INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Improving Zero-shot Cross-domain Slot Filling via Transformer-based Slot Semantics Fusion

Yuhang Li, Xiao Wei, Yuke Si, Longbiao Wang, Xiaobao Wang*, Jianwu Dang*

Tianjin Key Laboratory of Cognitive Computing and Application,
College of Intelligence and Computing, Tianjin University, Tianjin, China
{liyuhang00,weixiao,siyuke,longbiao_wang}@tju.edu.cn

Abstract

Slot filling is an essential component in task-oriented dialogue systems. Due to the scarcity of annotated data, zero-shot slot filling has been studied to transfer knowledge from source domains to a target domain. Previous methods adopt slot descriptions or questions as slot semantics, where they utilize slot descriptions to calculate similarity scores, or reformat the task as a question-answering problem. However, these methods do not fully exploit the token-level dependency between the slot semantics and utterances. In this study, we propose a **T**ransformer-based **S**lot semantics fusion method for **S**lot **F**illing (TSSF). We first adopt two encoders with shared weights to obtain the representations of utterances and slot semantics. Then, we design a transformer-based fusion module for effectively integrating slot semantics into utterances. Experimental results on the public benchmark SNIPS show that our model significantly outperforms the state-of-the-art model by 6.09% in terms of slot F1.

Index Terms: dialogue system, zero-shot slot filling, transformer-based slot semantics fusion

1. Introduction

In task-oriented dialogue systems, spoken language understanding (SLU) is an important component to capture the semantics of user utterances [1]. One pillar of SLU is the slot filling task, which needs to extract the slot entities in user utterances according to a set of predefined slots [2]. Fig. 1 shows an example of slot filling, given the utterance “what is the weather in koontz lake” in the “GetWeather” domain, models need to find the slot entity “koontz lake” corresponding to the slot type “city”. Conventional methods take slot filling as a sequence labeling problem (i.e., a token-level classification task), and such fully supervised learning methods require a large amount of annotated data [3, 4, 5, 6, 7]. However, the construction of such data in real life is labor-intensive and time-consuming. In addition, a dialogue system usually needs to be quickly developed and applied to a new domain [8], but this requirement is hard to be satisfied in the data scarcity scenario. One solution is to study zero-shot cross-domain slot filling, which aims to transfer knowledge from several source domains to a target domain [9, 10, 11].

In the zero-shot cross-domain slot filling task, the target domain contains two kinds of slots. The shared slots in both target and source domains are seen slots, and the slots that only appear in the target domain are unseen slots [12]. A challenge in the zero-shot slot filling is the prediction of the unseen slots, which results in a limited performance of conventional fully-supervised methods. And due to the strong bias between the

* denotes the corresponding author.

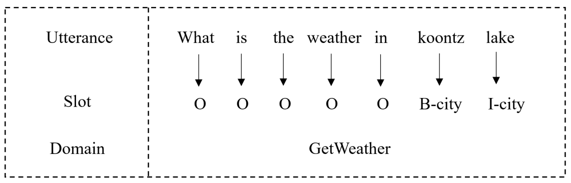


Figure 1: *An example of the slot filling task in the domain “GetWeather”. In this example, models need to find the slot entity “koontz lake” corresponding to the slot type “city”.*

source domains and the target domain, some slot entities have different semantics in the source and the target domain [13]. For example, in the SNIPS dataset [14], the slot entity “object_type” represents the type of book in the “RateBook” domain, but in the “FindScreenEvent” domain it represents the movie schedule. Previous studies introduce slot semantic information as label knowledge to avoid bias and predict both the seen and the unseen slots. These studies can be classified into two categories. On one hand, early studies predict the seen and the unseen slots by metric learning, which calculate the similarity scores between candidate slot entities and slot descriptions [15, 16, 17, 18]. In addition, slot examples are also added to slot descriptions to enhance the slot semantics representations [19]. However, these metric-based methods predict the slot label by calculating the similarity between the candidate slot entity and the sentence embedding of the slot descriptions, which does not consider the token-level dependency between the utterances and slot descriptions. And the short slot descriptions do not incorporate enough slot semantics [20]. On the other hand, later studies reformulate the zero-shot slot filling as a questioning & answering (QA) task [20, 21], which generates a question as a kind of slot semantics for each predefined slot type and then extract the answer spans using a QA model. These QA approaches simply integrate slot semantics into utterances by direct concatenation operation, yet making the model unable to fully focus on the question information, and unable to learn the dependency between the tokens in an utterance well.

