Interspeech 2022
18-22 September 2022, Incheon, Korea

mcBERT: Momentum Contrastive Learning with BERT for Zero-Shot Slot Filling

Seong-Hwan Heo[†*], *WonKee Lee*[‡*], *Jong-Hyeok Lee*[†‡]

[†]Graduate School of Artificial Intelligence,
[‡]Department of Computer Science and Engineering,
Pohang University of Science and Technology (POSTECH), Republic of Korea
{hursung1, wklee, jhlee}@postech.ac.kr

Abstract

Zero-shot slot filling has received considerable attention to cope with the problem of limited available data for the target domain. One of the important factors in zero-shot learning is to make the model learn generalized and reliable representations. For this purpose, we present mcBERT, which stands for **'m'**omentum **'c'**ontrastive learning with **BERT**, to develop a robust zero-shot slot filling model. mcBERT uses BERT to initialize the two encoders, the query encoder and key encoder, and is trained by applying momentum contrastive learning. Our experimental results on the SNIPS benchmark show that mcBERT substantially outperforms the previous models, recording a new state-of-the-art. Besides, we also show that each component composing mcBERT contributes to the performance improvement.

Index Terms: slot filling, zero-shot learning, momentum contrastive learning, task-oriented dialogue

1. Introduction

Slot filling, an essential module in a goal-oriented dialog system, seeks to identify contiguous spans of words belonging to domain-specific slot types in a given user utterance. For instance, given a user utterance *"play some 70's music in youtube music"* belonging to the `Play music` domain[1], the goal is to identify **slot entities**: *'70's'* and *"youtube music"* that correspond to the **slot types**, `year` and `service`, respectively.

Traditionally, slot filling models have relied on supervised learning by using labeled training data [1, 2, 3, 4]. Although having shown promising results for learned domains and slot types, these approaches require a significant amount of labeled data, which remains a chronic problem in developing robust systems. Moreover, from a practical perspective, it is very natural to expect that any new domains (or new slot types), on which the model has not trained, can be issued to the dialog system. Consequently, it is advisable for the model to maintain the ability to yield seamless predictions, even for domains and slot types that are rarely or even never learned.

In this regard, numerous recent studies focusing on **zero-shot** (and few-shot) slot filling have emerged to cope with limited training data. In particular, **cross domain framework** [5, 6, 7] has received much attention for zero-shot slot filling. A basic concept of this framework is to leverage general knowledge across diverse domains when making predictions on a target domain, and for which a slot filling model is trained using the collection of labeled data from all available domains.

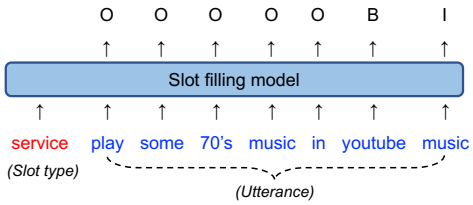


Figure 1: *Schematic overview of slot filling under the cross domain framework. 'B' and 'I' represent words corresponding to 'service', whereas 'O' represents a word not corresponding to 'service'.*

In addition, to enable zero-shot slot filling (especially to handle unseen slot types), a slot type and utterance are fed into the model simultaneously (Figure 1) so that the model uses their semantic relationship (i.e., joint representation) to discover the slot entities corresponding to the given slot type.

In recent years, great attention has been paid to **contrastive learning** for zero-shot learning in various research areas [8, 9, 10], and several previous studies [6, 7] have also shown that contrastive learning is beneficial for zero-shot slot filling. Basically, contrastive learning aims to construct contrastive samples, which include a single positive sample and several negative samples in response to the model input (also called anchor), and then let the model learn the representation of the anchor similar to the positive sample but dissimilar to all negative samples. Particularly, in the computer vision field, **momentum contrast learning** [11] has achieved great success in learning unsupervised visual representations; a key feature is to slowly update the **key encoder** (by which contrastive examples are encoded) using a momentum-based moving average of the **query encoder** (by which the anchor is encoded) to derive consistent representations between the anchor and contrastive samples.

