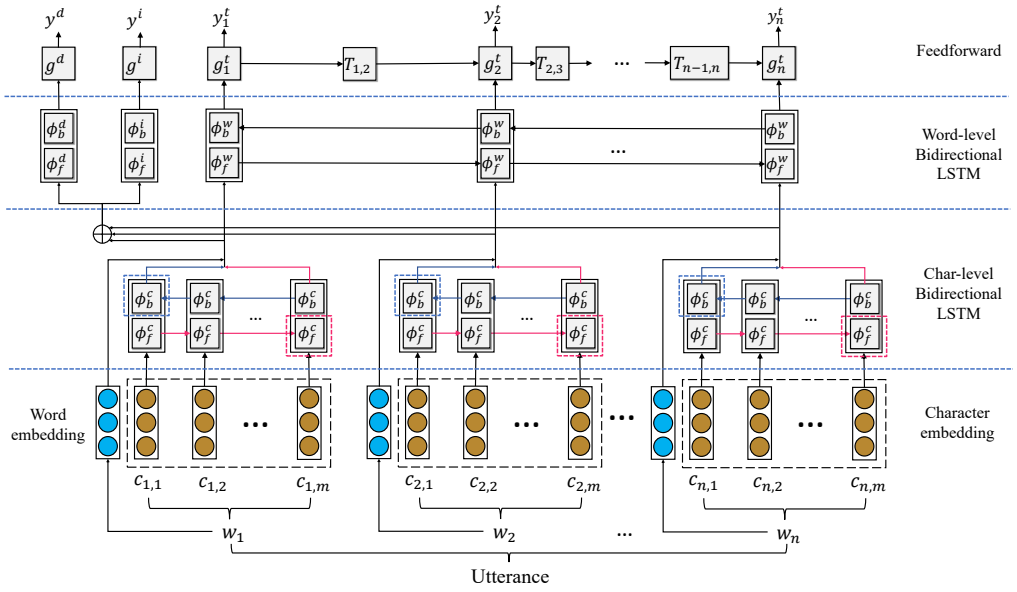

Fig. 2. The overall network architecture for joint modeling.

and define the conditional probability of the correct domain τ as

$$p(\tau|h_1 \dots h_n) \propto \exp\left(\mu_\tau^d\right) \tag{2}$$

The domain classification loss is given by the negative log likelihood:

$$L^d\left(\Theta, \Theta^d\right) = -\sum_l \log p\left(\tau^{(l)}|h^{(l)}\right) \tag{3}$$

where l iterates over the domain-annotated utterances.

3.4. Intent classification

The intent classification loss is given in an identical manner to the domain classification loss:

$$L^i\left(\Theta, \Theta^i\right) = -\sum_l \log p\left(\tau^{(l)}|h^{(l)}\right) \tag{4}$$

where l iterates over the intent-annotated utterances.

3.5. Slot tagging loss

Finally, we predict the semantic slots of the utterance using $(h_1 \dots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{E}$ denote the set of semantic types and $\mathcal{L}$ the set of corresponding BIO label types, that is, $\mathcal{L} = \{\texttt{B}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{I}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{O}\}$. We add a transition matrix $T \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{L}|}$ and a single-layer feed-forward network $g^t : \mathbb{R}^{200} \to \mathbb{R}^{|\mathcal{L}|}$ to the network; denote these additional parameters by Θ^t. The CRF tagging layer defines a joint distribution over label sequences of $y_1 \dots y_n \in \mathcal{L}$ of $w_1 \dots w_n$ as

$$p(y_1 \dots y_n|h_1 \dots h_n) \propto \exp\left(\sum_{i=1}^{n} T_{y_{i-1},y_i} \times g^t_{y_i}(h_i)\right) \tag{5}$$

The tagging loss is given by the negative log likelihood:

$$L^t\left(\Theta, \Theta^t\right) = -\sum_l \log p\left(y^{(l)}|h^{(l)}\right) \tag{6}$$

where l iterates over tagged sentences in the data.

3.6. Joint loss

The final training objective is to minimize the sum of the domain loss (3), intent loss (4), and tagging loss (6):

$$L(\Theta, \Theta^d, \Theta^i, \Theta^t) = \sum_{\alpha\in\{\Theta^d,\Theta^i,\Theta^t\}} L^\alpha\left(\Theta, \Theta^\alpha\right)$$

In stochastic gradient descent (SGD), this amounts to computing each loss l^d, l^i, l^t separately at each annotated utterance and then taking a gradient step on $l^d + l^i + l^t$. Observe that the shared layer parameters Θ are optimized for all tasks.