To address the above problems, we propose a **T**ransformer-based **S**lot semantics fusion method for **S**lot **F**illing (TSSF). Firstly, we adopt the siamese network architecture [22], where an utterance encoder and a slot semantics encoder are adopted to get the representations of utterances and slot semantics, respectively. The two encoders share weights for parameter and computation efficiency. Secondly, we propose a slot semantics fusion module by using stacked transformer encoder layers to integrate the slot semantics into the token-level utterance representations. Due to explicit fusion by stacked transformer encoders, we can obtain the utterance representations enhanced by slot semantics. Finally, two classifiers are used to predict the span boundary for each slot.

2. Transformer-based Slot Semantics Fusion for Slot Filling

The architecture of our proposed TSSF is depicted in Fig. 2. Firstly, TSSF uses an utterance encoder and a slot semantics encoder to obtain representations of utterances and slot semantics respectively. Note that the two encoders use shared weights. Then the stacked transformer encoders are used to integrate slot semantics into utterance representations. Finally, TSSF predicts the boundary of each slot type based on the slot semantics enhanced utterance representations. The details are shown in the following subsections.

Table 1: *Examples of slot types and their slot semantics, each slot type is converted to a natural language question as its slot semantics.*

slot type	slot semantics
playlist_owner	who is the owner?
object_select	which object to select?
best_rating	how many points in total?
num_book_people	how many people for booking?

2.1. Task Formulation

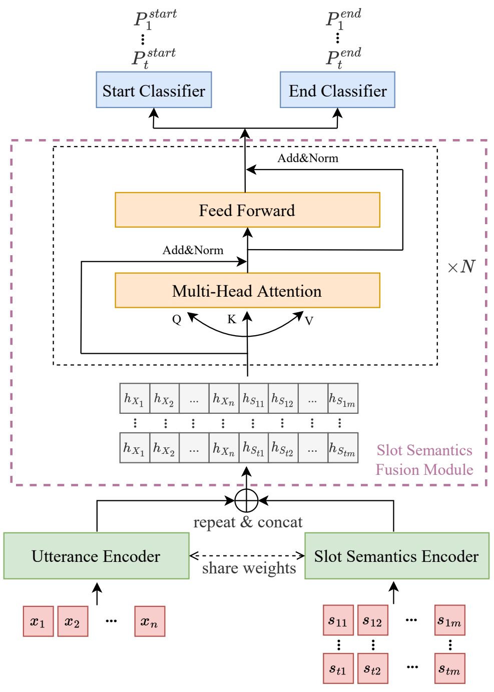
Given a user utterance $X = \{x_1, x_2, ..., x_n\}$ with n words and a set of predefined slot types L in domain D. The slot filling task is to extract a set of (slot_type, span) pairs (y_i, a_i), where y_i comes from the slot type set L, and $a_i = (j, k), 1 \le j \le k \le n$ is a span in utterance. There is a natural language description for each slot type, i.e., slot semantics, denoted as $S = \{s_{i1}, s_{i2}, ..., s_{im}\}_i^t$, where m is the length of slot semantics and t is the number of slot types. In this work, we adopt questions constructed in previous work [20] as the slot semantics, and Table 1 shows some examples. In the zero-shot cross-domain slot filling task, the model is trained with data in several source domains and evaluated with data in a target domain.

2.2. Utterance Encoder and Slot Semantics Encoder

We adopt the siamese network architecture [22] which contains the shared weights encoder to obtain the utterance and the slot semantics representations respectively. By encoding separately, we can make the encoder focus on the semantics of a sentence rather than other syntactic parts [23]. It can also improve computational efficiency by avoiding too many combinations of the utterances and the slot semantics [24, 25]. And due to the powerful representation capacity of the pre-trained language model [26, 27], which has been successfully applied on NER and slot filling task [18, 28], we use a shared weights BERT [26] as the utterance encoder and the slot semantics encoder. The operation can be denoted as follow:

$$h_X = \text{BERT}([x_1, x_2, ..., x_n]), \tag{1}$$
$$h_{S_i} = \text{BERT}([s_{i1}, s_{i2}, ..., s_{im}]), \tag{2}$$

where h_X and h_{S_i} are the output representations of utterance $\{x_1, x_2, ..., x_n\}$ and the i^{th} slot semantics $\{s_{i1}, s_{i2}, ..., s_{im}\}_i$, respectively. $h \in \mathcal{R}^{n\times d}$, $h_S \in \mathcal{R}^{t\times m\times d}$, where t is the size of slot set and d is the dimension of the output representations.