In this paper, we present **'m'**omentum **'c'**ontrastive learning with **BERT** (**mcBERT**) for the zero-shot slot filling. Beyond the visual representation, we hypothesize that momentum contrastive learning also has the potential to allow the model to learn adequate representations for zero-shot slot filling. Moreover, to maximize the zero-shot effect, we utilize BERT [12], a pre-trained language model with the capability of generating reliable and generalized linguistic representations, to initialize our query and key encoders. Also, we consider two methods to construct contrastive samples by modifying the utterance: (1) the first, inspired by [6], is **template sample**, in which the slot entities in the utterance are replaced with slot types. (2) the second, inspired by [13], is **synthetic sample**, in which the slot

[*] Equal contribution to this work
[1]Sometimes 'domain' is interchangeable with 'intent'. However, we use the term 'domain' throughout this paper.

10.21437/Interspeech.2022-839

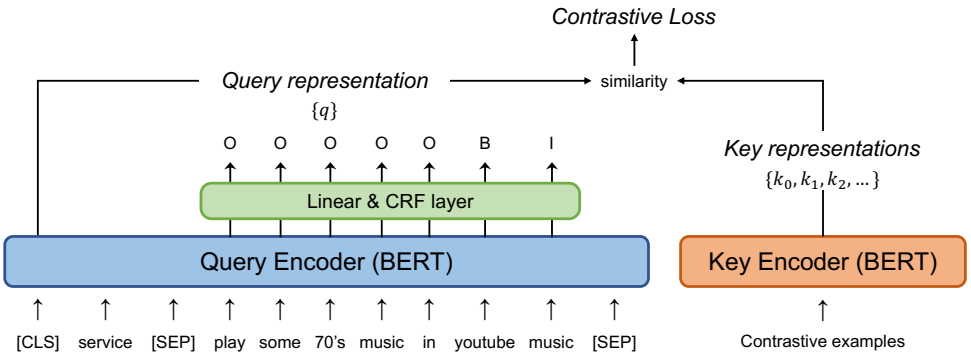


Figure 2: *The overall architecture of mcBERT.*

entities in the utterance are replaced with different slot entities.

Our experimental results on the SNIPS benchmark [14] revealed that mcBERT outperforms previous state-of-the-art models by a significant margin across all domains, both in zero-shot and few-shot settings, and we confirmed that each component we propose contributes to the performance improvement.

2. Approach

This section details our proposed method, mcBERT (Figure 2). First, we introduce the overall process of our BERT-based slot filling model, which operates under the cross-domain framework. Next, we introduce a method to train our slot filling model by applying momentum contrastive learning, and for which we also introduce a method to construct contrastive samples.

2.1. Slot Filling Model

Slot filling is performed by using the **query encoder**'s outputs. We use BERT as the query encoder because we expect that leveraging the knowledge transferred from BERT, which is a language model trained on a huge amount of plain text data, will significantly aid the model in learning proper representations for the slot filling task, especially for zero-shot slot filling.

Let t, $\mathbf{w} = (w_1, \cdots, w_n)$, and $\mathbf{y} = (y_1, \cdots, y_n)$ denote a slot type, an utterance including n words, and a sequence of B/I/O labels corresponding to each y_i, respectively. According to the BERT configuration, we configure the model input in the form of "`[CLS]` t `[SEP]` $\mathbf{w}$ `[SEP]`" to accept both t and $\mathbf{w}$. After receiving this input, the query encoder outputs the hidden states H with respect to $\mathbf{w}$, which also include information of t:

$$H = [h_1, \cdots, h_n]. \tag{1}$$

Then, H is forwarded to the linear projection layer to get the logits for the B/I/O labels:

$$[\ell_1, \cdots, \ell_n] = \text{Linear}(H). \tag{2}$$

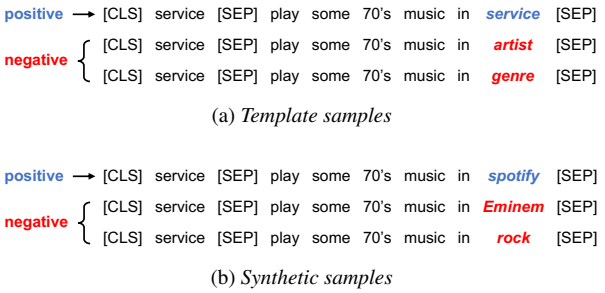
Subsequently, the CRF[2] layer receives these logits to compute the label sequence score:

$$score(\tilde{\mathbf{y}}|\mathbf{w}) = \text{CRF}\left([\ell_1, \cdots, \ell_n]\right). \tag{3}$$

At the inference, the model selects a label sequence that maximize the score:

$$\mathbf{y}^* = \arg\max_{\tilde{\mathbf{y}}} \left(score(\tilde{\mathbf{y}} \mid \mathbf{w})\right). \tag{4}$$

[2]To avoid clutter, we omit detailed calculation on CRF.