3.7. Curriculum learning

We find in experiments that it is important to pre-train individual models. Specifically, we first optimize the domain classifier (3), then the intent classifier (4), then jointly optimize the domain and intent classifiers (3) + (4), and finally jointly

System	Type	Model	Domain	Intent	Slot
1	Domain adapt	[21]	91.06%	87.05%	85.80
2	Independent mode	[23]	91.06%	89.24%	88.17
3	Independent domain Joint intent and slot	[15]	91.06%	86.83%	85.50
4		[16]	91.06%	89.46%	88.00
5		[17]	91.06%	89.99%	87.69
6		[18]	91.06%	90.75%	88.47
7		[19]	91.06%	90.84%	88.39
8	Full joint	[25]	92.05%	89.56%	87.52
9		[24]	-	90.11%	88.46
10	Independent models	*OneNet-Independent*	91.06%	86.47%	85.50
11	Oracle domain	*OneNet-Oracle*	100.0%	93.98%	90.83
12	Full joint	*OneNet*	**95.50**%	**94.59**%	**93.20**

Table 1. Comparative results of our joint model against other baseline systems. System 9 does not make domain prediction.

optimize all losses (3) + (4) + (6). Separately training these models in the beginning makes it easier for the final model to improve upon individual models.

4. EXPERIMENTS

4.1. Data

The data is collected from 5 domains of Microsoft Cortana, a commercial personal assistant: *alarm*, *calendar*, *communication*, *places*, *reminder*. Please refer to Appendix A for detailed data statistics. The numbers of utterances for training, tuning, and test are 10K, 1K, and 15K respectively for all domains.

4.2. Training setting

All models were implemented using Dynet [29] and were trained using Adam [30][2]. Each update was computed with Intel Math Kernel Library[3] without minibatching. We used the dropout regularization [31] with the keep probability of 0.4. We used pre-trained embeddings used by [26].

4.3. Results

To evaluate the *OneNet* model, we conducted comparative experiments with a rich set of prior works (Table 1) on real user data. We report domain, intent classification results in accuracy and slot tagging results in slot F1-score. For those systems which do not include the domain prediction task (marked as Indep. Domain and Pipeline for the Type category), we used the part for the domain prediction of the *OneNet* model trained only with the domain loss. The result clearly shows that our joint approach enjoys a higher performance than various groups of baseline systems: ones performing domain adaptation for three independent prediction models (1-2); ones exercising joint prediction only for intent detection and slot tagging (3-7); finally the full joint models (8-9). Particularly, the large performance gap between *OneNet-Independent* and *OneNet* demonstrates how much gain we could get through the proposed joint modeling. Interestingly, *OneNet* even outperformed *OneNet-Oracle* that does not incorporate the domain prediction part in training, instead referring to the oracle domain labels. This result displays an unintuitive power of shared multitask representation learning at a first glance. We found character-level modeling generally adds about 1.5% on average performance, better capturing sub-word patterns, eg. prefix, suffix, word shape, and making the model robust to spelling errors. Also, without curriculum learning, we saw about 1.7% performance drop in intent prediction. By placing intent prediction earlier in training, the latent representation is geared more toward intent prediction. A detailed performance breakdown for the *OneNet* variants by domain are provided in Appendix B. Furthermore, an illustrative example where our joint model get the prediction right while non-joint models do incorrectly can be found in Appendix C.

5. CONCLUSION

To address the disadvantages of the widely adopted pipelined architecture for most SLU systems, we presented a unified neural network, *OneNet*, that jointly performs domain, intent and slot predictions. Our model delivered significant improvements in all three tasks across all domains over strong baselines, including one using oracle prediction for domain detection, on real user data of a commercial personal assistant. Future work can include an extension of *OneNet* to take into account dialog history aiming for a holistic framework that can handle contextual interpretation as well.

[2]learning rate=4×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$.

[3]https://software.intel.com/en-us/articles/intelr-mkl-and-c-template-libraries

6. REFERENCES

[1] Ruhi Sarikaya, Paul Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiuahu Liu, et al., "An overview of end-to-end language understanding and dialog management for personal digital assistants," in *IEEE Workshop on Spoken Language Technology*, 2016.

[2] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya, "Scalable semi-supervised query classification using matrix sketching," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. August 2016, pp. 8–13, Association for Computational Linguistics.

[3] Young-Bum Kim, Karl Stratos, and Dongchan Kim, "Domain attention with an ensemble of expert," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. July 2017, pp. 643–653, Association for Computational Linguistics.

[4] Young-Bum Kim, Karl Stratos, and Dongchan Kim, "Adversarial adaptation of synthetic or stale data," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. July 2017, pp. 1297–1307, Association for Computational Linguistics.

[5] Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya, "Weakly supervised slot tagging with partially labeled sequences from web search click logs," in *Proceedings of the NAACL*. Association for Computational Linguistics, 2015.

[6] Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya, "Compact lexicon selection with spectral methods," in *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[7] Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras, "Task specific continuous word representations for mono and multi-lingual spoken language understanding," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.