P_1^{start}
P_t^{start}
P_1^{end}
P_t^{end}
Start Classifier
End Classifier
Add&Norm
Feed Forward
Add&Norm
Multi-Head Attention
Q
K
V
$\times N$
h_{X_1} h_{X_2} ... h_{X_n} $h_{S_{11}}$ $h_{S_{12}}$ ... $h_{S_{1m}}$
h_{X_1} h_{X_2} ... h_{X_n} $h_{S_{t1}}$ $h_{S_{t2}}$ ... $h_{S_{tm}}$
Slot Semantics Fusion Module
repeat & concat
Utterance Encoder
share weights
Slot Semantics Encoder
x_1 x_2 ... x_n
s_{11} s_{12} ... s_{1m}
s_{t1} s_{t2} ... s_{tm}

Figure 2: *Model structure of the proposed TSSF model. Firstly, the utterance and the slot semantics are encoded separately. Then the transformer-based slot semantics module is used to integrate the slot semantics into the utterance representation. Finally, the boundary of each slot is predicted by the start and the end classifier.*

2.3. Slot Semantics Fusion Module

Transformer-based slot semantics fusion module aims to integrate slot semantics into the utterance representation. Given the utterance representation h_X and slot semantics representations h_S from the output of utterance encoder and slot semantics encoder, we firstly repeat h_X for t times to get $h_X = \{h_{X_1}, h_{X_2}, ..., h_{X_n}\}_{i=1}^t$, then concatenate the utterance representation with each slot semantics representations and get:

$$h = \{h_{X_1}, h_{X_2}, ..., h_{X_n}, h_{S_{i1}}, h_{S_{i2}}, ..., h_{S_{im}}\}_{i=1}^t, \tag{3}$$

where $h \in \mathcal{R}^{t\times(n+m)\times d}$, then we take h as the input to the stacked transformer encoder layers [29], with the multi-head attention in transformer, slot semantics can be integrated into utterance representations explicitly. The attention mechanism can be formulated as follow:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{4}$$

and the output of the multi-head attention is calculated as:

$$h_i' = \text{MultiHead}(h_i), \tag{5}$$

$$h_i' = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_k\right) W^O, \tag{6}$$

$$\text{head}_\text{i} = \text{attention}\left(h_i W_i^Q, h_i W_i^K, h_i W_i^V\right), \tag{7}$$

where $W_i^Q, W_i^K, W_i^V \in \mathcal{R}^{d\times \frac{d}{k}}$ and $W^O \in \mathcal{R}^{d\times d}$ are trainable parameters during training step, k is the number of

Table 2: *Experimental results (slot F1) on the SNIPS dataset. Scores in each row represent the performance of the seven domains and the average slot F1-score. ATP, BR, GW, PM, RB, SCW, FSE denote the domain AddToPlaylist, BookRestaurant, GetWeather, PlayMusic, RateBook, SearchCreativeWork and FindScreeningEvent respectively. Bold numbers indicate the best performance.*

Model↓ Domain→	ATP	BR	GW	PM	RB	SCW	FSE	Average F1
CT [15]	38.82	27.54	46.45	32.86	14.54	39.79	13.83	30.55
RZT [19]	42.77	30.68	50.28	33.12	16.43	44.45	12.25	32.85
Coach [12]	50.90	34.01	50.47	32.01	22.06	46.65	25.63	37.39
QASF [20]	57.57	48.75	61.27	38.54	36.51	60.82	27.72	47.31
SLMRC [21]	**63.21**	**60.11**	65.23	50.16	32.78	55.17	30.67	51.05
TSSF (ours)	62.58	44.01	**71.92**	**51.77**	**39.34**	**78.74**	**51.59**	**57.14**

parallel heads. After multi-head attention, a feedforward neural network and residual connections are applied to get the output of the transformer encoder h', where $h' \in \mathcal{R}^{t\times(n+m)\times d}$ has the same dimension as the input feature h.

After N-layer stacked transformer encoders, we discard the slot semantics part and get the enhanced utterance representation $h' = \{h'_{i1}, h'_{i2}, ..., h'_{in}\}_{i=1}^{t}$ for each of the t slots.