(a) *Template samples*

(b) *Synthetic samples*

Figure 3: *Two types of contrastive samples given the anchor "`[CLS]` service `[SEP]` play some 70's music in youtube music `[SEP]`", whose slot entities are "youtube music".*

During training, the training object is to minimize the negative log likelihood with respect to the score as follows:

$$\mathcal{L}_{BIO} = -\log \frac{score(\tilde{\mathbf{y}} = \mathbf{y} \mid \mathbf{w})}{\sum_{\hat{\mathbf{y}}} score(\hat{\mathbf{y}} \mid \mathbf{w})} \tag{5}$$

2.2. Momentum Contrastive Learning

2.2.1. Contrastive sample construction

Given the model input (i.e., anchor, which is the input to the query encoder), it is necessary to construct contrastive samples with size[3] m, containing a single *positive* sample (similar to the anchor) and $m-1$ *negative* samples (dissimilar to the anchor) so as to conduct contrastive learning. We suggest two methods (Figure 3) for sample construction by modifying the given utterance.

The first approach, **template sample** (Figure 3a), replaces the slot entities in the utterance with a given slot type for the positive sample, but with a different slot type (randomly selected from data) for the negative samples. This is to construct an utterance representation in which the representation of the slot type is encoded explicitly by lexicalizing the slot type into the utterance. Thus, by mapping a given utterance to the positive sample yet straying from the negative samples, it is likely to create a representation of slot entities similar to its corresponding slot type. Consequently, we expect that the model will

[3]Following previous studies [6, 7], we set $m = 3$ in our experiments.

more easily identify the slot entities corresponding to a given slot type.

The second approach, **synthetic sample** (Figure 3b), replaces the slot entities in the utterance with different slot entities (randomly selected from data), the slot types of which are still identical to a given slot type for the positive sample but different for the negative samples. The intuition behind this approach is to construct positive samples containing different slot entities that are also likely to appear in a given utterance but to construct negative samples containing unnatural slot entities. By using these samples in training, we expect the model to eventually better distinguish whether each word in an utterance is convincing regarding a given slot type.

To utilize both approaches, we empirically configure the contrastive samples by taking either template type or synthetic type at random with equal probabilities; we also analyze the effect on individual types and a different configuration, as detailed in Section 4.2.

2.2.2. Training process

To perform contrastive learning, the **key encoder**[4] serving as a counterpart to the query encoder is necessary, being used to represent contrastive samples. The key encoder is architecturally equivalent to the query encoder (i.e., the key encoder is also BERT) to make the representation of contrastive samples consistent with the anchor.

We use `[CLS]`, which encompasses all information on slot type and utterance, to construct the representation for both the anchor and contrastive samples. Namely, the query encoder produces **query representation** q from `[CLS]` in the anchor, and the key encoder produces **key representations** $K = \{k_1, \cdots, k_m\}$ from m contrastive samples' `[CLS]`; the contrastive loss can be defined as

$$\mathcal{L}_{CL} = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{k \in K} \exp(q \cdot k/\tau)}, \tag{6}$$

where τ is a temperature hyperparameter and k_+ is a single key regarding the positive sample that matches q. In short, minimizing this loss is intended to classify q as k_+.

Finally, apart from the key encoder, the model is trained to minimize the combined loss by using both $\mathcal{L}_{BIO}$ (Eq. 5) and $\mathcal{L}_{CL}$ (Eq. 6):

$$\mathcal{L} = \lambda \mathcal{L}_{CL} + (1 - \lambda)\mathcal{L}_{BIO}, \tag{7}$$

where $\lambda \in [0, 1]$ is a hyperparameter to determine the ratio between two losses. Following the original work [11], the key encoder is trained with a momentum update as follows:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q, \tag{8}$$

where $m \in [0, 1)$ is a momentum coefficient; θ_q and θ_k denote the parameters of the query encoder and key encoder, respectively. This momentum update helps prevent θ_k from rapidly changing by keeping most of θ_k (the original work set $m = 0.999$) while taking only a small portion of θ_q so that the key encoder can produce a consistent representation.

3. Experiments

Dataset. We used the SNIPS benchmark [14] to train and evaluate our model. SNIPS consists of 39 slot types across 7 domains, including approximately 2K samples per domain. We tokenized all words in the dataset into sub-word units using the BERT tokenizer[5].