[8] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya, "Pre-training of hidden-unit crfs," in *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[9] Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong, "New transfer learning techniques for disparate label sets," *ACL. Association for Computational Linguistics*, 2015.

[10] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya, "A framework for pre-training hidden-unit conditional random fields and its extension to long short term memory networks," *Computer Speech & Language*, vol. 46, pp. 311–326, 2017.

[11] Minwoo Jeong and Gary Geunbae Lee, "Triangular-chain conditional random fields," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 7, pp. 1287–1302, 2008.

[12] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[13] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[15] Puyang Xu and Ruhi Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 78–83.

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

[17] Xiaodong Zhang and Houfeng Wang, "A joint model of intent determination and slot filling for spoken language understanding," IJCAI, 2016.

[18] Bing Liu and Ian Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *Interspeech 2016*, 2016, pp. 685–689.

[19] Bing Liu and Ian Lane, "Joint online spoken language understanding and language modeling with recurrent neural networks," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Los Angeles, September 2016, pp. 22–30, Association for Computational Linguistics.

[20] Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya, "Natural language model re-usability for scaling to different domains," in *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2016.

[21] Aaron Jaech, Larry Heck, and Mari Ostendorf, "Domain adaptation of recurrent neural networks for natural language understanding," *arXiv preprint arXiv:1604.00117*, 2016.

[22] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya, "Frustratingly easy neural domain adaptation," *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 2016.

[24] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya, "Domainless adaptation by constrained decoding on a schema lattice," *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 2016.

[25] Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang, "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm," in *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*, 2016.

[26] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[27] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[28] Alex Graves, "Neural networks," in *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 15–35. Springer, 2012.

[29] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al., "Dynet: The dynamic neural network toolkit," *arXiv preprint arXiv:1701.03980*, 2017.

[30] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *The International Conference on Learning Representations (ICLR).*, 2015.

[31] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

A. DATA STATISTICS

B. PERFORMANCE BREAKDOWN BY DOMAIN

A detailed performance breakdown for the *OneNet* variants by domain are provided in Table 2.

C. EXAMPLE PREDICTION

Table 3 shows an illustrative example where our joint model get the prediction right while non-joint models do it incorrectly.

	Oracle Domain			Pipeline			Joint		
Domain	Domain	Intent	Slot	Domain	Intent	Slot	Domain	Intent	Slot
alarm	100%	95.07%	93.93	94.10%	91.00%	89.65	93.77%	95.92%	94.37
calendar	100%	92.80%	81.81	83.99%	78.83%	74.46	96.89%	93.19%	83.12
comm.	100%	89.99%	85.72	92.57%	84.19%	81.52	94.41%	90.91%	88.89
places	100%	94.47%	77.30	94.83%	90.31%	75.27	95.84%	95.03%	81.36
reminder	100%	97.58%	86.27	89.80%	88.03%	79.21	96.61%	97.89%	88.43
AVG	100%	93.98%	85.01	91.06%	86.47%	80.02	95.50%	94.59%	87.23

Table 2. Performance breakdown for the *OneNet* variants by domain

Type	Sentence	Domain	Intent
Gold	inform maya$_{\text{contact name}}$ about change of lunch tomorrow from eleven thirty to twelve$_{\text{message}}$	comm.	send text
OneNet-Pipeline	inform maya about change of lunch$_{\text{title}}$ tomorrow$_{\text{start date}}$ from eleven$_{\text{start time}}$ thirty$_{\text{start date}}$ to twelve$_{\text{end time}}$	cal.	change cal entry
OneNet-Oracle	inform maya$_{\text{contact name}}$ about change of lunch$_{\text{message}}$ tomorrow$_{\text{date}}$ from eleven thirty$_{\text{time}}$ to twelve$_{\text{time}}$	comm.	send text
OneNet	inform maya$_{\text{contact name}}$ about change of lunch tomorrow from eleven thirty to twelve$_{\text{message}}$	comm.	send text

Table 3. Example predictions by different models. Gold represents the human labels for the sentence.

Domain	average utterance length	# slots	average #slots/utterance
alarm	5.7	15	2.12
calendar	5.9	42	2.02
communication	3.1	44	1.50
places	5.62	63	2.45
reminder	6.12	34	3.07

Table 4. Data additional stats

	alarm	calendar	communication	places	reminder
alarm	15	13	3	2	9
calendar		42	10	4	26
communication			44	10	8
places				63	4
reminder					34

Table 5. The number of overlapped slots between domains

	alarm	calendar	communication	places	reminder
alarm	3539	2253	2682	2512	2766
calendar		12563	8477	8148	8054
communication			77385	19847	16465
places				57494	15614
reminder					30707

Table 6. The number of overlapped vocabularies between domains