2.4. Slot Entity Prediction

Given the slot semantics enhanced representation $h' = \{h'_{i1}, h'_{i2}, ..., h'_{in}\}_{i=1}^{t}$for each slot, we use two linear classifiers to predict the boundary of every slot, i.e., predict the start position and end position for each slot. The process can be denoted as follow:

$$P_i^{start} = \sigma(h'_i W_{start} + b_{start}), \tag{8}$$

$$P_i^{end} = \sigma(h'_i W_{end} + b_{end}), \tag{9}$$

where $W_{start} \in \mathcal{R}^{d\times1}$ and $b_{start} \in \mathcal{R}^1$ are model parameters of the start position classifier, $W_{end} \in \mathcal{R}^{d\times1}$ and $b_{end} \in \mathcal{R}^1$ are parameters of the end position classifier, σ is the sigmoid function, and $P_i^{start} \in \mathcal{R}^{n\times1}$ reflects the probability of each word in the utterance becoming the start position of i^{th} slot, $P_i^{end} \in \mathcal{R}^{n\times1}$ reflects the probability that each word in the utterance becoming the end position of the i^{th} slot.

2.5. Train and Test

In training stage, we minimize the binary cross-entropy loss between the predicted start probability P^{start} and ground truth start position Y^{start} for each of the t slots, this loss function can be denoted as:

$$L_{start} = \sum_{i=1}^{t}\sum_{j=1}^{n} \mathrm{BCE}(P_{ij}^{start}, Y_{ij}^{start}), \tag{10}$$

where BCE is the binary cross-entropy loss function, likewise, we can get the loss function of end position predictions:

$$L_{end} = \sum_{i=1}^{t}\sum_{j=1}^{n} \mathrm{BCE}(P_{ij}^{end}, Y_{ij}^{end}), \tag{11}$$

we sum the above two functions as our final loss:

$$L = L_{start} + L_{end}. \tag{12}$$

Note that if a slot type doesn't exist in the utterance, we map the ground truth start and end position to $[cls]$ token.

In testing stage, we predict a start, end position (j, k) for the i^{th} slot with the maximum $P_{ij}^{start} + P_{ik}^{end}$ probability, and (j, k) should also satisfy $P_{ij}^{start} > P_{i0}^{start}$ and $P_{ij}^{end} > P_{i0}^{end}$, where the P_{i0}^{start} and P_{i0}^{end} are the start probability and end probability of the $[cls]$ token, respectively. And if there are overlaps among the predictions on slots, we take the one with a higher probability.

3. Experiments

3.1. Dataset

We evaluate our model on SNIPS [14], a public spoken language understanding dataset that contains 39 slot types across seven domains (intents) – "AddToPlaylist" (ATP), "BookRestaurant" (BR), "GetWeather" (GW), "PlayMusic" (PM), "RateBook" (RB), "SearchCreativeWork" (SCW), "FindScreeningEvent" (FSE). Each domain contains around 2000 samples. We use the same setting as previous work [12, 21]: each time we choose one domain as the target domain and the other six domains as the source domains, and the first 500 samples in the target domain are used as the validation set and the remainder is used as the test set. We train the model on the source domains and evaluate it on the target domain.

3.2. Baselines

We compare our approach with a number of representative baselines: **CT** [15] first introduces slot descriptions to improve the performance on unseen slots. **RZT** [19] adds several slot example values based on CT to improve the robustness of zero-shot slot filling. **Coach** [12] designs a two-step framework and predicts the specific slot types by metric-based computation between slot spans and slot descriptions. **QASF** [20] reformulates the slot filling task as a QA problem and adopts a BERT-based QA model to extract slot spans from utterances. **SLMRC** [21] is the current state-of-the-art method for zero-shot slot filling task, which addresses slot filling as a machine reading comprehension (MRC) problem. Additionally, it uses MRC dataset SQuAD [30] for data augmentation. For a fair comparison, we report its results without the additional data.