Model Configuration. We used the BERT model `bert-base-uncased`[5] to implement our query and key encoders. For training, we used the following hyperparameters: the AdamW optimizer [15] with $\beta = (0.9, 0.999)$, learning rate of 1e-5, 4K warm-up steps followed by linear decay with 400K maximum training steps, a dropout probability of 30%, and batch size of 128 samples. We followed the original work [11] to configure the hyperparameters belonging to the momentum contrastive learning: $\tau = 0.07$ (Eq. 6) and $m = 0.999$ (Eq. 8).

Training Details. For a fair comparison, we followed the data setup described in [6], which has also been adopted by other baselines: (1) for each domain (i.e., the target domain) in SNIPS, the other six domains are selected as the source domains used for training; (2) when conducting zero-shot learning, the data from the target domain are never used for training, 500 samples in the target domain are used for the development data, and the remainder are used as the test data; and (3) when conducting few-shot learning, 50 samples from the target domain are used along with those from source domains for training; the development and test data configurations are the same as for zero-shot learning. We trained mcBERT using a single NVIDIA RTX A5000 GPU, and it took approximately 2 hours to reach convergence.

Baselines. In our experiments, we compared our mcBERT with the following baselines, including the current state-of-the-art system:

- **Concept Tagger (CT)** [5]: a typical early work on cross-domain slot filling, where the RNN-based model receives both slot type and utterance simultaneously.
- **Robust Zero-shot Tagger (RZT)** [13]: a method to improve CT, where examples of the slot entities are additionally provided to the model.
- **Coach** [6]: a method that equips two separate layers for detecting slot entities and slot type, respectively, and additionally provides template samples (similar to ours) to compute the regularization loss for model training.
- **CZSL-Adv** [7]: current state-of-the-art (to our knowledge) model, which improved Coach by adopting contrastive learning (but not momentum contrastive learning) and by adding adversarial noise to the input.

4. Results

4.1. Main Results

The evaluation results (Table 1) reveal that our proposed mcBERT is superior to all the baselines by a substantial margin in both zero-shot and few-shot settings for all domains, achieving a new state-of-the-art. Note that the results for mcBERT used $\lambda = 0.5$ for the zero-shot learning and $\lambda = 0.1$ for the few-shot learning when configuring the combined loss (Eq. 7), which resulted in the best performing model. This indicates that momentum contrastive learning has a greater effect on zero-shot

[4]Note that the key encoder is not used for inference, only for training.

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

Table 1: *Experimental results on zero-shot and few-shot (with 50 samples) settings for the baselines and mcBERT in terms of F1-score. The best results in each row (each domain) for zero-shot and few-shot learning are highlighted in **bold**. The last rows are average F1-scores over all domains.*

Domains	**Zero-shot**					**Few-shot** (50 samples)				
	CT	RZT	Coach	CZSL-Adv	mcBERT	CT	RZT	Coach	CZSL-Adv	mcBERT
AddToPlaylist	38.82	42.77	50.90	53.89	**60.09**	68.69	74.89	74.68	76.18	**86.57**
BookRestaurant	27.54	30.68	34.01	34.06	**71.64**	54.22	54.49	74.82	76.28	**84.82**
GetWeather	46.45	50.28	50.47	52.24	**88.98**	63.23	58.87	79.64	83.28	**96.04**
PlayMusic	32.86	33.12	32.01	34.59	**78.50**	54.32	59.20	66.38	68.17	**89.62**
RateBook	14.54	16.43	22.06	31.53	**50.43**	76.45	76.87	84.62	87.22	**90.75**
SearchCreativeWork	39.79	44.45	46.65	50.61	**76.35**	66.38	67.81	64.56	66.49	**88.95**
FindScreeningEvent	13.83	12.25	25.63	30.05	**62.86**	70.67	74.58	83.85	83.26	**87.16**
Average F1	30.55	32.85	37.39	40.99	**69.84**	64.85	66.67	75.51	77.27	**89.13**

Table 2: *Results on ablation study for each component used in mcBERT in terms of average F1-score. The absence of BERT indicates that both encoders were trained from scratch. The absence of $MoCo$ indicates that contrastive learning is performed with gradient updates instead of momentum updates.*

Settings	Zero-shot	Few-shot
mcBERT	**69.84**	**89.13**
$-$ $MoCo$	68.45	88.99
$-\mathcal{L}_{CL}$	66.21	88.62
$-$ BERT	35.25	59.22
$-$ BERT $-$ $MoCo$	33.99	54.93
$-$ BERT $-\mathcal{L}_{CL}$	35.92	60.02

Table 3: *Comparison depending on the configuration of the contrastive samples. Each result represents average F1-score.*

Contrastive samples	Zero-shot	Few-shot
Template	67.05	**89.46**
Synthetic	68.51	88.83
Concat (Template, Synthetic)	68.29	88.94
Random (Template, Synthetic)	**69.84**	89.13

learning than few-shot learning. We surmise that contrastive learning is less helpful in few-shot learning because the model can learn a suitable representation to some degree using at least a few data from the target domain.