3.3. Implementation Details

We use "bert-base-uncased[1]" model for our utterance encoder and slot semantics encoder, followed by QASF and SLMRC. We set the number of transformer encoder layers N to 5, the hidden size to 768, and the number of multi-heads to 12. The model is fine-tuned for 10 epochs with the early stop patience of 5. The batch size is set to 8. We also set a dropout rate of 0.1 [31] to avoid over-fitting. The maximum length of utterance and slot semantics is 64 and 32 respectively. We train our model

[1]https://huggingface.co/bert-base-uncased

Table 3: *Average slot F1-scores for seen and unseen parts.*

Model↓ Part→	seen	unseen
CT [15]	44.18	27.10
RZT [19]	47.15	28.28
Coach [12]	51.93	34.09
QASF [20]	56.23	41.73
SLMRC [21]	64.41	42.76
TSSF (ours)	**69.98**	**43.16**

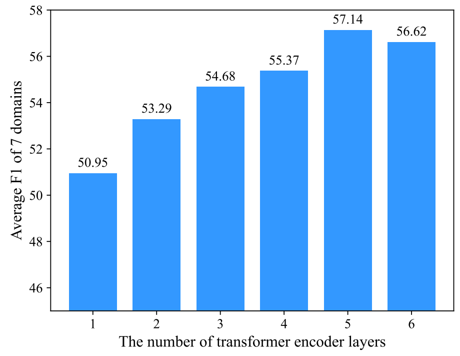


Figure 3: *Average slot F1-scores of the different number of transformer layers in transformer-based slot semantics fusion module.*

using Adam [32] optimizer with a learning rate of 1e-5 on a single RTX 2080Ti GPU. We use slot-F1 as the evaluation metric followed by [12, 20], and we fine-tune all hyper-parameters on the validation set then use the best checkpoint to test our model.

3.4. Experimental Results

Quantitative Analysis: We report the results of baselines and our approach in Table 2, which contains the F1-scores of each domain and the average F1-scores of seven domains. It can be seen that our TSSF model significantly outperforms the state-of-the-art model SLMRC in five of seven domains, and the average-F1 of seven domains is 6.09% higher than SLMRC, this demonstrates that our model can better utilize the slot semantics through the explicit fusion in the stacked transformer encoders.

Compared to SLMRC, our model has inferior performance in the "BookRestaurant" domain, we conjecture that this is because there are 14 slot types in the "BookRestaurant" domain, but only 2∼9 slot types in the other six domains, which causes inconsistency between training and testing for our model.

Analysis on Seen and Unseen Slots: To further analyze the effectiveness of our method on seen and unseen slots in the target domain, we split the test set for each domain into "seen" and "unseen" parts and test the model on both parts. An utterance is categorized into the "unseen" part as long as there is an unseen slot (i.e. the slot does not exist in the source domains). Otherwise, it is categorized into the "seen" part.

The results for the seen and unseen parts are shown in Table 3. We can observe that our model has a better performance on both seen and unseen parts. And for the seen part, our model significantly outperforms the best baseline SLMRC by 5.57%. It demonstrates that our model can better learn the different semantics of the same seen slot in different domains, such as the different semantics of the slot entity "object_name" in the "RataBook" domain and the "SearchCreativeWork" domain.

Table 4: *Ablation study of different fusion strategies. Concat denotes concatenation, DPA denotes dot-product attention, MHA denotes multi-head attention, and TSSF denotes our proposed transformer-based slot semantics module.*

Domain↓ Fusion→	Concat	DPA	MHA	TSSF
AddToPlaylist	12.66	60.23	58.71	**62.58**
BookRestaurant	1.21	38.72	38.17	**44.01**
GetWeather	5.76	64.45	62.60	**71.92**
PlayMusic	5.63	46.02	45.93	**51.77**
RateBook	3.38	33.18	30.32	**39.34**
SearchCreativeWork	41.28	69.36	**80.43**	78.74
FindScreeningEvent	10.22	34.48	35.46	**51.59**
Average F1	11.45	49.49	50.23	**57.14**

Analysis on the Number of Transformer Layers: Fig. 3 shows the performance of the different number of transformer encoder layers. As the number of layers increases from 1 to 5, we can observe that our model has a better performance from 50.95% to 57.14%, which indicates that the deeper stacked transformers have a better capacity to capture slot semantics. When the number of layers reaches 6, the performance decreases slightly, which we think is caused by the over-fitting issue due to too many introduced parameters.

3.5. Ablation Study

To further explore the impact of the transformer-based slot semantic fusion module, we compare our method with three other fusion strategies: 1) Concat: like RZT [19], we concatenate the sentence-level representations of slot semantics with the token-level utterance representations. 2) DPA (dot-product attention) [29]: we adopt the dot-product attention between token-level representations of slot semantics and utterances. 3) MHA (multi-head attention) [29]: we use the token-level utterances as the query vector, and the token-level slot semantics representations as the key and the value vector without the concatenation operation. The results are shown in Table 4, where we can observe that the Concat method has a poor performance, due to the lack of token-level interactions between utterances and slot semantics. And our transformer-based slot semantics fusion method outperforms DPA and MHA by 7.65% and 6.91% respectively, which indicates that through the stacked transformer encoders, our model can better integrate the slot semantics into the utterance representations.

4. Conclusions and Future Work

In this work, we propose a novel slot filling model with transformer-based slot semantics fusion, where the stacked transformers can integrate slot semantics into utterance representations effectively. Experimental results show that our model significantly outperforms the previous state-of-the-art method. In the future, we would like to build a joint model for slot filling and intent detection in resource-scarce scenarios.

5. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62176182.

6. References

[1] G. Tur and R. De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

[2] L. Qin, T. Xie, W. Che, and T. Liu, "A survey on spoken language understanding: Recent advances and new frontiers," in *IJCAI*, 2021.

[3] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2014.

[4] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm." in *Interspeech*, 2016, pp. 715–719.

[5] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, "Slot-gated modeling for joint slot filling and intent prediction," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 753–757.

[6] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[7] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, "A stack-propagation framework with token-level intent detection for spoken language understanding," 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)*, 2019, pp. 2078–2087.

[8] A. Siddique, F. Jamour, and V. Hristidis, "Linguistically-enriched and context-aware zero-shot slot filling," in *Proceedings of the Web Conference 2021*, 2021, pp. 3279–3290.

[9] E. Ferreira, B. Jabaian, and F. Lefevre, "Zero-shot semantic parser for spoken language understanding," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[10] S. Schuster, S. Gupta, R. Shah, and M. Lewis, "Cross-lingual transfer learning for multilingual task oriented dialog," 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)*, 2019, pp. 3795–3805.

[11] A. Siddique, F. Jamour, L. Xu, and V. Hristidis, "Generalized zero-shot intent detection via commonsense knowledge," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1925–1929.

[12] Z. Liu, G. I. Winata, P. Xu, and P. Fung, "Coach: A coarse-to-fine approach for cross-domain slot filling," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 19–25.

[13] Y. Yan, J. Ye, Z. Zhang, and L. Wang, "Aisfg: Abundant information slot filling generator," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 4180–4187.

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

[15] A. Bapna, G. Tür, D. Hakkani-Tür, and L. Heck, "Towards zero-shot frame semantic parsing for domain scaling," *Proc. Interspeech 2017*, pp. 2476–2480, 2017.

[16] K. He, J. Zhang, Y. Yan, W. Xu, C. Niu, and J. Zhou, "Contrastive zero-shot learning for cross-domain slot filling with adversarial attack," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 1461–1467.

[17] L. Wang, X. Li, J. Liu, K. He, Y. Yan, and W. Xu, "Bridge to target domain by prototypical contrastive learning and label confusion: Re-explore zero-shot learning for slot filling," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 9474–9480.

[18] X. Wei, Y. Si, S. Wang, L. Wang, and J. Dang, "Hierarchical Tagger with Multi-task Learning for Cross-domain Slot Filling," in *Proc. Interspeech 2022*, 2022, pp. 3273–3277.

[19] D. Shah, R. Gupta, A. Fayazi, and D. Hakkani-Tur, "Robust zero-shot cross-domain slot filling with example values," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5484–5490.

[20] X. Du, L. He, Q. Li, D. Yu, P. Pasupat, and Y. Zhang, "Qa-driven zero-shot slot filling with weak supervision pretraining," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2021, pp. 654–664.

[21] J. Liu, M. Yu, Y. Chen, and J. Xu, "Cross-domain slot filling as machine reading comprehension: A new perspective," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 673–685, 2022.

[22] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[23] J. Hong, J. Park, D. Kim, S. Choi, B. Son, and J. Kang, "Tess: Zero-shot classification via textual similarity comparison with prompting using sentence encoder," *arXiv preprint arXiv:2212.10391*, 2022.

[24] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 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)*, 2019, pp. 3982–3992.

[25] P. Yang, X. Cong, Z. Sun, and X. Liu, "Enhanced language representation with label knowledge for span extraction," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 4623–4635.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, 2019, pp. 4171–4186.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[28] J. Ma, M. Ballesteros, S. Doss, R. Anubhai, S. Mallya, Y. Al-Onaizan, and D. Roth, "Label semantics for few shot named entity recognition," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 1956–1971.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[30] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 784–789.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.