4.2. Analysis

We first conducted an ablation study to examine the effect of each component applied to mcBERT. As shown in Table 2, we confirmed that both BERT and momentum contrastive learning contributed to performance improvement. The improvement gain was significant when BERT was used, which can be reasonable given that BERT has already been shown to be helpful in various tasks. Furthermore, we observed that contrastive learning with momentum updates gave a greater performance enhancement than with gradient updates. On the other hand, we found that contrastive learning was ineffective when training the encoders from scratch without using BERT, speculating that the given data were insufficient to train both the query and key encoder simultaneously.

Next, we investigated the effects of using individual types (i.e., template and synthetic types) for constructing contrastive samples and also using a different combination from that described in Section 2.2.1. In Table 3, $\mathrm{Concat}(\cdot)$ simply combines two types of samples, but the positive sample is randomly selected from one of the two types (as only a single positive sample is required); $\mathrm{Random}(\cdot)$ is what we used to train mcBERT. Consequently, we found that regardless of the configuration used, the performance was improved consistently; $\mathrm{Random}(\cdot)$ achieved the best result for zero-shot learning, but using template type solely showed a slightly better result for few-shot compared to $\mathrm{Random}(\cdot)$.

5. Related Work

Coach [6] introduced *template regularization*, which can be conceptually similar to contrastive learning (but is not the same), and for which they constructed template examples in a similar manner to ours. CZSL-Adv [7] applied contrastive learning for training; their method differs from ours in that they did not apply momentum updates and their contrastive loss is calculated based on distance.

6. Conclusions

In this paper, we propose mcBERT especially focusing on zero-shot slot filling. We attempt to apply momentum contrastive learning and use BERT to initialize our encoders. To apply contrastive learning, we suggest constructing contrastive samples of two types: template and synthetic samples. We found that our findings successfully contribute to the performance improvement and eventually achieved new a state-of-the-art. This may stem from learning suitable representations, that is, we believe that our proposed model stably learns to cluster the slot types and their corresponding slot entities and captures semantic patterns precisely by momentum contrastive learning and BERT's representation. We believe that our approach is applicable in various sequence labeling tasks, such as named entity recognition, which remains future work.

7. Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

8. References

[1] S. J. Young, "Talking to machines (statistically speaking)," in *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*, J. H. L. Hansen and B. L. Pellom, Eds. ISCA, 2002.

[2] J. R. Bellegarda, "Spoken language understanding for natural interaction: The siri experience," in *Natural Interaction with Robots, Knowbots and Smartphones*, J. Mariani, S. Rosset, M. Garnier-Rizet, and L. Devillers, Eds. New York, NY: Springer New York, 2014, pp. 3–14.

[3] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, "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, 2015.

[4] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder LSTM for semantic slot filling," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2077–2083.

[5] A. Bapna, G. Tur, D. Hakkani-Tur, and L. Heck, "Towards zero shot frame semantic parsing for domain scaling," in *Interspeech 2017*, 2017.

[6] 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*. Association for Computational Linguistics, Jul. 2020, pp. 19–25.

[7] 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*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 1461–1467.

[8] J. Wang and B. Jiang, "Zero-shot learning via contrastive learning on dual knowledge graphs," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 885–892.

[9] M. U. Anwaar, R. A. Khan, Z. Pan, and M. Kleinsteuber, *A Contrastive Learning Approach for Compositional Zero-Shot Learning*. New York, NY, USA: Association for Computing Machinery, 2021, p. 34–42.

[10] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer, "VideoCLIP: Contrastive pre-training for zero-shot video-text understanding," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6787–6800.

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[12] 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[13] D. J. Shah, R. Gupta, A. A. Fayazi, and D. Hakkani-Tur, "Robust zero-shot cross-domain slot filling with example values," in *Proceedings of ACL, 2019*, 2019.

[14] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *CoRR*, vol. abs/1805.10190, 2018.

[15] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.